UNIVERSIDAD CARLOS III DE MADRID

PROYECTO FIN DE CARRERA

INGENIERÍA TECNICA INDUSTRIAL: ELECTRÓNICA INDUSTRIAL

ACELERACIÓN HARDWARE PARA LA EXTRACCIÓN DE CARACTERÍSTICAS EN SEÑALES PROCEDENTES DE ECG

AUTOR: Dª Laura García Mateos

DIRECTOR: Dª Marta Portela García

CO-DIRECTOR: Dª Marta Ruiz Llata

ESCUELA POLITECNICA SUPERIOR

Departamento de Tecnología electrónica

Leganés, Julio de 2013.

RESUMEN

En este trabajo se presenta un estudio de la implementación hardware del cálculo de los cumulantes de segundo y cuarto orden, para ser empleados en la extracción de características dentro de un proceso completo de selección, y extracción de características para la clasificación de señales procedentes de electrocardiogramas (ECG).

Estos cumulantes son los que se derivan de estudios como el de [Gua 12], que implementa un sistema completo de selección y clasificación de características en sistemas embebidos.

Para la realización de este estudio se emplean datos reales procedentes de una de las mayores bases de datos que proporcionan señales procedentes de electrocardiogramas (MIT/BIH Arrythmia). A partir de estas señales se llega a la obtención de las características más significativas de cada uno de los datos reduciendo el volumen de información a través del cálculo de los cumulantes de segundo y cuarto orden.

Con este estudio lo que se pretende es comprobar si es posible una aceleración a través de la implementación Hardware del cálculo de estos dos cumulantes ya mencionados.

Finalmente se presentarán los resultados obtenidos de la aceleración que se ha conseguido gracias al diseño propuesto a través de VHDL (Hardware Description Language), que ha permitido realizar, a través de una herramienta de simulación como es ModelSim, diferentes simulaciones para comprobar si la aceleración era posible o no.

Además se añaden las líneas de investigación futuras, que se consideran más importantes basándose en las posibles mejoras que se puedan realizar al diseño propuesto.

ÍNDICE

Capítulo 1: Motivación y Objetivos

1.1. Motivación

El desarrollo de aplicaciones para el diseño de sistemas automáticos de detección y clasificación de señales es un área en la que se pueden encontrar diversos estudios, ya que el aumento de la aparición de herramientas para el procesamiento de datos ha supuesto un gran avance para este campo de investigación, debido a que estas herramientas agilizan el procesamiento de grandes cantidades de datos.

Los estudios de estos sistemas automáticos de detección y clasificación de señales se encuentran enfocados a la clasificación de señales procedentes de diferentes fuentes de información.

Una de los ámbitos de estudio con mayor influencia es el relativo a la clasificación de señales procedentes de un electrocardiograma (ECG), este es un tema de gran interés, ya que hay ocasiones en que un paciente necesita un control de larga duración del comportamiento eléctrico de su corazón. Este control de las pulsaciones cardíacas del paciente en un periodo de tiempo elevado, implica un gran volumen de información que posteriormente será necesaria interpretar por expertos. Al tratarse de una cantidad considerable de señales esto requerirá por parte del experto muchas horas de trabajo.

Para solucionar el problema de interpretación y complejidad de grandes cantidades de información, se estudia el uso de estos sistemas automáticos de detección y clasificación de las señales. Con esto se podría reducir el tiempo de espera en el diagnóstico de una anomalía cardíaca en el paciente, por lo que el inicio de un posible tratamiento sería más rápido y en ocasiones de vital importancia.

Además es interesante poder llegar a obtener una aceleración hardware del sistema, para que contribuya a que el diagnostico pueda ser más eficaz desde el punto de vista del tiempo que se tarda en diagnosticar. Si se puede conseguir que una parte del sistema se acelere de forma hardware, se podría trasladar el sistema completo.

1.2. Objetivos.

Aunque hay muchos estudios destinados al estudio de las señales procedentes de electrocardiogramas como [Meh 07], [Jal 09], [Son 05], [Kar 10], [Yin 12], [Can 10], en esta ocasión el trabajo descrito en este caso se basará en la investigación desarrollada por [Gua 12], ya que en este se trata de manera detallada la importancia del uso de estadísticos de alto orden (HOS) para la extracción de las características más importantes de las señales a analizar.

El objetivo de este Proyecto Fin de Carrera es llevar a cabo la parte de la extracción de características, realizando un diseño en VHDL (Hardware Description Language) que permita obtener una buena aceleración a través de hardware.

Es importante tener en cuenta que únicamente se realizará la implementación de los estadísticos de alto orden descritos por los estudios previos. Con esto se pretende conseguir la aceleración de una parte importante del proceso que engloba la selección y la extracción de características de forma automática.

Para realizar esta parte de la extracción de características se usará el cálculo de los cumulantes de segundo y cuarto orden, empleando las ecuaciones a las que [Gua 12] hace referencia en su trabajo.

Estas ecuaciones son un caso particular para la extracción de las características en el tratamiento de señales procedentes de electrocardiogramas, ya que otras señales pueden requerir un tratamiento y unos cálculos totalmente diferentes.

1.3. Estructura del documento.

El documento se encuentra dividido en 5 capítulos, que pretenden dar un conocimiento del estudio realizado desde los conceptos más generales hasta llegar a las particularidades del diseño que se propone, para terminar con los resultados experimentales y las conclusiones finales.

En primer lugar se presenta en el *Capítulo 1: Motivación y Objetivos* que muestra la motivación que ha llevado a desarrollar este proyecto, así como los objetivos que se quieren alcanzar.

Se continuará con el *Capítulo 2: Conceptos básicos y trabajo* previo. que explica los conocimientos necesarios para entender de forma más concreta cual será el diseño que se quiere realizar.

En el *Capítulo 3: Diseño* se desarrolla todo el diseño que se ha realizado, se explica cada parte intentando hacerlo lo más detallado posible.

El documento acaba con el *Capítulo 4: Resultados experimentales* y el *Capítulo 5: Conclusiones*, donde en el primero de ellos se muestran todos los resultados obtenidos del diseño. Y en el último se muestran las conclusiones a las que se ha llegado y se muestran cuáles pueden ser alguna de las líneas de investigación futuras.

Capítulo 2: Conceptos básicos y trabajo previo.

2.1. Situación general.

Una de las formas de diagnóstico de anomalías del sistema circulatorio se realiza a través de la interpretación de las señales que produce el corazón. Estas señales se pueden representar a través del electrocardiograma (ECG), que hace una interpretación del comportamiento eléctrico que tiene el corazón. El ECG realiza una representación gráfica de la actividad electica del corazón, se trata de una representación de la contracción cardíaca para la que se emplean pequeños electrodos que captan, amplifican y registran las señales del latido del corazón. Un ejemplo real de esta representación gráfica, puede verse en la Figura 1.

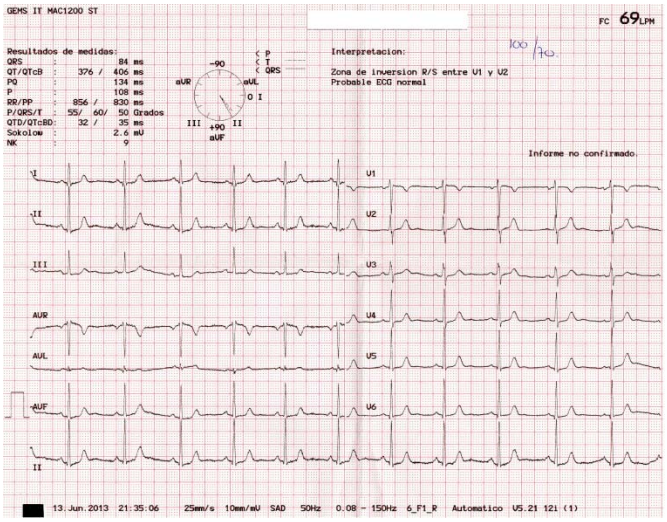

Figura 1. Representación gráfica de un ECG real

Estas señales contienen multitud de pulsos, cada uno de los picos que se observan en la gráfica representada en la Figura 1 se corresponden con una pulsación del corazón.

El ECG es una prueba médica que debe llevarse a cabo por profesionales y suele ser el primer examen que se realiza a cualquier paciente que sea susceptible de padecer alguna cardiopatía.

La señal normal de un ECG se representa a continuación (Figura 2), donde podemos ver las zonas más significativas en una señal de ECG. Al inicio del pulso se observa una onda P, en la parte central se encuentra el trazado QRS y por último la onda T, estas zonas se corresponden con la representación gráfica de actividades electicas concretas del corazón.

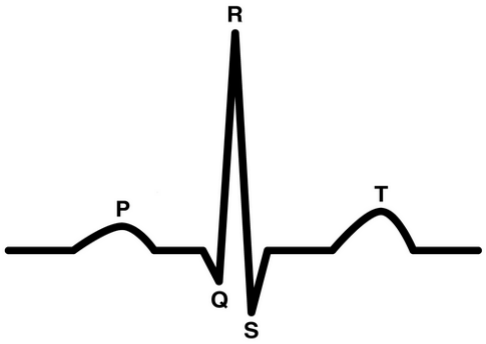

Figura 2. Representación de pulsación normal de un ECG

Una anomalía en el comportamiento de alguno de los factores de la pulsación, derivan en un diagnóstico de trastorno cardíaco.

Los trastornos en el ritmo cardíaco derivados de la frecuencia cardíaca (pulso) o un patrón irregular de las pulsaciones, son conocidos como arritmias. Estos trastornos pueden derivar en enfermedades importantes si no son tratados a tiempo, por ello la detección de estas anomalías es muy importante para poder dar un diagnóstico preciso, rápido y fiable.

La aparición de una arritmia puede deberse a varios factores diferentes. Se puede presentar una arritmia si los impulsos que controlan los latidos del corazón sufren alguna alteración, un retraso o bloqueo, esto ocurre si las células que producen estos impulsos no funcionan correctamente o si los impulsos no se desplazan normalmente a través del corazón.

Otra causa de arritmia es la aparición en otra parte del corazón de impulsos eléctricos. Estos impulsos sumados con los pulsos generados de forma normal por el corazón, lo que hace que los latidos normales del corazón se vean alterados.

Por ejemplo, el hecho de que los ventrículos no se contraigan de forma simultánea, genera una pulsación con bloqueo de ramal izquierdo o derecho, esto es

debido a que se produce un fallo en la transmisión de las señales eléctricas del corazón.

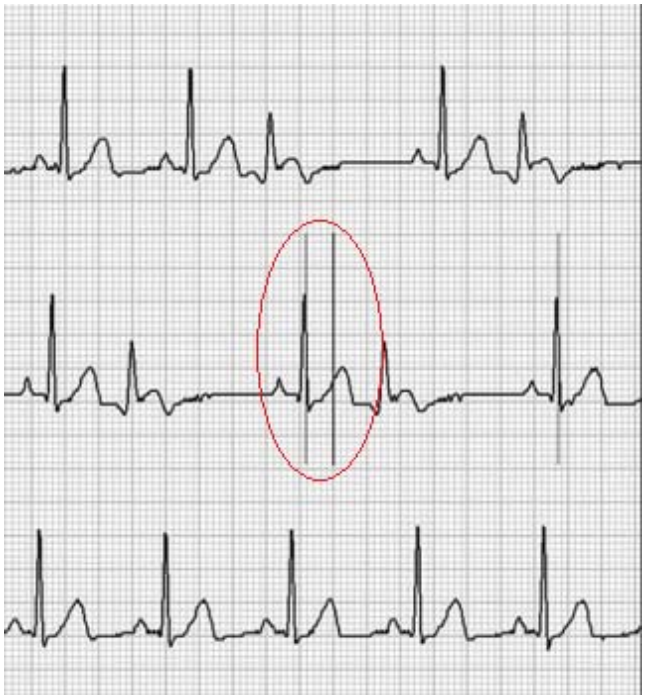

Figura 3. Señal de ECG con anomalía

En la Figura 3anterior se muestra la señal de un electrocardiograma realizado a un paciente, y se puede observar la aparición de anomalías en el comportamiento eléctrico del corazón.

Si comparamos esta figura con la Figura 1. Representación gráfica de un ECG real, se aprecia una diferencia en los picos de pulsación P y en la onda final T. Estos resultados deben ser valorados por un experto cardiólogo, pero lo que se pretende con este estudio es hacer una aportación a la obtención de un sistema automático de clasificación de anomalías cardíacas.

2.2. Sistemas automáticos para la detección y la clasificación de anomalías cardíacas.

Son numerosos los estudios realizados sobre sistemas de detección y clasificación de señales, los datos de estas señales de estudio son muy diversos, pero muchos de ellos se basan precisamente en señales procedentes de los resultados obtenidos de electrocardiogramas.

Autores como [Yin 12], [Jal 09], [Son 05], [Can 10] entre otros, realizan un diseño del proceso de detección y clasificación de anomalías en las señales del

electrocardiograma, pero también hay estudios que realizan estos diseños de forma automática basados en la utilización de una FPGA.

Estudios como es el caso de [Pao 12] se centra en realizar el diseño de la extracción y clasificación de características implementándolo en una FPGA para poder realizar el proceso de clasificación de forma más rápida y eficaz. Este estudio utiliza el reconocimiento automático de ECGs para la identificación de personas.

O también como es el caso de [Mat 10] donde desarrolla una aplicación Hardware - Software(HW/SW) y demostrando que esta aplicación desarrollada hace posible el uso de dispositivos FPGA de bajo coste en los sistemas donde se requiera una alta velocidad de procesamiento.

2.2.1. Proceso de diseño de un sistema automático de detección de anomalías cardíacas.

El análisis de las señales de ECG para la obtención de información ha generado diversos estudios al respecto a través de herramientas computacionales como [Meh 07], [Jal 09], [Son 05], entre otros.

Todo proceso de identificación de señales está dividido en dos grandes bloques: la obtención de los datos y el reconocimiento de estos (Figura 4yFigura 5).

La obtención de datos, se encuentra dividida en tres fases Figura. La primera de ellas consiste en la adquisición de señales, en el caso de detección de anomalías cardíacas estas señales serán las procedentes de un electrocardiograma, en donde por medio de electrodos y circuitos de instrumentación se hace una medida de los cambios del comportamiento eléctrico del corazón. La mayoría de los estudios de señales de ECG toman sus datos de bases de datos ya existentes.

A continuación se realiza una selección de las características de forma que estas características sean las que mejor definan las diferentes señales del electrocardiograma. Esta selección de características supone una forma de reducir la señal y así eliminar la información redundante, manteniendo todas las propiedades relevantes sin perder información crítica. En la última fase se realiza un entrenamiento para la clasificación de las señales a partir de las características anteriores.

Figura 4. Descripción de los pasos a seguir para la obtención de datos

En el segundo bloque de pasos dentro del proceso de identificación de señales, se tiene una visión general de cómo llevar a cabo el reconocimiento de las distintas clases.

Las fases que componen el reconocimiento de las distintas clases de una señal se muestran en la Figura 5. En este bloque se lleva a cabo la obtención de las características más representativas (primera fase), para extraer de los datos el menor número de características que nos proporcione la mayor cantidad de información (segunda fase) y que nos permita discriminarlos correctamente en el proceso de reconocimiento de datos.

Para finalizar el proceso se emplea un clasificador (tercera fase del proceso de reconocimiento de señales), que será el encargado de brindar un diagnóstico del tipo de clase al que pertenece la señal analizada.

Figura 5. Esquema de bloques de la etapa de reconocimiento de clases.

Estas dos figuras (Figura 4 y Figura 5) son las etapas que deben plantearse a la hora de desarrollar cualquier tipo de sistema basado en la obtención de señales y su posterior clasificación, ya que todos estos sistemas deben realizar un procesamiento de la información de entrada, de manera que se obtengan unas características significativas para poder realizar una clasificación fiable.

2.2.2. Descripción del sistema de referencia para este proyecto.

Para la extracción de características hay diversos estudios, que se basan en el uso de la estadística de alto orden, o High Order Spectral Analysis (HOSA), como puede ser el caso de [Kar 10] [Sar 13] y [Gua 12]. Las características más conocidas en el ámbito de las estadísticas de alto orden son los cumulantes.

El estudio de [Gua 12] es en el que nos basaremos para cumplir nuestros objetivos. Este estudio hace un seguimiento completo de todo el proceso de selección y extracción de características, desde la base de datos empleada, pasando por el procesamiento previo de los datos, donde explica las diferentes técnicas que se pueden emplear, hasta llegar al estudio de las Máquinas de Vectores Soporte (SVM) con las que se pretende realizar un entrenamiento para que partiendo de un conjunto de muestras se pueda llegar a identificar las distintas clases y entrenar una SVM para obtener un modelo que prediga cual es la clase de una nueva muestra. Como todo sistema de clasificación y detección se divide en dos bloques.

En el primero de ellos se realiza la extracción de características a través de HOS de una pulsación cardíaca para ello emplea el cálculo de cumulantes de segundo y cuarto orden, a continuación se realiza la selección de las características más importantes utilizando la función de costo del discriminante de Fisher (FDR), y por último con estas características obtenidas realiza un entrenamiento enfocado a la fase de clasificación de características. Con esta etapa lo que se consigue es tener los clasificadores finales ya entrenados con las características seleccionadas. Ver Figura 6. Etapa 1

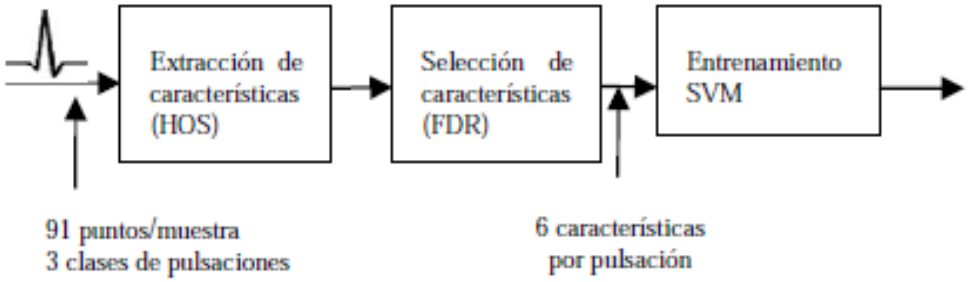

Figura 6. Etapa 1

En la segunda etapa se explica el procedimiento de la clasificación de las pulsaciones de entrada. Para ello, primero se realiza el cálculo de los cumulantes elegidos en la etapa anterior. Con los resultados obtenidos de estos cumulantes

procede a realizar un entrenamiento para la clasificación de las pulsaciones de entrada en tres clases diferentes. Ver Figura 7

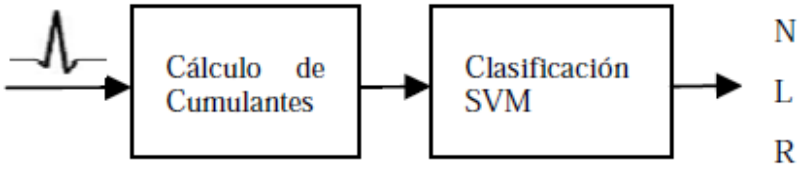

Figura 7. Etapa 2

Es fácil darse cuenta de la relación existente entre las Figura 4yFigura 6, por un lado, y las Figura 5 y Figura 7 por otro. En los dos casos se describe un sistema de selección y clasificación de características.

El estudio que se realiza en este Proyecto Fin de Carrera se centrará en la obtención del cálculo de los cumulantes de segundo y cuarto orden, partiendo de la obtención de características realizada por [Gua 12] y por [Sar 13], y empleando las expresiones que se han obtenido de los dos cumulantes a tratar, las cuales se explicarán en el apartado siguiente, se realizará un estudio para comprobar si es posible una mayor aceleración a través de su implementación hardware.

2.2.3. Extracción y selección de características.

Dentro de todo el proceso de extracción y selección de características para la detección de arritmias, este Proyecto Fin de Carrera se centrará únicamente en el cálculo de los cumulantes de segundo y cuarto orden para la extracción de características, cuyas expresiones vienen descritas a continuación.

Estos cumulantes son el resultado de un estudio realizado entre otros por [Gua 12], donde se llega a la conclusión de que para la obtención de características de señales procedentes de electrocardiogramas, la mejor forma de hacerlo es usando el cálculo de estos cumulantes para unos valores determinados.

Para el cálculo del cumulante de segundo orden se emplea la siguiente ecuación:

$$C_{2,x}(i) = \frac{1}{N}\sum_{n-0}^{N-1} x(n)x(mod(n+i,N))$$

Ecuación 1: Formula cumulante segundo orden

Y para el cumulante de cuarto orden:

$$C_{4,x}(i) = \frac{1}{N}\sum_{n=0}^{N-1} x(n)x^3(mod(n+1,N)) - \frac{1}{N^2}\sum_{n=0}^{N-1} x(n)x(mod(n+i,N))\sum_{n=0}^{N-1} x^2(n)$$

Ecuación 2: Formula cumulante cuarto orden

Donde: *N* será siempre el número de puntos que tenemos por cada muestra.

n varía desde 0 hasta N-1

i su valor varía desde 0 hasta N-1.

mod(n+i,N) esta función es el módulo del cociente entre *n+i* y *N*, cuyo resultado devuelve un número entero.

Para llevar a cabo este estudio partiendo de señales de ECG es necesario tener accesible una buena base de datos, ya que se trata de las señales complejas que poseen mucha información. La base de datos que se empleará para el estudio será la misma que ya han usado otros autores como [Gua 12] y [Sar 13].

Se trata de una de las bases de datos de este tipo de señales más empleada para estudios, esta base de datos es la MIT-BIH ARRHYTHMIA DATABASE [MIT]. Se trata de una de las principales fuentes de registros electrocardiográficos, ya que incluyen numerosos registros correspondientes a un amplio conjunto de patologías diferentes, que han sido estudiadas y etiquetadas por expertos cardiólogos, para que puedan ser utilizadas, entre otros, en el estudio de algoritmos de detección automática de anomalías cardíacas. En la Figura 8 se muestra la representación gráfica de una de estas señales procedentes de electrocardiogramas.

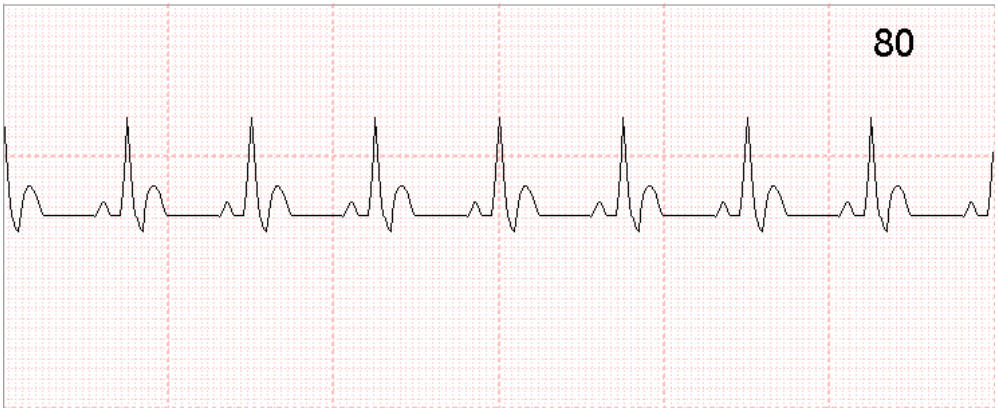

Figura 8 Señales de un ECG

A continuación se representan las señales de los cumulantes de segundo y cuarto orden de cada una de las pulsaciones que se pueden obtener, pulsación normal, pulsación con bloqueo circulatorio en ramal derecho y pulsación con bloqueo circulatorio en ramal izquierdo.

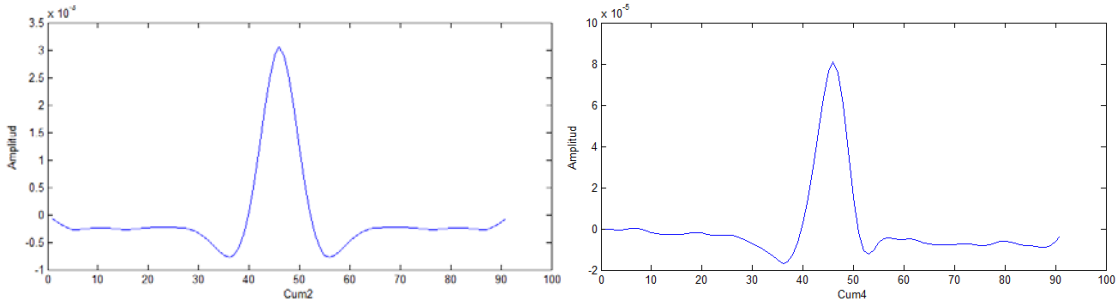

Figura 9. Cumulante segundo y cuarto orden de pulsación normal

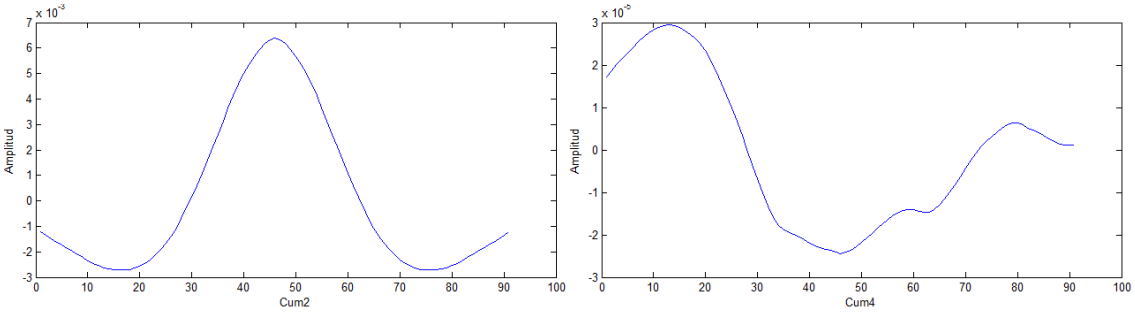

Figura 10. Cumulante de segundo y cuarto orden pulsación tipo L

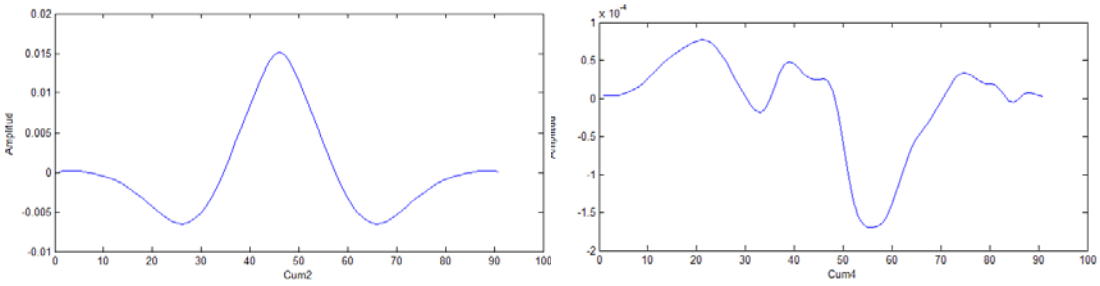

Figura 11. Cumulante de segundo y cuarto orden, pulsación tipo R

Según explica [Gua 12], todas las señales de las pulsaciones con las que se trabaja tienen relacionada una señal por cada cumulante, con lo que aparecen dos nuevas señales por pulsación. . Esto es, que cada cumulante se corresponde con un vector de *N* elementos. Pero de estos *N* elementos por cumulante selecciona solamente los más representativos.

Para realizar esta selección [Gua 12] describe el uso del discriminante de Fhiser (FDR). Una vez obtenido el FDR y aplicándolo conjuntamente con los estadísticos de alto orden (HOS), obtiene un valor para cada cumulante y clase que viene dado por el mayor valor del FDR, y que será el valor que nos permita identificar la variable *i* más representativa perteneciente al vector de *N* elementos de cada cumulante de todos los datos pertenecientes al vector de n elementos.

Estos valores de *i* se muestran en la siguiente tabla.

Cumulante	**Pulsación N vs (L y R)**	**Pulsación L vs (N y R)**	**Pulsación R vs (N y L)**
Segundo	36	35	6
Cuarto	56	6	37

Tabla 1. Valores de i por clase y cumulante

Estos valores vienen dados por la selección de características haciendo una comparación de cada una de las clases con las otras dos analizadas.

En este momento se tienen señales con 91 puntos por muestra, y al aplicar las ecuaciones de los cumulantes de segundo y cuarto orden (Ecuación 1 y Ecuación 2) a cada una de las señales para un único valor de i, el que más información aporta (Ver Tabla 1. Valores de i por clase y cumulante)

Si se realiza una representación gráfica de estos valores se obtiene un punto por cada muestra en un espacio de dos dimensiones, de modo que todos los valores que representan a un mismo tipo de clase se encontrarán agrupados, por lo que visualmente se podría distinguir las diferentes zonas pertenecientes a las distintas anomalías.

A continuación se muestra esta representación gráfica de los valores obtenidos de los dos cumulantes estudiados (Figura 12), tomando como valores de *i* los descritos en la Tabla 1 para la comparación de una pulsación normal frente a las pulsaciones con bloqueo circulatorio en los ramales derecho e izquierdo, pulsación N vs (L y R).

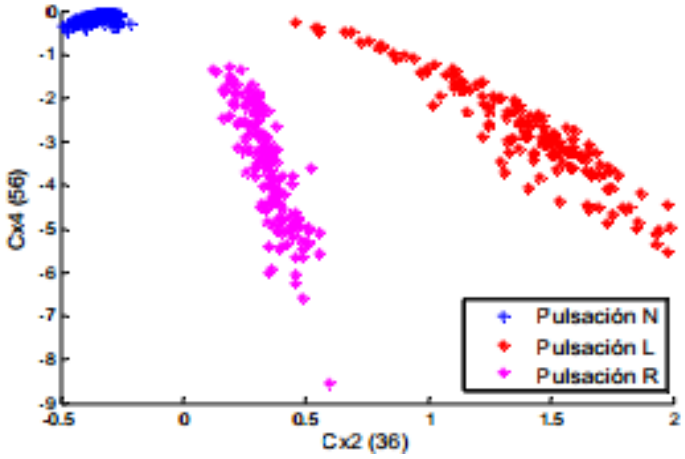

Figura 12. Comparación de características $C_{2,x}(36)$ y $C_{4,x}(56)$

De una forma visual y rápida se comprueba que los distintos tipos de pulsación o anomalías, están perfectamente diferenciadas.

Lo que se pretende con este PFC es realizar los cálculos de los cumulantes de segundo y cuarto orden para unos valores determinados de i, los correspondientes a la comparación de pulsación normal frente a las dos pulsaciones con anomalías, es decir para el cumulante de segundo orden el valor de *i* será 36, mientras que en el caso del cumulante de cuarto orden *i* tomara como valor 56, ver Tabla 1. Valores de i por clase y cumulante.

Una vez que se tengan clasificadas todas las señales, a partir de una nueva señal de pulsaciones se podría clasificar esta última en una de las tres clases representadas en la Figura 12. Comparación de características C2,x(36) y C4,x(56) puede observar que hay grandes diferencias entre una clase de pulsación y otra, las zonas en las que se encuentran representadas cada una de las clases se encuentra bien definida, pudiendo emplear umbrales numéricos para la clasificación de dichas señales.

2.3. Estudio realizado.

Hasta ahora se ha descrito un sistema de clasificación de las señales de pulsaciones procedentes de electrocardiogramas. Este sistema se ha basado en el análisis de la amplitud y la forma de estas señales a través de herramientas específicas para el procesamiento de señales. Las herramientas escogidas han sido HOS y FDR, para el cálculo y selección de características respectivamente.

Este estudio se ha desarrollado e implementado sobre Matlab como herramienta principal de trabajo. Lo que aquí se propone es una implementación hardware del diseño del cálculo de los cumulantes de segundo y cuarto orden para que nos proporcione una respuesta a mayor velocidad.

Al tratarse de una implementación en hardware será necesario emplear un lenguaje de descripción hardware de alto nivel.

El diseño de la obtención de los cumulantes de segundo y cuarto orden se hará en base a las ecuaciones proporcionadas por los estudios previos ya mencionados (ver las fórmulas en Ecuación 1 y Ecuación 2).

Para conseguir esto se partirá del lenguaje de diseño hardware VHDL (VHSIC Hardware Description Language.

Como son dos los cálculos que deberán realizarse, también serán dos los diseños que a continuación se propongan. Cada uno de estos diseños comprende el cálculo de un cumulante a partir de las formulas (Ecuación 2 y Ecuación 3) de las que ya disponemos gracias a los estudios previos realizados por otros autores, donde además indican cuales son los órdenes de los cumulantes que deben emplearse para este caso concreto de señales procedentes de electrocardiogramas.

Capítulo 3: Diseño

El diseño que a continuación se detalla trata de dar una respuesta de mayor velocidad al proceso de extracción de las características de las señales proporcionadas por un ECG, basándose en el cálculo de los cumulantes de segundo y cuarto orden.

3.1. Diseño cumulante de segundo orden.

Ya conocemos cuál es la expresión correspondiente al cumulante de orden dos, pero a continuación la recordaremos:

$$C_{2,x}(i) = \frac{1}{N}\sum_{n=0}^{N-1} x(n)x(mod(n+i,N))$$

Ecuación 1. Cumulante de segundo orden.

La Figura 13representa el diagrama de bloques del cumulante de segundo orden, con las entradas y salidas, algunas de las señales de control, y tres bloques diferentes que explicaremos detalladamente.

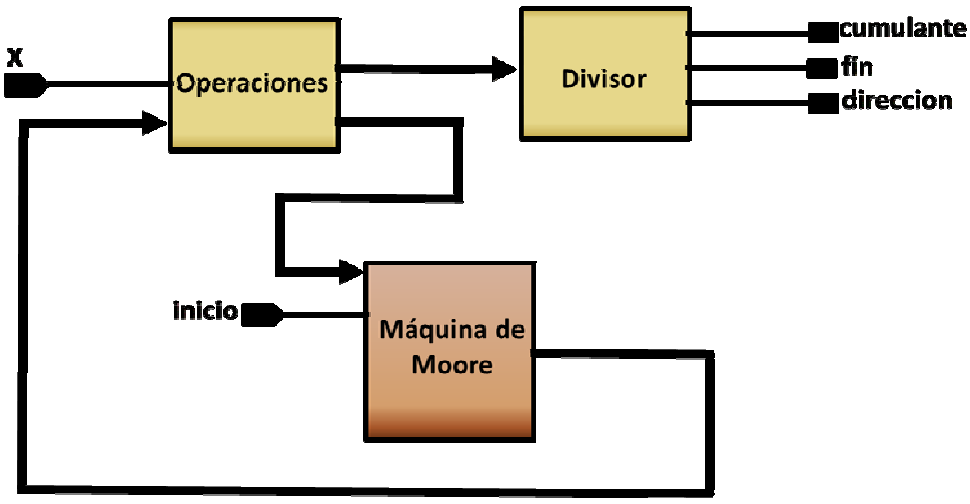

Figura 13 Diagrama de Bloques del Cumulante de Segundo Orden

El diseño realizado consta de 3 bloques principales:

- Bloque de operaciones: en este bloque se encuentran todas las operaciones necesarias para el cálculo (sumas, productos y contadores)
- Bloque divisor: este bloque representa un divisor.
- Bloque Máquina de Moore: máquina de estados.

A continuación se va a proceder a explicar la interfaz del circuito realizado explicando las características de los puertos de entrada y salida.

Se ha realizado un diseño síncrono que posee una señal de reloj *clk* y una señal de inicialización asíncrona, *reset*, activa a nivel alto.

Se dispone además de una entrada *inicio* que será la encargada de activar el proceso de obtención del cumulante de segundo orden.

La entrada *x* se utiliza para leer las muestras de la señal temporal a caracterizar. Esta entrada tiene un tamaño de 32 bits para poder representar todos los valores que están incluidos en la base de datos empleada (MIT-BIH ARRHYTHMIA DATABASE[MIT]). El tamaño de esta entrada se especifica en el diseño como un genérico, *ancho*, de forma que el circuito diseñado sea fácilmente aplicable con otra base de datos en la que las muestras requieran de un mayor número de bits para su representación digital.

La entrada *ena_dato* será un bit que permitirá saber en qué momento ha llegado cada uno de los datos introducidos a través de la entrada *x*, estará activa durante un ciclo de reloj cada vez que llegue un dato. Cuando tenga un nivel lógico alto, indicará el momento en que ha llegado el dato que estaba esperando.

Para saber en cada momento qué valor es el que se tiene que leer de la memoria donde se encontrarán almacenados todos los datos procedentes de la selección de características, se dispondrá de una salida *dirección*.

El resultado final obtenido tras realizar todos los cálculos indicados en la expresión que representa al cumulante de segundo orden, se almacenará en la señal *cumulante* que deberá tener un tamaño de 64 bits (*2*ancho,* siendo *ancho* el genérico anteriormente explicado). Esto es así, porque el valor de *cumulante* se obtiene a partir de multiplicaciones, sumas y divisiones.

> *Siendo x de tamaño ancho, x(n) y x(mod(n+i,N)) también tendrán el mismo tamaño. Por lo que al multiplicar estos se obtiene un tamaño de 2*ancho, la suma posterior debida al sumatorio no influirá en el tamaño final así como tampoco lo hará la división puesto que se desea realizar de forma genérica para que se pueda aplicar a distintas bases de datos.*

También se ha añadido una salida *fin* que se activará durante un ciclo de reloj cuando se haya finalizado el cálculo del cumulante de segundo orden.

A parte de estas entradas y salidas se han especificado como genéricos el valor de *N* que representa el número de puntos por muestra y el valor de *i* para que pueda ser fácilmente modificado en caso de que las especificaciones requirieran una modificación de este parámetro.

En la Figura 14se muestra el diagrama de flujo del funcionamiento general del circuito.

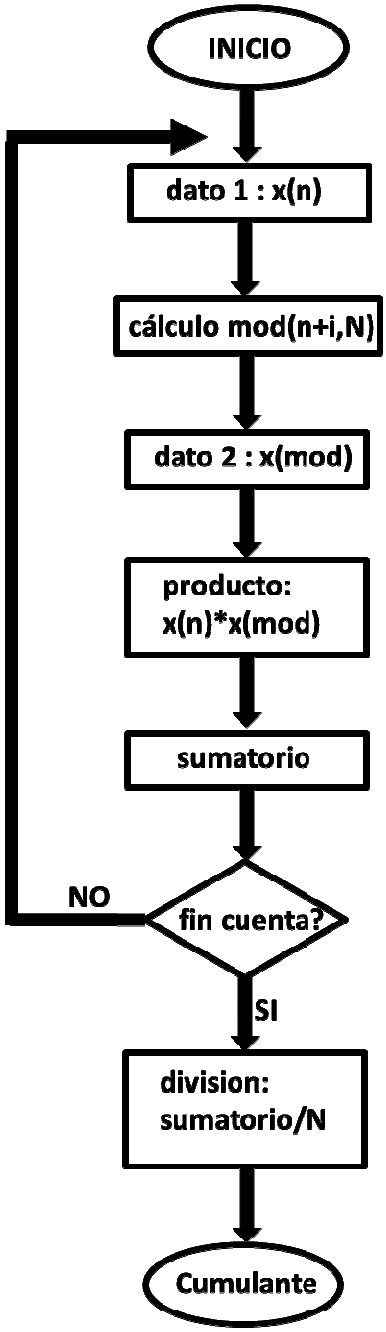

Figura 14 Diagrama de flujo cumulante de orden dos

De forma general, el sistema se encuentra inicialmente en reposo a la espera de la señal que le indique el momento en que debe comenzar a funcionar. Con la llegada de esta señal, *inicio*, el siguiente paso es esperar la llegada del primer dato, lo que con la fórmula del cumulante de orden dos se corresponde con *x(n)*.

Una vez que se dispone de este primer valor de x, ya se puede proceder a calcular el modulo, es decir *mod(n+i,N)*, con este valor del módulo, se accederá a la dirección de memoria que indique dicho valor y así se obtiene el siguiente dato, *x(mod(n+i,N))*. Con estos dos valores registrados es el momento de realizar el producto de ambas. Con esto solamente se consigue el primer valor del sumatorio final, para obtener la totalidad de los valores del sumatorio, si la cuenta no ha llegado a su fin el sistema debe volver al momento en que llega el primer dato y obtener el nuevo valor de *x(n)*, de *mod(n+i,N)* y en consecuencia de *x(mod(n+i),N)*.

En caso de que la cuenta haya finalizado, es decir que el contador tenga como valor 91, número de puntos por muestra, se conocerá cual es el valor del sumatorio y ya se puede proceder a realizar la división, y con ella obtener el resultado final, valor que tomará la señal *cumulante*.

3.1.1. Máquina de estados.

La máquina de estados va a ser la encargada de controlar el correcto funcionamiento del diseño del cálculo del cumulante de segundo orden. A través de esta se van a activar las distintas señales de control necesarias para llevar a cabo el diseño.

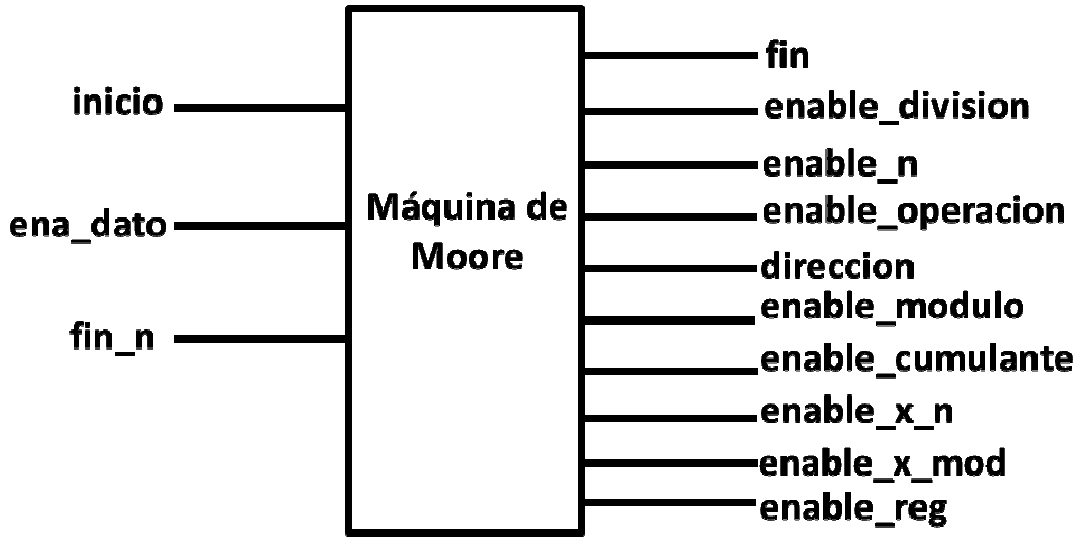

Figura 15. Máquina de estados cumulante segundo orden (entradas y salidas, izquierda y derecha respectivamente)

Las entradas a la máquina de Moore, son las del diseño inicial las cuales ya han sido explicadas en el apartado 3.1. de este capítulo, pero además tiene otra entrada *fin_n* que se activa durante un ciclo de reloj cuando se ha realizado un recorrido por todas las direcciones de la memoria. Nos avisa de cuando se han leído todos los datos, y con ello decidir si volvemos al estado de reposo o continuamos porque aún no ha finalizado el proceso.

Las señales *fin* y *dirección* coinciden con dos de las salidas del diseño completo, también explicadas en el apartado 3.1. de este capítulo. Además de estas salidas, se han empleado:

- *Enable_division*: se trata de un bit que activa el proceso del bloque división, cuando ya se ha obtenido el resultado del sumatorio.
- *Enable_ modulo*: es un bit que activa el cálculo del módulo, en el bloque operaciones.
- *Enable_cumulante*: activa el registro del valor final del cumulante, esto se hace justo después de recibir el valor de la división.
- *Enable_x_n*: activa el registro de *x(n)*, como esta señal va tomando diferentes valores hay que registrar su valor para que esté disponible cuando sea necesario usarlo.
- *Enable_x_mod*: activa el registro de *x(n+i,N)*, al igual que *x(n)* esta señal tiene que ser registrada.
- *Enable_n*: activa la cuenta del *indice*, que se realiza a través de un contador que se encuentra en el bloque operaciones.
- *Enable_operacion*: activa todo el proceso suma y producto, que se encuentran en el bloque operaciones.
- *Enable_reg*: habilita el registro de los datos que serán usados en la división.

Se ha optado por realizar el diseño utilizando una máquina de estados de tipo Moore, de manera que las salidas estén determinadas únicamente por el estado en el que se encuentre en ese momento el circuito, no dependerá directamente del valor que tomen las entradas.

Desde la máquina de estados podremos controlar el inicio de todas y cada una de las operaciones que son necesarias para realizar el cálculo del cumulante. Además también se controla el contador encargado de direccionar la memoria donde se almacenan las muestras de la señal a caracterizar. Con el valor de la cuenta de dicho contador, la máquina de estados detecta cuando se alcanza el final de la memoria. El control del contador se realiza con la salida *enable_n* que es la señal que habilita la cuenta.

Además, el inicio de la división también viene controlado por la máquina de estados. Es necesario saber en qué momento se debe realizar la división, y para ello contamos con la señal de control *enable_division* que se activará para decirnos cuál es el valor que debe tomar el *numerador* y cuál el *denominador*, para así poder realizar la división.

En la siguiente figura se muestra el diagrama de estados de la máquina de Moore empleada.

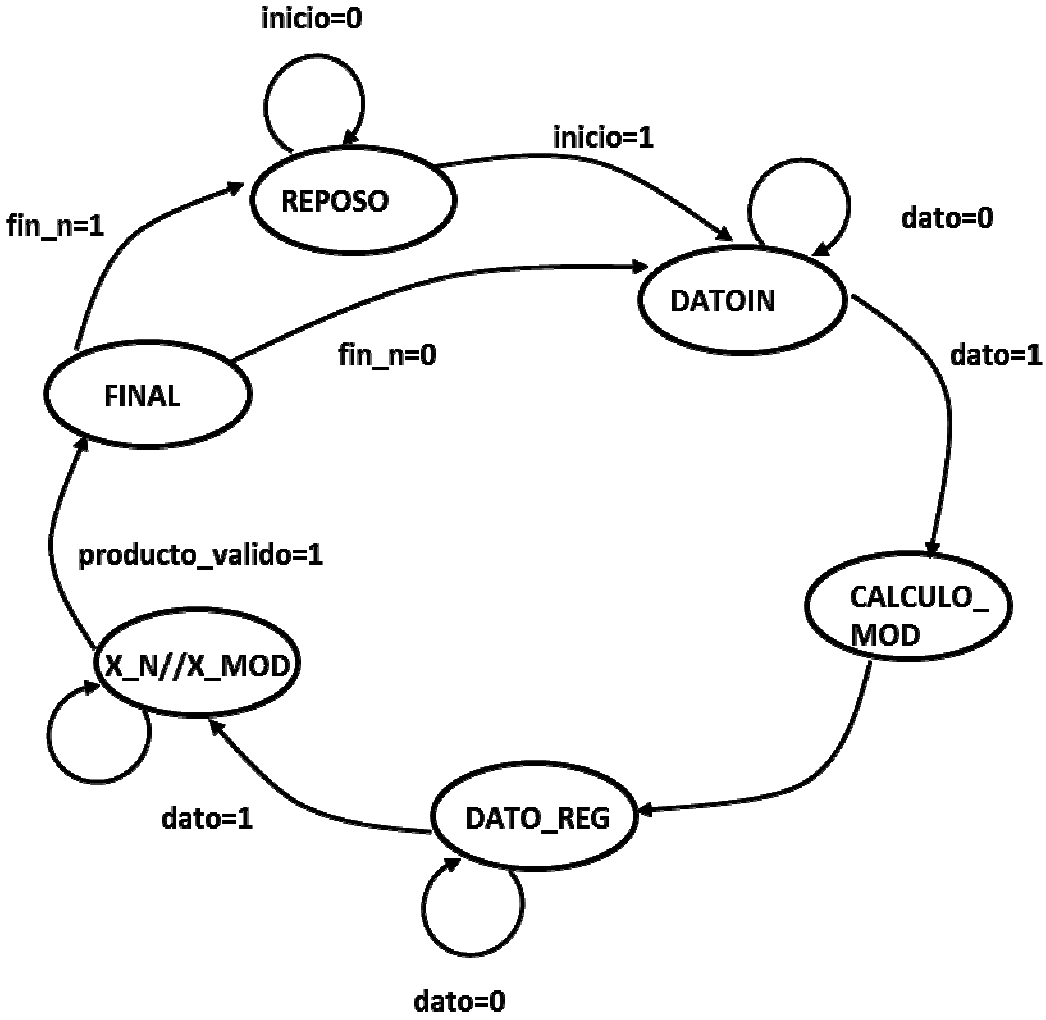

Figura 16 Diagrama de Estados máquina Moore Cumulante de segundo orden.

La máquina de Moore empleada consta de seis estados diferentes, inicialmente se partirá de un estado inicial de reposo desde el que se irá avanzando a medida que las señales vayan tomando los diferentes valores.

De forma sencilla, lo que queremos conseguir con esta máquina de estados es el control para ir obteniendo los valores de *n* y *mod(n+i,N)* para poder acceder a la

posición de la memoria correspondiente y obtener así los valores de *x_n_reg* y *x_mod_reg* que serán los valores que deberán ser multiplicados y almacenados para posteriormente realizar el sumatorio de estos productos. Para finalizar se realizará una división entre *N*, que nos dará el resultado final del cumulante de segundo orden.

REPOSO: es el estado inicial en el cual el sistema queda a la espera de recibir la señal para comenzar a realizar el proceso. Esta señal de comienzo vendrá proporcionada por la entrada del diseño *inicio*, en el momento en que esta entrada esté activa a 1 dará comienzo la primera transición entre estados.

DATO_IN: en este estado es donde se espera la llegada del primer dato. Cuando ena_*dato* pase a tomar el valor 1, nos indicará que el dato ha llegado correctamente y que podemos continuar el proceso.

CALCULO_MOD: en este momento es cuando se llevará a cabo el cálculo del módulo que viene definido por la expresión:

mod(n+i,N)esta función es el módulo del cociente entre n+i y N, cuyo resultado devuelve un número entero

DATO_REG: aquí la máquina de estados se encarga de registrar el valor de *x* o, lo que es lo mismo, nuestro dato de entrada. Además con el valor obtenido en el cálculo del módulo, que recordemos viene dado por la señal *modulo*, busca en la memoria del sistema diseñado el valor del dato que se encuentra en la dirección que se corresponde con el valor de la señal *modulo*, una vez que este valor queda localizado se vuelve a activar a nivel lógico alto la señal ena_*dato*, para indicarnos que el siguiente valor necesario ya se encuentra disponible.

X_N_X_MOD: es aquí donde llega el valor del dato correspondiente a la dirección *módulo*, este valor junto con el primer dato obtenido pasan a ser registrados. Es decir, tenemos *x_n_reg* que almacena el valor de *x* cuya dirección en la memoria es *indice*, y por otro lado *x_mod_reg* cuyo valor se corresponde con el valor de *x* cuya dirección en la memoria viene determinado por *modulo*.

Todos estos pasos hasta obtener los valores de *x_n_reg* y *x_mod_reg* se tiene que repetir tantas veces como muestras tenga la base de datos con la que se esté trabajando en ese momento, en nuestro caso se trata de 91 puntos por muestra, por lo que debemos realizar este proceso 91 veces. La señal *fin_n* nos avisa de cuándo se llega al último dato de la muestra.

FINAL: es aquí donde se comprueba si ya se ha realizado el proceso para todos los puntos de la muestra o no. Para ellos se comprueba el valor de *fin_n*, en caso de que esta señal se encuentre activa a 1, el proceso habrá finalizado y se

volverá al estado inicial de REPOSO, donde ya se podrá llevar a cabo la división necesaria. Sin embargo en el caso de que la señal *fin_n* se encuentre a 0, la transición debe realizarse hasta el estado DATO_IN para seguir realizando las operaciones correspondientes hasta llegar a completar el ciclo de los 91 puntos de la muestra.

3.1.2. Bloque operaciones.

A continuación se explicarán cada una de estas operaciones realizadas en el bloque de operaciones.

- El cálculo del módulo, se activa a través de la señal *enable_mod* procedente de la máquina de estados descrita en el apartado 3.1.1. Este cálculo viene definido por la expresión *mod(n+i,N)*.

  Donde: *n* toma los valores de la señal *indice* desde 0 hasta 90

  N es siempre 91 que es el número de puntos que tenemos por cada muestra.

  i al tratarse del cumulante de segundo orden tomará siempre el valor de 36, tal y como se explicó en el apartado 2.2.

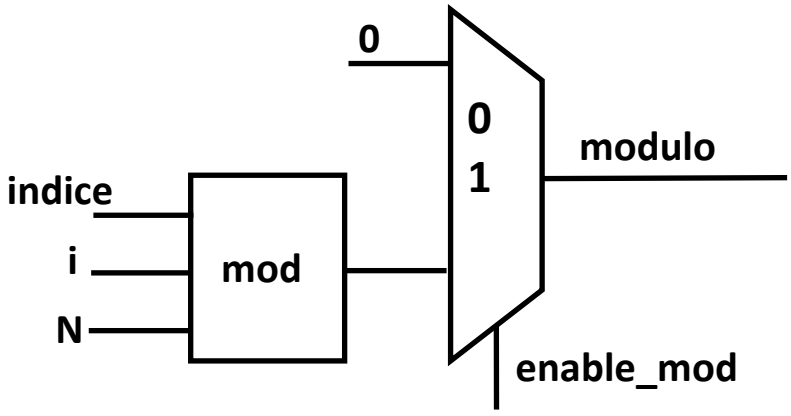

Figura 17 Operación mod

El valor obtenido del cálculo del módulo, queda reflejado en la señal *modulo* que será un entero comprendido entre 0 y *N*. El control de cuándo se debe realizar este cálculo, viene dado por *enable_modulo* que es una señal que se activa en el primer estado, este valor nos proporciona la siguiente dirección de memoria que deberá ser leída en el caso de tratarse de la señal *x_mod_reg*.

El producto de *x_n_reg* y *x_mod_reg* se almacena en la señal *producto*, que tiene un tamaño de *2*ancho*, ya que tanto *x_n_reg* como *x_mod_reg* tienen un tamaño de *ancho*, recordemos que *ancho* estará definido como un genérico con valor 32 que dependiendo de las necesidades podremos modificar.

- El producto de *x_n_reg* y *x_mod_reg* pasa por la señal *producto*, que tiene un tamaño de *2*ancho*, ya que tanto *x_n_reg* como *x_mod_reg* tienen un tamaño de *ancho*, recordemos que *ancho* estará definido como un genérico con valor 32 que dependiendo de las necesidades podremos modificar.

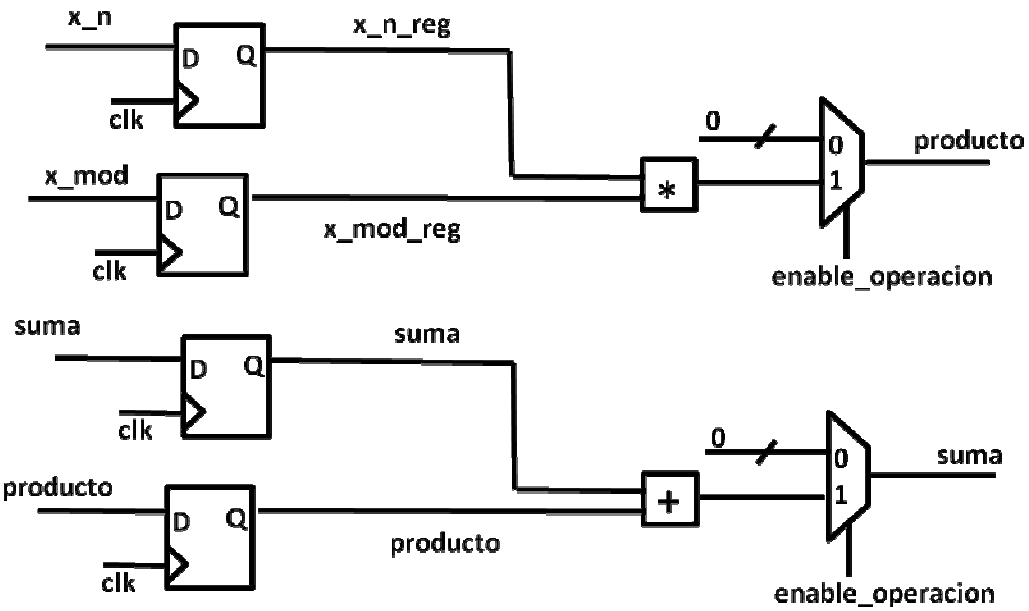

Figura 18 Operaciones de producto y suma

El momento en que se lleva a cabo esta multiplicación viene dado por la activación de la señal *enable_operacion*, lo que también dará pie a que se realice la suma asociada, esto es, que la señal *suma* almacenará el valor del sumatorio de los distintos valores que vaya tomando la señal *producto.*

- Como ya hemos dicho, es necesario el empleo de un contador de 0 a 90, cuya función será avisar a la máquina de estados en el momento en que llegue al final de la cuenta, para que esta sepa que ha finalizado el recorrido por toda la memoria, y que ya puede proporcionar el valor final del cumulante. La señal *indice* será la que se vaya incrementando cada vez que pase por el estado FINAL, o lo que es lo mismo la señal *enable_n* esté activa a 1. En el momento en que *indice* tome como valor *N-1*, el contador habrá llegado al final y podrá finalizarse el proceso.

3.1.3. Divisor

El bloque del divisor, es un bloque algo más complejo, ya que dentro de este se encontrará una máquina de estados y un divisor serie con restauración, este divisor se ha realizado a partir de un divisor realizado por Eric Gutiérrez Fernández. Aunque los divisores se pueden diseñar a través de diferentes algoritmos, en este caso se decide emplear un divisor serie con restauración, encontrando la posibilidad de usar el ya diseñado por Eric Gutiérrez que eligió este algoritmo en concreto.

Para dividir *a/b*, se pone *a* en el registro *A*, b en el registro *B*, 0 en el registro *P*, y después se procede de la siguiente manera:

1. Desplazar el par de registros (*P,A*) un bit a la izquierda.
2. Restar el contenido del registro *B*(que es B_{n-1}, b_{n-2}.....b_0) del registro *P*.
3. Si el resultado del paso anterior es negativo, poner el bit de orden inferior de *A* a 0, en cualquier otro caso a 1.
4. Si el resultado del paso 2 es negativo, restaurar el valor antiguo de *P* sumando el contenido del registro *B* de nuevo a *P*.

Este algoritmo será un proceso iterativo de *n+1* ciclos.

P	A	
00000	1110	Divide 14= 1110 por 3=11. B siempre contiene 0011
00001	110	Paso (1): desplaza
-00011		Paso (2): resta
-00010	1100	Paso (3): resultado es negativo, pone bit de cociente a 0
00001	1100	Paso (4): restaura
00011	100	Paso (1): desplaza
-00011		Paso (2): resta
00000	1001	Paso (3): resultado es negativo, pone bit de cociente a 1
00001	001	Paso (1): desplaza
-00011		Paso (2): resta
-00010	0010	Paso (3): resultado es negativo, pone bit de cociente a 0
00001	0010	Paso (4): restaura
00010	010	Paso (1): desplaza
-00011		Paso (2): resta
-00001	0100	Paso (3): resultado es negativo, pone bit de cociente a 0
00010	0100	Paso (4): restaura. El cociente es 0100 y el resto es 00010

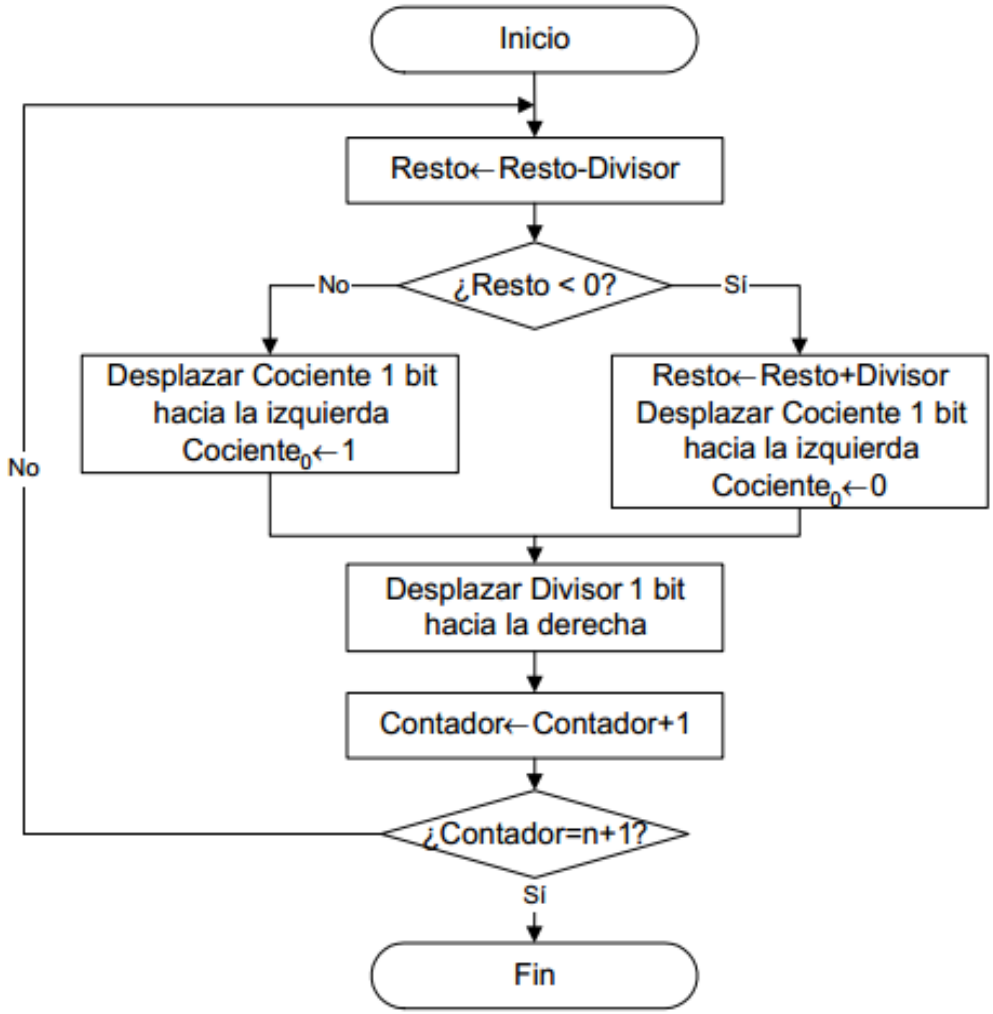

Figura 19. Diagrama de Flujo de Divisor con restauración

En la Figura 19 se muestra un diagrama de flujo de cuáles son los pasos que se deben seguir a la hora de implementar un divisor con este algoritmo.

En el algoritmo de división con restauración partimos de un registro *A* y un registro *B*, donde *A* está formado por a_{n-1}, a_{n-2},… a_0, y *B* es b_{n-1}, b_{n-2},… b_0, además hay otro registro intermedio *P* que se inicializa a 0 donde se irá acumulando el resto de la división. Los pasos a seguir en el algoritmo de división con restauración son los siguientes:

1- Desplazar el par de registros (*P,A*) un bit a la izquierda.

2- Restar el contenido del registro *B* del registro *P*.

3- Si el resultado del paso 2 es negativo, poner el bit de orden inferior de *A* a 0, en cualquier otro caso a 1.

4- Si el resultado del paso 2 es negativo, restaurar el valor antiguo de *P* sumando el contenido del registro *B* de nuevo a *P*.

Tras repetir este proceso n veces, el registro *A* contendrá el cociente, y el registro *P* el resto.

El divisor con restauración de Eric Gutiérrez, ha sido una base para llegar a obtener el divisor que realmente se necesita, es decir, algunas de las características iniciales del divisor han sido modificadas para poder adaptarlo a las necesidades del circuito que nos interesa.

Inicialmente se trataba de un divisor con un tamaño fijo de 16 bit que realizaba divisiones de forma continua y no tenía la capacidad de dividir números con signo negativo.

Para poder usar este divisor, se han realizado una serie de modificaciones. Para empezar había que cambiar el tamaño de las señales, ya que este divisor con restauración estaba diseñado para un tamaño diferente, de modo que se ha realizado de forma genérica indicando el tamaño a través de un genérico *ancho*, que en nuestro diseño tomará como valor 64. Además se ha añadido una señal de control que permite elegir el momento en que se realizaba la división, ya que para nuestro diseño era muy importante el momento en que se realizaba la división.

A parte de estas modificaciones se ha realizado un control a través de una máquina de estados, ya que no solamente interesa controlar el momento en que se realiza la división, sino que además hay que tener en cuenta otros factores, como puede ser el signo tanto del dividendo, como del divisor.

Cómo en nuestro caso se puede dar la existencia de números negativos, para poder realizar la división hay que transformar el número negativo en su correspondiente complemento a 2, y así eliminar el obstáculo que representa tener un signo negativo.

Todos estos factores son los que se tienen en cuenta a la hora de diseñar el control del divisor con restauración realizado por Eric Gutiérrez.

A continuación se representa el diagrama de bloques de nuestro divisor, que incluye el divisor con restauración, un bloque que realiza el complemento a 2 de los números negativos y el control realizado por una máquina de estados tipo Moore.

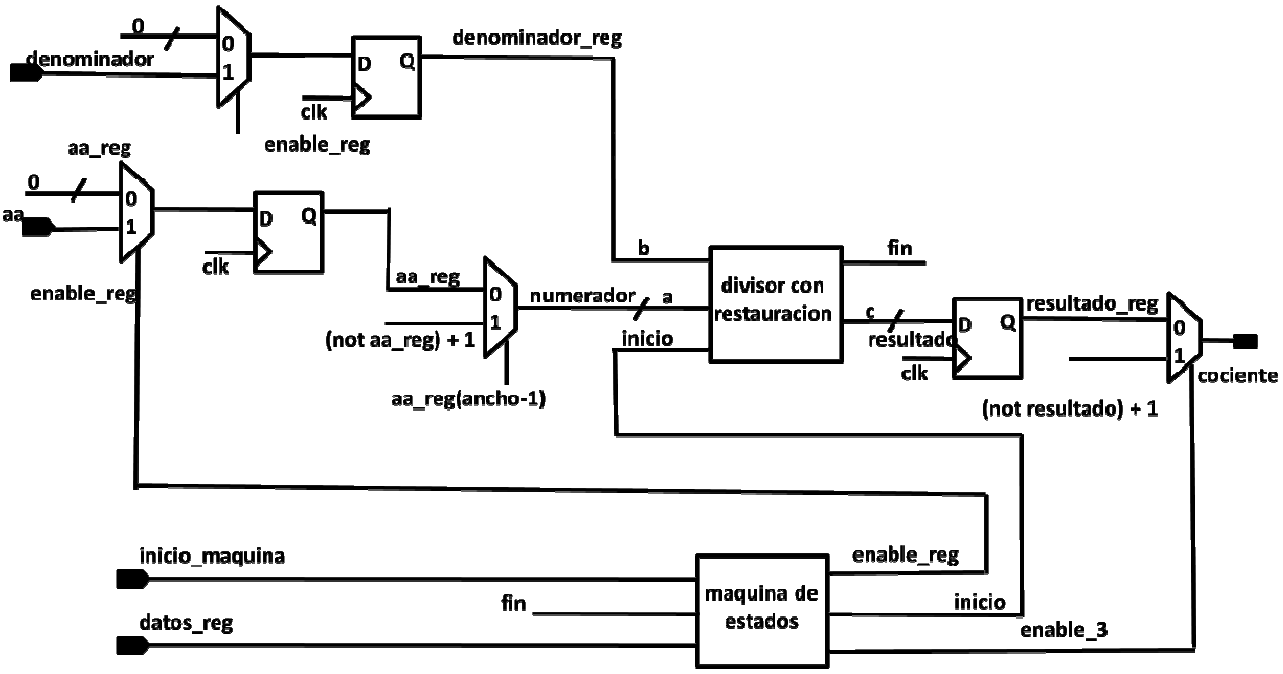

Figura 20Diagrama de bloques del divisor

El objetivo que debe cumplir nuestro divisor es, en un primer lugar, comprobar si el dato que le llega es un dato positivo o negativo, en caso de tener signo negativo el siguiente paso es realizar la transformación a complemento a 2, una vez que tengamos un número positivo se procede a realizar la división empleando el divisor con restauración.

En el diagrama de bloques se pueden apreciar dos bloques diferentes, uno es el divisor, donde se realiza el cálculo de la división, y el otro el control a través de la máquina de estados.

En el caso del divisor con restauración se parte de dos entradas, *numerador* y *denominador*, ambas entradas tendrán un tamaño *ancho*, en el caso del divisor con restauración el tamaño de *ancho* tendrá un valor de 64, es decir el doble que en el resto de bloques del diseño del cumulante de segundo orden, ya que los valores que se introducirán en la señal *numerador* son los resultados obtenidos en la señal *suma* que tiene un tamaño de 64 bits. La salida obtenida de este bloque será la señal *resultado* que tendrá un tamaño de *ancho* (64 bits). La señal que nos ayudará a controlar el momento en que debe realizarse la división es *inicio* que durará un solo ciclo de reloj, y se activará a través de la máquina de estados.

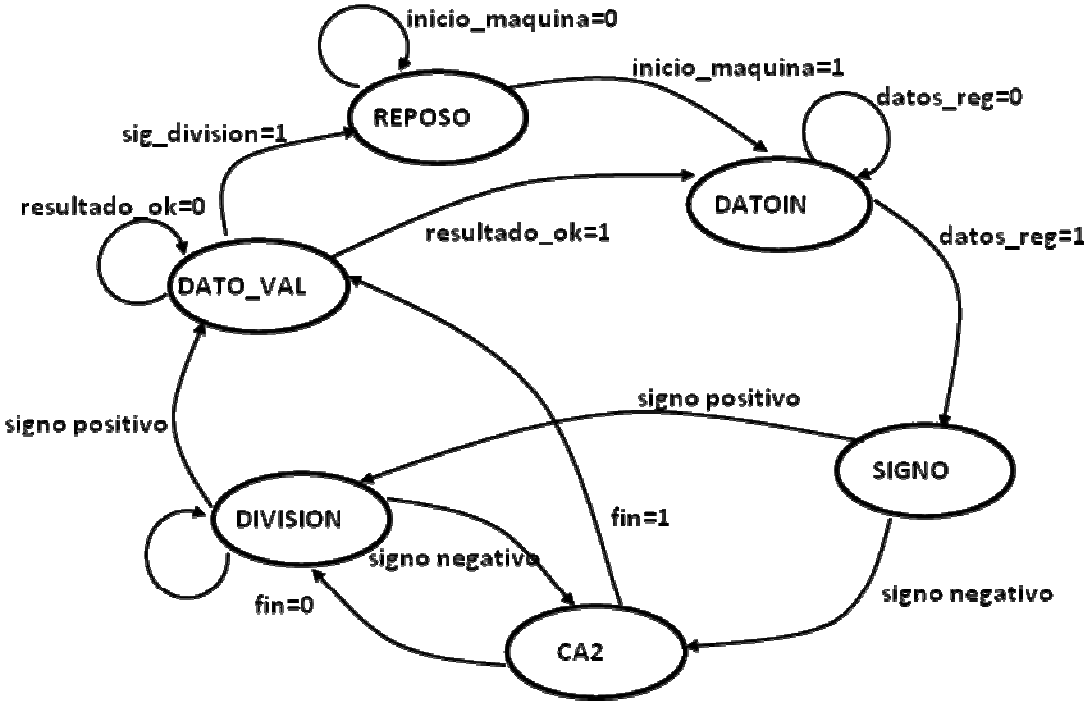

Figura 21 Diagrama de estados del divisor

REPOSO: es el estado de inicio, en el cual todas las señales, entradas y salidas toman su valor inicial, para salir de este estado se debe activar a 1 la señal *inicio_maquina* que es la encargada de avisar del momento en el cual debe activarse todo el proceso que engloba la división. Este momento viene determinado por la máquina de estados del cumulante de segundo orden, cuando se ha finalizado el cálculo del sumatorio, es decir, que tenemos el valor final de este almacenado en *suma*, es cuando esta señal debe activarse y se corresponde con la señal del cumulante de segundo orden *enable_division*.

DATOIN: una vez que se ha dado comienzo al proceso de la división debe quedar claro qué valores son los que debemos emplear en la división. Estos valores son los que introduciremos a través de *aa_reg* y *denominador*, como la división siempre tiene el mismo valor en el divisor para la base de datos utilizada en este proyecto, *N* o lo que es lo mismo 91 (número de muestras), el valor de *denominador* siempre será el mismo y no habrá posibilidad de error, pero en el caso de *aa_reg* el valor que debemos introducir será el obtenido en la señal *suma* del cumulante de segundo orden. Una vez que estos valores sean registrados, se activará *datos_reg* y pasaremos al siguiente estado.

SIGNO: aquí es donde se tiene que evaluar el signo del dato introducido, como ya hemos dicho el divisor siempre va a ser el mismo y por lo tanto sabemos que siempre será un número positivo, por lo que no es necesario hacer ningún tipo de comprobación referente al valor de la señal *denominador*. Sin embargo, la señal *aa_reg,* que se corresponde con el dividendo, sí que puede ser un

número negativo, por lo que hay que poner especial interés en ella, y comprobar el signo que tiene, dependiendo de este signo, en el diagrama de bloques vemos que podemos hacer la transición a un estado o a otro. El signo se comprobará viendo si el bit más significativo es 1 o 0, en el caso de que sea 1 sabremos que el signo es negativo, de lo contrario será positivo.

CA2: a este estado llegaremos en el caso de tener un dato negativo, por lo que habrá que realizar el complemento a 2 de dicho número. Para realizar el cálculo del complemento a dos de un número negativo, lo primero que hay que hacer es invertir el valor de cada una de los bits, es decir, calcular el complemento a uno y al resultado sumarle 1.

En este estado se nos pueden presentar dos situaciones diferentes dependiendo del valor que tenga la señal *fin*, si esta activa a 1 quiere decir que la división se ha realizado previamente y que el valor que estamos convirtiendo en complemento a 2 es el resultado obtenido de la división, y por lo tanto el siguiente estado sería DATO_VAL. De lo contrario, es decir *fin* toma el valor 0, indica que la división aún no se ha realizado y que debemos pasar al estado DIVISION, donde se procederá a realizar la operación de dividir.

DIVISION: en este estado es donde debemos tener claro qué valor es el que va a tomar el resultado de la división, o lo que es lo mismo, nuestra señal *cociente*. Si el signo del resultado obtenido en la división es positivo, directamente pasamos al estado DATO_VAL, si no (si se trata de un signo negativo), debemos volver al estado CA2.

DATO_VAL: es el estado en que se guarda el valor de la división, es decir se almacena en *cociente* el valor final correspondiente, o bien directamente *resultado_reg* en el caso de que se trate de un número positivo, o el complemento a dos de esta señal si el signo es negativo. Si la división ha finalizado se activará *sig_division* y pasara al estado de REPOSO donde espera la llegada de un nuevo dato para dividir.

3.2. Cumulante de cuarto orden.

Recordemos que la expresión correspondiente al cumulante de cuarto orden viene definida por:

$$C_{4,x}(i) = \frac{1}{N}\sum_{n=0}^{N-1} x(n)x^3(mod(n+1,N)) - \frac{1}{N^2}\sum_{n=0}^{N-1} x(n)x(mod(n+i,N))\sum_{n=0}^{N-1} x^2(n)$$

Ecuación 2. Cumulante de cuarto orden.

La siguiente Figura 1representa el diagrama de bloques del cumulante de cuarto orden, con las entradas y salidas, algunas de las señales de control, y a diferencia del cumulante de segundo orden cinco bloques diferentes que explicaremos detalladamente.

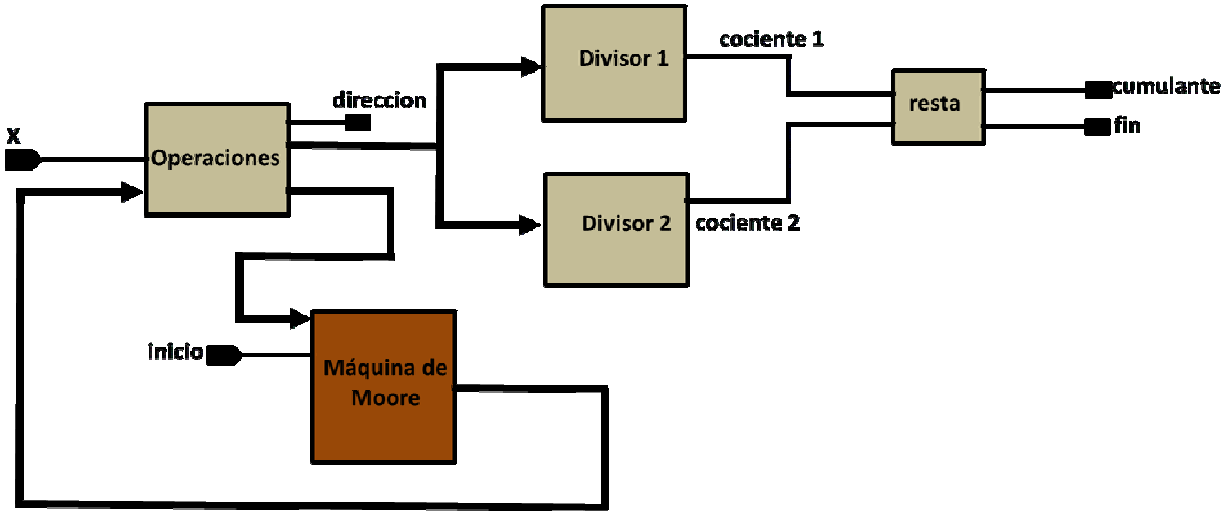

Figura 22 Diagrama Bloques cuarto cumulante

Mas adelante se explicará cada uno de los bloques en detalle, pero de forma general:

- Bloque de operaciones y bloque resta: estos dos bloques se explicaran juntos, ya que en ellos se encuentran todas las operaciones necesarias para el cálculo (sumas, productos y contadores)
- Bloque divisor 1 y divisor2: estos dos bloques representan los dos divisores que se van a emplear para el diseño.
- Bloque Máquina de Moore: máquina de estados.

Tanto en el diagrama de bloques como en la expresion del cumulante de cuarto orden, se puede observar que hay bastantes similitudes entre el calculo de ambos cumulantes.

Si nos centramos en las expresiones Ecuacion 2 y Ecuacion 2, del segundo y cuarto cumulante respectivamente, vemos que una parte de esta formula es practicamente igual a la del cumulante de segundo orden.

$$C_{2,x}(i) = \frac{1}{N}\sum_{n=0}^{N-1} x(n)x(mod(n+i,N))$$

$$C_{4,x}(i) = \frac{1}{N}\sum_{n=0}^{N-1} x(n)x^3(mod(n+1,N)) - \frac{1}{N^2}\sum_{n=0}^{N-1} x(n)x(mod(n+i,N)) \sum_{n=0}^{N-1} x^2(n)$$

Por este motivo muchos de los cálculos empleados en el cumulante de segundo orden serán similares a los del cumulante de orden cuatro.

Exactamente igual que en el caso del cumulante de segundo orden se va a trabajar con un diseño síncrono que posee una señal de reloj *clk* y una señal *reset* de inicialización asíncrona activa a nivel alto.

En realidad las entradas y salidas de este diseño global son exactamente las mismas que en el cumulante de orden dos. Estas entradas y salidas ya quedaron explicadas en el apartado 3.1. de este capítulo.

La única diferencia que existe entre ambos cumulantes es el tamaño de la salida *cumulante*, ya que en este caso el tamaño de dicha señal es de cuatro veces el genérico *ancho*, esto se debe a que en el cumulante de cuarto orden se realizan mayor número de multiplicaciones, lo que hace que se incremente el tamaño de los datos.

La entrada *inicio* seguirá siendo la encargada de decidir el momento en que se va a llevar a cabo el proceso de obtención del cumulante de cuarto orden. Los datos se seguirán leyendo a través de la entrada *x*, que conservará el mismo ancho que en el diseño anterior, 32 bits, ya que los valores que se emplean siguen siendo los procedentes de la base de datos MIT-BIH Arrhythmia DataBase [Mit].

Al igual que en el cumulante de segundo orden, el tamaño de esta entrada se especifica con un genérico para poder aplicar este diseño a cualquier tipo de base de datos.

La entrada *ena_dato* será un bit que permitirá saber en qué momento ha llegado cada uno de los datos introducidos a través de la entrada *x*, estará activa durante un ciclo de reloj cada vez que llegue un dato. Cuando tenga un nivel lógico alto, indicará el momento en que ha llegado el dato que estaba esperando.

Para conocer el valor de la dirección de memoria en la que se debe buscar el siguiente dato se empleará la salida *dirección*, al igual que en el diseño anterior.

El resultado final obtenido tras realizar todos los cálculos indicados en la expresión que representa el cumulante de cuarto orden, se almacenará en la salida *cumulante* que tal y como acabamos de explicas dispondrá de un tamaño de cuatro veces *ancho*.

También se ha añadido una salida *fin* que se activará durante un ciclo de reloj cuando se haya finalizado el cálculo del cumulante de segundo orden.

A parte de estas entradas y salidas se han especificado como genéricos el valor de *N* que representa el número de puntos por muestra y el valor de *i* para que pueda ser fácilmente modificado en caso de que las especificaciones requirieran una modificación de este parámetro.

En la Figura 23 se muestra el diagrama de flujo del funcionamiento general del circuito.

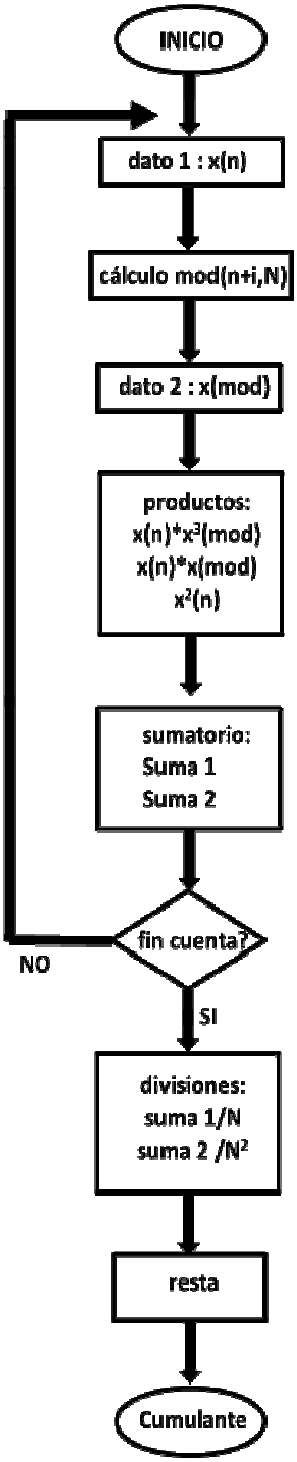

Figura 23 Diagrama de flujo

De forma general, el sistema se encuentra inicialmente en reposo a la espera de la señal que le indique el momento en que debe comenzar a funcionar. Con la llegada de esta señal, *inicio*, el siguiente paso es esperar la llegada del primer dato, lo que con la fórmula del cumulante de orden dos se corresponde con *x(n)*. Una vez que se dispone de este primer valor de x, ya se puede proceder a calcular el modulo, es decir *mod(n+i,N)*, con este valor del módulo, se accederá a la dirección de memoria

que indique dicho valor y así se obtiene el siguiente dato, *x(mod(n+i,N)*. Una vez registrados estos dos valores es el momento de realizar los diferentes productos que aparecen en la expresión del cumulante de cuarto orden: $x(n)*x^3(mod)$, *x(n)*x(mod)* y $x^2(n)$. Al igual que en el diseño anterior, con esto solamente conseguimos el primer termino de los sumatorios, para obtener la totalidad de los valores del sumatorio, si la cuenta no ha llegado a su fin el sistema debe volver al momento en que llega el primer dato y obtener el nuevo valor de *x(n)*, de *mod(n+i,N)* y en consecuencia de los diferentes productos antes mencionados. En caso de que la cuenta haya finalizado, es decir que el contador tenga como valor 91, número de puntos por muestra, se conocerá cual es el valor de lo sumatorios y se puede proceder a realizar las divisiones, para finalmente realizar la resta de los valores resultantes de estas dos divisiones y con ello obtener el resultado del cumulante de cuarto orden.

3.2.1. Máquina de estados.

La descripción de la máquina de estados de este diseño va a ser prácticamente igual a la del diseño del cumulante de segundo orden que ya hemos detalladlo en el apartado 3.1.1.

La Figura 24 representa el bloque de la máquina de estados para el cálculo del cuarto cumulante.

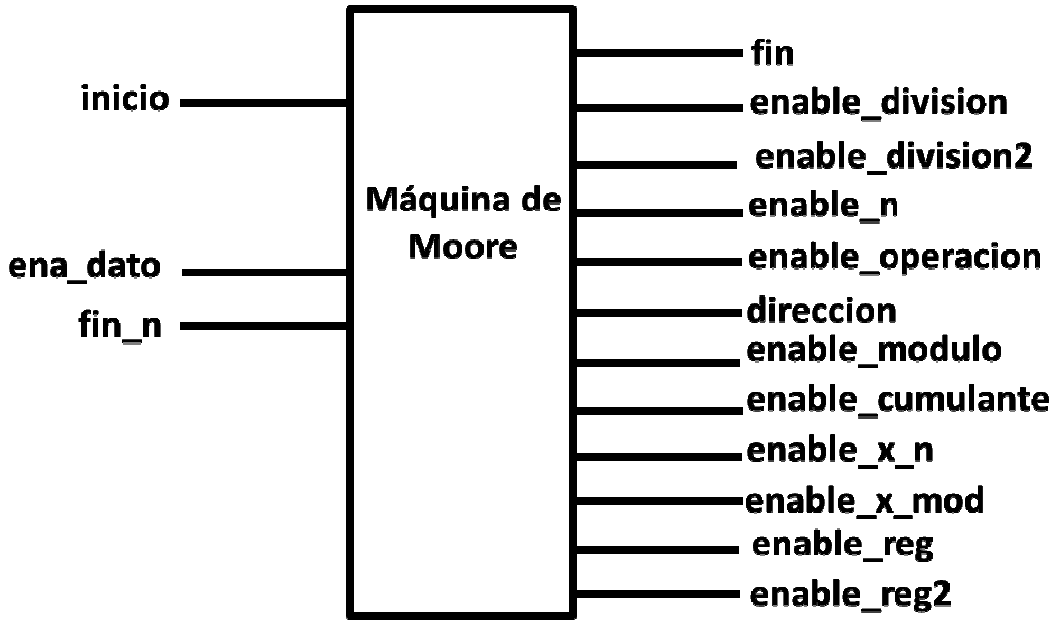

Figura 24 Diagrama de bloques maquina estados cumulante cuarto orden

Vemos que existen muy pocas diferencias entre esta figura y la correspondiente al cumulante de segundo orden. Una de estas diferencias es que en lugar de una señal *enable_division* aparece también otra señal *enable_division2*, estas señales activan el proceso de la primera y segunda división respectivamente.

Además aparece otra señal más *enable_reg2* que de forma análoga a la señal *eanble_division2* será la encargada de habilitar el registro de los valores de la segunda división.

Por último también se ha añadido una señal que habilitará la obtención del resultado final. Si se observa la expresión del cumulane de cuarto orden, se aprecia que hay que realizar una resta de los dos resultados de las divisiones, esto es precisamente lo que posibilita la señal *enable_cumulante*, ya que habilita la resta de estos dos valores intermedios.

El resto de las entradas y salidas de la máquina de estados son iguales a las del cumulante de segundo orden, por ellos sus funcionalidad la podemos recordar en el apartado 3.1.1 de este capítulo.

En el caso del diagrama de estados si comparamos el del segundo cumulante con este que nos ocupa ahora, no se verá ninguna diferencia, ya que el proceso es exactamente igual en ambos casos. Las transiciones se realizarán de la misma forma en cualquiera de los dos cumulantes.

Aun así la Figura 25 muestra el diagrama de estados del cumulante de cuarto orden.

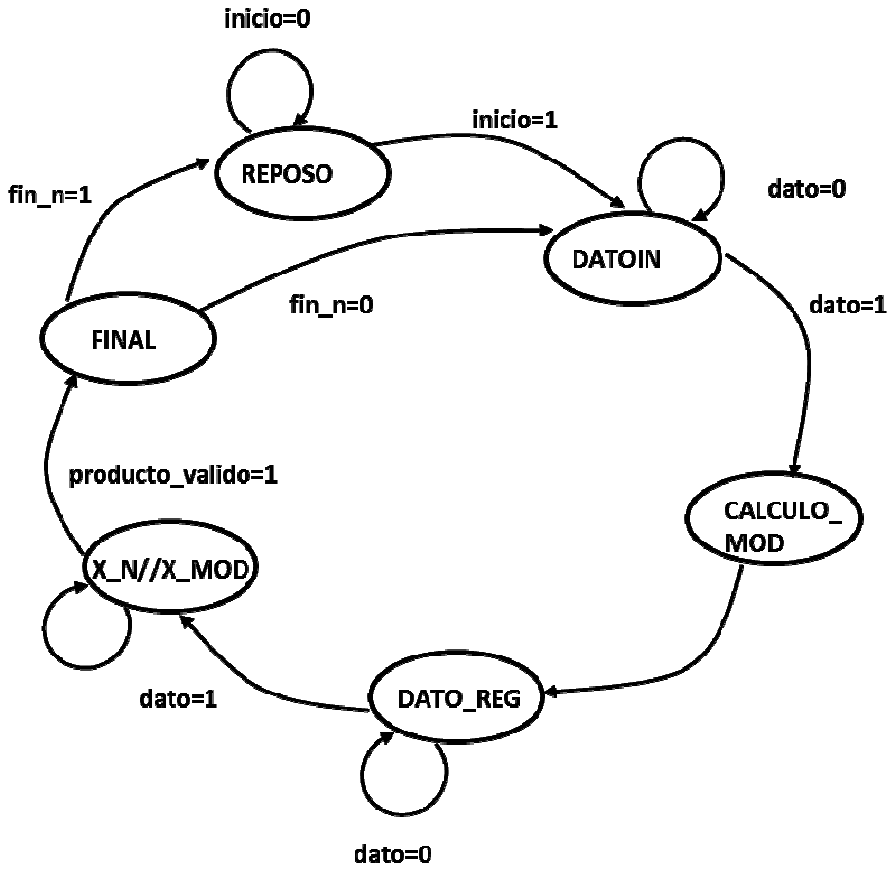

Figura 25 Diagrama estados maquina estados cumulante cuarto orden

Por lo tanto los estados y las transiciones entre ellos se realizará de forma totalmente similar al diseño anterior, estos estados quedarón definidos en el apartado 3.1.1. de este capítulo.

Como ya se ha explicado al inicio de este punto, la mayor diferencia entre ambos diseños radica en la necesidad de realizar dos divisiones en el calculo del cumulante de cuarto orden. Para realizar estas divisiones se ha optado por utilizar dos divisores en lugar de uno, ya que el fin de este estudio es obtener una aceleracion en la obtención del resultado final.

No obstante los dos divisores se activarán de forma simultánea, por lo que las señales *enable_division* y *enable_division2* se activarán a la vez, para permitir que los procesos de las dos divisiones se realice de forma simultanea.

La otra diferencia que obtenemos en este cumulante es que habrá mayor número de operaciones, por lo que también aumenta el número de valores intermedios que deberemos almacenar.

3.2.2. Bloque operaciones.

Dentro de este bloque se realizarán operaciones de suma y resta, productos, cálculo modulo y un contador.

Explicaremos el control que se ha realizado sobre estas operaciones y el sentido de cada una de las señales empleadas.

- El cálculo del módulo, de nuevo es un cálculo idéntico al ya realizado en el diseño del cumulante de segundo orden, véase apartado 3.1.2.

- De la expresión que define el cumulante de orden cuatro, se sabe que hay múltiples operaciones y para poder guardar los valores de todas ellas, se ha necesitado definir una serie de señales destinadas a este propósito.

  La señal *producto_uno* será la encargada de dar el resultado obtenido de multiplicar *x_n_reg* y *x_mod_reg*al cubo, donde esta última es la encargada de guardar el valor de memoria indicado por la dirección a donde apunta el *modulo* elevada al cubo, tal y como indica la fórmula de este cumulante. La señal *suma_uno* indica el valor del primer sumatorio, es decir la suma de todos los valores de *producto_uno*. La siguiente Figura 26 representa un diagrama de bloques de estas operaciones.

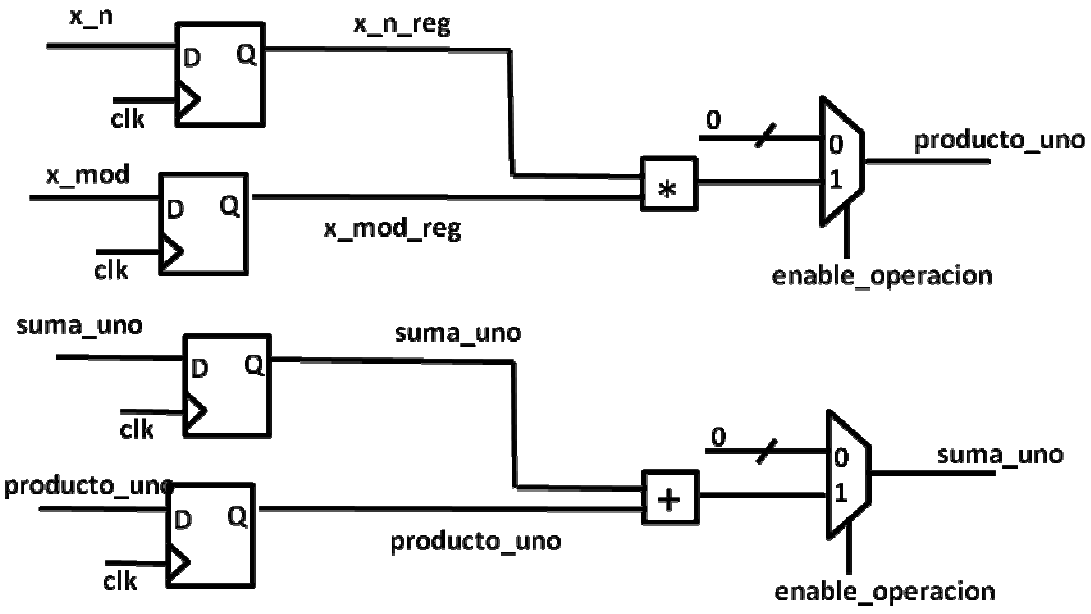

Figura 26 Bloques operaciones *suma_uno y producto_uno*

De forma similar la señal *producto_dos* indica los valores del producto de *x_n_reg* y *x_mod_reg*, que se correspondería con el valor de *producto* del diseño del cumulante de segundo orden. Y la señal *suma_dos* tendrá como resultado el valor resultante de realizar el sumatorio de los valores de *producto_dos*. La Figura 27 representa un diagrama de bloques de esta situación.

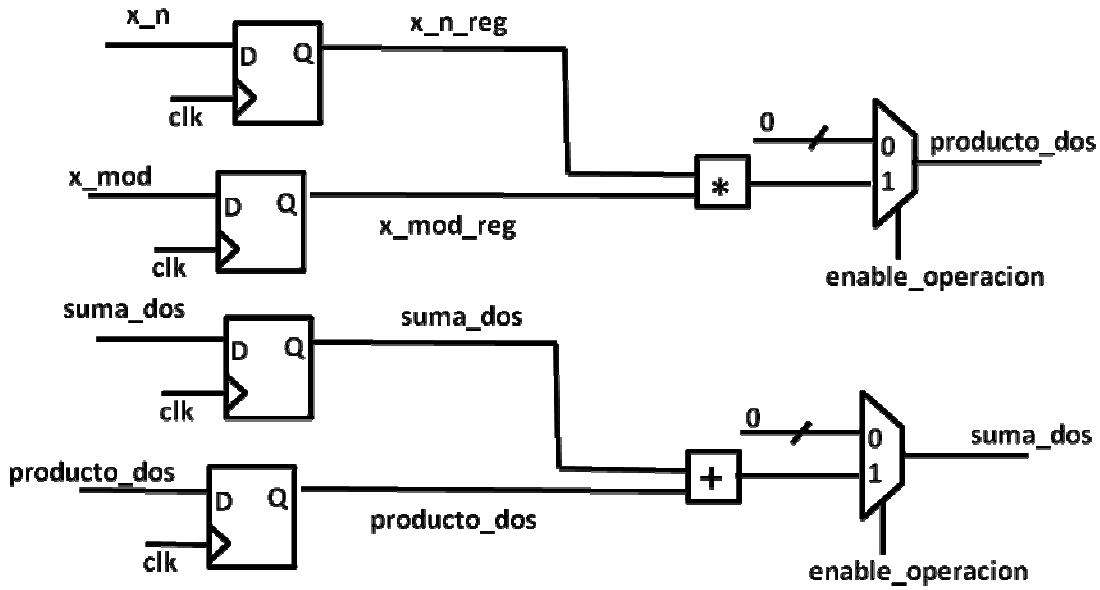

Figura 27 Diagrama bloques *suma_dos y producto_dos*

Por último tenemos la señal *suma_tres*, encargada de almacenar el sumatorio de todos los valores de *x_n_reg*, que almacena el cuadrado del valor de *x_n_reg*. Ver la Figura 28.

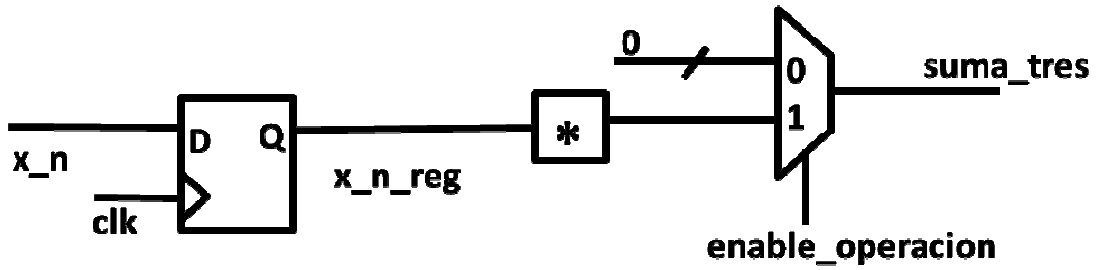

Figura 28 Diagrama de bloques *suma_tres*

En el momento en que la señal *enable_operacion* se activa con un nivel lógico alto, es cuando se realiza el control de todas las señales descritas anteriormente.

- Igual que en el diseño del cumulante de orden dos, también es necesario un contador que nos indique cuando ha llegado el final del proceso, y este contador es exactamente igual al descrito anteriormente en el capítulo 3.1.2.

- Por último hay un bloque que representa la resta la resta de los dos valores que se obtendrán del cálculo de las dos divisiones a realizar. Esto es, en el momento en que se active la señal *enable_cumulante* se realizará la resta de estos dos valores.

3.2.3. Divisores

En el bloque de los divisores, se van a utilizar dos divisores exactamente iguales al empleado en el apartado 3.1. Por lo que aquí solamente se detallarán las diferencias existentes en el control de los divisores.

En el caso de los divisores tanto el diagrama de bloques como el diagrama de estados son exactamente iguales a los ya descritos en el apartado 2.1.3., no hay ningún tipo de diferencia respecto al empleado en el diseño del segundo cumulante. La única diferencia es que las señales del segundo divisor se nombrarán igual que en el diseño anterior, pero con un "2" final para poder distinguir entre los dos divisores empleados.

La activación del inicio del proceso de la división global viene dado al igual que en el caso del cumulante de segundo orden por las señal *inicio_maquina* que se encarga de avisar del momento en el cual debe activarse todo el proceso. Como ya hemos explicado anteriormente este momento viene determinado por la máquina de estados del cumulante, en este caso, de cuarto orden, cuando se ha finalizado el cálculo de los distintos sumatorios, es decir, que tenemos el valor final de este almacenado en *suma_uno*, *suma_dos* y *suma_tres* es cuando esta señal debe activarse y se corresponden con las señales del cumulante de cuarto orden *enable_division* y *enable_division2,* que darán lugar a la activación del proceso de los dos divisores. Es decir que ahora tendremos dos señales que activen los dos procesos de control de los dos divisores.

La gran diferencia que existe entre los dos diseños es que en este caso, al tener dos divisores con restauración hay que tener claro que valores son los que deben entrar en cada uno de los divisores con restauración.

Estos valores dependerán de que las señales *enable_division* y *enable_division2* se encuentren activas o no, estas señales estarán vinculadas al primer y segundo divisor con restauración respectivamente.

A diferencia del cumulante de segundo orden, en este caso se realizan dos divisiones, los valores de los divisores (denominadores) serán diferentes, en la primera división que se debe realizar el divisor es *N, denominador*, mientras que en la segunda el divisor es N^2, *denominador2*, tal y como podemos ver en la Ecuación 2. Ecuación Cuarto Cumulante.

En el caso del primer divisor con restauración el valor que debe tener *numerador* es el resultado del primer sumatorio, es decir *suma_uno*. Y el valor de *denominador* debe ser *N*, que coincide con el número de puntos por muestras. Este último valor es un entero, por lo que se debe realizar la conversión a vector, indicando *ancho* como el tamaño del vector final.

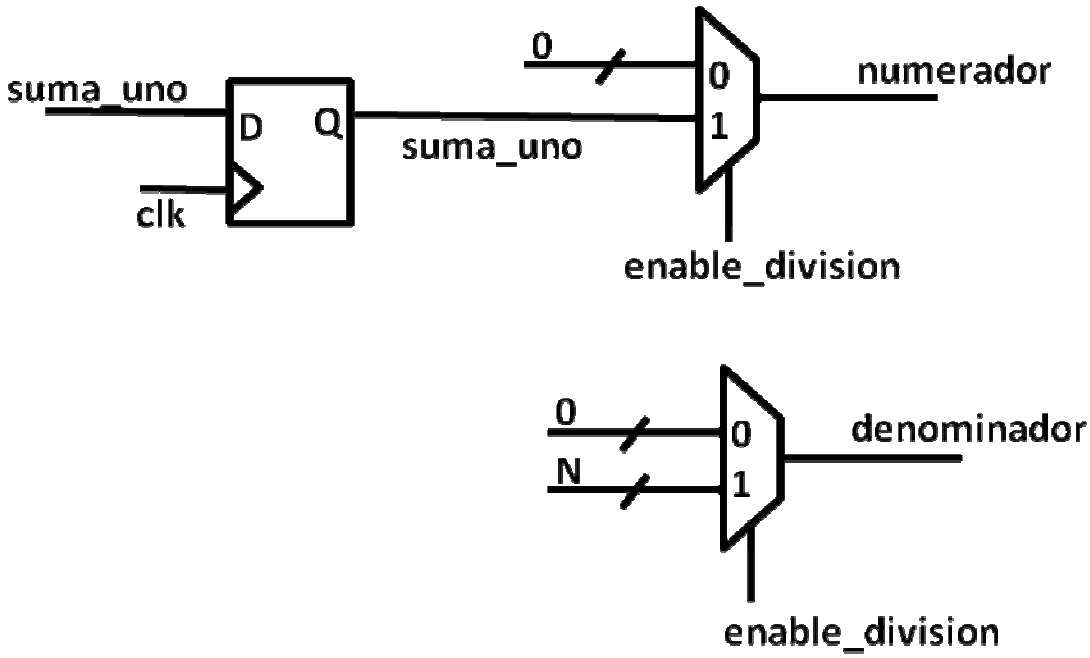

Figura 29 Entradas al Divisor 1

En la segunda división el valor que toma la señal *numerador* es el valor del producto obtenido entre *suma_dos* y *suma_tres*. Y como *denominador*N^2.

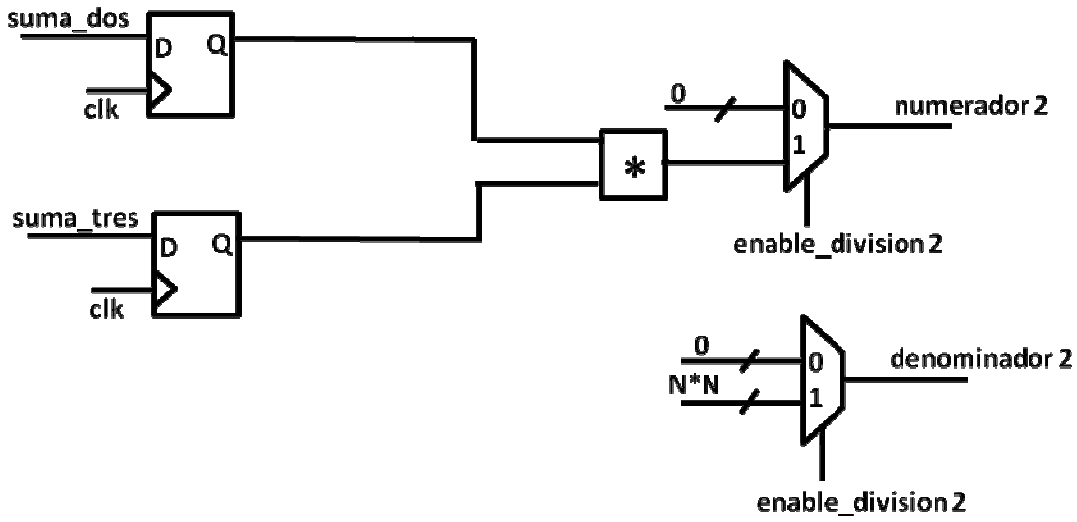

Figura 30 Entradas Divisor 2

El resultado final del valor del cumulante de cuarto orden, *cumulante*, será la diferencia existente entre los cocientes resultantes de los dos divisores con restauración.

Capítulo 4: Resultados experimentales

4.1. Herramientas empleadas.

Han sido varias las herramientas empleadas para llevar a cabo tanto el diseño del sistema como los resultados experimentales del estudio descrito en los apartados anteriores.

En primer lugar mencionar que para realizar el diseño del sistema propuesto, era necesario utilizar como herramienta de diseño un lenguaje de descripción hardware de alto nivel apto para este tipo de diseños y un programa que nos facilitara su simulación.

Se decidió emplear como lenguaje de descripción hardware de alto nivel el VHDL y la herramienta de simulación de circuitos digitales sobre la que se ha trabajado durante todo el trabajo ha sido ModelSim de Mentor Graphics, desde la página web de Xilinx, www.xilinx.compuede descargarse gratuitamente una versión de evaluación de prueba de tres meses. Para realizar la implementación del diseño se ha utilizado el entorno de desarrollo ISE de Xilinx.

Con ModelSim, lo que se ha conseguido es implementar el diseño y realizar las simulaciones que han permitido comprobar el correcto funcionamiento del diseño planteado.

Además del diseño, es necesario saber si los resultados obtenidos a través de la simulación coinciden con los que se obtendrían realizando los cálculos de forma manual. Y para poder llevar a cabo esta comprobación se ha utilizado una hoja de cálculo de Excel, ya que a través de las fórmulas correctas es posible saber de manera rápida cuales deberían ser los resultados obtenidos en nuestro diseño.

Estos resultados también se podían haber realizado con Matlab, pero a la hora de hacer comprobaciones de los cálculos intermedios para poder encontrar los distintos errores que han ido apareciendo a lo largo del diseño, era mucho más sencillo hacerlo con Excel ya que este permitía de forma más rápida comprobar valores intermedios.

Sin embargo Excel no aporta datos en cuanto a la duración del cálculo, de forma que también se ha tenido que implementar en Matlab las funciones de los cumulantes estudiados, para poder obtener datos referentes al tiempo que se tarda en realizar el proceso.

4.2. Resultados experimentales.

Como se ha dejado claro durante todo el documento, el objetivo es conseguir una aceleración hardware del sistema propuesto, es decir conseguir que este diseño sea más rápido en el cálculo de los cumulantes que la función en Matlab del cálculo de cumulantes.

Antes de entrar a ver los tiempos empleados por las dos herramientas mencionadas anteriormente, lo primero que se debe comprobar experimentalmente es el correcto funcionamiento del diseño descrito, ya que si el diseño que se ha realizado en VHDL no proporcionase los resultados correctos, esto no tendría ningún sentido.

Para realizar esta comprobación de los resultados se ha empleado una hoja Excel, en la que se muestran todos y cada uno de los cálculos intermedios necesarios para realizar los cálculos de los cumulantes de segundo y cuarto orden.

Estos cálculos intermedios se corresponden con el cálculo de *modulo*, *suma* y *producto* con respecto al cálculo del cumulante de segundo orden. Y el cálculo de *modulo*, *suma_uno*, *producto_uno, suma_dos, producto_dos,* suma_tres, *cociente_uno y cociente_dos*, para el caso del cumulante de cuarto orden.

Para obtener estos resultados, se deben introducir los datos de la señal que se quiere analizar, estos datos han sido facilitados por el trabajo realizado en [Gua 12], quien ha proporcionado un fichero con todos los datos que se debían emplear en el estudio. Estos datos se corresponden a los resultados que obtuvo en el procesamiento previo de las señales procedentes de electrocardiogramas.

Aquí se debe aclarar que los datos que se han empleado son directamente los procedentes de este fichero, no se ha realizado ninguna manipulación para su posterior uso; pero sin embargo es necesario comprender, tal y como se indica en [Mar 06], que los datos que se deberían usar para obtener valores reales, tendrían que ser tratados de forma que:

$$X' = X - \mu(X)$$

Donde: *X*, los datos obtenidos directamente del fichero facilitado por [Gua 12],

$\mu(X)$, la media de cada uno de las muestras que componen *X*,

X', los valores que realmente se deberían emplear.

Estas señales disponen de 91 puntos por cada muestra tomada y estarán almacenados en la primera columna de la tabla Excel empleada. En las sucesivas columnas se han introducido las fórmulas correspondientes a los cálculos intermedios mencionados anteriormente.

Finalmente se obtiene el valor de cada uno de los cumulantes. Recordemos que este valor parte de un total de 91 puntos, y que al tener constancia de que la mejor caracterización de estos cumulantes viene dado por un valor determinado de *i* (i=36 en el caso del segundo cumulante y *i*=56 para el cumulante de orden cuatro) el resultado será un único valor, que se podría representar en una gráfica de dos dimensiones, tal y como se puede apreciar en la Figura 12.

Un pequeño ejemplo de los valores obtenidos en Excel, se representan en las siguientes tablas:

MUESTRA	RESULTADOS CON MODELSIM	RESULTADOS CON EXCEL
Muestra 1	6,7168E+16	6,7168E+16
Muestra 2	6,0865E+16	6,0865E+16
Muestra 3	5,7469E+16	5,7469E+16
Muestra 4	5,0644E+16	5,0645E+16
Muestra 4	5,3519E+16	5,3519E+16

Tabla 2 Valores cumulante segundo orden

MUESTRA	RESULTADOS CON MODELSIM	RESULTADOS CON EXCEL
Muestra 1	2,90468E+31	2,9047E+31
Muestra 2	2,9624E+31	2,9624E+31
Muestra 3	5,0056E+31	5,0056E+31
Muestra 4	3,6516E+31	3,6516E+31
Muestra 4	2,8949E+31	2,8949E+31

Tabla 3 Valores cumulante cuarto orden

Estos resultados obtenidos en Excel solamente nos sirven para comprobar que los cálculos de los cumulantes se realizan de forma correcta. Una vez que se confirma que los datos son correctos se realiza una comprobación del tiempo que tarda nuestro diseño en realizar los cálculos y el tiempo que tarda en hacerlo en Matlab.

Para comprobar el tiempo se ha realizado la función de cada uno de los cumulantes en Matlab y se ha empleado la función que nos indica el tiempo que tarda en realizar los cálculos de una función.

El tiempo medio que emplea Matlab para llevar a cabo estos cálculos vienen expresados en la siguiente tabla:

	Tiempo medio de Matlab en ns
Cumulante de segundo orden	27.000
Cumulante de cuarto orden	52.000

Tabla 4 Tiempo medio Matlab

Los valores de esta tabla se han obtenido realizando una media del tiempo que tarda en realizar los cálculos de una muestra de 300 elementos.

Sin embargo a continuación indicamos los tiempos que nos proporciona la herramienta ISE de Xilinx para el diseño propuesto anteriormente, y para una frecuencia máxima de 74.304MHz.

	Tiempo diseño a través de ISE en ns
Cumulante de segundo orden	6.625
Cumulante de cuarto orden	6.818

Tabla 5 Tiempo medio ISE

Es fácil darse cuenta de que la aceleración que proporciona el uso de la implementación de este sistema en Hardware es más que significativa. En el caso del cumulante de segundo orden, la aceleración es del orden de unos 20µs, mientras que para el caso del cumulante de cuarto orden hablamos de una aceleración de unos 45µs.

Además hay que tener en cuenta que al emplear la herramienta de Matlab, esta sólo permite realizar el cálculo de uno de los cumulantes, es decir que para obtener el resultado de los dos cumulantes el tiempo que emplearía sería la suma del tiempo que tarda en calcular el primer cumulantes más el tiempo del segundo. Sin embargo con el diseño propuesto se pude realizar el cálculo de forma simultánea, lo

que implica que el tiempo que se tardaría en realizar el cálculo de los dos cumulantes sería el tiempo del cálculo más lento, esto es el del cumulante de cuarto orden.

	Tiempo calculo simultáneo de los dos cumulantes
Diseño propuesto	6.818 ns
Función Matlab	79.000ns

Tabla 6 Comparativa tiempo de cálculo de los dos cumulantes

Esto proporciona una ventaja adicional a la simple aceleración del cálculo de un cumulante. De esta forma se pueden realizar cálculos de manera simultánea y acelerar aún más el proceso de cálculo.

Esto es simplemente la aceleración de una de las partes que componen todo un proceso de selección y extracción de características para una posterior clasificación, si esta implementación hardware se puede ampliar al resto del sistema de clasificación de señales, puede suponer un avance muy importante en cuanto a la estimación de recursos y tiempo de espera para el diagnóstico de la clasificación.

Por otro lado, se ha obtenido el volumen de recursos que se emplean para implementar este diseño en una FPGA Spartan 3 XC3S1000 FT256:

	Recursos segundo cumulante
LUTs	665 de 15320 (4%)
Flip Flop	571 de 15320 (3%)

	Recursos cuarto cumulante
LUTs	6953 de 15320 (45%)
Flip Flop	1731 de 15320 (11%)

Se puede observar que los recursos empleados para la implementación del cumulante de cuarto orden es muy superior al caso del cumulante de segundo orden, esto es debido a que los cálculos del cumulante de cuarto orden son mucho más complejos, por lo que el tamaño de los datos que se van a tratar en este cumulante llegan incluso a cuadriplicarse con respecto a los del cálculo del cumulante de segundo orden. Por este motivo es por el que los recursos empleados son muy superiores, ya que los datos que se deben registrar tienen mayor tamaño.

Capítulo 5: Conclusiones

5.1. Conclusiones

Para realizar la aceleración Hardware del cálculo de los cumulantes de segundo y cuarto orden, se ha desarrollado un diseño a partir de las ecuaciones de estos cumulantes desarrollados por [Gua 12], donde se demuestra que estos cumulantes son los más significativos a la hora de la extracción de las características de las señales procedentes de un electrocardiograma.

Si nos basamos en los resultados experimentales obtenidos en el capítulo anterior, vemos que hay una gran diferencia entre desarrollar el cálculo de estos dos cumulantes en Matlab, y hacerlo con el diseño propuesto implementado a través de ISE.

Los tiempos mejoran de forma considerable realizando la implementación Hardware del cálculo de estos dos cumulantes.

Además de la aceleración se obtiene la ventaja de que con este diseño se puede llegar a realizar varios cálculos simultáneamente, ahorrando más tiempo aun. Esto es, si queremos realizar el cálculo de los dos cumulantes simultáneamente, con el diseño Hardware podemos ahorrar tiempo ya que se pueden lanzar los dos cálculos al mismo tiempo, de forma que el tiempo total que se emplearía en los dos cálculos sería el tiempo que tarde el cálculo más lento.

Sin embargo en Matlab primero se debe desarrollar uno de los cumulantes y luego el otro, esto implica que el tiempo final para los dos cálculos es el equivalente a la suma de los dos cálculos.

Con esto vemos que es mucho más beneficioso emplear una implementación Hardware, ante el obvio ahorro de tiempo que esto supone para el desarrollo del sistema.

5.2. Trabajos futuros

Hay varias formas de hacer que el diseño realizado sea más eficaz desde el punto de vista temporal.

La primera de ellas, y además totalmente necesaria, se encuentra en el modo en que se obtiene la división de los distintos valores. En este estudio se ha empleado un divisor serie para realizar estos cálculo; por falta de tiempo se adaptó el sistema a un divisor que ya estaba diseñado.

Si se realizara el cambio del divisor serie por otro paralelo de las mismas características se podría obtener, sin duda alguna, una mayor aceleración del proceso de cálculo. Esto generaría un mayor margen de tiempo para poder emplear en otras partes del sistema de clasificación y detección de características.

Además sería interesante realizar pruebas experimentales sobre una FPGA para evaluar el efecto que tiene la lectura de los datos a procesar sobre la velocidad del proceso completo.

Finalmente, se propone continuar con el diseño e implementación hardware de otras fases del proceso de identificación de señales.

REFERENCIAS

[Meh 07] SVM for beat detection.

[Jal 09] Nasiri ECG classification with SVM and Genetic Algorithm.

[Son 05] SVM Classification using reduced features.

[Kar 10] Classification ECGs SVM.

[Can 10] Arrhythmia Classification.

[Yin 12] Classification of Cardiac Arrhytmia via SVM.

[Gua 11] Sistema de Identificación de señales arrítmicas.

[Gua 12] Estudio de máquinas de vector soporte en sistemas embebidos para instrumentación.

[Pao 12] An embedded system for on field testing of human identification using ECG biometric.

[Mat 10] FPGA-oriented HW/SW implementation of ECG beat detection and classification algorithm.

[Sar 13] Estudio de máquinas de vectores de soporte para clasificación de electrocardiogramas.

[Mar 06] Power quality disturbances detection using HOS.

[MIT] MIT-BIH ARRHYTHMIA DATABASE Base de datos de señales de electrocardiogramas, http://ecg.mit.edu/dbinfo.html