Projecte de Fi de Carrera

Enginyer Industrial

Desarrollo de un prototipo de un Sistema en un Chip Programable (SOPC) destinado a aplicaciones de tratamiento digital de imágenes en movimiento

Autors: Carles Rellán Martínez
Miguel Ángel Sánchez López

Director: Emili Lupón Roses

Convocatòria: Setembre 2004 (pla 94)

Escola Tècnica Superior
d'Enginyeria Industrial de Barcelona

Índice

A Imágenes en dominios espectrales

A.1 Introducción

En este anexo se tratarán de exponer los conceptos y las técnicas relacionadas con el tratamiento de imágenes en dominios espectrales, utilizados especialmente a la hora de comprimir la imagen tal y como está comentado en el apartado 2.4 de la memoria. La información contenida en este anexo es tan significativa como la que hay en el resto de capítulos de la memoria, pero al no estar tan directamente relacionada con lo que es el sistema en si, se ha decidido separarla en formato de anexo.

A.2 Acerca de los dominios espectrales

Tal y como está comentado en el apartado 2.4 (página 15 de la memoria), es necesario hacer una clasificación categórica de la información contenida en la imagen a fin de poder seleccionar aquella que se considere de mayor importancia.

En cierto modo, esto es lo que proporciona la transformación de señales a dominios espectrales; se puede descomponer cualquier señal periódica en la suma de otras señales periódicas más sencillas permitiendo de este modo el poder discriminar aquellas componentes en las que se tenga interés. Esta discriminación es lo que se conoce como filtrado.

En el mundo del tratamiento digital de señales es de sobras conocido el hecho de que se pueden tomar fragmentos de una señal discreta compuesta de un conjunto de muestras determinado y aplicar sobre dicho conjunto operaciones de transformación a dominios espectrales. Como resultado de dichas operaciones se obtiene un conjunto de tantos parámetros como muestras originales, que indican cómo son las funciones en las que se descompone el conjunto original [Proakis, 1998; pág.235:399].

Estas funciones componentes suelen ser las funciones trigonométricas seno y coseno, con lo que la información que contienen los parámetros resultantes de la operación de transformación son simplemente las características de una onda: **la frecuencia, la amplitud** y **la fase**.

A.3 Interpretación de los parámetros de las componentes

Una imagen digitalizada no es más que una matriz de parámetros que definen las propiedades ópticas de cada punto. En este proyecto con la información referente a la luminosidad de las imágenes es más que suficiente; se trabajará con imágenes en blanco y negro. Con esto, el valor de cada posición de la matriz será proporcional a la intensidad de la luminosidad que le corresponda. Como tal conjunto de valores, una imagen digitalizada también es tratable mediante técnicas de dominios espectrales; simplemente hay que interpretar los parámetros de las componentes.

Lo que se quiere es clasificar la información que contiene una imagen según su orden de importancia para poder prescindir de los detalles más insignificantes. Este sistema no pretende hacer localización exacta de objetos mediante métodos de búsqueda de patrones, sino que simplemente lo que se quiere es seguir un objeto a grosso modo. Por lo tanto, estos detalles

insignificantes estarán basados principalmente en la forma de los objetos contenidos en la imagen.

No se desea saber si el objeto que se está reflejando es una mano, un objeto redondo, un objeto cuadrado o lo que sea. Simplemente se quiere hacer un seguimiento de algo que se mueva cerca del objetivo.

Para mostrar la viabilidad del tratamiento de la imagen a través de dominios espectrales, se realizaron pruebas que trataban de mostrar como la información en una imagen podía clasificarse en mayor o menor importancia según caracterice de forma más o menos detallada a un objeto de la misma. Para ello se hizo uso del software SPIP (Scanning Probe Image Processor) 2.30 para Windows (incluido en el CD), el cual es un programa de tratamiento de imágenes con varios módulos de gran utilidad para el cometido de ese Proyecto. Se usó una opción que permite obtener superficies 3D a partir de imágenes en 2D. En estas superficies 3D pueden percibirse de forma más clara los detalles de la imagen (Fig. A.1). Mediante esta transformación se puede observar que la interpretación de la imagen en 3D permite visualizar detalles que no son perceptibles en 2D. Son precisamente este tipo de componentes las que pretenden ser eliminadas para evitar utilizar información innecesaria.

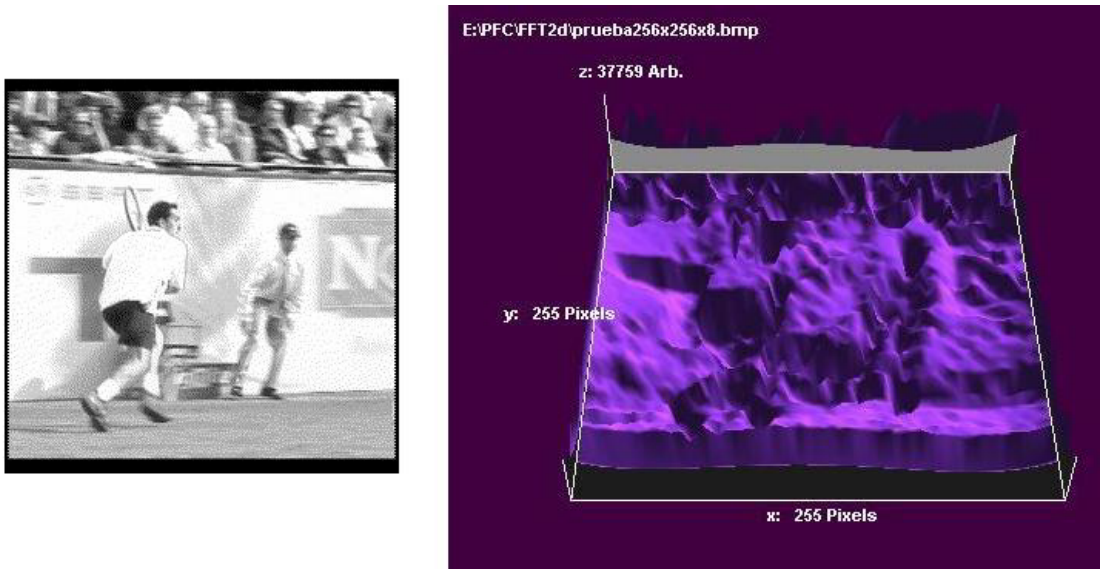


Fig. A.1 Representación 3D de una imagen generada por SPIP.

En el apartado previo se comentó la descomposición de las muestras en formas senoidales. En el caso de un conjunto de muestras en dos dimensiones, como es una imagen digital, se puede aplicar exactamente lo mismo, pero pensando en superficies de cumbres y valles, tal como se ve en la Fig. A.1. Con ésto, los parámetros de cada onda se pueden interpretar de la siguiente forma:

- **Frecuencia** o longitud de onda: Esto nos indica la extensión de una componente en el espacio delimitado por las muestras. A menor frecuencia, mayor longitud de onda, es decir, mayor extensión geométrica de la componente.
- **Amplitud**: Límites de nivel de gris de la componente.
- **Desfase**: Determinación de la posición de la componente dentro del espacio delimitado por las muestras.

Según esta interpretación, está claro que los detalles geométricos de los que se quiere prescindir se encuentran en las componentes de alta frecuencia (pequeña longitud de onda). La Fig. A.3 muestra como al ir añadiendo componentes de alta frecuencia a partir del espectro de la señal *X(ω)* mostrada en la figura Fig. A.2, la señal (en este caso unidimensional) va ganando en detalle.

A.4 La transformada discreta de Hartley

Es de sobras conocida en el mundo del tratamiento digital de señales la utilización de la Transformada Discreta de Fourier. En este tipo de transformada la frecuencia de la componente viene determinada por la posición de dicha componente dentro del vector resultante de la transformación. Los parámetros de amplitud y fase están en forma de número complejo, siendo la parte real la correspondiente a la amplitud y la parte imaginaria la correspondiente al desfase.

A parte de la conocida Transformada de Fourier existen otras como por ejemplo: la transformada coseno, la seno, la de Slant, la de Haar, la de Walsh y la de Hartley. Entre todas estas transformadas se ha decidido utilizar para este proyecto la Transformada de Hartley.

La Transformada Discreta de Hartley (TDH) es una transformación frecuencial discreta muy similar a la Transformada Discreta de Fourier. El resto de este apartado está extraído de [Wikipedia.org, 2004], ya que este documento ofrece una introducción bastante clara sobre esta transformada.

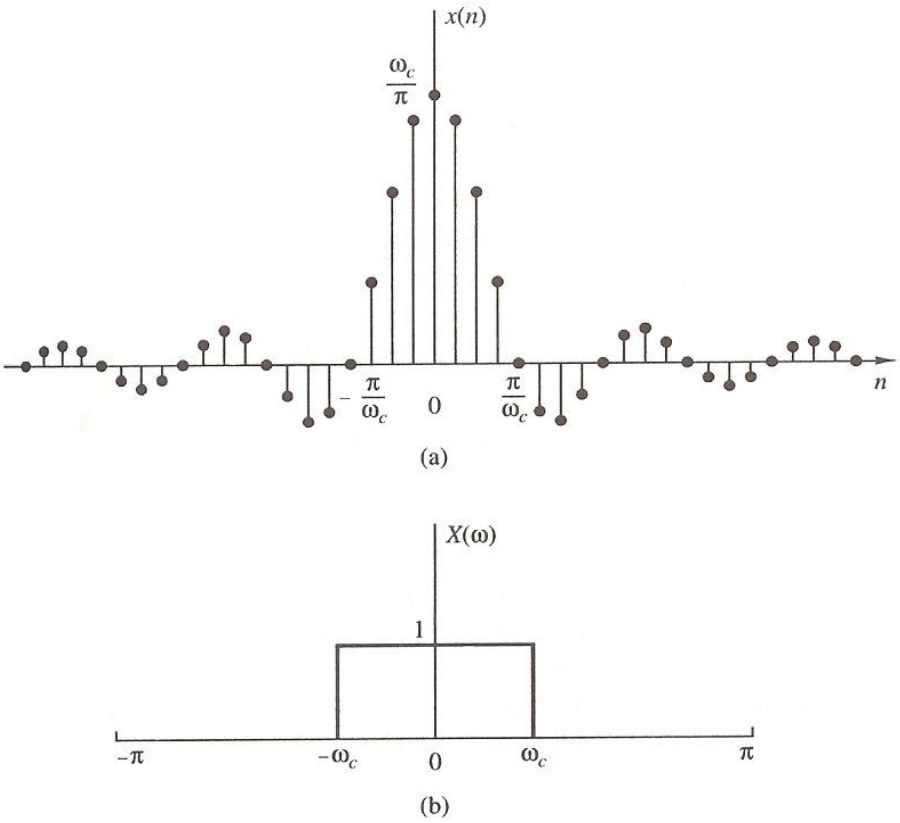


Fig. A.2 Transformada de Fourier *x*(*n*) de *n* muestras de la señal *X*(ω) [Proakis, 1998; p.263]

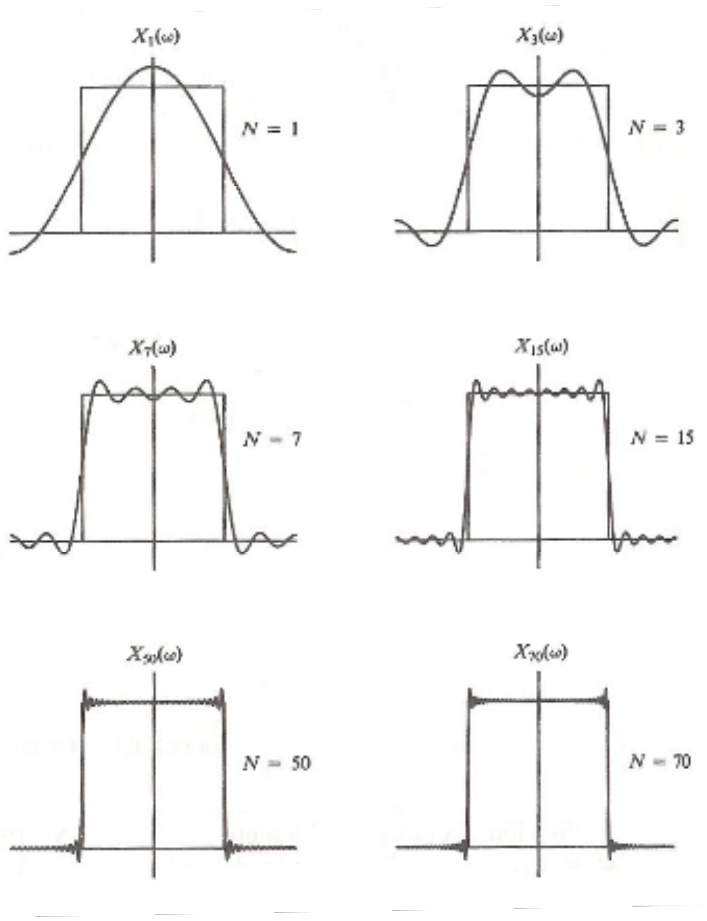


Fig. A.3 A medida que se añaden componentes, la reconstrucción de la señal se aproxima cada vez más a la forma original [Proakis, 1998; p.264].

A.4.1 Definición

Formalmente, la Transformada Discreta de Hartley (TDH) es una función linear reversible $H:\Re^N \to \Re^N$. Los N números reales $x_0,\ldots,x_{N-1}$ son transformados en los N números reales $X_0,\ldots,X_{N-1}$ según la ecuación

$$X_k = \sum_{n=0}^{N-1} x_n \operatorname{cas}\left(2\pi \frac{kn}{N}\right), \quad 0 \le k \le N-1$$

siendo $\operatorname{cas}(\theta) = \cos(\theta) + \operatorname{sen}(\theta) = \sqrt{2}\cos(\theta - \pi/4)$. Esta función debe ser contrastada con la función $\cos(\theta) - i\operatorname{sen}(\theta)$ que aparece en la definición de la Transformada Discreta de Fourier (TDF).

A.4.2 Propiedades

La transformada puede ser interpretada como la multiplicación del vector $(x_0,\ldots,x_{N-1})$ por una matriz cuadrada de $N \times N$, cosa que hace de la TDH un operador lineal. La matriz es reversible; la transformación inversa permite recuperar x_n a partir de X_k simplemente multiplicando la TDH de X_k por $1/N$. Es decir, la TDH es su propia inversa multiplicada por un factor de escala.

La TDH puede ser utilizada para calcular la TDF, y viceversa. Para entradas reales x_n, la salida F_k de la TDF tiene una parte real $(X_k - X_{N-k})/2$ y una parte imaginaria $(X_{N-k} - X_k)/2$. De forma opuesta, la TDH es equivalente a calcular la TDF de x_n multiplicada por 1+*i*, y quedarse con la parte real del resultado.

Del mismo modo que con la TDF, una convolución cíclica **z = x*y** de dos vectores **x**=(x_n) e **y**=(y_n) para obtener un vector **z**=(z_n), todos de longitud *N*, resulta una operación sencilla con el uso de la TDH. En particular, supóngase que los vectores **X, Y** y **Z** denotan la TDH de **x, y** y **z** respectivamente. Entonces los elementos de **Z** se expresan como:

$$\begin{aligned} Z_k &= \left[\left(X_k Y_{N-k} + X_{N-k} Y_k\right) + \left(X_k Y_k - X_{N-k} Y_{N-k}\right)\right]/2 \\ Z_{k-1} &= \left[\left(X_k Y_{N-k} + X_{N-k} Y_k\right) - \left(X_k Y_k - X_{N-k} Y_{N-k}\right)\right]/2 \end{aligned}$$

tomándose todos los vectores con periodo *N* ($X_N = X_0$, etcétera). Así pues, ya que la TDF transforma una convolución en una multiplicación punto a punto de números complejos (*pares* de partes reales e imaginarias), la TDH transforma una convolución en una sencilla combinación de *pares* de componentes frecuenciales reales. La inversa de la TDH ofrece pues el valor **z** deseado. Esta propiedad de la convolución es en la que se basan los algoritmos para calcular la transformada rápida de Hartley [Ullmann, 1984].

De hecho, lo que realmente ha determinado la elección de esta transformada ha sido:

- Poder trabajar solamente con números reales.
- Poder pasar a Fourier de una forma muy sencilla.

A.5 Transformación y filtrado de imágenes con la TDH

Los filtros para señales bidimensionales se pueden entender como una revolución sobre el eje de ordenadas de los filtros convencionales para señales unidimensionales. Con esto se pueden utilizar filtros como los de Butterworth o filtros ideales, por citar algunos ejemplos.

Como se ha dicho en muchas ocasiones, el objetivo es intentar simplificar las operaciones en la medida de lo posible. Tras algunos experimentos con la aplicación SPIP (*Scanning Probe Image Processor*) 2.30 para Windows (véase Fig. A.4), se llegó a la conclusión que trabajar con filtros ideales sería suficiente para los propósitos que se querían alcanzar. De esta forma, se evita el tener que hacer operaciones sobre las componentes de la imagen transformada y, simplemente enmascarando dicha matriz de componentes con una matriz que defina el filtro, se pueden obtener las componentes deseadas.

Por otro lado, a la hora de hacer la transformada de una imagen hay que plantearse algunas estrategias. Según las propiedades de la TDH comentadas en el apartado anterior, resulta que el vector transformado por la TDH es de un valor N veces mayor que el vector original, siendo N la dimensión del vector. Esto quiere decir, que a medida que la matriz que se quiera convertir crezca en tamaño, se necesitara mayor cantidad de información (mayor cantidad de bits) por componente.

Cuando se trabaja con máquinas de recursos, por decirlo de algún modo, ilimitados y sin ninguna restricción de tiempo, esto no supone demasiado problema, pero en el sistema que se quiere implementar es algo a tener en cuenta: el objetivo es que cada componente no supere

los 8 bits, siendo cada valor del vector original también de 8 bits. Por otro lado, en el apartado 2.5 se habla de la necesidad de hacer regiones para facilitar el sistema de seguimiento. Con todo es obvio que en lugar de transformar la imagen completa, se van a transformar y a filtrar fragmentos o regiones de dicha imagen.

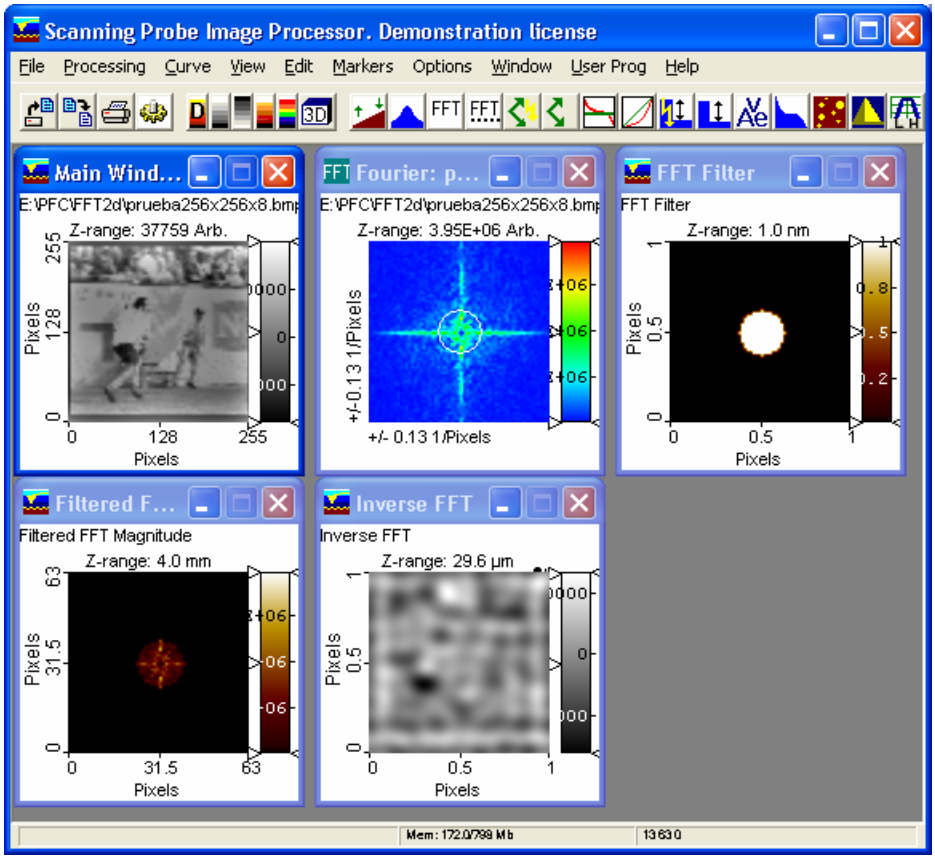


Fig. A.4 Espacio de trabajo de la aplicación SPIP

Un poco más adelante se mostrarán los resultados de experimentar con regiones cuadradas de 16 y 32 píxeles de lado, pero previamente véase el tipo de filtrado que se ha hecho. A pesar que, tal y como está implementada la máquina, se puede definir prácticamente cualquier tipo de máscara para actuar como filtro (véase 4.8, pág. 52 vol. memoria), los experimentos que se muestran a continuación han sido *filtrados* con filtros de forma cuadrada en lugar de circular como se puede ver en la Fig. A.4.

Otra cosa a tener en cuenta es que según la definición de la transformada de Hartley que se está utilizando, las componentes de baja frecuencia en las que se está interesado se encuentran en los extremos del vector. En el caso de dos dimensiones son las esquinas de la matriz, con lo cual el filtro estará centrado en las esquinas.

Por último, para hacer los experimentos se ha utilizado la imagen en blanco y negro de 256x256 píxeles mostrada en la Fig. A.1 y se ha utilizado el un software de diseño propio que se muestra en en anexo K, apartado K.4 (Pág. 127). Las tablas de las siguientes páginas muestran, la longitud del lado de la “banda pasante” del filtro, el espectro de Hartley filtrado y la reconstrucción de la imagen a partir de dicho espectro.

A.5.1 Divisiones de 16x16 píxeles

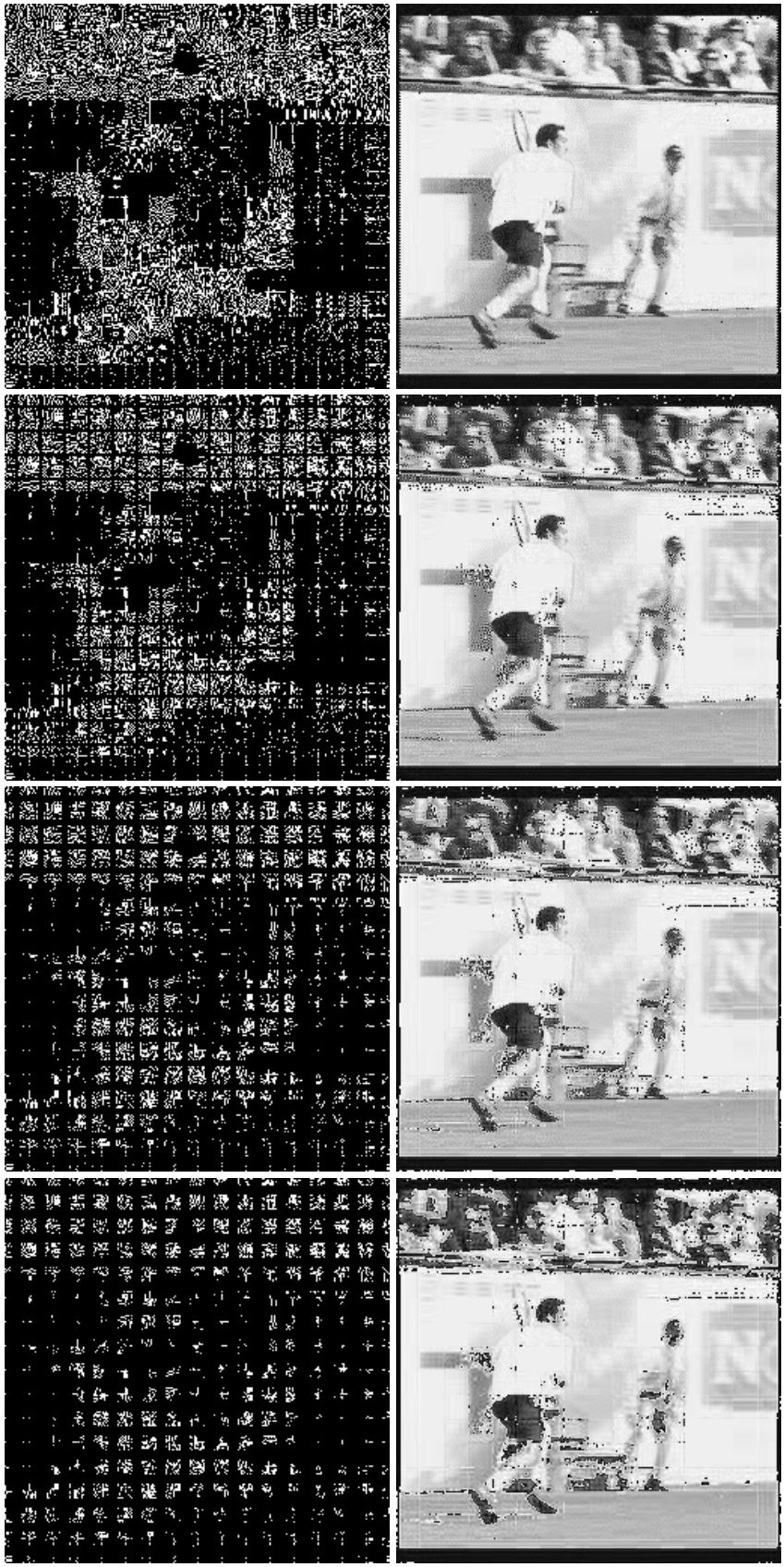


Tabla A.1 Pruebas con regiones de 16x16 píxeles (1/2)

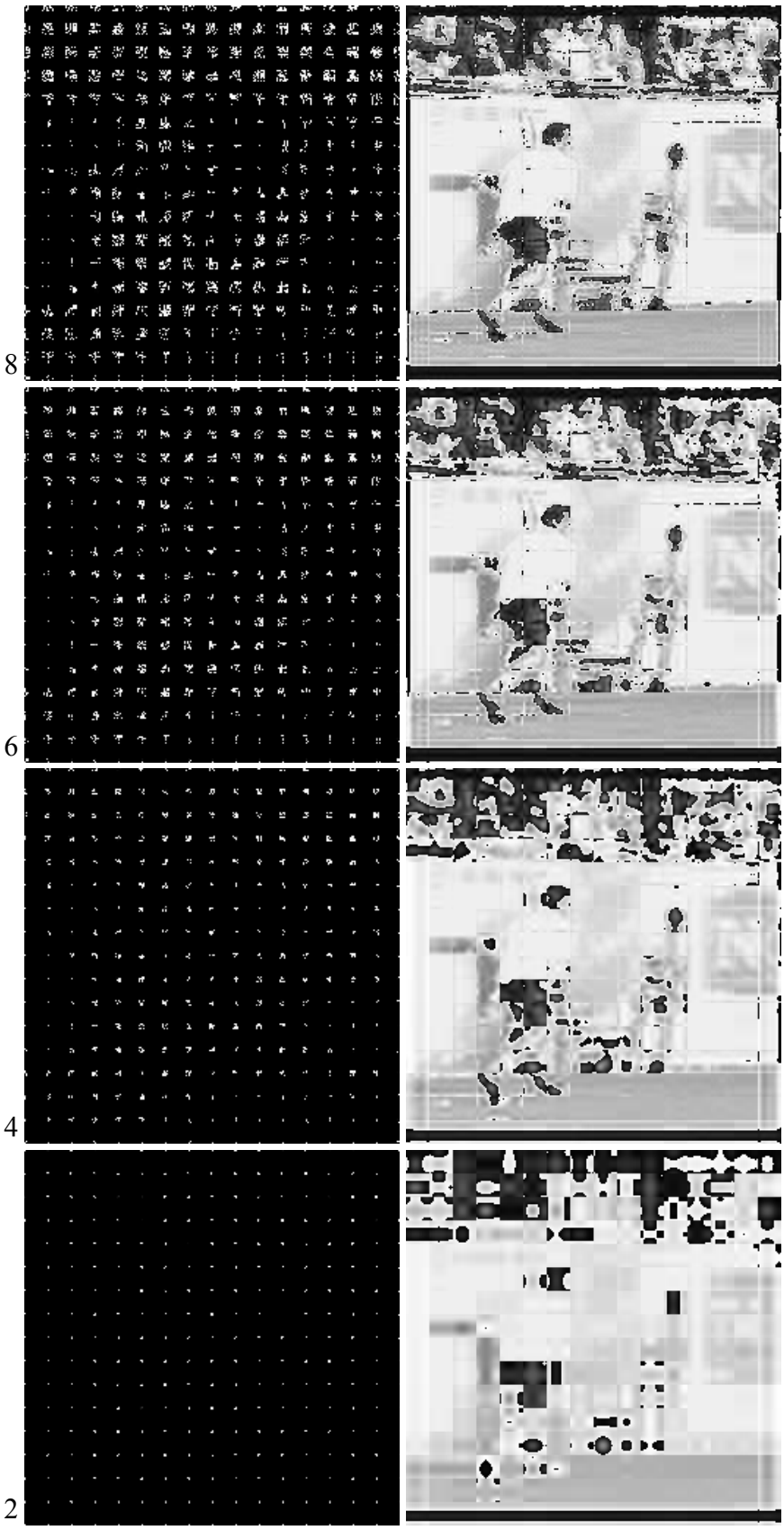


Tabla A.2 Pruebas con regiones de 16x16 píxeles (2/2)

A.5.2 Divisiones de 32x32 píxeles

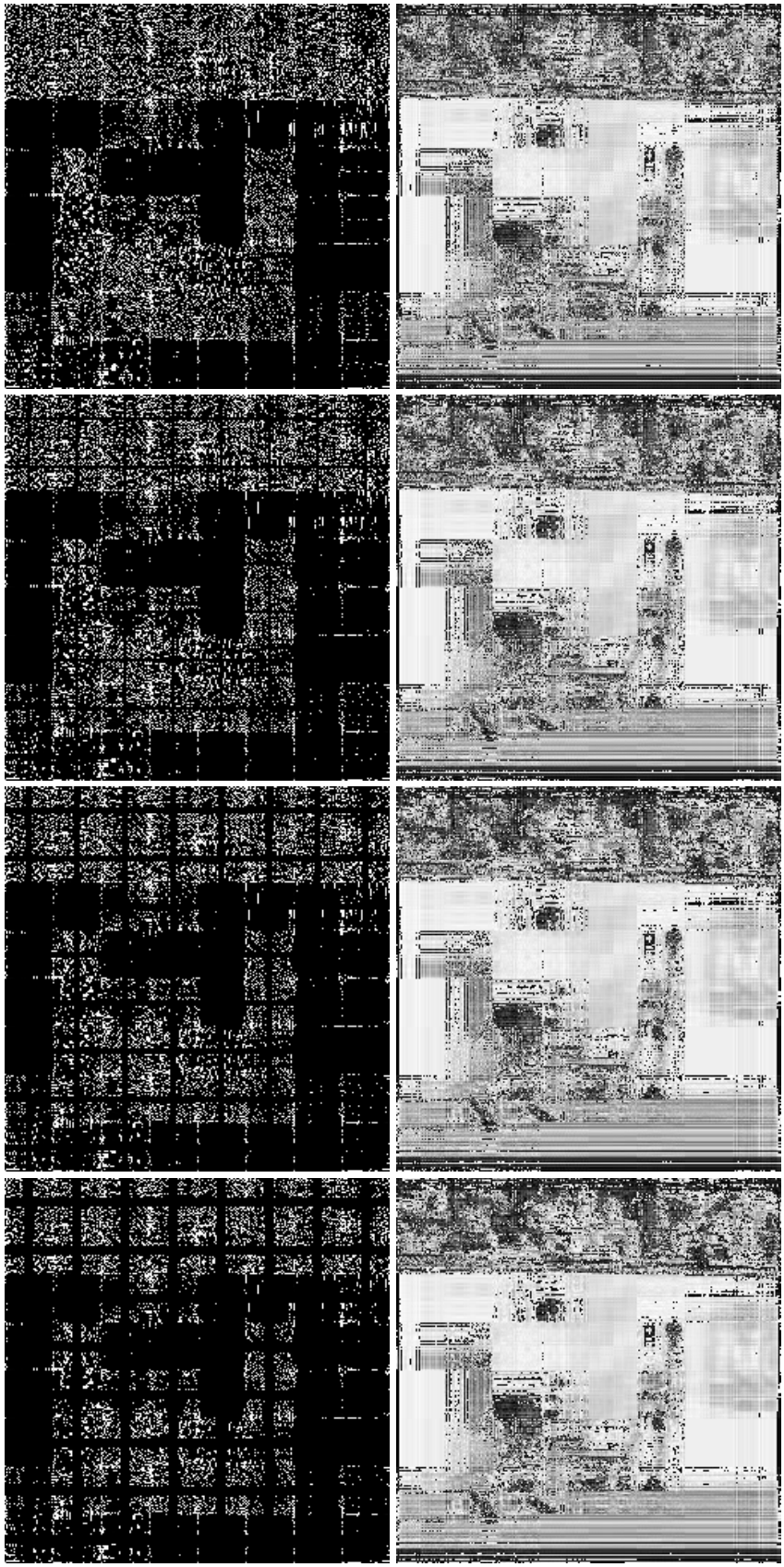


Tabla A.3 Pruebas con regiones de 32x32 píxeles (1/4)

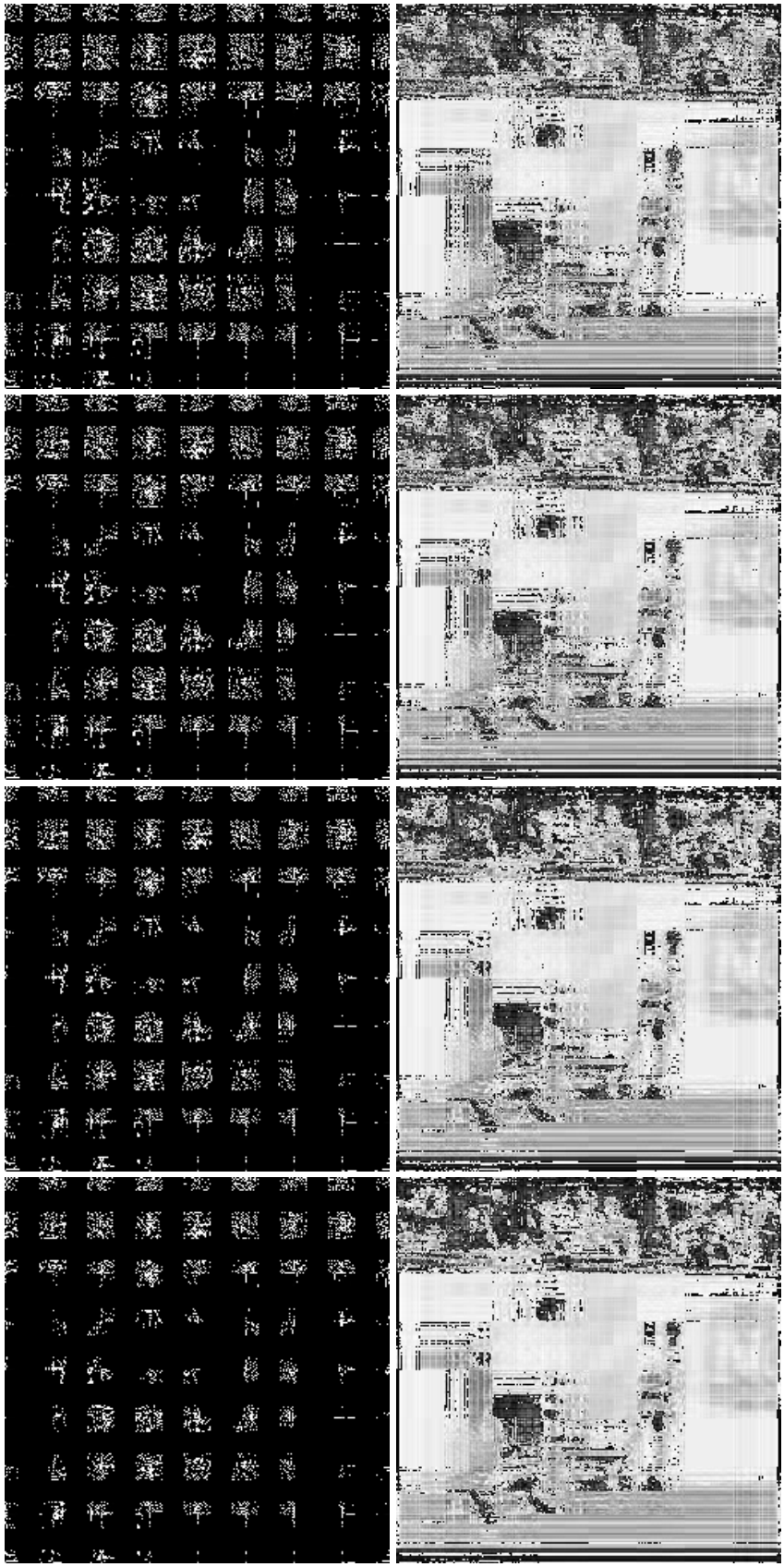


Tabla A.4 Pruebas con regiones de 32x32 píxeles (2/4)

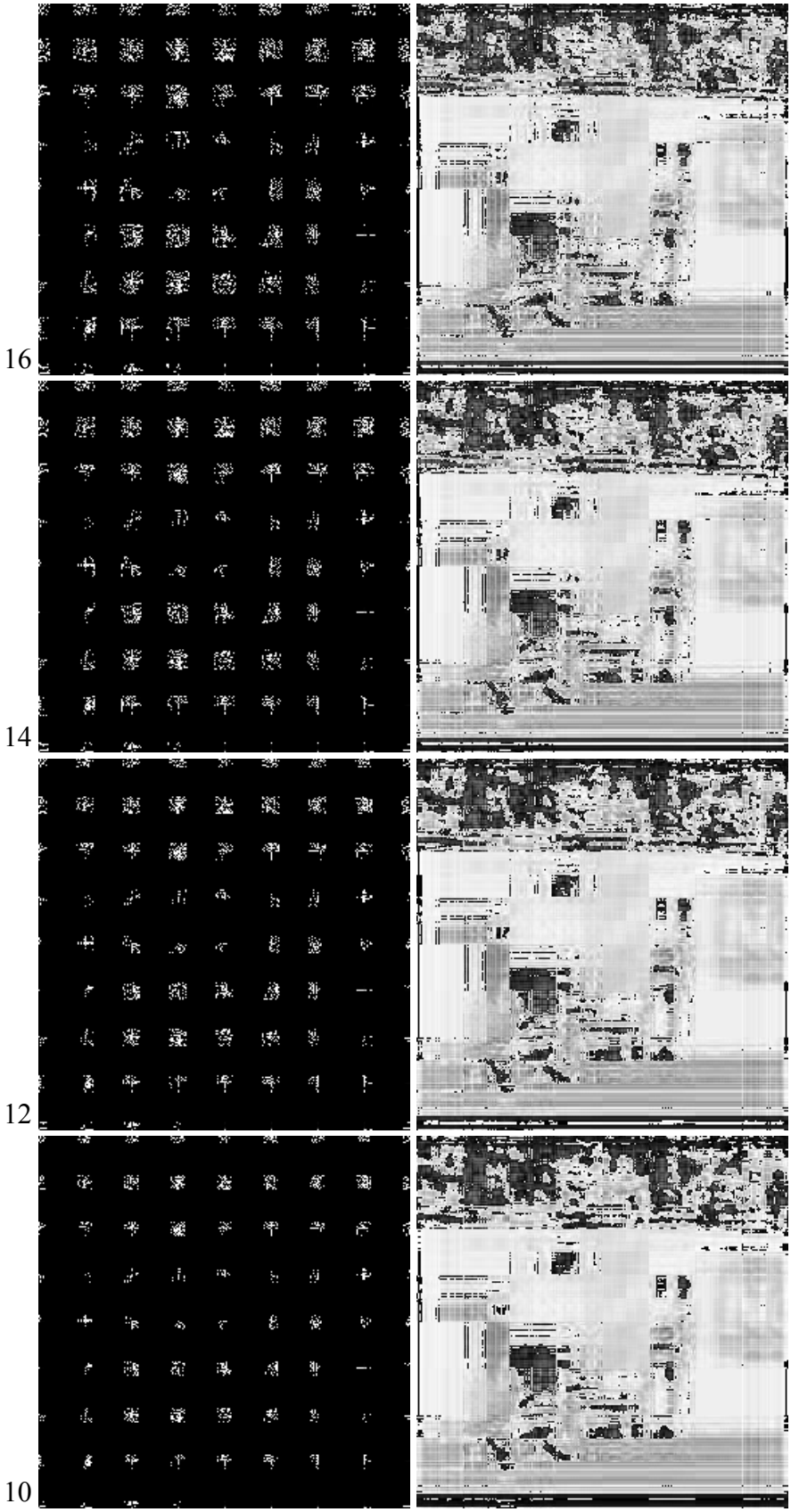


Tabla A.5 Pruebas con regiones de 32x32 píxeles (3/4)

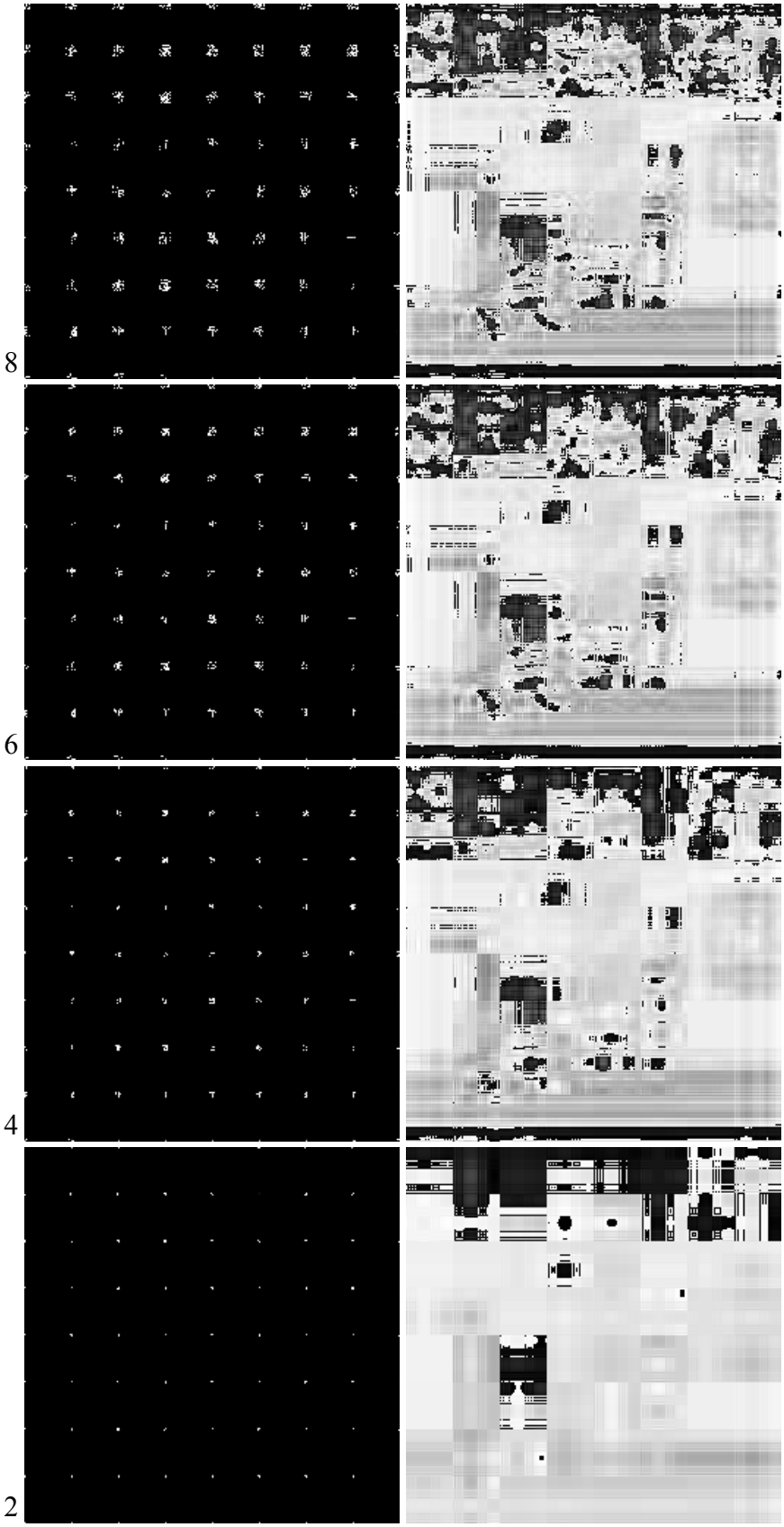


Tabla A.6 Pruebas con regiones de 32x32 píxeles (4/4)

A.5.3 Análisis de los resultados

En ambos casos, en la recuperación de la imagen original hay cierta pérdida de información causada por el redondeo y la adaptación del resultado a 8 bits, aunque en el caso de 16 no se vea a simple vista.

También comparando ambos casos se ve claramente lo que se ha comentado antes: al aumentar el tamaño de la muestra, los valores de las componentes del espectro son de mayor rango; en este caso, al tratar de ajustar a valores de 8 bits, tanto las componentes del espectro como el resultado de la destransformación de la imagen se ve que las regiones de 32x32 píxeles están mucho más afectadas que las de 16x16.

Por último, se puede ver hasta qué tamaño de banda pasante se está dispuesto a tolerar para el correcto funcionamiento de la aplicación. Si en el caso de las regiones de 16x16 se tolerase un tamaño 4, querría decir que se habrían tomado los 4 píxeles de las esquinas de la imagen tal como se muestra en la Fig. A.5, según el tipo de filtro que se ha utilizado y teniendo en cuenta tal como se ha comentado que las componentes de baja frecuencia se encuentran en las esquinas de la matriz que representa la imagen. De esta forma, se habría pasado de necesitar 256 bytes (1 byte por pixel) para representar la región a necesitar solamente 16. Tomando todas las regiones de la imagen, se pasaría de 65536 bytes (256^2) a sólo 4096 (16^3), es decir, una cantidad de datos 16 veces menor que la original.

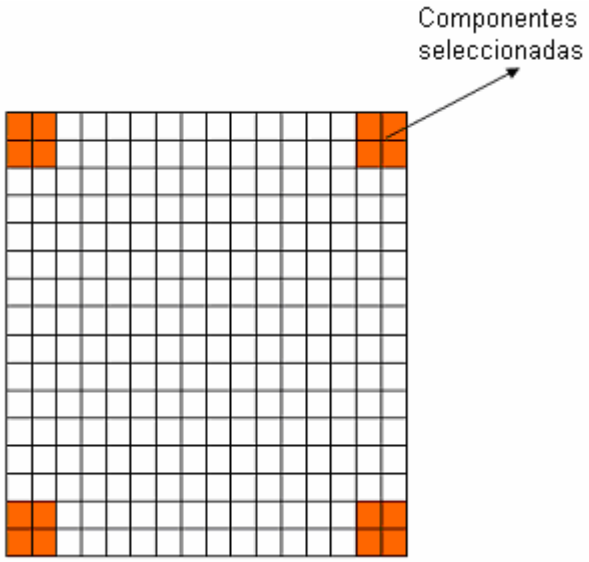


Fig. A.5 Componentes del espectro 16x16 que se conservarán tras aplicarse el filtro.

Por otro lado, para implementar por hardware la transformación de Hartley, tal y como está propuesta en [Dimitrov, 1999], para el caso de vectores de 32 elementos existirían los problemas de necesitar más elementos de proceso y el utilizar coeficientes polinomiales que ya **no serían potencias de dos**, con lo que los elementos de proceso serían mucho más complicados que los que se han implementado para el caso de 16 elementos, tal y como están presentados en el apartado 3.4.2.

Aunque la transformación no es perfecta se acepta el resultado como satisfactorio porque cumple la función esperada: cada espectro nos muestra un resumen de su región respectiva con una cantidad mínima de información a partir de la cual, si luego interesara, se podría reconstruir algo parecido a la imagen original y, en la medida de lo posible, hacer un análisis

más preciso sobre la geometría de la región en cuestión con tal de determinar de una forma más o menos precisa la posición del objeto a seguir.

El hecho de haber elegido regiones pequeñas en lugar de hacer la transformada de toda la imagen completa se debe al hecho de que, a parte de la ventaja que supone el saber dónde se ha podido producir la diferencia entre dos imágenes de forma general, cuanto menor sea la dimensión del vector a transformar las componentes de la transformada tendrán un orden de magnitud menor con lo cual su manipulación será más sencilla.

Por otro lado el haber elegido regiones de 16x16 y no de 8x8 o 4x4 (por ejemplo) se debe a lo siguiente: utilizando un sistema sistólico, la operación de la Transformada Discreta de Hartley se podría asemejar a un algoritmo de orden N, siendo N el tamaño del vector. Por otro lado el algoritmo de la Transformada Rápida de Hartley (FHT) es de orden $N\log_2(N)$. La velocidad del sistema sistólico frente a la FHT vendría determinada por la relación $\log_2(N)$

Cuanto mayor fuera N, mayor será el valor del logaritmo en base 2 de N. Elegir N=16 es un punto de compromiso entre velocidad (4 veces mayor que FHT) y orden de magnitud del espectro. Tal y como se ha comprobado en las pruebas con regiones de 16 y de 32 píxeles de lado, se ve claramente que en el caso de 32 la saturación a la que se somete el espectro y la imagen recuperada es más traumática que en el caso de 16.

Seguramente en el caso de 8 no habría ningún problema, pero solamente tendríamos una velocidad 3 veces mayor que la FHT. Por esto, la solución más adecuada es la de utilizar regiones de 16x16 píxeles.

B Seguimiento de Objetos en imágenes digitales.

B.1 Introducción

Este anexo pretende ser una pequeña introducción del modo en como se ha enfocado el problema de seguimiento de objetos a partir de una secuencia de imágenes en este proyecto. Es evidente que dadas las particularidades con la que se ha pretendido tratar la información, una aplicación directa de los diferentes algoritmos desarrollados en este sentido no sería de demasiada utilidad. Es por ello que se ha intentado analizar de forma conceptual las diferentes metodologías existentes en lo que a seguimiento visual de objetos se refiere, para con ello intentar desarrollar una metodología propia.

Existe un gran número de áreas y tareas dentro del campo de la industria y la ciencia, en las cuales el seguimiento de un objeto que se encuentra capturado en una secuencia de imágenes es un problema crucial a resolver. Algunos ejemplos de este tipo de sistemas son los sistemas de búsqueda y vigilancia, de análisis sanguíneo, de navegación autónoma, etc.

Los problemas o asuntos a tener en cuenta en cualquier sistema de seguimiento se pueden agrupar en tres aspectos generales. Los aspectos **relacionados con la visión**, los relacionados con el **posicionamiento y estabilización**, y los aspectos referentes al **dispositivo de captura** empleado. Tales aspectos no son de ningún modo independientes entre si, con lo que en el desarrollo del sistema global deberán tenerse en cuenta en mayor o menor medida cada uno de ellos.

En lo que se refiere a los aspectos de visión, cabe centrarse en el análisis y procesamiento de la imagen para poder realizar el seguimiento al objeto deseado dentro de la secuencia de imágenes. Se debe considerar en qué condiciones se captura la imagen, pues cualquier alteración de éstas podría modificar el modo en qué se capturan los datos.

Por lo que se respecta al posicionamiento y estabilización, se puede realizar mediante sistemas de control que prevén el estado futuro de los sensores de captura en relación a como evoluciona el objeto a seguir.

Por último, el dispositivo de captura empleado debe ser capaz de determinar el alcance del sistema, aspecto a tener en cuenta en la capacidad de resolución de la imagen capturada y la velocidad de captura.

Dadas las características del sistema, se ha creído conveniente no analizar de forma exhaustiva los aspectos que hacen referencia al posicionamiento y dispositivo de captura, con el objetivo de centrar el análisis del procesamiento de los datos capturados por el sistema y sus diferentes metodologías.

Con este objetivo, se ha pretendido ejemplificar el funcionamiento del sistema implementado mediante una aplicación que trate de forma aproximada el seguimiento de un objeto determinado a través de una secuencia de imágenes correspondientes al movimiento de éste a través del tiempo.

B.2 Concepto

La entrada de un sistema de este tipo es una secuencia de imágenes consecutivas tomadas del mundo real.

El movimiento aparente captado en la secuencia de imágenes es lo que se denomina **"Flujo Ótico",** el cual describe la dirección y velocidad del movimiento de las características y/o puntos de la imagen. En este sentido cabe mencionar la definición que se da en [Peregrina, 2002], donde se define el flujo óptico como *"registro de velocidades en el plano de la imagen debidas al movimiento del observador, al movimiento de objetos en la escena, o al movimiento aparente dados por los cambios de intensidad entre cada imagen que generan falsos movimientos de objeto u observador"*.

Según esta definición, se puede clasificar un sistema de captura según sea el dispositivo de captura empleado (en este caso cámara) o los objetos los que se muevan. En este sentido podrían distinguirse cuatro tipos de situaciones que cabría tener en cuenta antes de abordar cualquier estudio en este tipo de sistemas. Se pueden distinguir las siguientes situaciones:

- **SCSO** : Cámara estacionaria, objetos estacionarios.
- **SCMO** : Cámara estacionaria, objetos en movimiento.
- **MCSO** : Cámara en movimiento, objetos estacionarios.
- **MCMO** : Cámara en movimiento, objetos en movimiento.

En este caso, a modo de facilitar en cierta medida los procesos de cálculo, se ha analizado el caso **SCMO**, basado en el movimiento del objeto y la posición estática del dispositivo de captura y que tan importantes aplicaciones tiene en el campo del análisis de imágenes dinámicas.

El proceso de seguimiento puede implicar a cualquier objeto móvil presente en la escena que se esté analizando, sin especificar de qué objeto se trata, o el seguimiento de un objeto determinado del cual se tiene un modelo predefinido.

Tal como se ha comentado en el párrafo anterior, uno de los aspectos más importantes a tener en cuenta en un sistema de seguimiento es el relacionado con la visión, en lo que se refiere al procesamiento y tratamiento de la imagen recibida (aspecto sobre el cual versa la mayor parte del desarrollo de este proyecto).

En este sentido y con el objetivo de llegar a determinar el *"flujo óptico"* (necesario para el seguimiento de objetos) de una secuencia de imágenes capturadas, se citan las metodologías más comúnmente utilizadas para ello.

- Métodos basados en características.
- Métodos basados en gradiente.
- Métodos basados en correlación.

B.3 Descripción de las metodologías

A continuación se pasará a describir las características principales de cada una de estas metodologías para el desarrollo e implementación de los sistemas de seguimiento.

B.3.1 Métodos basados en características

En el análisis de imágenes se considera una característica como *un punto o conjunto de ellos que satisface una serie de relaciones (similitud, consistencia, etc.) dadas de antemano*. Así pues, esta metodología consiste en encontrar en cada una de las imágenes de la secuencia las características estipuladas inicialmente (por ejemplo: lados, esquinas o alguna otra estructura reconocible) del objeto u objetos que se quieran analizar. Mediante este método podrá encontrarse el objeto deseado mediante las características definidas en cualquier imagen de la secuencia.

Normalmente este tipo de métodos constan de dos etapas fundamentales, la primera consistente en la detección de la o las características de los objetos en la imagen. La segunda etapa consiste en seguir la misma característica en todo el grupo de imágenes con el objetivo de realizar el proceso de seguimiento a lo largo del tiempo.

La detección de características en la imagen, debe poder realizarse de forma rápida y concisa. Por lo que respecta a la correspondencia entre las características detectadas en las diferentes imágenes, cobran gran importancia pues es a partir de ellas que podrá calcularse el desplazamiento temporal del objeto.

La ventaja que presenta este tipo de métodos es que al tener que trabajar con una cantidad reducida de información (tan sólo aquellas características predefinidas), el coste en cuanto a tiempo de cálculo se reduce notablemente.

B.3.2 Métodos basados en gradiente

Los métodos basados en gradiente trabajan con la relación entre el espacio y las transiciones de valores en el tiempo. Se utilizan diferentes técnicas de estimación diferencial, aplicadas al cálculo del movimiento de cada uno de los píxeles de la imagen. La estimación del movimiento a lo largo de la secuencia, está basada en la observación de los cambios de intensidad de brillo con respecto al tiempo, registrando este valor como velocidades de cambio de las características de brillo de la imagen.

Este tipo de metodología de seguimiento presenta un tipo de problema fundamental en cuanto a la estimación del flujo óptico. Las discontinuidades de brillo ocasionadas por la presencia de ruido o posibles interferencias en la captura de los objetos en movimiento provocan presencia de discontinuidades en los campos de flujo ópticos.

Analizando las razones indicadas, puede considerarse a esta metodología bastante costosa en cuanto a tiempo de cálculo se refiere, por lo que podría considerarse como inviable para aquellas aplicaciones de seguimiento en las que los algoritmos no sean implementados a través de hardware. En el caso presente, como este procedimiento se ha decidido desarrollar vía software, se ha descartado esta alternativa.

B.3.3 Métodos basados en correlación

Estos tipos de métodos consisten en calcular la relación de igualdad que existe entre dos tipos de datos, en este caso unos datos que describen de forma más o menos precisa una imagen digital. La correlación aplicada al mundo de seguimiento de objetos en secuencias de imágenes consiste en comparar porciones o regiones representativas de una imagen en un instante determinado, con estas mismas regiones en instantes posteriores o anteriores, para de esta forma hallar el grado de similitud entre ellos. Cabe destacar que, pese al alto grado de

precisión que presenta este tipo de tecnología, puede llegar a resultar computacionalmente muy costosa.

B.4 Proceso de Tracking

Sin duda, uno de los aspectos más importantes y sobre el que se ha prestado el máximo interés en este proyecto de desarrollo de un sistema de seguimiento de objetos, es el procesado y análisis de la imagen. Este proceso es de vital importancia ya que a partir de él se generarán y derivarán el resto de procesos.

Se entiende como *Tracking* al seguimiento de un objeto por medio de visión. Restando importancia a la cuestión de la captura y transmisión de la imagen, se va a analizar en este apartado un método eficiente en cuanto a tiempo y resultados para el análisis de la imagen.

El tiempo empleado en el procesado de la imagen se relaciona directamente con la capacidad del procesador usado, el algoritmo de búsqueda usado y el área de tamaño de la imagen tratada. Dado que sobre el microprocesador no se puede actuar, se va a intentar, por un lado, minimizar la información a tratar de la imagen y, por otro lado analizar el algoritmo más conveniente para el tratamiento de estos datos.

Dentro de los diferentes tipos de seguimiento que se muestran en [Peregrina, 2002], se van a analizar dos que podrían considerarse como los más adecuados para implementar en este proyecto.

B.4.1 Seguimiento relativo a diferencias

La idea principal que caracteriza a este tipo de seguimiento es la de calcular la diferencia existente entre los píxeles de una imagen en un instante determinado y las del instante anterior. En este caso el valor o la presencia de movimiento será fácil de calcular, dado que si las imágenes son iguales la diferencia será nula y en caso contrario la diferencia existente entre una imagen y otra delatará la existencia de movimiento en un entrono determinado.

Para la correcta implementación de este tipo de procedimiento, es conveniente considerar que el resto de la imagen (a parte del objeto del cual se quiere calcular el movimiento) permanece inmóvil. Dado que esta situación se da en el caso que se plantea, puede llegar a considerarse como válida.

B.4.2 Seguimiento relativo a correlación

Los procesos basados en correlación buscan la similitud de una imagen con otra imagen, en este caso otra imagen de la secuencia analizada. Mediante este método se analiza el movimiento del objeto en la medida que se mueva dentro de la imagen estimando el grado de correlación existente entre dos imágenes consecutivas en la secuencia.

La correlación de valor cero denota que no existe ningún tipo de similitud entre las dos imágenes, mientras que una correlación unitaria describe una similitud perfecta entre imágenes. Así pues se puede definir como correlación entre dos conjuntos de valores X e Y la expresión indicada en la ecuación B.1, donde x e y son dos vectores que representan una serie de valores genéricos cuyas medias aritméticas son $\bar{x}$ e $\bar{y}$.

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{B.1}$$

Este tipo de seguimientos presenta dos ventajas claras que lo hacen bastante óptimo para implementar algoritmos de seguimiento. Por un lado, se comportan de forma bastante eficaz frente a cambios debidos al ruido y, por otro lado no presenta problemas aunque haya una expansión o compresión del área de imagen capturada.

Un ejemplo de este tipo de algoritmos se presenta en [Sawasaki, 2002]**,** donde se muestra una posible aplicación de este tipo de metodologías de correlación. Dicho artículo describe un algoritmo basado en el estudio de la correlación existente entre una imagen plantilla o referencia y todo el resto de imágenes de la misma dimensión que pueden llegar a existir en la región en la que se pretende analizar si ha existido o no algún tipo de movimiento.

El objetivo del algoritmo es hallar la posición de un objeto que posea el mayor grado de correlación posible con la plantilla de la que se dispone. Concretamente el sistema que se propone consiste en hallar la mínima diferencia absoluta (MDA), mediante la cual la diferencia absoluta entre píxeles para una imagen plantilla de dimensiones NxN se calcularía según la ecuación B.2, en la cual $T_{k,l}$ representa el valor del píxel en la imagen plantilla y $S_{k+i,l+j}$ el valor del píxel en el área de la imagen de la secuencia que se analice en cada instante de tiempo.

$$D_{i,j} = \sum_{k=0}^{n}\sum_{l=0}^{n}\left|T_{k,l} - S_{k+i,l+j}\right| \tag{B.2}$$

El parámetro $D_{i,j}$ es conocido como *coeficiente de distorsión* y representa el mayor o menor grado de correlación entre la plantilla y la imagen que se analice en ese preciso instante. Cuanto menor sea este valor existirá un mayor grado de correlación entre ambas imágenes. Una vez analizado este valor de correlación entre todos los calculados, se puede conocer la nueva posición del objeto analizado mediante las coordenadas *i,j* del punto donde ha existido la máxima correlación que corresponderá al que tenga un coeficiente de distorsión menor. El proceso que se seguiría este algoritmo se muestra en la Fig. B.1.

Uno de los principales problemas de esta metodología es el excesivo tiempo de computación que se necesita si para ello se quieren analizar todas las posibles correlaciones que pudieran generarse entre la imagen patrón y la imagen que se use como referencia.

Con el objetivo de resolver este tipo de problemas, referente a la búsqueda de correspondencia entre bloques de imágenes, se han desarrollado a lo largo de los últimos años gran número de algoritmos con el objetivo de optimizar este tipo de procesos. Sin embargo, y pese a las mejoras en cuanto a computación puedan obtenerse, si se requiere un alto grado de precisión este tipo de metodología no presentará nunca las mejores prestaciones.

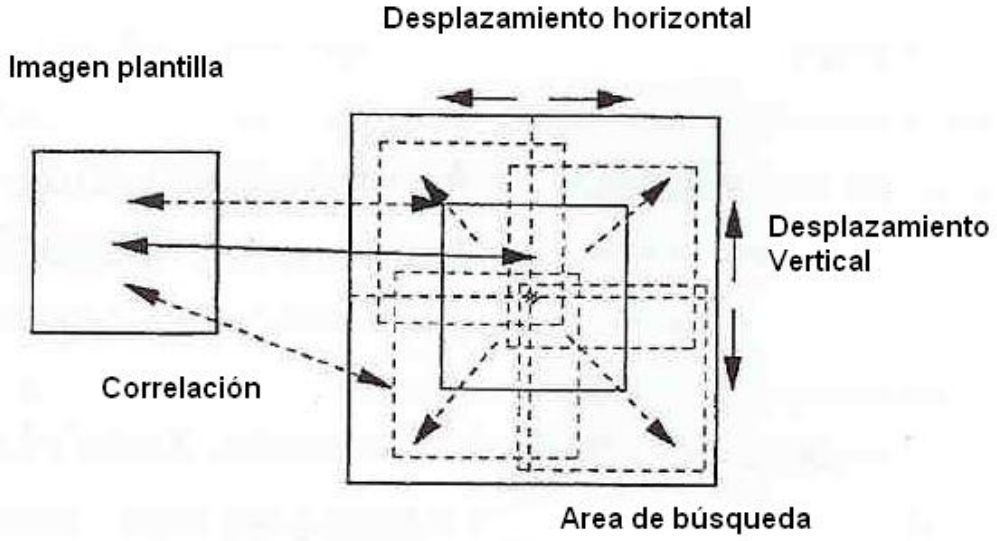


Fig. B.1 Esquema del proceso de correlación.

B.5 Conclusiones

Una vez analizado de forma más o menos general el tipo de metodología que podría seguirse en el proceso de seguimiento, este proyecto se ha decantado por el uso de un algoritmo basado en el seguimiento por diferencias, ya que tanto la situación de la captura de la imagen (fondo de imagen y dispositivo de captura fijos), como el tratamiento de la misma (tratamiento frecuencial de los datos de la imagen capturados) llevan a pensar en la óptima aplicación de los mismos. Es por ello que se ha desarrollado un algoritmo basado en el cálculo de las diferencias de los tipos de datos, que de forma comprimida representan la información que describe la imagen que realmente ha sido capturada inicialmente. Mediante el tratamiento de estos datos se puede llegara predecir de forma mas o menos precisa cuál ha sido el movimiento del objeto en la secuencia de imágenes.

C Software de Aplicación para el seguimiento de la Imagen

C.1 Introducción

Tal como se ha comentado en la memoria se ha decidido desarrollar una pequeña aplicación desde la que el usuario pueda controlar fácilmente el sistema desarrollado.

El objetivo de la misma ha sido el de desarrollar una interfaz a través de la cual el usuario del sistema pueda interactuar y definir cómo y cuando van a tener lugar las diferentes operaciones que desde la aplicación y a través del sistema desarrollado, permitirán realizar el proceso de seguimiento del objeto en movimiento.

La aplicación debe permitir al usuario interaccionar por tanto con los diferentes procesos que forman el proceso de seguimiento del objeto. De este modo, el usuario debería ser capaz de poder configurar el modo en que va a realizarse la captura de información (definición de las características imagen, filtro a emplear, calibración, etc.) así como visualizar el proceso de seguimiento a través de una interfaz gráfica.

C.2 Estructura

La aplicación ha sido desarrollada en Visual Basic. Se ha estructurado en dos partes fundamentales: una primera parte desarrollada para conformar las acciones de configuración, y una segunda parte en la que se visualiza y controla el propio proceso de seguimiento. Además, se ha incluido unas interfaces de consulta de información del sistema respecto a los diferentes sistemas que conforman la estructura global (Fig. C.1).

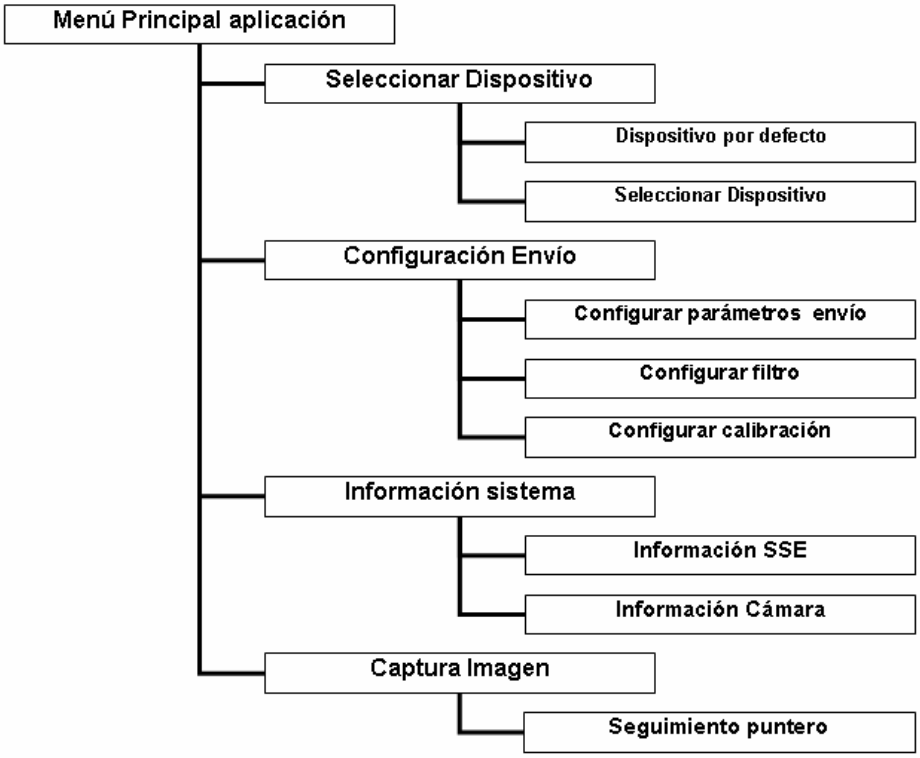


Fig. C.1 Diagrama del menú principal de la aplicación

Cada uno de los submenús que dependen del menú principal corresponde a una pantalla mediante la cual el usuario puede interaccionar para llevar a cabo las diferentes funciones. Desde el sistema de menús, siempre visible durante la ejecución de la aplicación, el usuario puede acceder de forma directa a cualquiera de los submenús que le llevarán al formulario pertinente.

Existe una pantalla principal que sirve como presentación del software desarrollado y desde la cual se tiene acceso a la barra de menús que nos guiará a través de la aplicación. En esta primera pantalla (Fig. C.2) ya se puede visualizar como va a estar estructurado el software. Por un lado, se tiene la barra de comandos desde donde se accederán a los diferentes formularios, y por otro lado, la propia interfaz de la pantalla, donde estarán los diferentes controles para cada caso concreto.

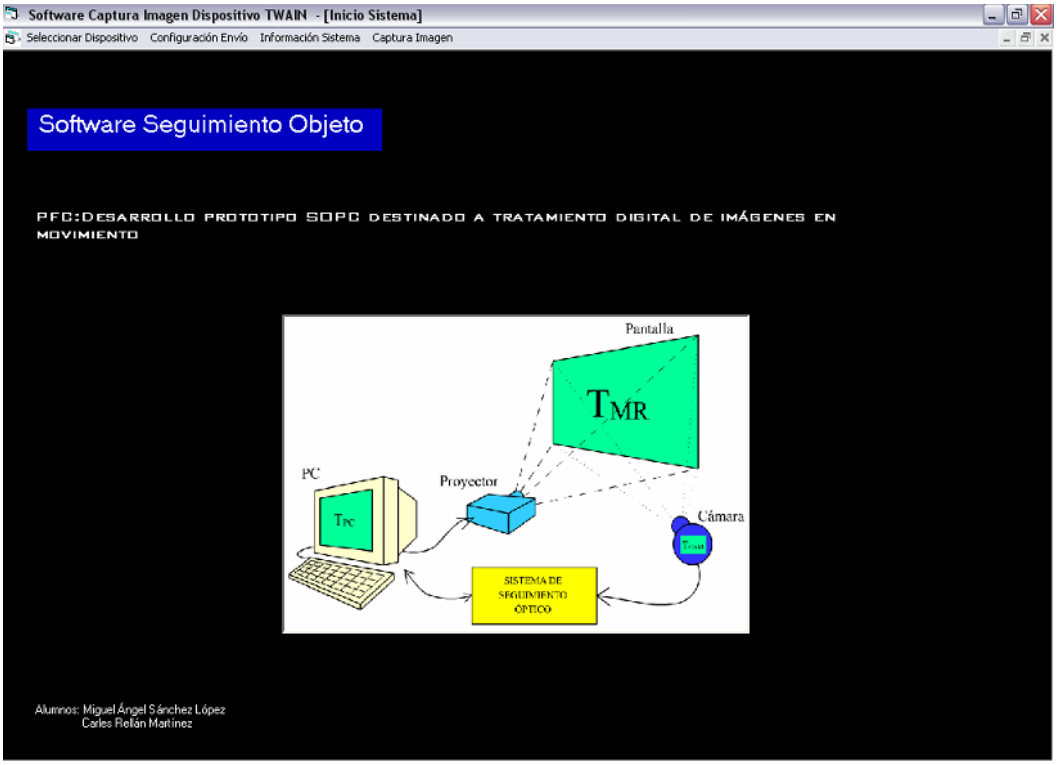


Fig. C.2 Pantalla inicial

C.3 Esquema de la aplicación

Para un correcto manejo del software deben seguirse unas directrices para que el proceso de seguimiento se desarrolle de forma óptima. Tal como se ha comentado, previamente al proceso de seguimiento se deben configurar las condiciones de envío de información. Por tanto, en el modo de funcionamiento de la aplicación, antes de ejecutar el propio seguimiento, se deberá realizar los pasos de configuración del envío y recepción de información.

Para establecer este orden en cuanto a acceso a los diferentes formularios se presenta el esquema representado en la Fig. C.3 que especifica el proceso a seguir en la configuración del envío en lo que a la navegación por los menús de la aplicación se refiere.

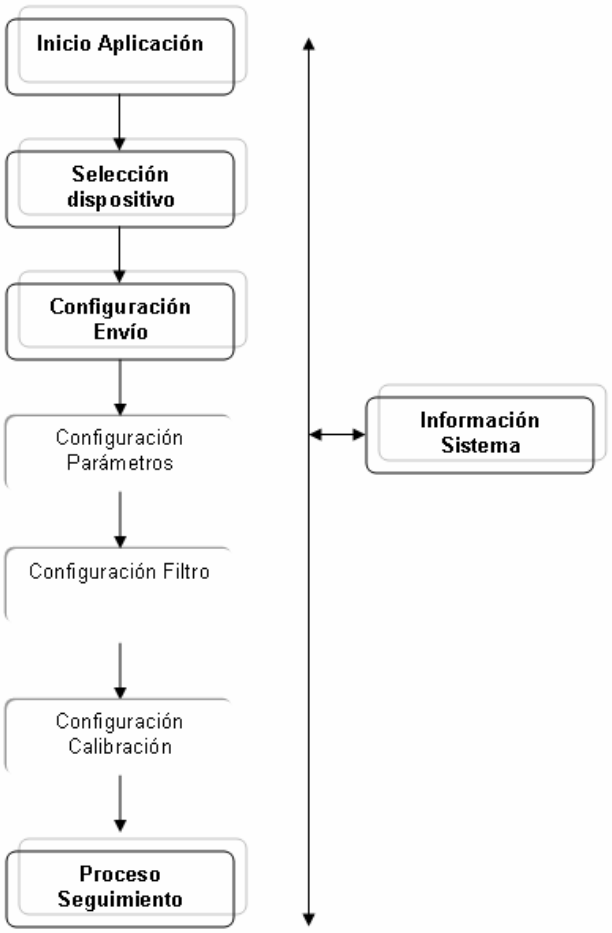


Fig. C.3 Esquema del proceso

Como se puede observar en la Fig. C.3, previamente al proceso de seguimiento deben configurarse tanto el dispositivo que va a ser utilizado para la captura de imágenes como los parámetros que configurarán el envío. Posteriormente ya puede ejecutarse el procedimiento para el seguimiento del objeto. El formulario que accede tanto a información del Subsistema Externo (SSE) como a información de la cámara sobre el formato de envío de imagen es accesible durante todo el desarrollo del proceso.

C.4 Formularios de configuración

Los formularios de configuración son aquellos con los que el usuario podrá definir los parámetros con los cuales quiere que sea tratada la información en el sistema, tanto a nivel de imagen, filtro, etc. Para ello se han desarrollado tres formularios que se detallarán a continuación.

C.4.1 Formulario de configuración de parámetros

El objetivo de este formulario es el de definir algunas de las características o propiedades principales que definen una imagen capturada mediante un dispositivo de adquisición. Entre ellas se han seleccionado el brillo, contraste, color y unidad de dimensión de la imagen.

Mediante el formulario el usuario podrá seleccionar estos valores para las imágenes que sean capturadas dentro de unos márgenes preestablecidos.

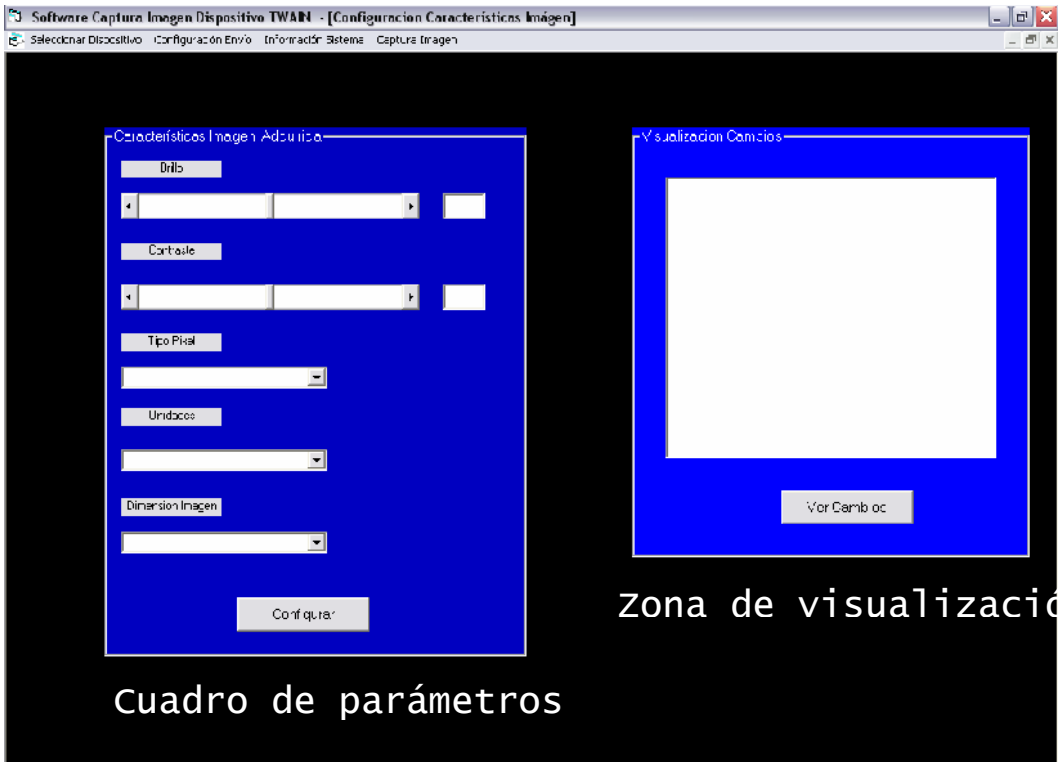


Fig. C.4 Formulario de configuración

Como se ve en la Fig. C.4, existen dos cuadros fundamentales que conforman el formulario. Por un lado está el cuadro de parámetros, donde es posible configurar algunas de las características que van a definir la imagen a capturar. En este sentido, se dispone de dos barras de desplazamiento para configurar el nivel de brillo y contraste que se desea. Por otro lado, existen tres cuadros de diálogo desde donde es posible configurar parámetros como el tipo de píxel, las unidades utilizadas y la dimensión de la imagen. Para este caso tan sólo se han configurado algunas de las posibilidades, mientras que las demás son una muestra más de futuras ampliaciones que pudieran realizarse dada la versatilidad del sistema.

C.4.2 Formulario de configuración del filtro

El objetivo de este formulario es el de configurar el filtro que va a ser enviado al SSE con el objetivo de seleccionar aquellas partes de la imagen realmente representativas en cuanto a información de las mismas.

Tal como se ha comentado en el apartado 4.8 (pág. 52 de la memoria), lo que se pretende es definir el filtro mediante la elección del número de píxeles de interés dentro de un cuadrante de la imagen de 8x8 píxeles; que corresponde a un cuarto de la imagen de 16x16 quees enviada al SSE para ser procesada. Dado que el filtro va a ser simétrico en la parte de la imagen de 16x16, puede ser definido éste en la submatriz de 8x8 píxeles.

La estructura del formulario es sencilla. El usuario dispone de una representación esquemática del mapa de bytes (8x8) en el que se visualizará la parte de los mismos que se seleccionan para formar parte del filtro. Para ello el usuario dispone de una *barra de desplazamiento* al

costado izquierdo del mapa, con la que podrá configurar el porcentaje de la imagen que va a ser seleccionada para su posterior envío al SSE. Éste valor es visualizado mediante un cuadro de texto en la parte inferior de la matriz, tal como se muestra en la Fig. C.5.

Una vez establecido este valor, pulsando el botón visualizar puede verse sobre la representación de la matriz de píxeles, cuales han sido realmente seleccionados, lo que muestra una representación más visual de cómo va a ser configurado el filtro.

Si el filtro mostrado es el deseado, éste será enviado al SSE una vez el usuario seleccione la opción de *Configurar Filtro*. En caso que el filtro seleccionado no sea de interés, se pueden generar cuantos filtros se desee siguiendo el procedimiento descrito previa selección de la opción *Borrar*.

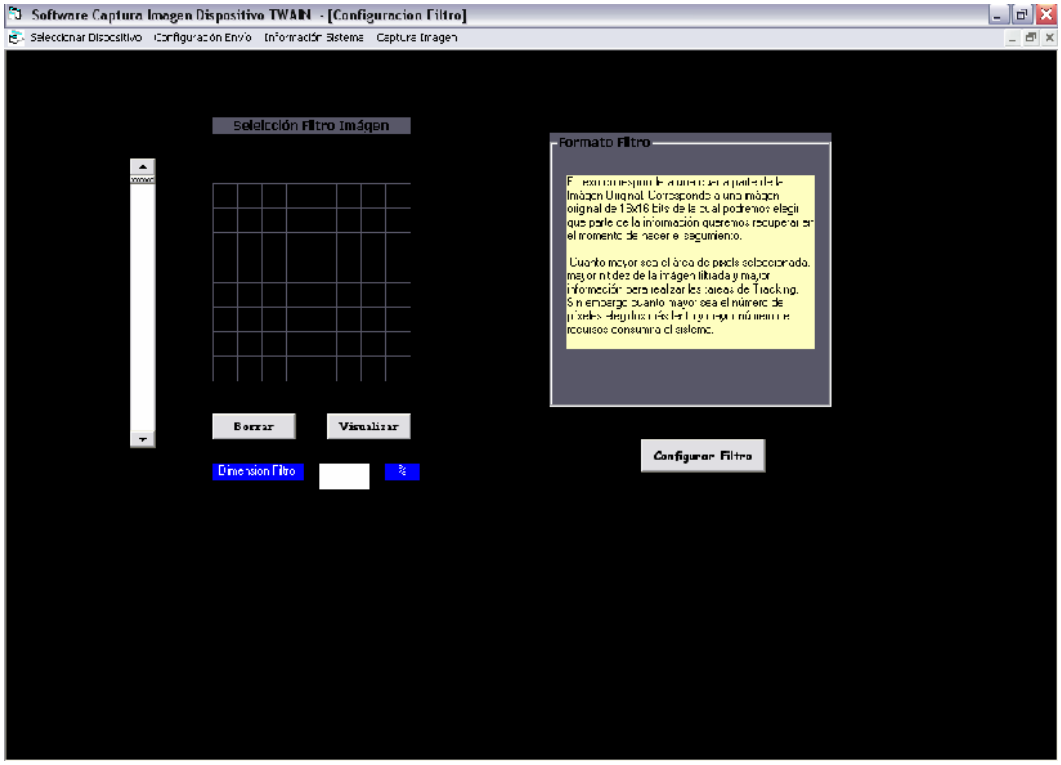


Fig. C.5 Formulario de configuración del filtro.

C.4.3 Formulario de calibración

Con este formulario, representado en la Fig. C.7, lo que se pretende es calibrar a través de la cámara la parte de la imagen que interesa y a su vez, dentro de ésta, restringir la zona de interés a lo que es la pantalla del PC que va a ser proyectada en T_{MR} . El formulario consta de un control donde se visualiza la imagen capturada (Picture Box).

Una vez capturada la imagen, el usuario pude determinar las coordenadas de la pantalla en el control, las cuales se visualizarán en controles del formulario. Una vez se tienen las coordenadas y el usuario considera que éstas son las correctas, se procede a la validación de las mismas, con el objetivo de ubicar éstas en la imagen total capturada para de este modo reducir el área de análisis de la imagen y poder calcular el factor de escalado para la posterior visualización del seguimiento. El esquema del proceso a seguir se muestra en la Fig. C.6.

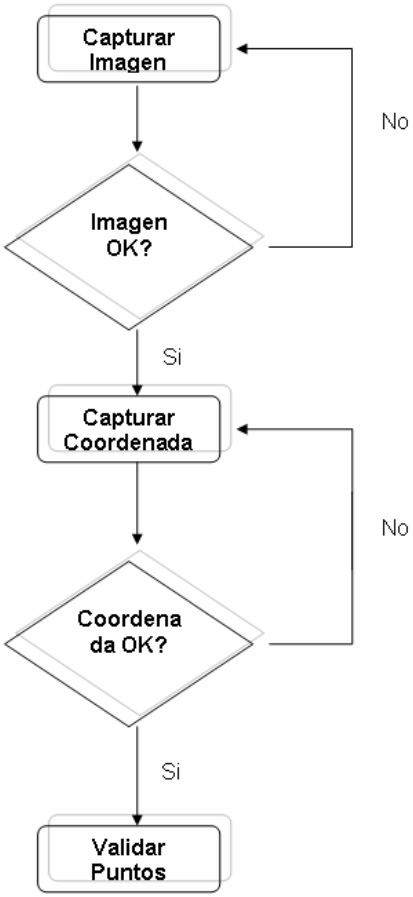


Fig. C.6 Proceso de calibración de la imagen.

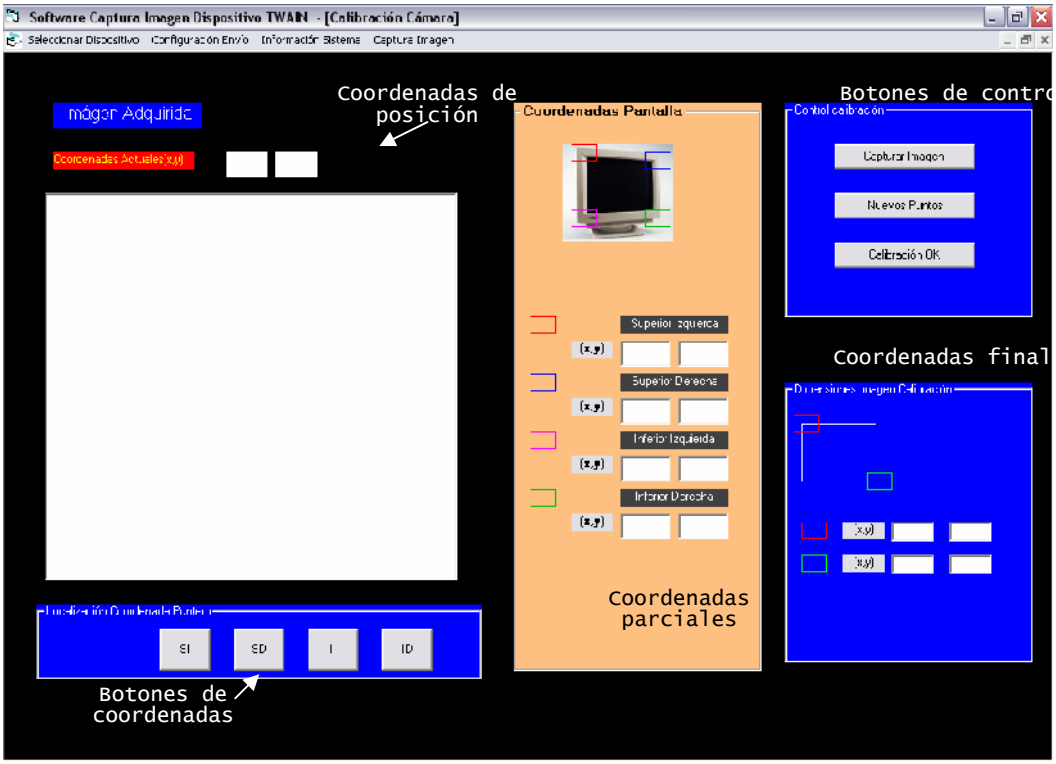


Fig. C.7 Estructura del formulario de calibración de la imagen

En el formulario (Fig. C.7) se pueden distinguir varias partes. Por un lado, se tienen los botones de control, que son los que ejecutarán las acciones pertinentes para llevar a cabo varias fases del procedimiento, como son la captura de la imagen y la validación de los puntos.

Por otro lado, para visualizar la imagen capturada desde la cámara, se utiliza un control Picture Box, componente éste estándar del entorno de programación Visual Basic. Una vez se visualiza la imagen en el control, en las coordenadas de posición se visualizarán las coordenadas de cualquier punto sobre el que pulsemos el ratón en coordenadas del control, tomando como origen de coordenadas la esquina superior izquierda del mismo.

El usuario podrá conocer las coordenadas de cualquiera de los puntos que pulse con el ratón. En el momento en que crea que alguno de los marcados, pueda coincidir con uno de los puntos cuyas coordenadas puedan identificar las esquinas del monitor del PC en la imagen, debe proceder a pulsar los botones de coordenadas. Cada uno de ellos identificará parcialmente a las coordenadas del monitor del PC dentro del Picture Box y con referencia a éste mismo objeto según se ha comentado antes. La nomenclatura de los botones de coordenadas sigue la siguiente relación:

- SI : Coordenadas esquinas superior izquierda.
- SD : Coordenadas esquinas superior derecha.
- II : Coordenadas esquinas inferior izquierda.
- ID : Coordenadas esquinas inferior derecha.

Una vez se han capturado las coordenadas de la pantalla, el usuario podrá decidir si está de acuerdo o no con las mismas. Si lo está, validará los puntos y de este modo se generará un registro con el área de interés y el factor de escala. En caso contrario, mediante los Botones de Control, tiene la posibilidad de recalcular los puntos cuantas veces quiera.

C.4.4 Formulario de Seguimiento

El formulario en el que propiamente se va a desarrollar el proceso de seguimiento del objeto característico se presenta en la Fig. C.8. Consta de un PictureBox sobre el que se visualizará la trayectoria del movimiento seguido por el objeto de interés en el mundo real. Cuenta con dos botones de comando, uno para el inicio del proceso de captura y otro para reiniciar el proceso de descripción de trayectorias.

Tal como se ha comentado en el apartado 5.3.3, el proceso de seguimiento se basa en un autómata al cual ahora habrá que añadir un nuevo proceso, que no es otro que el de detectar la presencia del objeto que actúa como puntero en el mundo real en la zona de influencia donde se tiene el objeto de interés, que en el formulario presentado en la Fig. C.8 se presenta en color rojo.

Existen por tanto en lo que respecta al proceso de seguimiento dos partes esenciales. Por un lado, detectar que se produce un cambio en el entorno de la ubicación del objeto a mover y, por otro lado, una vez sucedido esto, detectar cambios de posición del objeto puntero que serán las nuevas posiciones donde deberá desplazarse el objeto de atención.

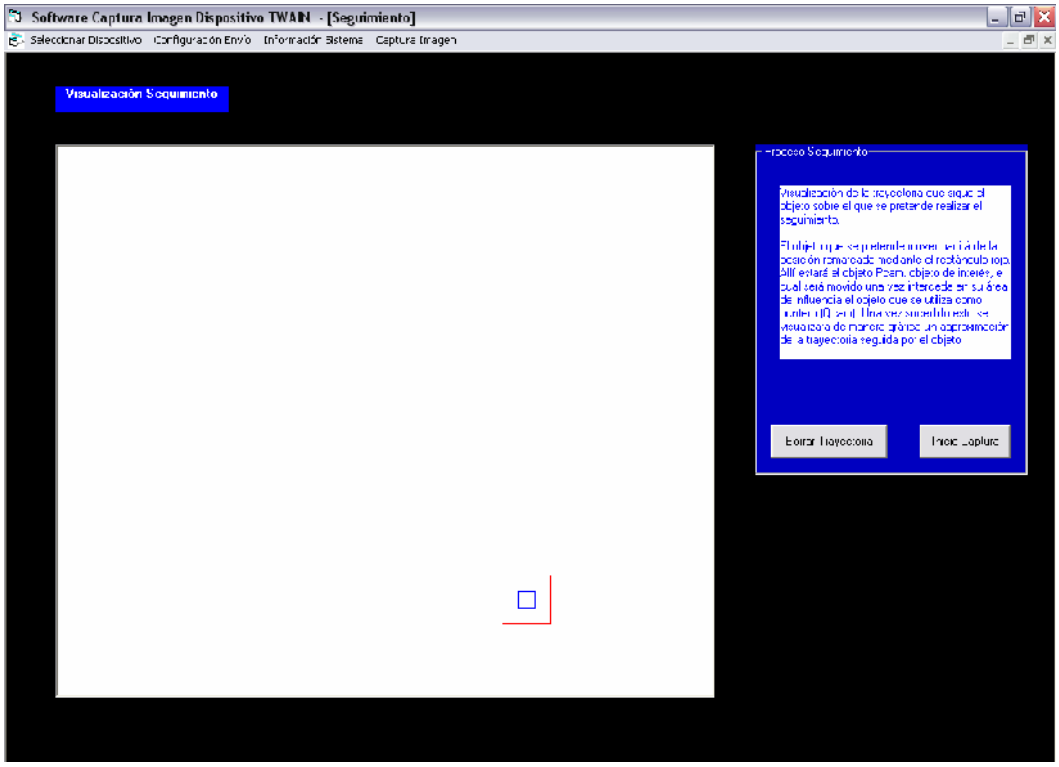


Fig. C.8 Formulario de seguimiento

Un aspecto a tener en cuenta será la posición y las coordenadas de referencia de los diferentes objetos o controles con los que se está trabajando. En este sentido ya se sabe gracias al proceso de calibración las coordenadas de la pantalla del PC dentro de una imagen capturada de dimensiones MxN píxeles.

Estas coordenadas servirán ahora para ubicar el control PictureBox que se utiliza para visualizar las trayectorias seguidas por el objeto. Para ello se va a analizar de forma precisa cómo Visual Basic 6.0 hace referencia a la posición y tamaño de los formularios y los demás controles.

En un formulario y en los controles PictureBox las propiedades asociadas a la escala y dimensiones, agrupadas de cuatro en cuatro, son las que se presentan a continuación: (top, left, height, width) y (scaleTop, scaleLeft, scaleHeight, scaleWidth) (Fig. C.9). Las primeras posicionan al Picture Box dentro de las coordenadas con referncia al formulario, y las segundas posicionan cualquier objeto presente en el control a partir de las coordenadas de su esquina superiro izquierda.

En ella podemos ver que podemos conocer la posición de cualquier punto dentro de un control PictureBox si conocemos sus dimensiones y las distancias que separan a su coordenada superior izquierda de los márgenes del formulario. Conociendo estos datos y aproximando que las coordenadas obtenidas en la calibración pueden asociarse a los límites del formulario en el mundo real, se puede de forma sencilla interpretar movimientos sucedidos en el MR a través del control PictureBox presente en el formulario de seguimiento.

Un esquema de la posición y referencia de estos parámetros se muestran en la Fig. C.9.

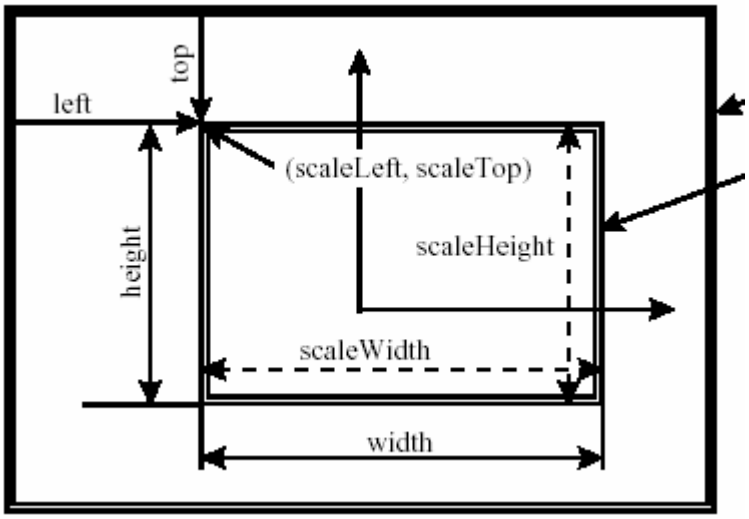


Fig. C.9 Parámetros de una imagen capturada.

El objetivo será interpretar el desplazamiento que el objeto ha padecido en el mundo real, analizando la imagen que de éste se ha tomado. Dado que la región de interés por calibración ya se ha capturado y se saben sus dimensiones, tan sólo hará falta escalar los desplazamientos en ese espacio con las dimensiones que el control Picture Box tenga en el formulario de seguimiento.

De este modo, como se ha comentado, el primer paso es detectar la presencia del puntero en la zona de influencia del objeto que se quiere. Una vez se ha detectado su presencia, se procede a activar el proceso de seguimiento, calculando las variaciones de posición del puntero referencia y desplazando el objeto de interés a esa posición (Fig. C.10).

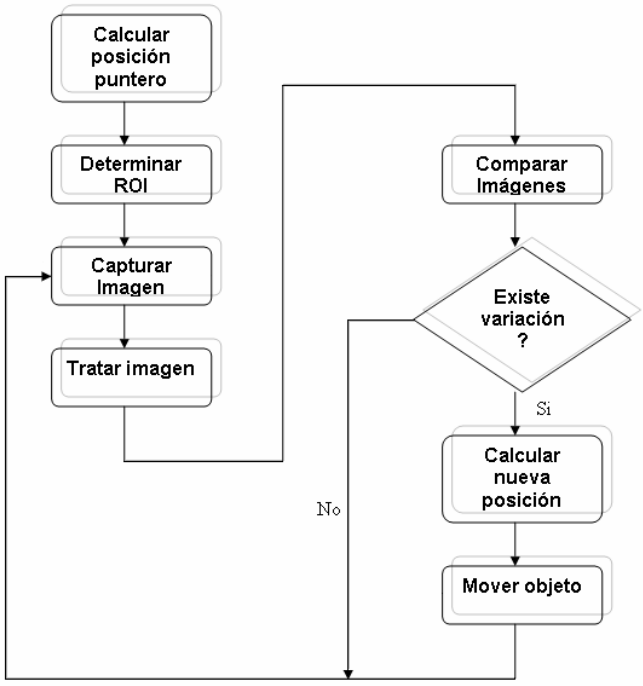


Fig. C.10 Esquema del algoritmo de seguimiento

D Introducción a los SOPC

D.1 Introducción

El hecho de haber elegido la tecnología basada en SOPC (Sistema en un Chip Programable) para la implementación del sistema se debe a razones que se intentarán analizar a través de este anexo.

Lo que se mostrará en este anexo es una comparación de las características de los diferentes dispositivos digitales, analizando las ventajas de cada uno de ellos, sobretodo haciendo hincapié en su optimización para aplicaciones que se van a desarrollar a gran escala, como pueda ser dispositivos para aplicaciones industriales o comerciales.

D.2 Origen de los SOPC

El orígen de los dispositivos SOPC se fundamenta en los llamados dispositivos SoC (Sistema en un Chip). Este tipo de dispositivos pueden definirse como circuitos integrados de mayor o menor complejidad, los cuales poseen al menos una unidad de proceso (CPU) así como otros periféricos como memorias, puertos etc. A medida que la complejidad de este tipo de dispositivos fue incrementándose y que las aplicaciones de los mismos también aumentaban, surgió la necesidad de convertir estas estructuras fijas, en circuitos programables, configurables por el usario según las necesidades del mismo.

Así pues combinando la estructura de los SoC e introduciendo la programabilidad de los dispositivos lógicos integrados en el mismo chip, surgieron los SOPC (Sistema en un Chip Programable). Tales dispositivos pueden definirse como circuitos programables de complejidad muy elevada que, de igual modo que los SoC, permiten integrar en el mismo chip una o varias CPUs y sus dispositvos periféricos asociados.

Existen dos tipos fundamentales de CPUs que pueden llegar a integrarse en estos dispositivos. Por un lado, están las CPUs definidas mediante software, las cuales pueden ser sintetizables en la lógica programable del dispositivo. Por otro lado, están las CPUs definidas mediante hardware, las cuales actúan como bloques especiales empotrados junto a la lógica programable.

La aparición de los SOPCs supuso una mejora en el desarrollo de las tecnologías digitales, permitiendo escalas de integración muy elevadas (submicrónicas, nanométricas etc.) con la ventaja adicional de la programabilidad en contraste con los SoC.

Hasta llegar al desarrollo de los SOPC, tal como se ha comentado, se desarrollaron de forma progresiva una serie de tecnologías, cada una de las cuales suponía una evolución respecto a la otra, pero todas ellas caracterizadas por ofrecer prestaciones diferentes en los diferentes parámetros que se utilizan para valorar estas tecnologías.

Los SOPC nacieron con el objetivo de aglutinar las mejores propiedades de cada una de las diferentes tecnologías, en los diferentes parámetros que definen el grado de optimización de estos sistemas.

D.3 Características Tecnológicas

Como se ha comentado, antes de la aparición de los SOPC como dispositivos válidos para el desarrollo de sistemas electrónicos existían una serie de tecnologías que competían en el mercado como dispositivos capaces de implementar sistemas electrónicos digitales complejos.

Cada una de estas tecnologías poseía una mayor o menor valoración en los diferentes parámetros que sirven para valorar este tipo de tecnologías, como pueden ser la fiabilidad del dispositivo, el coste de fabricación por unidad, los costes fijos, el tiempo de lanzamiento al mercado, la flexibilidad y el nivel de integración del dispositivo.

En la tabla de la Fig. D.1 se muestra una caracterización de las diferentes tecnologías en relación a los parámetros comentados.

	ASIC/SoC	ASSP/Micro	PLD
RISK	✗	✓	✓
UNIT COST	✓	~	~
FIXED COST (NRE)	✗	✓	✓
TIME-TO-MARKET	✗	✓	✓
FLEXIBILITY	✗	✗	✓
INTEGRATION	✓	~	~

Fig. D.1 Tabla comparativa de propiedades según la tecnología [Mead, 2001].

Como se puede apreciar cada una de las tecnologías de desarrollo de dispositivos tienen valoraciones diferentes según sea el criterio o parámetro que se quiera valorar.

Se puede observar que la gran ventaja de los dispositivos ASIC/SoC es su alto nivel de integración, superior al resto de dispositivos analizados. Por el contrario, los costes fijos son altos a la vez que son dispositivos poco flexibles.

Por otro lado están los dispositivos llamados PLD, los cuales sí que ofrecen unas buenas características en lo que a costes, tiempo en mercado y flexibilidad se refiere. En cambio este tipo de dispositivos no alcanza el nivel de integración que puede llegar a lograrse con los ASIC's.

Por tanto lo que se intentó desarrollar a partir de esta situación, es un dispositivo que pudiera aglutinar las mejores características de cada uno de estos dispositivos. De este modo, se buscó que estos nuevos dispositivos tuviesen la flexibilidad de los PLD's y la capacidad de

integración y el bajo coste de fabricación por unidad de los ASIC's y otros dispositivos como los ASSP.

Con este objetivo fueron desarrollados lo que hoy se conocen como SOPC (Sistema en un Chip Programable).

D.4 Características de la tecnología SOPC

Tal como se ha comentado, la tecnología SOPC nace con el objetivo de aglutinar las mejores propiedades que caracterizan a las diferentes tecnologías o dispositivos electrónicos digitales que habían aparecido hasta el momento.

En la siguiente tabla se muestran las características de estos dispositivos haciendo referencia a las diferentes propiedades que comúnmente se utilizan para la evaluación de los mismos.

RISK	✓	■ Risk is very low as design and requirements mistakes can be fixed. SOPC offers true architectural exploration
UNIT COST	~	■ Unit cost higher than SoC / ASIC, but is offset by low cost of ownership. New high volume programs help lower cost further
FIXED COST (NRE)	✓	■ No fixed costs or royalties
TIME-TO-MARKET	✓	■ Design tools and IP available to assist productivity – existing tools used in a new way
FLEXIBILITY	✓	■ High flexibility – Hardware can implement many different functions and implement evolving standards – Easy migration of functions from S/W <-> H/W
INTEGRATION	✓	■ Embedded microprocessors and high-speed I/O offer exceptional integration

Fig. D.2 Propiedades de la tecnología SOPC [Mead, 2001].

Como se puede observar en la tabla, se trata de una tecnología que, a diferencia de los dispositivos PLD, tiene mayor capacidad de integración que estos. Además se reducen los costes de fabricación unitarios, pese a que no se alcanza el nivel de los SoC/ASIC, aspecto que se ve compensado por la gran optimización en el resto de parámetros.

Lo que básicamente nos ofrece este tipo de dispositivos es una interacción absoluta entre la parte del diseño que concierne al hardware y la parte propiamente referida al software. Un esquema que muestra esta clase de interacción entre los diferentes dispositivos que conforman el hardware y el software de este tipo de sistemas se muestra a continuación en la Fig. D.3. En ella puede verse la interacción entre los difeerntes elementos de hardware que componene este tipo de sistemas (dispositivos PLD, puertas, periféricos etc.), con los elementos de software (algoritmos, procesadores etc.) que serán los que propiamente le darán una funcionalidad al sistema. La integración de ambas tareas, es una de las características esenciales de este tipo de sistemas.

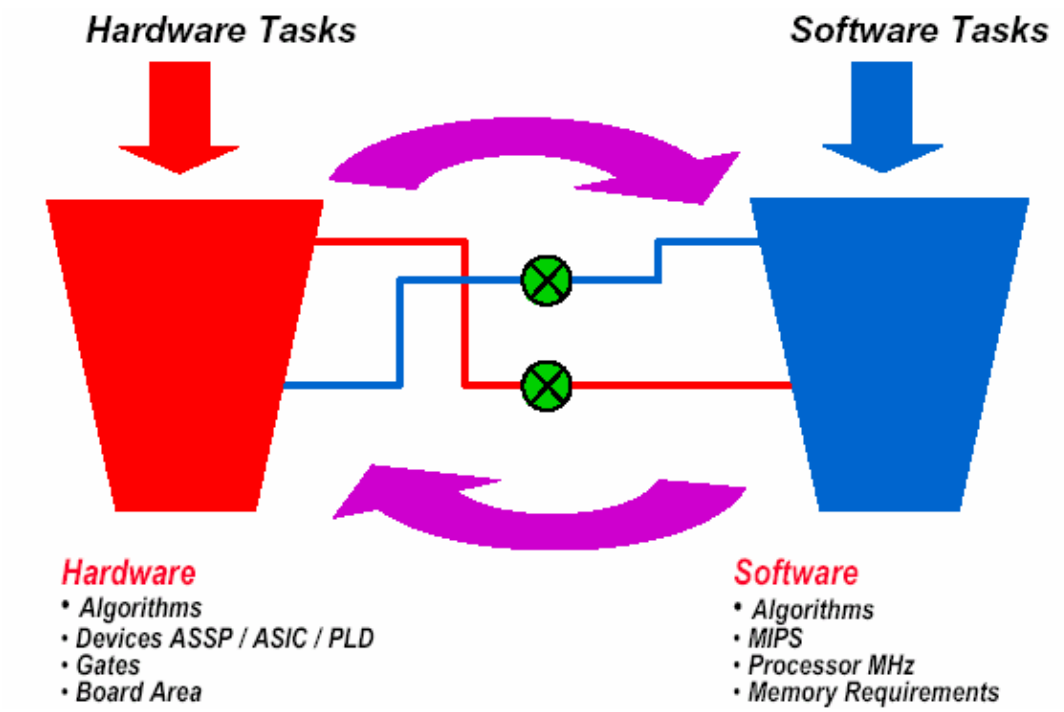


Fig. D.3 Modo de trabajo de un SOPC [Mead, 2001]

D.5 Soluciones Excalibur para dispositivos SOPC

El paquete ***Excalibur*** es una herramienta diseñada por **Altera**. Este paquete ha sido utilizado para la elaboración y desarrollo de este proyecto, con el objetivo de facilitar una aproximación al diseño e implementación de SOPC.

Este sistema se basa en un procesador diseñado para poder desarrollar conjuntamente las tareas implementadas a través de software y hardware de la palicación del usuario. Además esta herramienta proporciona otras herramientas para la configuración de sistemas en chip programables. Esta herramienta de desarrollo fue creada con el objetivo de facilitar la flexibilidad de los dispositivos lógicos programables con el potencial de los dispositivos de proceso.Un esquema del procesador y la interacción con otros periféricos del sistema se muestran en la Fig. D.4.

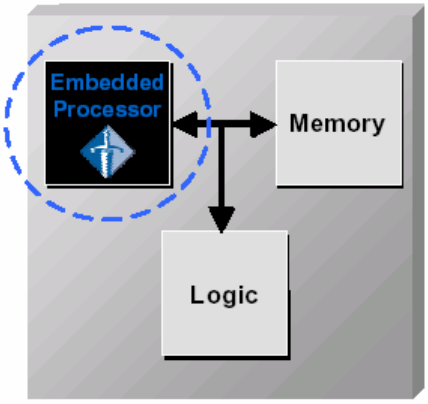


Fig. D.4 Esquema del SOPC Excalibur [Mead, 2001]

Este tipo de tecnología es capaz de combinar estructura de hardware y software implementada por el usuario, de forma que todas ellas conformen el sistema con aplicación específica que se quiera desarrollar.

Quartus II es un software para el diseño de dispositivos lógicos programables que vayan a ser implementados en chips de Altera. Como añadido existe un programa llamado ***SOPC Builder*** destinado a diseñar mediante una interfaz sencilla un SOPC sobre Quartus II. En el anexo E se describe con más detalle las herramientas de Altera utilizadas en este proyecto.

Un esquem ade la forma de actuación de este programa se muestra en la Fig. D.5. en ella puede verse como el programa SOPC a través de librerías de componentes y periféricos configura el sistema en chip deseado. Una vez configurados los componentes, el programa genera los archivos VHDL para la configuración del hardware del sistema, así como los archivos C y drivers para la configuración del software del mismo sistema.

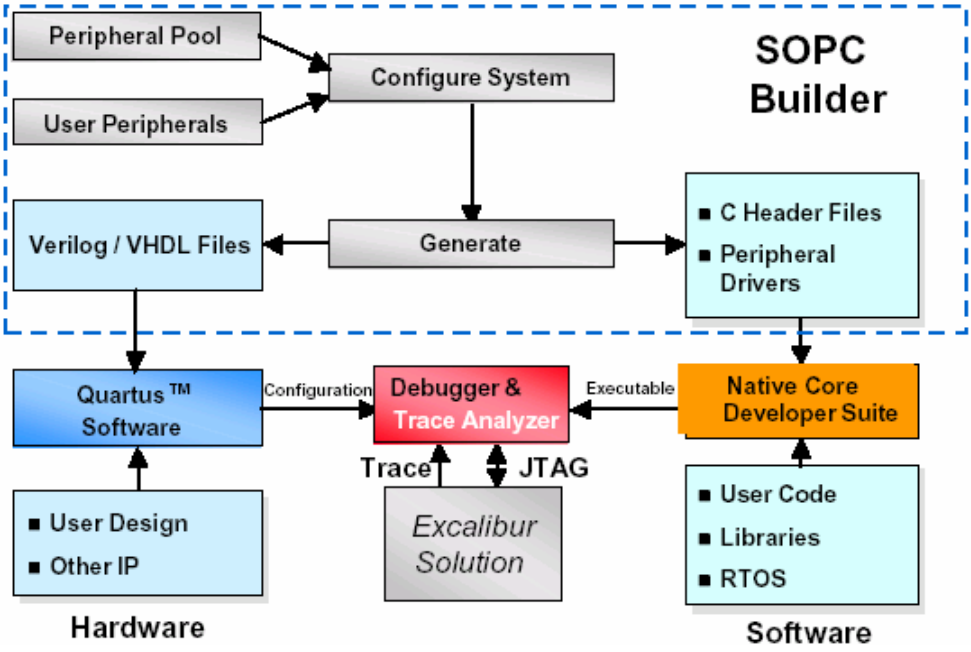


Fig. D.5 Etapas del desarrollo de un SOPC utilizando el software de Altera [Mead, 2001].

D.6 Conclusiones

Una vez realizada esta pequeña síntesis, primero analizando las ventajas de este tipo de tecnología de desarrollos y posteriormente viendo el grado de implantación de éstas mediante las placasutilizadas de Altera, se concluye que los SOPC son un sistema óptimo para el desarrollo e implementación de sistemas electrónicos digitales.

La implantación de este tipo de tecnología para el desarrollo de este tipo de sistemas ha venido superponiéndose al uso de los SoC, por las mayores prestaciones que ofrecen en los diferentes parámetros de estudio. El uso de SoC para la implementación de este tipo de sistemas tan sólo está justificado si la aplicación de los mismos va a suponer una producción a gran escala, debido a sus costes unitarios menores. Para este caso, en que se trabaja en una solución particular capaz de ofrecer grandes prestaciones, los SOPC son la solución más óptima desde el punto de vista técnico.

E Herramientas de Altera utilizadas en el proyecto.

E.1 Introducción

Para el desarrollo del proyecto y tal como se ha ido comentando a lo largo del mismo se han utilizado principalmente herramientas de la casa Altera, mediante las cuales ha sido posible implementar los diferentes sistemas que han permitido el desarrollo de este proyecto.

Puesto que el objetivo del proyecto era desarrollar una aplicación (en este caso una aplicación de tratamiento digital de imágenes) mediante el diseño de un Sistema en un Chip Programable, fue necesaria la elección de una placa de desarrollo que sirviera como base para todos los módulos de hardware y software del sistema. Para este cometido se eligió una placa de desarrollo *Excalibur*, basada en el microprocesador empotrado *Nios*, de Altera (véase Fig. E.1).

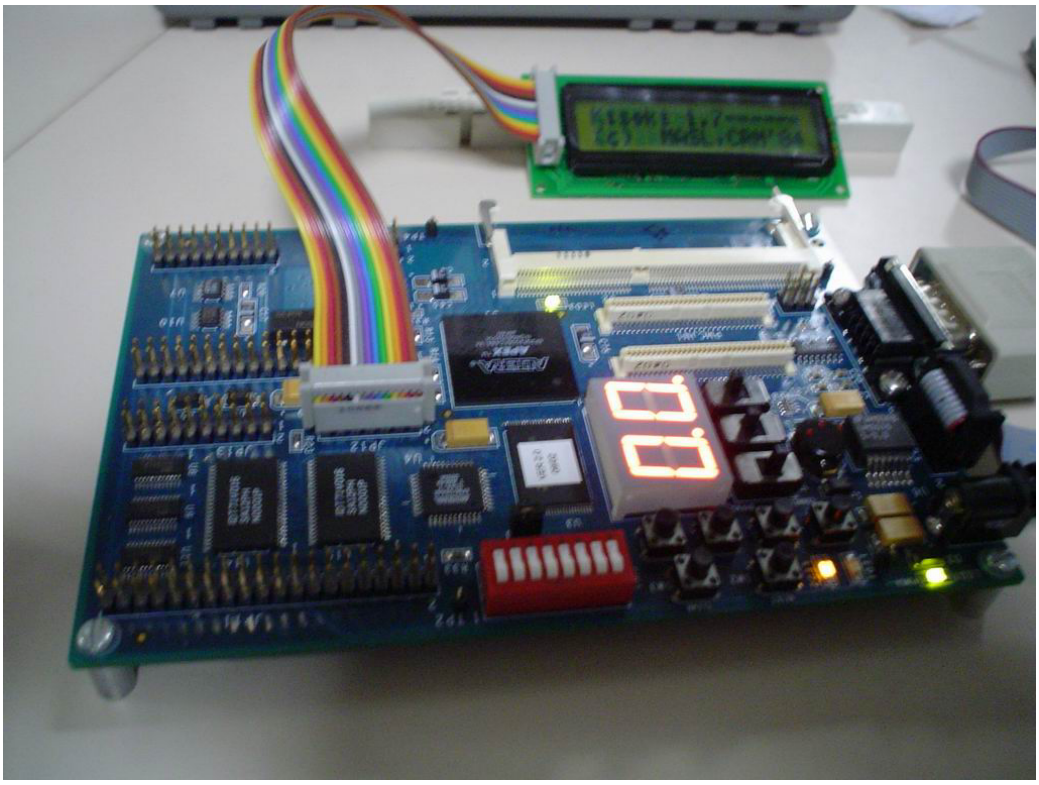


Fig. E.1 Placa de desarrollo Excalibur.

Para trabajar con esta placa de desarrollo se ha utilizado el software de diseño de hardware *Quartus II*, desarrollado también por Altera y que consiste en una aplicación para el desarrollo de forma precisa de sistemas en dispositivos programables de Altera. Este software ha permitido el diseño, la sintetización, la compilación y la verificación de los diferentes módulos desarrollados.

Una herramienta también utilizada e integrable en el paquete Quartus II, ha sido la aplicación *SOPC Builder*. Mediante esta aplicación, cuyo objetivo es la generación de Sistemas en un Chip Programable, se ha implementado el Subsistema Externo (SSE). Esta aplicación permite definir de forma óptima los tipos de bloques que van a utilizarse en el sistema así como una fácil interconexión de éstos con el hardware desarrollado mediante la arquitectura de bus

Avalon, propia de Altera. Mediante la determinación de estas especificaciones, SOPC Builder es capaz de generar de forma sencilla de cara al usuario un sistema con microprocesador empotrado mediante el cual se puede poner en marcha cualquier sistema funcional desarrollado.

Así pues, éstos han sido las tres herramientas esenciales con las que se ha trabajado para el desarrollo del hardware del SSE: la placa de desarrollo *Excalibur*, el paquete de software *Quartus II* y la aplicación *SOPC Builder*.

E.2 La placa de desarrollo Excalibur

E.2.1 Introducción

La placa de desarrollo *Excalibur* utilizada es una plataforma hardware para el desarrollo de forma rápida de sistemas basados en dispositivos FPGA APEX de Altera. La FPGA contenida en esta placa de desarrollo está preparada para configurarse con un sistema con procesador empotrado Nios de 32-bit, a la vez que está conectada a diversos módulos de memoria y de entrada/salida propios de la máquina de Von Newmann. Uno de los directorios de proyectos de Quartus II contiene un ejemplo de referencia de diseño junto al software de desarrollo para Nios. Tanto el diseño de referencia como el software están grabados en la memoria flash que incorpora la propia placa. El software de referencia de diseño incluye un monitor que puede ser utilizado para descargar y depurar programas.

E.2.2 Componentes

En la Fig. E.2 se muestran los componentes más significativos que se hallan en la placa de desarrollo Excalibur, muchos de los cuales han sido utilizados para la implementación del sistema de este proyecto.

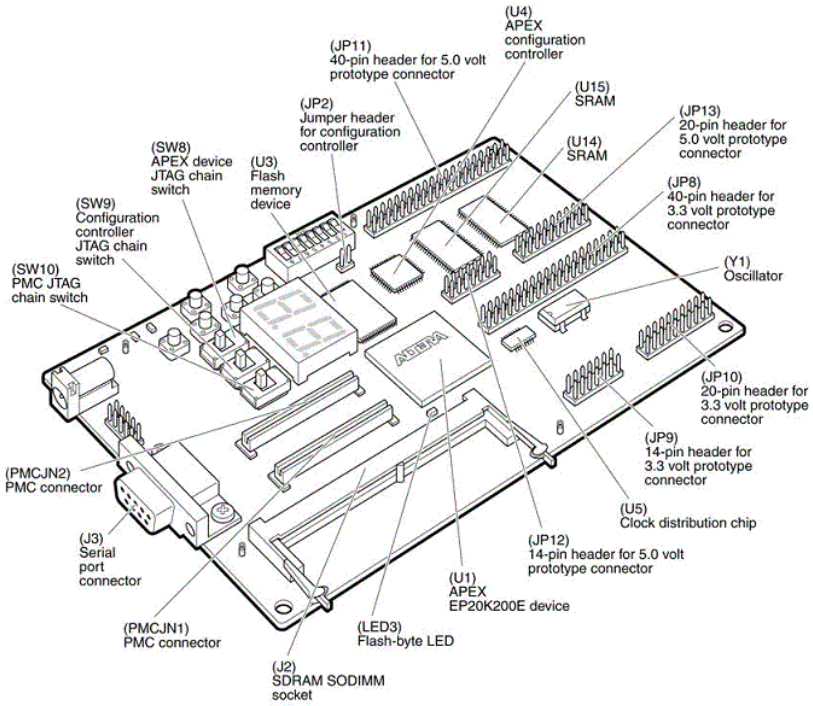


Fig. E.2 Esquema de la placa de desarrollo Excalibur.

A continuación se describirá con más detalle algunos de los componentes más significativos presentes en la placa y que sin duda son la parte fundamental para la implementación del diseño desarrollado.

- **Dispositivo APEX**: El dispositivo etiquetado como U1 es una FPGA APEX 20K200E. La configuración por defecto de este dispositivo, indicada por la FLASH, contiene una CPU Nios y otros periféricos que ocupan entorno al 25% o 35% de la lógica del dispositivo. Sus características más significativas se muestran en la Fig. E.3.

Table 1. APEX20K200E Device Features	
Maximum system gates	526,000
Typical gates	211,000
LEs	8,320
ESBs	52
Maximum RAM bits	106,496
Maximum macrocells	832
Maximum user I/O pins	382

Fig. E.3 Características del dispositivo APEX.

Como puede observarse se trata de un dispositivo con una considerable capacidad para la implementación de hardware, a través del más de medio millón de puertas equivalentes de las que se puede llegar a disponer.

Para cambiar la configuración del hardware implementado en este dispositivo se ha conectado el conector JTAG (JP3 en Fig. E.2) con el cable de descarga ByteBlaster MV, conectado a su vez al puerto paralelo del PC para poder descargar la nueva configuración del dispositivo desde Quartus II.

- **Memoria Flash**: El componentes U3 en la Fig. E.2 es un chip de memoria Flash de AMD (AM29LV800BB 1 Mbyte). Esta memoria está conectada al dispositivo APEX y se utiliza con dos propósitos: por un lado, como memoria de programa para el procesador Nios empotrado en el dispositivo APEX, y por otro lado, como contenedor de un archivo de configuración del hardware del dispositivo APEX, ya que éste al tratarse de un dispositivo FPGA basado en tecnología SRAM está totalmente desprogramada al dar tensión a la placa.

- **Dual SRAM Chips**: Los componentes U14 y U15 son dos chips de memoria SRAM de 256 Kbytes (128 K x 16-bits). Están conectados al dispositivo APEX y pueden ser utilizados por el procesador Nios para cualquier tarea de almacenamiento de datos.

- **Conectores de expansión 5V**: Los conectores JP11, JP12 y JP13 representados en la Fig. E.2 representan un sistema estándar de conexiones que puede ser utilizado por ejemplo como interfaz para la conexión de tarjetas. El procesador Nios utiliza el conector JP12 como interfaz para comunicarse con el display LCD de salida de texto, con el que han sido realizadas diferentes pruebas en este Proyecto. La Fig. E.4 muestra un esquema del conector utilizado para la comunicación con dicho display.

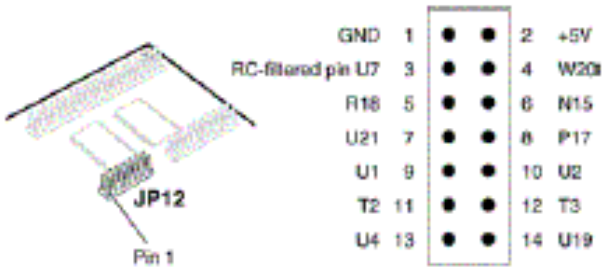


Fig. E.4 Conector JP12

- **Conector RS-232**: El conector JP3 del esquema general de la placa corresponde a un conector serie estándar DB-9. Éste es el típico conector utilizado para comunicaciones en serie utilizando el protocolo estándar RS-232. Utilizando un cable estándar de 9 pines puede utilizarse para las comunicaciones con un puerto COM de un PC. Las señales TXD (transmisión desde la placa), RXD (recepción en la placa), CTS y RTS (preparado para transmisión) utilizan los niveles lógicos de voltaje que marca el estándar RS-232 (Fig. E.5).

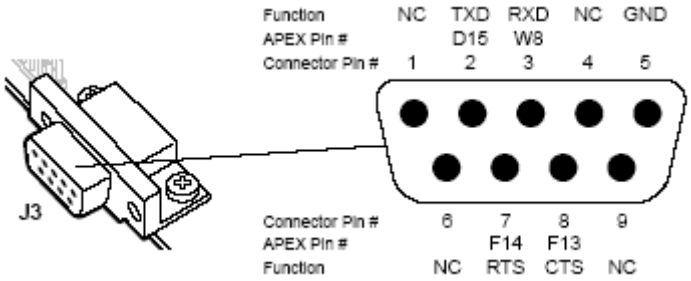


Fig. E.5 Conector RS-232

- **Conector JTAG**: El conector que en el esquema general se referencia mediante JP3, es un conector JTAG de 10 pines compatible con los cables de descarga MasterBlaster y ByteBlasterMV de Altera. Mediante este conector se consigue configurar el dispositivo desde el PC utilizando uno de estos cables. Para ello debe ser conectado el interruptor SW8 y debe situarse en posición de desconexión los interruptores SW9 y SW10, los cuales permiten otros usos para dichos cables que en este proyecto no ha interesado utilizarlos.

- **Controlador de Configuración APEX**: El dispositivo U4 indicado en el esquema de la placa es el controlador de configuración del dispositivo APEX antes mencionado. Se trata de un PLD de Altera (EPM7064). Viene configurado de fábrica con una lógica que configura el dispositivo APEX (U1) con los datos almacenados en la memoria Flash (U3) al alimentar la placa. Los tres dispositivos están interconectados de forma que los datos desde la memoria Flash puedan configurar a través del controlador el dispositivo APEX.

  A través de Quartus II pueden ser generados también archivos hexout capaces de configurar de forma personalizada el dispositivo APEX. Estos archivos son

almacenados en la memoria Flash y acostumbran a tener un tamaño inferior a los 256 Kbytes con lo que llegan a ocupar sólo un cuarto de la capacidad con la que cuenta la memoria Flash.

- **Pulsadores e interruptores**: El bloque SW1 (Fig. E.6) identificado en el esquema de la placa, es un sistema de interruptores 8-DIP conectados directamente al dispositivo APEX mediante los cuales se pueden asignar funciones específicas mediante el envío de señales lógicas al dispositivo APEX.

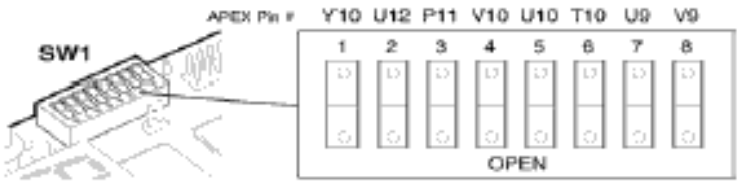


Fig. E.6 Interruptor SW1.

Existen en la placa dos pulsadores utilizados para unas funciones específicas y ya previamente configurados. Se trata por un lado del pulsador SW2, que se considera el **reset** del sistema. Cuando este pulsador es accionado, un valor lógico 0 es cargado en U7, el reset del controlador de configuración. Cuando esto ocurre el controlador vuelve a configurar el dispositivo APEX con el archivo que esté cargado en ese momento en la memoria Flash. El pulsador SW3, es el llamado **clear**. Cuando es pulsado un valor lógico 0 es configurado en el pin DEV_CLRn del dispositivo APEX. El resultado de esta acción dependerá de cómo se haya configurado en cada caso el dispositivo APEX.

E.3 El software de desarrollo Quartus II.

E.3.1 Introducción

El software de diseño Quartus II de Altera proporciona una completa y funcional plataforma para el diseño de sistemas electrónicos digitales que fácilmente se adapta a las necesidades de diseño de cada usuario particular. Es una herramienta, tal como se ha comentado, especialmente indicada para el desarrollo de Sistemas en Chips Programables (SOPC). El software incluye soluciones para todas las fases de diseño utilizando diferentes tipos de dispositivos lógicos programables tales como FPGA o CPLD. El proceso de desarrollo de un sistema digital mediante el software, sigue el proceso que se detalla en la Fig. E.7.

Se pueden utilizar las diferentes aplicaciones con las que cuenta el software Quartus II para implementar los diferentes procesos que conforman el proceso de diseño de un sistema digital. Para ello Quartus II provee al usuario de una interfaz gráfica con la que se pueden desarrollar los diferentes procesos de forma óptima.

A continuación se intentará describir de forma general los medios y los pasos seguidos para la implementación de cada uno de éstos pasos durante el desarrollo del Proyecto.

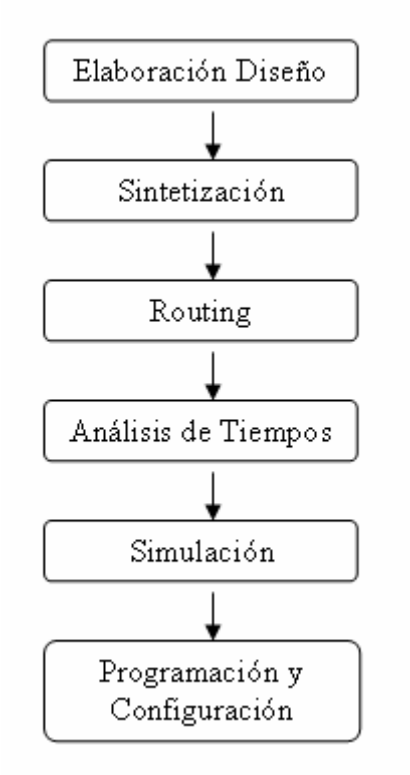


Fig. E.7 Esquema del proceso de diseño en Quartus II.

E.3.2 Diseño

Un proyecto gestionado desde Quartus II incorpora todos los archivos de diseño y archivos de software necesarios para el buen hacer a la hora de implementar el diseño del sistema que se quiera elaborar. Para el diseño de éste, se puede usar la herramientas Quartus II Block Editor o Text Editor, mediante las cuales se podrá implementar de forma óptima los diferentes diseños digitales haciendo uso de las funciones predefinidas en las librerías de Altera.

Se han utilizado dos metodologías básicas para la elaboración de los sistemas digitales. Por un lado, módulos editados a través de lenguajes de descripción de hardware VHDL (.vdh) y, por otro lado, utilizando la aplicación de diseño de sistemas digitales que incorpora Quartus II mediante la utilización de archivos tipo diagramas de bloque (.bdf).

E.3.3 Síntesis

Quartus II ofrece una herramienta para sintetizar los diseños implementados en la etapa de diseño. Para ello dispone de una gran número de opciones para realizar el proceso a través del módulo Analysis & Syntesis Settings, mediante la cual se pueden elegir las diferentes metodologías para sintetizar el diseño.

Este módulo dispone de un compilador con el que se pueden analizar el nivel de corrección del diseño implementado (archivos VHDL) y proceder así a crear la base de datos y archivos con la que se desarrollará el proyecto iniciado en Quartus II.

E.3.4 Placement & Routing

La herramienta Quartus II Fitter se encarga de localizar y plasmar el sistema que ha sido diseñado sobre los dispositivos lógicos presentes en la placa de desarrollo utilizada. Utilizando los datos que se han generado a través del proceso de síntesis, este adaptador se

encarga de de combinar las precondiciones en cuanto a lógica y tiempos derivadas del diseño creado con los recursos de los que dispone la placa de desarrollo o el dispositivo programable utilizado en cada caso.

El adaptador se encargará de asignar cada una de las funciones lógicas generadas en la posición óptima del sistema de celdas lógicas del dispositivo hardware, seleccionando al mismo tiempo las interconexiones y asignaciones de pines más apropiadas. Esto es lo que se conoce como *Placement & Routing* (Fig. E.8).El *Placement* es como se denomina al proceso de asignación de celdas, mientras Que EL Routing se encargará del conexionado de las mismas.

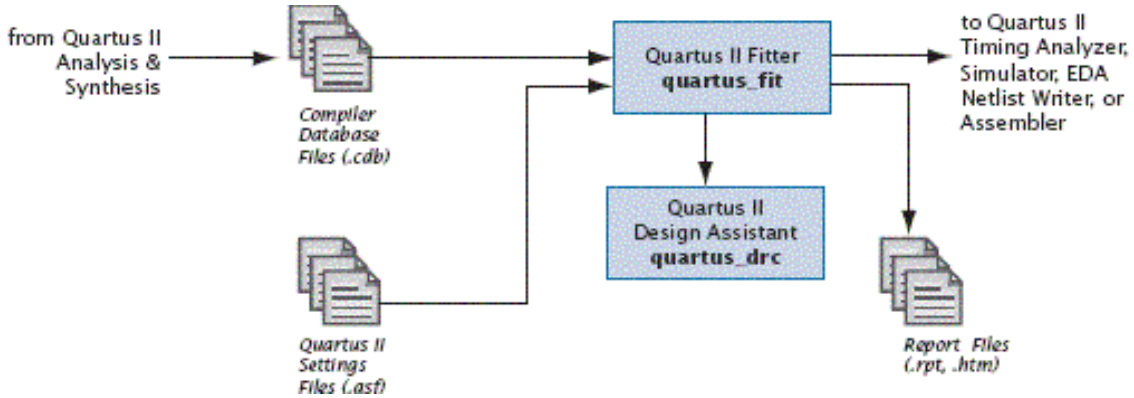


Fig. E.8 Proceso de *Routing*.

E.3.5 Simulación

Para comprobar que el funcionamiento de los diferentes dispositivos es el deseado, Quartus II incorpora una herramienta de simulación, el Quartus II Simulator. Mediante esta herramienta se puede simular el comportamiento de cualquier diseño generado en un proyecto. Dependiendo de las necesidades de cada usuario, puede configurarse una simulación de tipo funcional para los diferentes diseños o bien una simulación de tipo temporal, para analizar tanto las operaciones lógicas del diseño así como el peor caso en cuanto al retardo o comportamiento temporal del diseño en el dispositivo.

Esta aplicación permite tanto la simulación de diseños completos como de partes o módulos separados, facilitando de esta forma la progresión en el desarrollo del sistema.

De entre los diferentes parámetros que pueden configurarse en las simulaciones destacan el periodo temporal que debe cubrir la simulación, el vector de estímulos generados, etc.

Antes del inicio de cualquier simulación, se debe generar el archivo netlist apropiado indicando si se destina a simulación temporal o funcional. En consecuencia, se deberá crear el vector apropiado de señales que regulen la simulación. El simulador utilizará estos vectores de entrada de señales para generar el valor de las salidas que el sistema digital da como resultado.

Para la generación de estos vectores de entrada de datos, existe una aplicación en el software que permite generar estos vectores a través de un formato de ondas. Se trata de la aplicación *Waveform Editor* (Fig. E.9), la cual, como se ha comentado, representa los vectores mediante formas de onda que describen el comportamiento del dispositivo lógico en el diseño implementado.

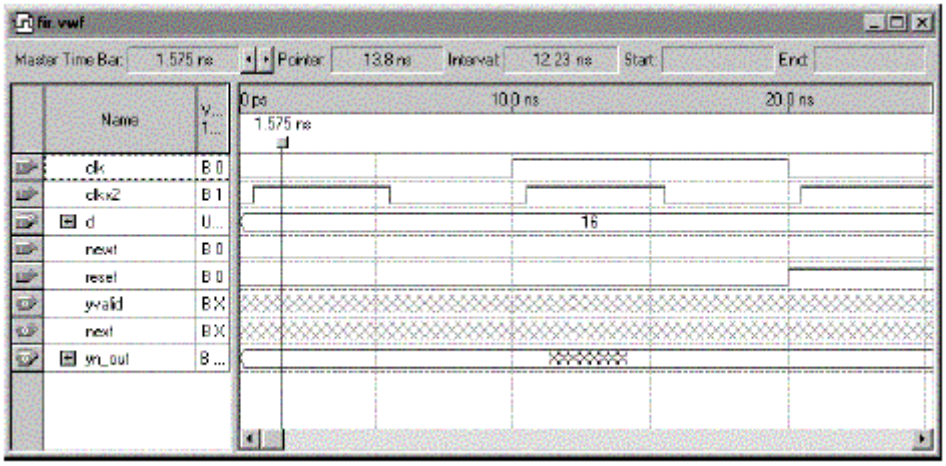


Fig. E.9 Pantalla del *Waveform Editor*.

E.3.6 Programación

Una vez el proyecto generado mediante Quartus II ha sido compilado mediante las herramientas que el mismo software dispone para ello, se puede pasar a programar o configurar el dispositivo de Altera. El módulo de ensamblaje del compilador de Quartus II genera los pertinentes archivos de programación que el módulo Quartus II Programmer utilizará para configurar un dispositivo a través del hardware de Altera.

El proceso que se sigue en la programación de dispositivos mediante este sistema se muestra en la Fig. E.10.

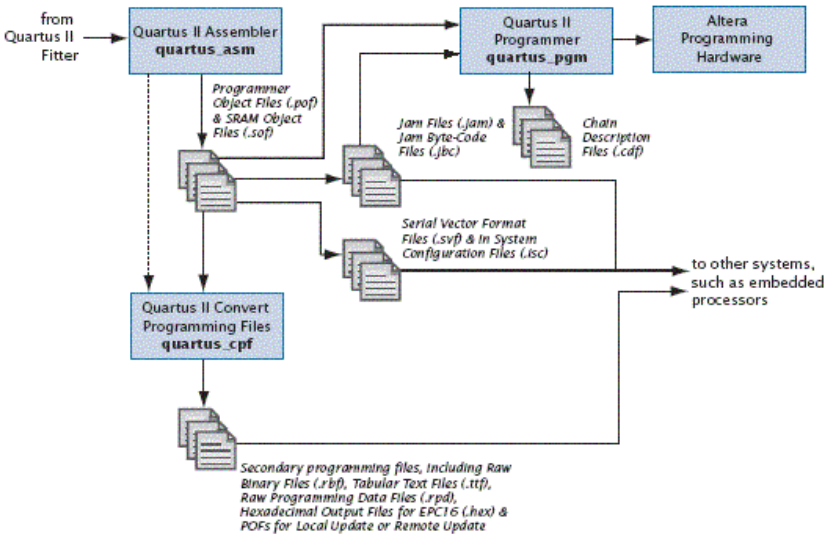


Fig. E.10 Esquema del proceso de programación.

E.4 La aplicación SOPC Builder

E.4.1 Introducción

SOPC Builder es una herramienta para el desarrollo de sistemas basados en bus como es éste el caso, utilizando componentes almacenados en librerías de la propia aplicación como pueden ser CPU's, interfaces de memoria y periféricos. Con SOPC Builder se puede llegar a

generar un módulo del sistema simple que automáticamente genera un listado de componentes e interfaces estándar que componen este tipo de sistemas.

E.4.2 Descripción de la aplicación

SOPC Builder, tal como se ha dicho, permite generar un sistema digital complejo a partir de la inclusión e interconexión de los diferentes módulos o componentes que van a conformar el mismo. La aplicación es accesible desde el entorno del software Quartus II, el cual deberá iniciarse una vez ya se haya creado un proyecto a través de Quartus II.

El sistema SOPC se compone de dos partes: por un lado, posee una interfaz gráfica (GUI) mediante la cual se lista los posibles componentes que podrían formar parte del sistema electrónico que se quiera idear. Dentro de esta interfaz gráfica, cada uno de los componentes seleccionables posee su propia interfaz, mediante la cual se podrá configurar las propiedades de cada uno de ellos. Esta interfaz gráfica (GUI) crea un archivo descriptivo del sistema llamado archivo PTF del sistema.

Por otro lado, existe un generador que convierte el archivo de descripción del sistema (PTF) que se ha generado mediante la elección y configuración de componentes, en su implementación a nivel de hardware. Este generador crea un archivo en un lenguaje de descripción de hardware (para este proyecto VHDL) que describe de igual forma el sistema desarrollado y que es capaz de sintetizar éste en el dispositivo elegido.

Tal como se ha comentado con la interfaz gráfica (GUI), se podrá especificar que componentes contendrá el sistema y como estos componentes serán dispuestos y generados. Esta interfaz por si misma no genera ningún tipo de lógica ni software, simplemente se trata de un editor para generar un archivo que sea capaz de describir de forma más o menos detallada el sistema que quiera generarse mediante un archivo PTF.

Esta interfaz gráfica consta de cuatro partes fundamentales; cada una de las cuales desarrolla una función determinada en la generación del sistema deseado.

- **Índice de Sistema**: esta página consta de dos secciones fundamentales: un listado de todos los componentes disponibles en la librería, tal y como muestra la Fig. E.11, y otra lista con todos los módulos que han sido incluidos ya en el actual sistema. Mediante esta parte, se puede describir que componentes e interfaces están incluidas en el sistema y que dispositivos maestros están conectados a qué dispositivos esclavos. A su vez, también se describen el mapa de direcciones del sistema y las asignaciones IRQ.

- **Asignación de Prioridades**: En el caso que dos dispositivos maestros compartan o tengan a acceso al mismo dispositivo esclavo, se debe determinar un orden de prioridades de acceso para evitar problemas. Para ello SOPC Builder incorpora esta funcionalidad. Mediante esta aplicación se puede determinar el orden de prioridad de los dispositivos maestros a la hora de acceder o intercomunicarse con un componente esclavo del sistema. Los conflictos que hacen referencia a estas prioridades son resueltos de manera sencilla por el sistema. Si cada componente maestro posee una prioridad *Pi,* y la suma de todas sus prioridades es *Ptotal,* entonces el dispositivo maestro i tendrá mayor o menor prioridad con respecto a los demás maestros en función del valor total de esta prioridad.

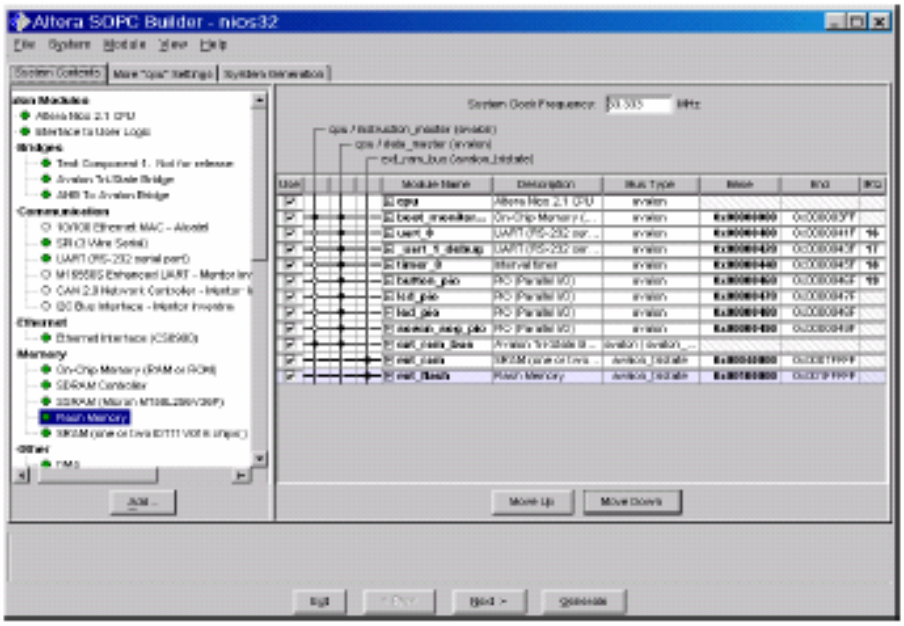


Fig. E.11 Aspecto del GUI de SOPC Builder.

- **Páginas CPU**: Cada CPU que se haya incluido en el sistema posee una tabla asociada. Esta tabla permite especificar las relaciones o uniones internas entre la CPU y el resto de módulos del sistema. Por ello, cada CPU manejará los diferentes módulos del sistema asociados según la programación que por defecto tenga en memoria.

- **Página de generación del Programa**: El programa de generación de la herramienta SOPC Builder comprende diferentes tareas. Lo primero que hace es leer el archivo PTF que ha sido generado mediante la selección y caracterización de componentes. Seguidamente se encarga de generar los archivos de software necesarios (drivers, librería, etc.) para cualquier componente del sistema que necesite un software de soporte para su correcto funcionamiento. Crea a continuación el programa generador para cada una de las componentes descritas en el sistema. Cada componente del sistema deberá tener su propio sistema de generación (el cual, por ejemplo, será capaza de generar los archivos HDL que describan al componente en cuestión). El programa generador principal del SOPC Builder activará estos subprogramas generadores asociados a cada uno de los componentes. Finalmente generará el archivo HDL (en este caso en VHDL) que describirá de forma general todo el sistema desarrollado. Una vez generado esto, el sistema generará un archivo bsf para el módulo del sistema. Para comprobar el correcto funcionamiento del sistema generado, a continuación se generará un proyecto de simulación *ModelSim* con el cual se podrá verificar el correcto funcionamiento del sistema. Una vez generado esto, se sintetiza el módulo que representa al sistema entero y todo su contenido usando la herramienta *LeonardoSpectrum.* El resultado de ello es la generación de un archivo en el lenguaje de descripción de netlist EDIF (extensión .edf) con el cual se puede hacer la adaptación e inclusión del sistema completo en el dispositivo electrónico utilizando Quartus II.

F El dispositivo FPGA APEX-20K200E

F.1 Introducción

Para el desarrollo del Sistema en un Chip Programable, es imprescindible que dentro de la placa de desarrollo exista un dispositivo lógico programable mediante el cual puedan implementarse los diferentes sistemas desarrollados. En este caso y mediante la utilización de la placa Excalibur, el dispositivo FPGA configurado por defecto es el APEX 20K200E de la familia de dispositivos APEX de Altera.

La familia de dispositivos APEX fue desarrollada por Altera en 1999, para ser adaptados en el desarrollo de Sistemas en chip programables, permitiendo a los diseñadores integrar en ello sistemas de modo eficiente, pudiendo usar éstos en gran cantidad de aplicaciones.

Este tipo de dispositivos que oscilan entre las 30000 y los 1,5 millones de puertas y ciclos de reloj de hasta 840 MHz, son capaces de ofrecer un alto grado de integración en cualquier dispositivo que decida usarse, como en este caso la placa de desarrollo Excalibur. En su momento, la arquitectura MultiCore de los dispositivos APEX, supuso un gran avance en cuanto a integración y eficiencia con respecto a los dispositivos FPGA aparecidos hasta entonces.

Esta familia de dispositivos posee tres variantes: APEX 20K, APEX 20KE y APEX 20KC. En este proyecto ha sido utilizado el APEX 20K, las características del cual se detallarán más adelante.

F.2 Evolución

La familia de este tipo de dispositivos supuso una evolución bastante considerable, tanto en lo que se refiere a los dispositivos que hasta entonces existían, como entre los diferentes dispositivos que fueron desarrollándose dentro de la familia APEX.

De entre los diferentes dispositivos de la familia, el que supuso un mayor avance en cuanto a integración y prestaciones fue el dispositivo APEX 20KC, el cual es capaz de combinar capas de 0,15 μm y tener las interconexiones entre sus diferentes bloques configuradas a partir de hilos de cobre, lo que permite aumentar en fiabilidad y eficiencia con respecto a las interconexiones realizadas a través de capas de aluminio.

La evolución dentro de la familia APEX, en cuanto a mejora de prestaciones a partir de los diferentes modelos se muestra en la Fig. F.1.

Sin duda una de las mejoras más importantes que aportaron esta nueva familia de dispositivos fue la inclusión en su modelo APEX 20KC, de una tecnología de interconexión de bloques mediante cobre, el cual es utilizado como material para realizar las conexiones entre bloques en todas sus capas. Mediante el uso de esta tecnología, se consiguió incrementar la velocidad con que los datos se distribuían en este tipo de dispositivos, debido a la menor resistencia que ofrece el cobre en relación a otro tipo de materiales con los que se realizan este tipo de interconexiones como pudiera ser aluminio. Esta mayor conductividad minimiza los retrasos en las transmisiones de señales en este tipo de dispositivos tal como se muestra en la Fig. F.2.

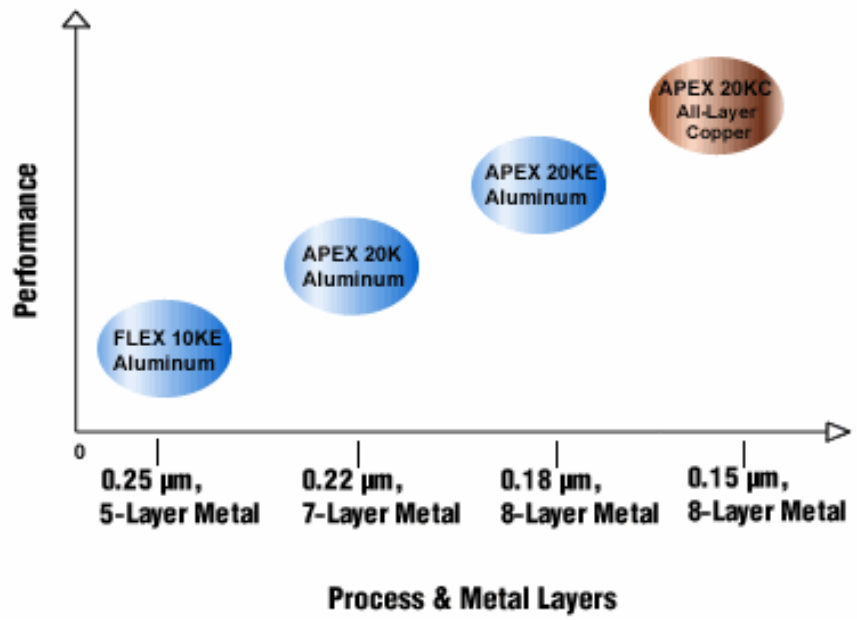


Fig. F.1 Evolución de los dispositivos APEX.

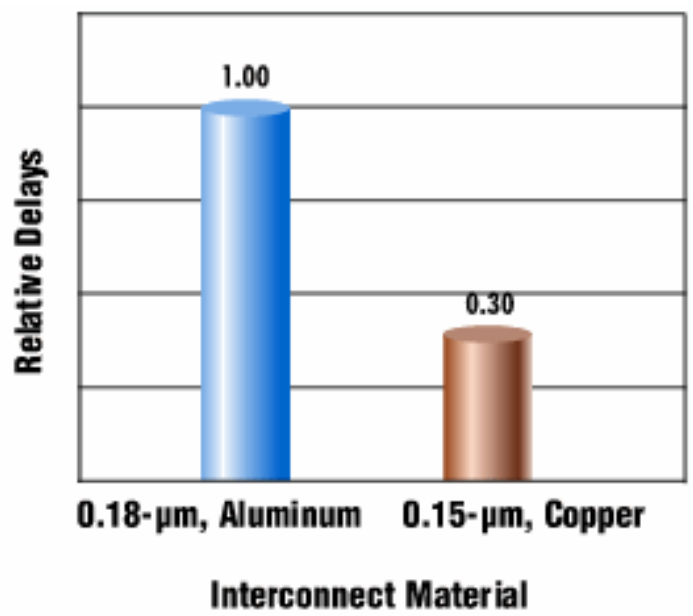


Fig. F.2 Evolución de los retrasos según el material.

F.3 Arquitectura APEX 20K

El objetivo de este apartado no es otro que el de mostrar de forma general la estructura lógica que presenta este tipo de dispositivos de Altera.

Tal como se ha comentado antes, esta familia de dispositivos FPGA supuso un gran desarrollo en referencia al resto de dispositivos desarrollados hasta entonces por la arquitectura interna con la que fueron diseñados, denominada MultiCore. Se pensó en un dispositivo que fuera capaz de aglutinar en un mismo chip los beneficios que hasta entonces ofrecían por separado

tres familias de dispositivos también de Altera: FLEX 10K, FLEX 6000 y MAX 7000. Se intentó desarrollar una arquitectura que fuera capaz de combinar las mejores propiedades de cada una de estas familias para poder así aprovechar todas estas ventajas mediante la incorporación de un único dispositivo en los diferentes sistemas.

En la Fig. F.3 se muestra la arquitectura que presenta este tipo de dispositivos:

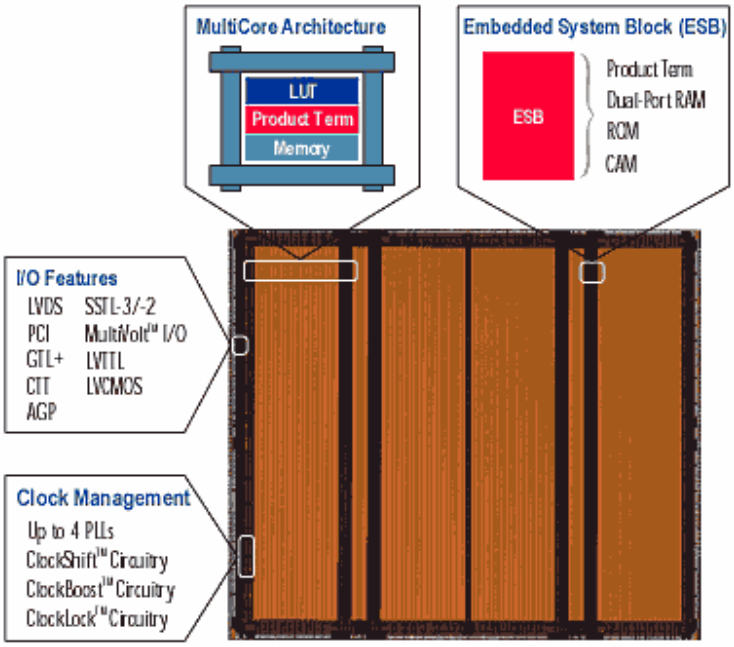


Fig. F.3 Arquitectura del APEX-20K

Como puede observarse esta arquitectura resulta de la combinación de tres tipos de arquitecturas PLD's diferentes: tablas lógicas tipo LUT como las que podrían hallarse en dispositivos FLEX 10K y FLEX 6000; bloques de términos producto como los implantados en dispositivos MAX 7000 y bloques de memoria como los hallados en dispositivos FLEX 10KE; todas ellas familias que fueron utilizadas como fuentes para el desarrollo de esta tipo de arquitectura. Todas estas estructuras desarrolladas en un mismo dispositivo permitieron la integración de complejas funciones de forma eficiente.

En la Fig. F.4, presentada a continuación, se muestra de forma más detallada la estructura que presenta la arquitectura MultiCore dentro del dispositivo APEX 20K.

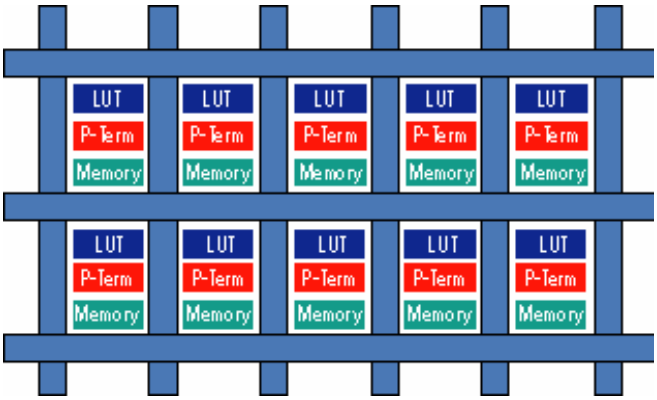


Fig. F.4 Arquitectura MultiCore APEX 20K

Tal como se puede contemplar en la Fig. F.4 las arquitectura MultiCore está fundamentada en una estructura de bloques lógicos (LAB's), cada uno de los cuales conformado por 10 elementos lógicos del tipo FLEX 6000. Todos ellos son combinados en una nueva estructura denominada MegaLAB, la cual debe considerarse como una estructura de LAB's. Cada una de estas nuevas estructuras está conformada a través de 16 bloques de LAB's, una estructura avanzada insertada denominada (ESB) y un sistema de interconexiones que comunica a los diferentes dispositivos.

Las interconexiones entre todo este conjunto de dispositivos lógicos aglutinados en una sola estructura tampoco resultan sencillas. Existe un canal de inteconexiones que comunica cada uno de los LAB's con el resto, y a su vez otro canal de comunicación que comunica a estos dispositivos con la estructura ESB. Este tipo de conexionado puede contemplarse en la Fig. F.5.

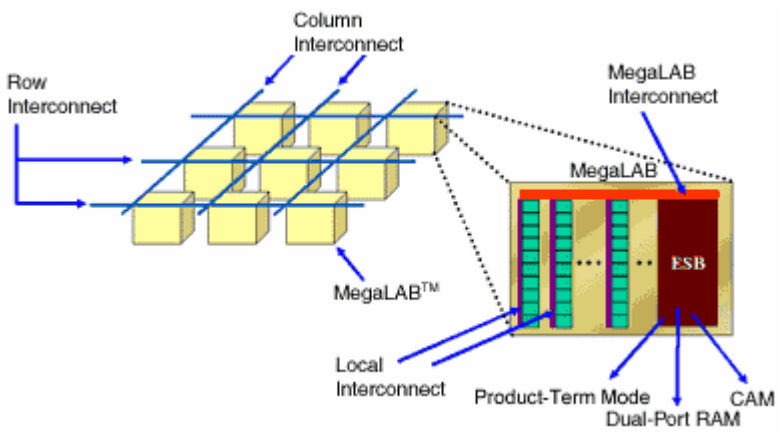


Fig. F.5 Conexiones de la estructura MegaLAB

Como puede comprobarse existen canales de interconexión entre las matrices de bloques lógicos (LAB's) organizados en filas y columnas. A su vez existe otro canal de comunicación que permite unir las matrices de bloques con la estructura ESB.

Para la implementación de estas interconexiones entre las estructuras, se utiliza habitualmente aluminio excepto para la familia APEX 20KC, la cual posee todas las líneas de interconexión hechas a base de cobre, el cual tener una menor resistencia que el aluminio permite optimizar las comunicaciones entre bloques.

La estructura ESB es el núcleo de la arquitectura MultiCore. Cada uno de estos bloques contiene 2048 bits programables que pueden ser configurados para el desarrollo de diferentes funciones: bien como dispositivos lógicos o bien como diferentes tipos de memoria como ROM, RAM o memoria de contenido direccionable (CAM).

Cada pin de entrada/salida de este dispositivo es alimentado mediante elementos llamados IOE localizados al final de cada una de las filas y columnas que forman las interconexiones entre los bloque lógicos. Cada IOE contiene un buffer bidireccional I/O y un registro.

La estructura completa de este tipo de dispositivo, con los bloques lógicos, los canales de interconexión y todos los pines de entrada/ salida se muestran en la Fig. F.6:

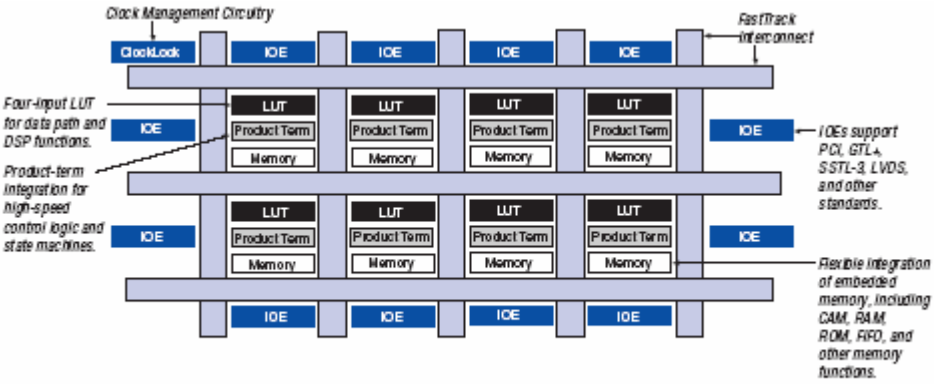


Fig. F.6 Diagrama general de bloques del dispositivo APEX 20K

Cada uno de los bloques que conforman la estructura MegaLAB, está compuesta como se ha comentado de una serie de bloques lógicos que han sido denominados LAB.

Para detallar de forma más precisa como están estructurados cada uno de estos dispositivos lógicos, debe pensarse que cada uno de ellos resulta ser también un conjunto de elementos lógicos más sencillos, que se denominarán LE's. Cada LAB estará conformado por un conjunto de 10 de estos bloques, el acarreo asociado a cada uno de los bloques, las señales de control y las conexiones locales, las cuales permiten comunicar a LE's del mismo bloque o bien con LAB's adyacentes. En la Fig. F.7 se puede observar la estructura que presenta este tipo de dispositivos.

Como puede observarse cada uno de los LAB's contiene lógica para manipular las señales de control de sus propios elementos lógicos (LE's). Entre estas señales de control se encuentran el clock, señal asíncrona de clear, señal síncrona de clear, señal asíncrona de preset etc. Hasta un máximo de seis señales pueden llegar a ser utilizadas al mismo tiempo.

Como parte de los componentes de esta estructura LAB, se ha comentado que existen unos elementos lógicos de menor complejidad denominados a partir de ahora LE. Estas unidades lógicas son las de menor complejidad que conforma un dispositivo del tipo APEX 20K. Cada uno de estos bloques posee cuatro entradas del tipo LUT, la cual es un generador de funciones que de forma sencilla puede generar cualquier función que implique a cuatro variables. Además cada uno de estos elementos posee un registro programable y un acarreo.

La estructura que presentan estos tipos de dispositivos lógicos se presenta en la Fig. F.8. Como puede observarse, cada uno de estos bloques posee dos salidas hacia los canales de intercomunicación, uno hacia el MegaLAB y otro hacia la estructura que interconecta los diferentes elementos LAB, llamada FastTrack. Además de los canales de interconexión entre dispositivos, los LE's poseen dos métodos para poder conectarse entre ellos de forma rápida y precisa: son los puertos denominados como *carry chains* y *cascade chains*. El primero permite dar soporte a funciones de tipo aritmético, tales como contadores o sumadores; mientras que el segundo permite implementar funciones con gran número de entradas como comparadores.

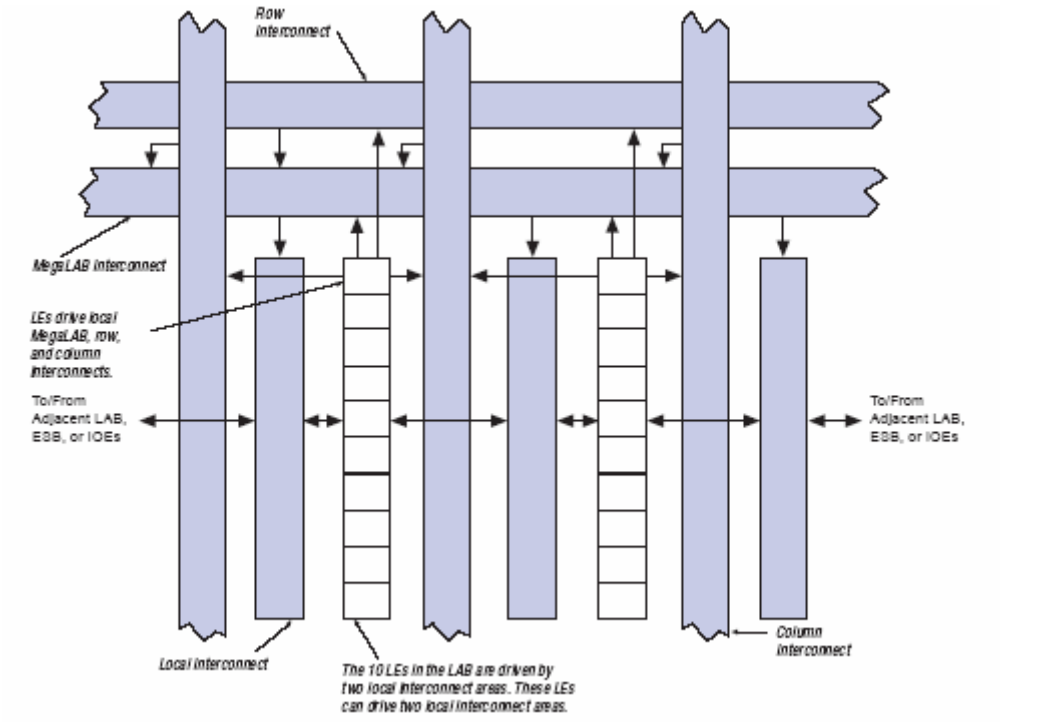


Fig. F.7 Estructura de un bloque lógico LAB

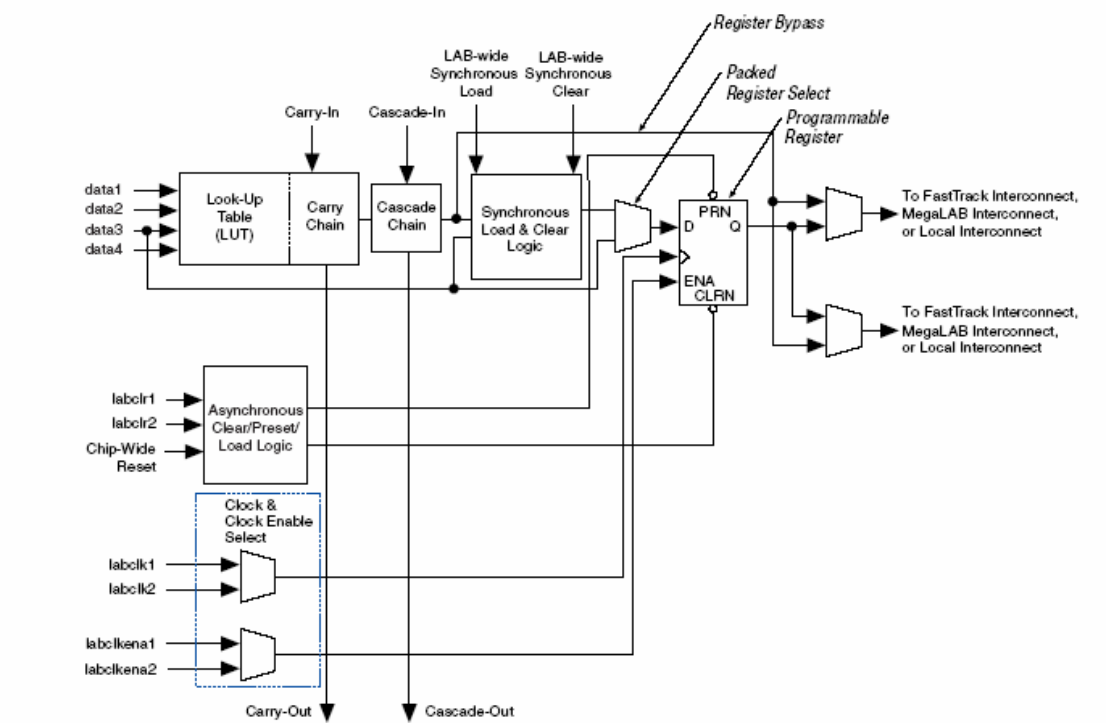


Fig. F.8 Estructura de un elemento lógico LE.

F.4 Dispositivo utilizado

Una vez analizada la estructura general que presentan este tipo de dispositivos en cuanto a su arquitectura y estructura interna, se pretende analizar las características del dispositivo de esta familia utilizado (APEX 20K200E), al mismo tiempo que se comparan sus características con otros dispositivos de las misma familia.

A modo de comparación se presenta en la tabla Fig. F.9, una comparación entre los diferentes dispositivos FPGA que conforman la familia APEX 20KE. Como puede contemplarse, se comparan valores como le máximo número de puertas del dispositivo, los elementos lógicos LE presentes, los números de pins I/O etc.

Como se observa en la tabla, el dispositivo utilizado en este caso, posee las mejores prestaciones en la gran mayoría de los parámetros analizados.

Table APEX 20KE Device Overview

Device	EP20K30E	EP20K60E	EP20K100E	EP20K160E	EP20K200E
Typical Gates	30,000	60,000	100,000	160,000	200,000
Maximum System Gates	122,704	161,792	262,912	404,480	525,824
Logic Elements (LEs)	1,200	2,560	4,160	6,400	8,320
Maximum RAM Bits	24,576	32,768	53,248	81,920	106,496
Phase-Locked Loops (PLLs)	2	2	2	2	2
Speed Grades(1)	-3, -2, -1	-3, -2, -1	-3, -2, -1	-3, -2, -1	-3, -2, -1
Maximum User I/O Pins	128	196	246	316	376

Fig. F.9 Comparativa de disposivos APEX 20KE.

Sin duda y pese a no poseer el nivel de integración de los dispositivos desarrollados actualmente, que poseen hasta un nivel de integración 10 o 20 veces superior, se puede considerar a esta FPGA como un dispositivo óptimo para desarrollar la aplicación que se ha venido definiendo a lo largo del proceso.

G Acerca de TWAIN

G.1 Introducción

Debido, en su momento, a la incipiente necesidad de los diferentes tipos de usuarios con respecto a la inclusión de imágenes en diferentes archivos o trabajos, los fabricantes de dispositivos de captación y adquisición de imágenes y los diseñadores de software para el tratamiento de estas imágenes decidieron desarrollar un estándar que fuese capaz de comunicar dispositivos y aplicaciones de software.

Este estándar beneficiaría a los fabricantes de dispositivos, los cuales verían que sus productos podrían comunicarse con el mayor número de aplicaciones, así como a los diseñadores de éstas que verían como no deberían preocuparse por si el dispositivo utilizado para la adquisición de imágenes era uno u otro.

Con este propósito nació TWAIN, con el objetivo de dar más consistencia y simplicidad a la interrelación entre dispositivos y aplicaciones de software.

Como uno de los objetivos de este proyecto es la adquisición de imágenes a través de una cámara digital para posteriormente tratar los datos obtenidos, se pensó que sería una buena solución el usar esta tecnología.

G.2 Descripción de los sistemas TWAIN

TWAIN define un protocolo estándar para la comunicación entre las diferentes aplicaciones de software y los múltiples dispositivos de adquisición de imágenes

Los sistemas TWAIN se componen de tres partes esenciales tal como se muestra en la Fig. G.1.

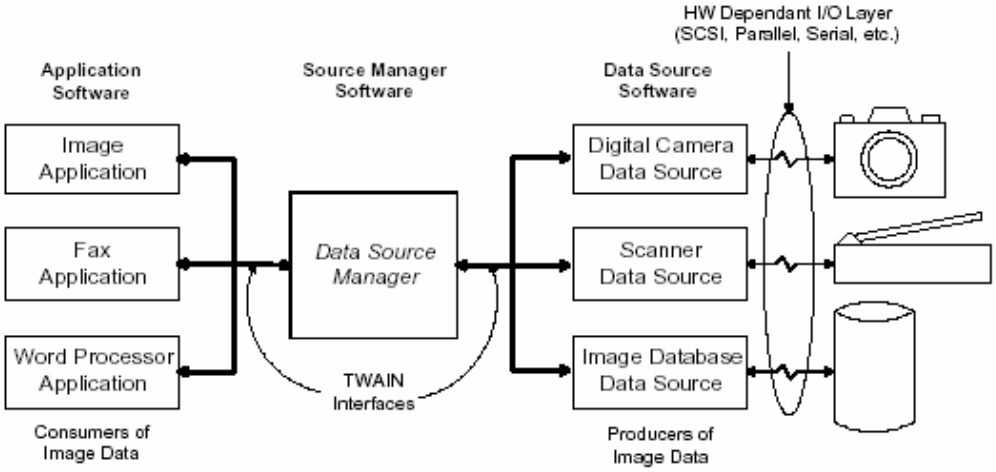


Fig. G.1 Composición de un sistema TWAIN

- *Aplicación de software*: la aplicación debe ser modificada y adaptada para permitir su interrelación con TWAIN.

- *Software director de Fuente*: es el encargado de manejar las interacciones entre la aplicación y la fuente u origen de datos.
- *Software de Fuente:* este software controla al dispositivo de adquisición de imágenes y es diseñado por el fabricante del dispositivo, de acuerdo con las especificaciones que marca el protocolo TWAIN.

G.3 Arquitectura de TWAIN

La transferencia de datos entre el dispositivo y la aplicación mediante el uso del protocolo TWAIN es posible gracias a la combinación de tres elementos de software, comentados previamente, presentes en la arquitectura general del sistema.

Dichos elementos usan la arquitectura TWAIN para comunicarse. Esta estructura consta de cuatro capas (Fig. G.2).

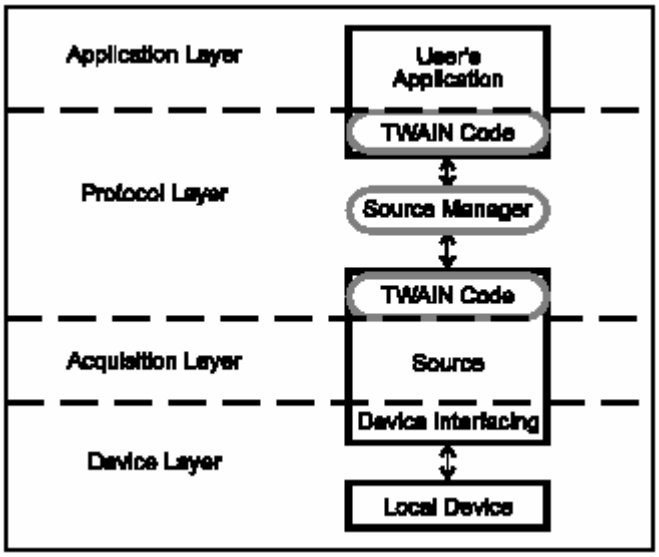


Fig. G.2 Capas de TWAIN

- **Aplicación:** Corresponde a la aplicación que es ejecutada por el usuario. TWAIN describe las directivas generales para la aplicación con respecto a como los usuarios deben acceder a las funcionalidades de TWAIN y como seleccionar entre los diferentes dispositivos.
- **Protocolo:** El protocolo es el "lenguaje" y la sintaxis utilizada por TWAIN para comunicarse. Es aquí donde se implementan instrucciones precisas y las comunicaciones requeridas para la transferencia de datos entre las diferentes partes.
- **Adquisición:** Los sistemas de adquisición de datos pueden ser físicos, tales como scanners o webcams), así como lógicos (por ejemplo una secuencia de imágenes en una base de datos). Los elementos de software escritos para controlar las capturas o adquisiciones de imágenes son llamados Fuentes y residen principalmente en esta capa.
- **Dispositivo:** Esta es la localización de los drivers de bajo nivel habitualmente usados para los diferentes dispositivos. Se encarga de traducir comandos recibidos desde el dispositivo a comandos hardware y acciones específicas para cada uno de los drivers y dispositivos, de forma que el dispositivo en particular pueda interpretarlos.

G.4 Comunicación entre elementos de TWAIN

La comunicación entre los diferentes elementos que conforman TWAIN es posible gracias a dos puntos de entrada. Éstos son llamados *DSM_Entry()* y *DS_Entry()* (Fig. G.3).

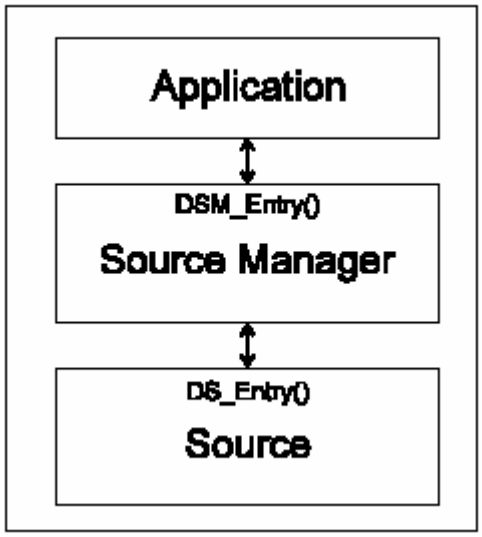


Fig. G.3 Esquema de la comunicación entre la aplicación y la fuente.

Aplicación: El objetivo de la aplicación no es otro que el de adquirir datos de la fuente o dispositivo en cuestión. Sin embargo, es imposible que la aplicación pueda controlar de forma sencilla el dispositivo directamente.

Es por ello que todos los requerimientos, bien sean de datos, información sobre capacidades o propiedades del dispositivo, información de errores, etc. deben ser conducidos a través del Source Manager.

Aproximadamente hay definidas unas 140 operaciones en TWAIN, las cuales pueden ser enviadas desde la aplicación hacia el Source Manager para su transmisión. La aplicación debe definir que elemento, si el Source Manager o el Source es el destinatario final de la información que se envía.

La aplicación se comunica con el Source Manager a través de la función *DSM_Entry()*, cuyos parámetros de entrada siguen la siguiente estructura:

- Una estructura de identificación que provee información sobre la aplicación que genera la llamada a la función.
- El destino de esta función (Source Manager o Source)
- Un "triplet" que describe la operación requerida. Este triplet debe especificar:
    - Grupo de datos para la operación (DG_).
    - Tipo de datos para el argumento de la operación (DAT_).
    - Mensaje para la operación (MSG_).

Esta función retorna un valor que indica el éxito o el fracaso en la operación de envío de la información.

Source Manager: este elemento del sistema TWAIN facilita la comunicación entre la aplicación y el Source, da soporte a la selección de Source por parte del usuario y configura el Source para el acceso por parte de la aplicación.

Las comunicaciones desde la aplicación al Source Manager llegan a través de el *DSM_Entry()*, cuyo destino es el Source Manager. Éste procesa el mismo la operación.

Si el destino del *DSM_Entry()* es el Source, el Source Manager traduce la lista de parámetros de información, reemplaza los parámetros de destino y llama a las funciones adecuadas del Source para las operaciones.

Source: El Source recibe operaciones a realizar desde la aplicación vía el Source Manager, o directamente desde el Source Manager. Éste se encarga de procesar la información y de devolver la respuesta adecuada (Código de Retorno - este código está prefijado con TWRC_) indicando los resultados de la operación hacia el Source Manager.

Si el elemento original que ha efectuado la operación es la aplicación en si, entonces el Código de retorno se envía hacia la aplicación como el valor de retorno de la función *DSM_Entry()*.

Si la operación no se ha llevado a cabo con éxito, entonces un Código de Condiciones (cuyo prefijo es TWCC_) es enviado por el Source, donde se especifica información sobre los posibles fallos.

G.5 Uso de los "Triplets"

Las funciones *DSM_Entry()* y *DS_Entry()* son utilizadas para ordenar las operaciones que deben realizarse. Las operaciones no son más que acciones que son llamadas bien por la aplicación o bien por el Source Manager.

La operación que se desea realizar viene indicada por un "triplet", que no es más que un trío de parámetros pasados en la llamada a las funciones. Cada "triplet" que se pasa a la función especifica una y sólo una operación particular que deberá realizarse.

Los tres parámetros que conforman esta estructura son:

Data Group (DG_XXXX):
Todas las posibles operaciones que pueden llegar a realizarse, están divididas según el valor que tome este identificador.

En principio y para este caso, se tendrán en cuenta dos tipos de operaciones:

- **CONTROL** (el identificador es *DG_CONTROL*): llevan a cabo el control de la sesión de adquisición de datos TWAIN.
- **IMAGE** (el identificador es *DG_IMAGE*): estas operaciones trabajan con datos de tipo imagen. Un ejemplo del uso de este tipo de identificador es para el requerimiento de inicio de una transferencia de datos de imagen.

Data Argument Type (DAT_XXXX):

Este parámetro indica el tipo de dato que va a ser enviado u operado. El tipo de argumento puede hacer referencia a una estructura de datos o a una variable. Existen numerosos tipos de datos que pueden llegar a pasarse como argumentos.

Message ID (MSG_XXXX):
Este parámetro identifica la acción que la aplicación o el Source Manager desean que tenga lugar. Existen también numerosos tipos de mensajes que se analizarán posteriormente.

G.6 Protocolo de Comunicación

Las tres partes descritas anteriormente (Aplicación, Source Manager y Source) deben configurar su comunicación entre si para así asegurarse la correcta adquisición de datos del dispositivo en cuestión. Esta comunicación se lleva a cabo mediante un protocolo compuesto de siete estados descritos a continuación (Fig. G.4):

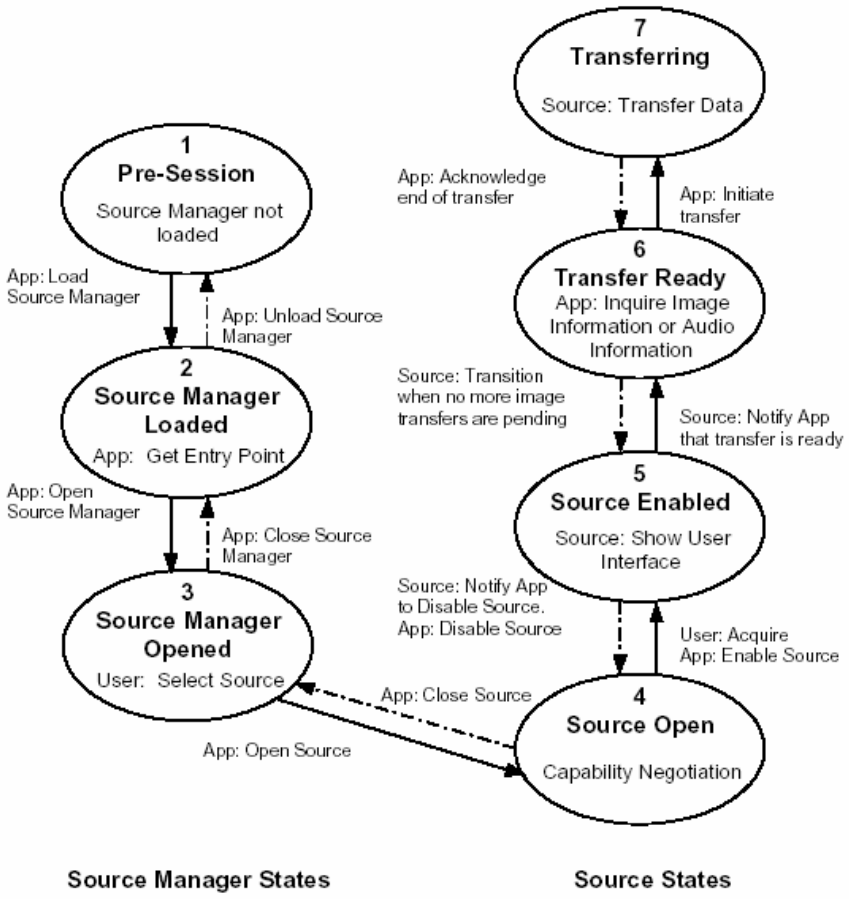


Fig. G.4 Esquema del protocolo TWAIN

1. **Pre-Session:** Es el estado en el que se encuentra el sistema Source Manager antes que la aplicación se decida a iniciar una nueva comunicación con él. En este estado el código que hace referencia al Source Manager ha sido instalado en el disco. En el caso de utilizar un sistema operativo como Windows, como es éste caso, el código también está instalado en memoria, pues su implementación no es más que una DLL.
2. **Source Manager Loaded:** En caso de no haberse utilizado Windows como sistema operativo, el código que hace referencia al Source Manager es cargado en memoria.

En este momento el sistema está preparado para aceptar operaciones a través de "triplets" que provengan de la aplicación.

3. **Source Manager Open:** En este estado el Source Manager está preparado para proveer un listado de posibles dispositivos, así como para abrirlos y cerrarlos. El Source Manager permanecerá en este estado durante el resto de la sesión. A su vez, este sistema rechazará cualquier intento de cierre de sesión mientras la sesión abierta tenga cualquier tipo de dispositivo abierto.
4. **Source Open:** En este estado, el dispositivo ha sido cargado e iniciado por el Source Manager en respuesta a un requerimiento que se le envía desde la aplicación. A partir de aquí el dispositivo está ya preparado para recibir operaciones. El dispositivo debe asegurarse de que existen diferentes recursos en el sistema para llevar a cabo la comunicación, a la vez que la aplicación puede configurar las características que van a definir a la imagen capturada (niveles de resolución, tipo de color, etc.)
5. **Source Enabled:** En este estado el dispositivo ha sido habilitado a través de una instrucción desde la aplicación a través del Source Manager, con lo que está ahora preparado para transferencias de datos.
6. **Transfer Ready**: Este es el estado en que el dispositivo indica si está preparado o no para el envío de datos. En este estado el dispositivo se encuentra a la espera de recibir los datos desde la aplicación en los que se defina las características de la imagen a capturar. Una vez conseguido esto, el dispositivo envía señal a la aplicación conforme inicia la transferencia de datos.
7. **Transferring**: Este es el estado en el que se está realizando propiamente la transferencia. Una vez ésta finalice o bien haya ocurrido un error en la misma, según haya informado el dispositivo a la aplicación, será la aplicación la que decida hacia que estado evoluciona el sistema de comunicación.

H El Hardware del Subsistema Externo

H.1 Introducción

En este anexo se mostrarán tanto los esquemas como los códigos VHDL que han servido para implementar el Subsistema Externo (SSE), pero antes de ello, véase un resumen sobre su estructura.

H.1.1 Visión general del SSE

Se decidió denominar como *Subsistema Externo* (SSE) a aquella parte del sistema que está implementada fuera del PC, tal y como se muestra en la Fig. H.1.

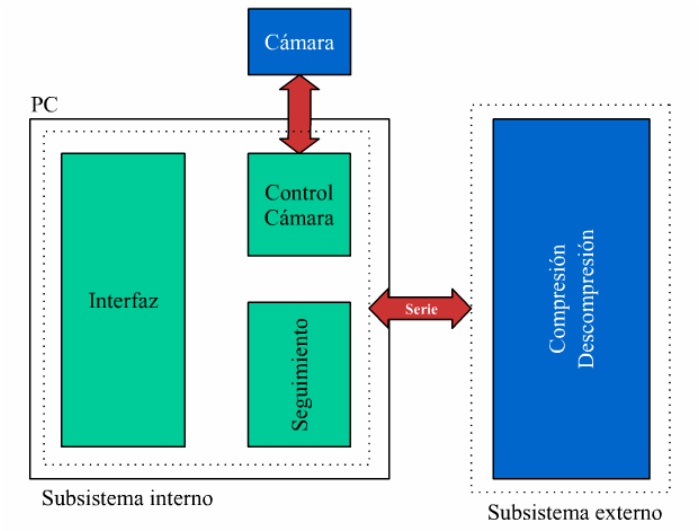


Fig. H.1 Esquema general del sistema.

El SSE es un sistema generado basado en un SOPC (*System On a Programmable Chip*) implementado sobre una FPGA (*Field Programmable Gate Array*) incluida en una placa de desarrollo *Excalibur* (Fig. H.2) de la casa Altera.

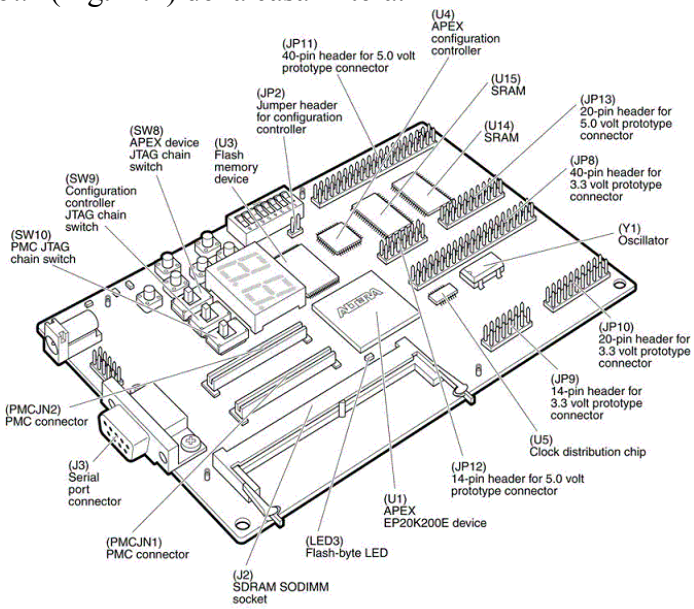


Fig. H.2 Placa de desarrollo Excalibur

El SOPC contenido en la FPGA APEX EP20K200E con la que se ha trabajado (U1 según la Fig. H.2) consiste en un bus (de tipo *Avalon*, un formato de Altera) al que van conectados varios módulos (Fig. H.3). La mayoría de estos módulos corresponden o bien a hardware también incrustado en esta FPGA o bien a dispositivos externos a la misma, como ahora los LEDs o los 'Siete Segmentos' incluidos en la placa.

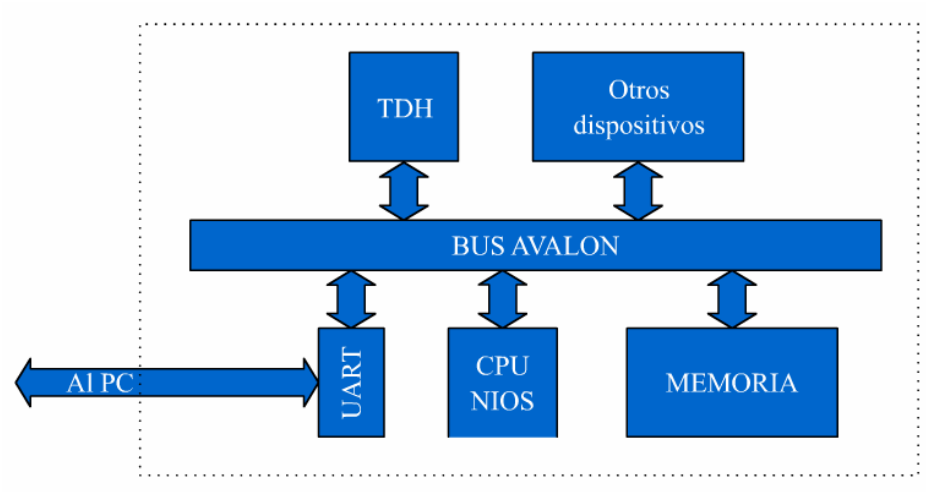


Fig. H.3 Esquema del SOPC en el que está basado el SSE.

De todos los módulos que van conectados al bus, solamente el módulo TDH ha sido el que se ha tenido que desarrollarpartiendo de cero. El resto, como ahora la UART, la CPU y demás venían prefabricados, y lo único que se ha tenido que hacer con ellos ha sido ajustar ciertos parámetros y acoplarlo al bus. Este acoplamiento se ha llevado a cabo utilizando el software *SOPC Builder*, incluido en el paquete de desarrollo de Altera (véase E.4, pág. 48).

H.1.2 El módulo TDH

Ésta, ha sido sin lugar a dudas, la parte más complicada de este proyecto. La estructura de este módulo se puede ver reflejada en la Fig. H.4.

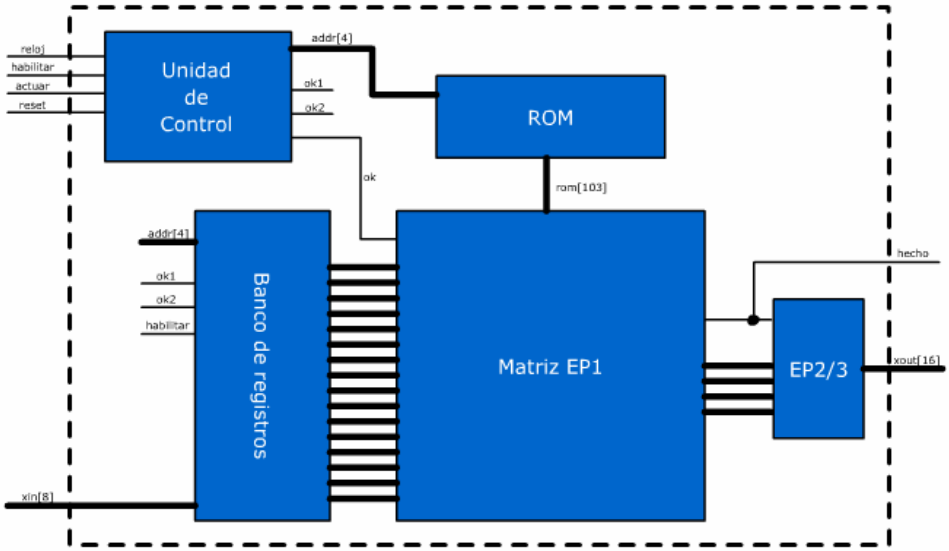


Fig. H.4 Esquema del módulo TDH

Tanto el **banco de registros** como la **ROM** son elementos prefabricados, aunque el contenido en la ROM se ha tenido que calcular tal y como se explicó en 3.4.1 y en 3.4.2.

La llamada **Unidad de Control** es el conjunto de dispositivos sencillos, como ahora un contador y unos *latches,* que proporcionan las señales de control adecuadas al resto del módulo.

Y por último, la parte más importante es la matriz sistólica que se presenta en la Fig. H.5 se ha decidido dividir en los bloques **Matriz EP1** (elementos de proceso ep1, en color azul) y **EP2/3** (elementos de proceso ep2 en verde y ep3 en rojo).

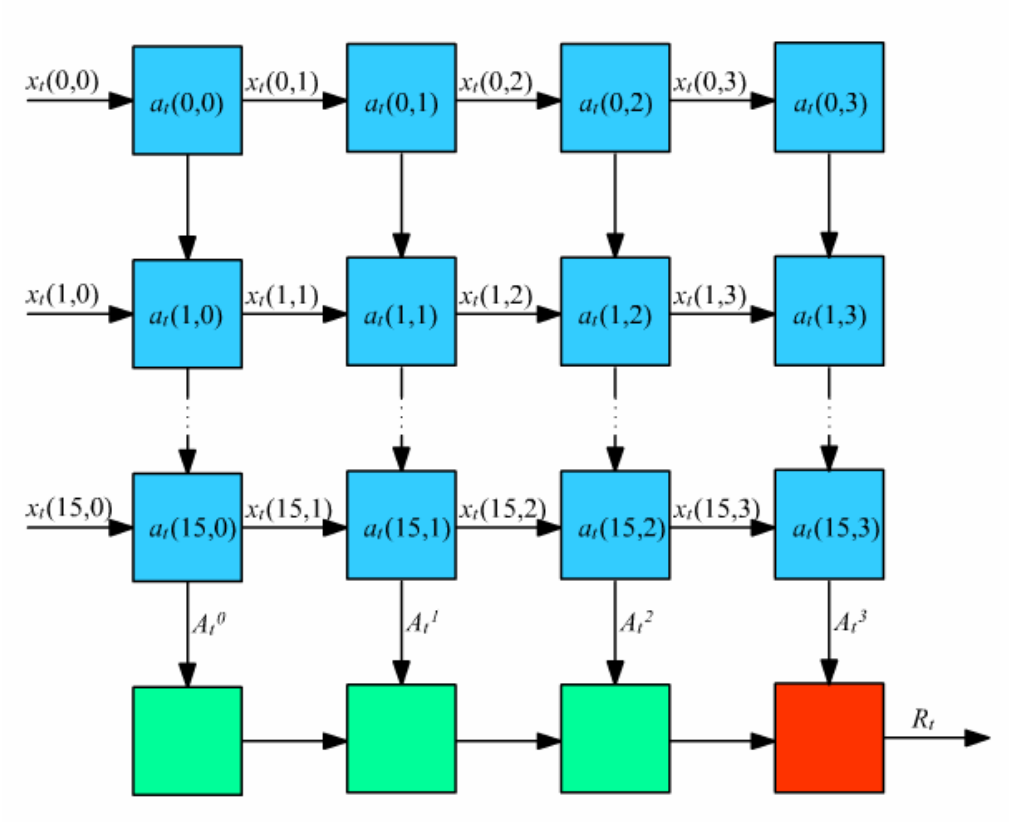
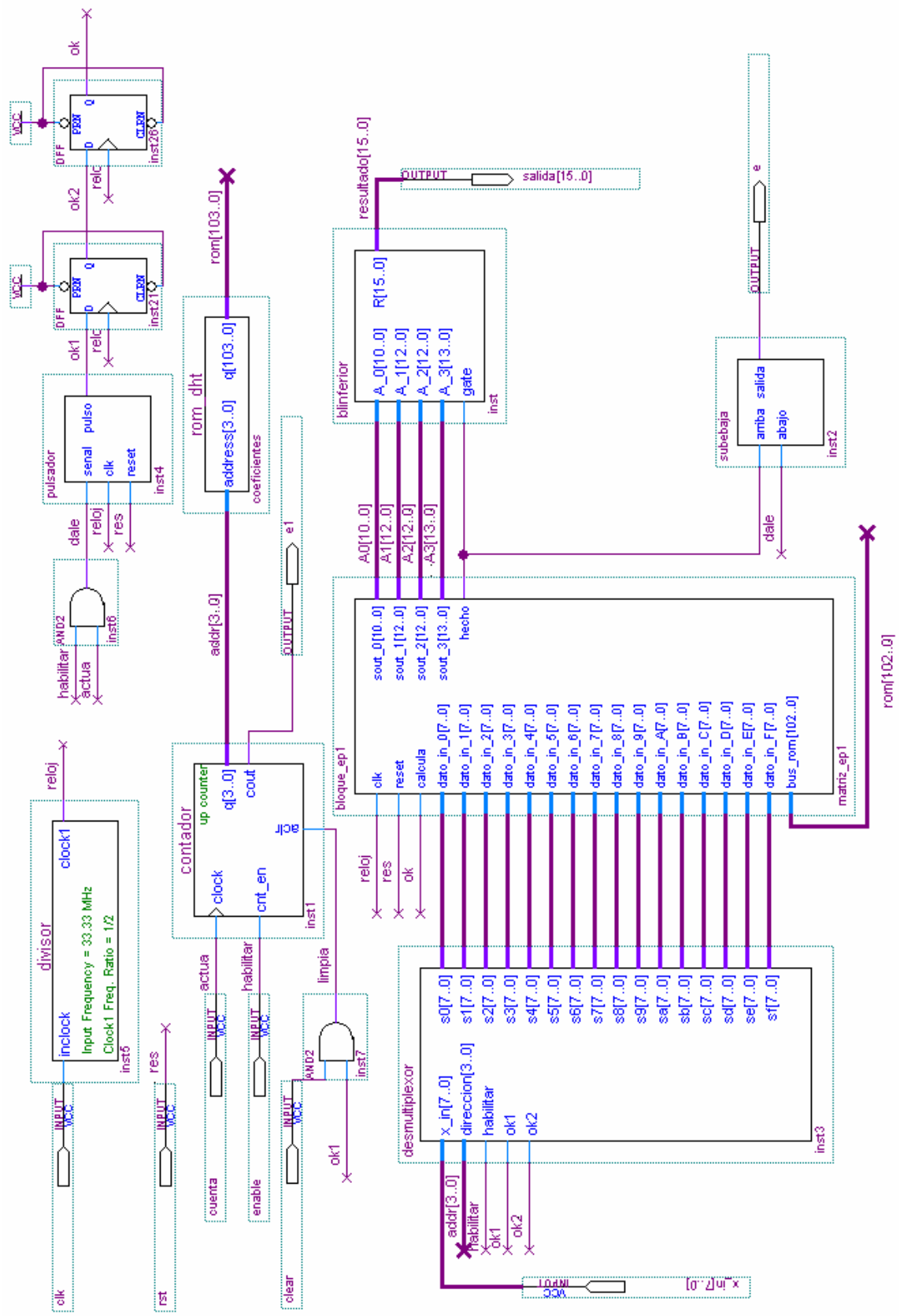


Fig. H.5 Esquema de la matriz sistólica del módulo TDH.

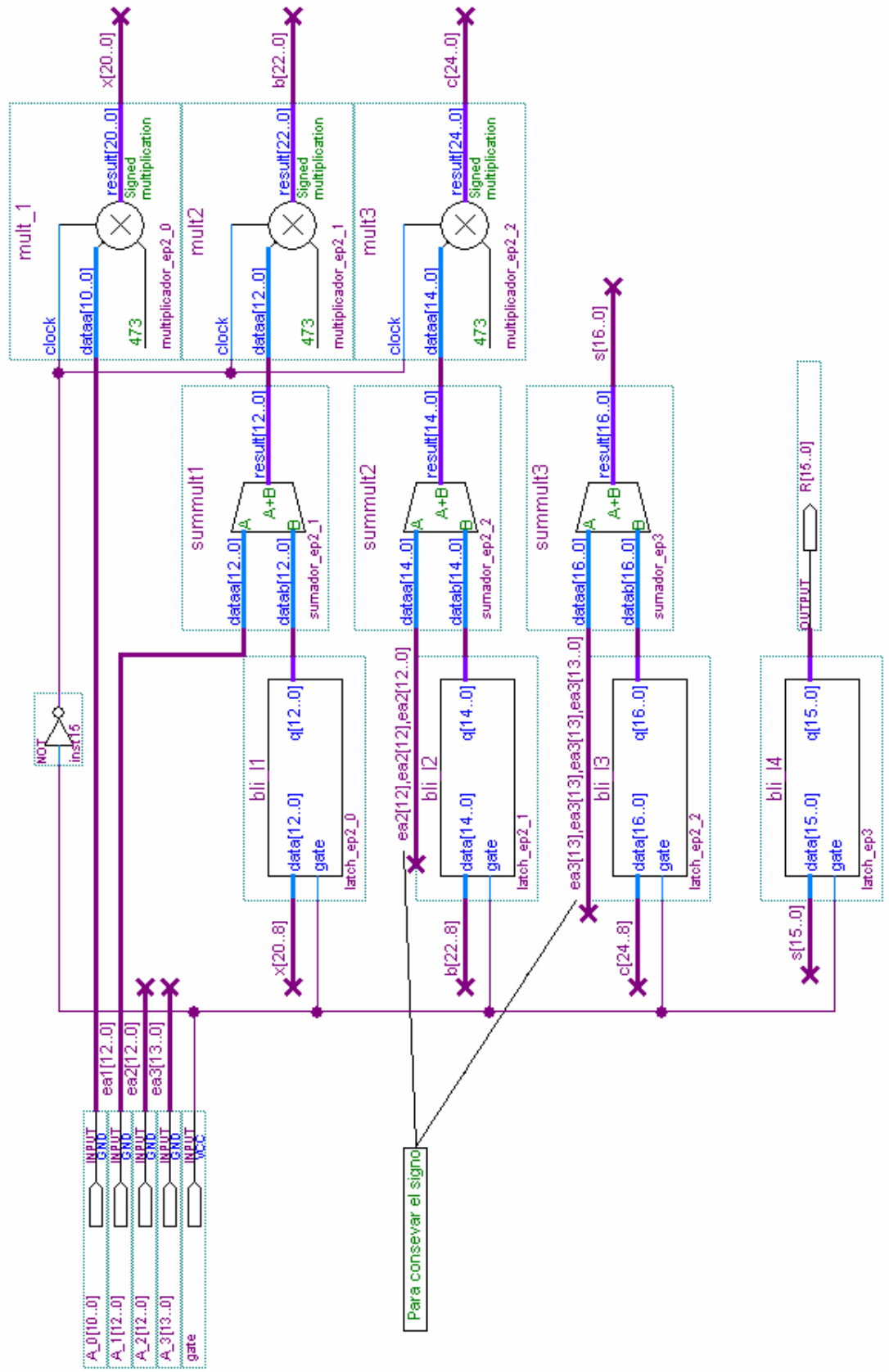
A continuación se mostrarán tanto los códigos fuente como los esquemas del hardware que componen el módulo TDH, empezando por el esquema general del módulo y siguiendo con el contenido de la ROM, el esquema del bloque EP2/3, el código fuente del bloque Matriz EP1, los elementos de proceso que componen Matriz EP1 y; por último, otros módulos de menor importancia mostrados en el esquema general del Módulo TDH.

Para el resto de módulos, al tratarse de módulos prefabricados, no se ha visto adecuado el incluir aquí sus códigos fuente.

H.2 El módulo TDH



H.2.1 El bloque EP2/3 (blinferior)



H.2.2 El contenido de la ROM (rom_dht.mif)

```
--
-- Este es el contenido de la ROM que gobierna el sistema sistólico
-- La distribución de los elementos de 'bloque_ep1.vhd' está íntimamente
-- relacionada con este código.
--
DEPTH=16;

WIDTH=104;

ADDRESS_RADIX = HEX;
DATA_RADIX = HEX;



CONTENT BEGIN
0:02816088B2984429122A8B2084;
1:24AD494929525A4AD494929525;
2:38C881A45054618CA018C14546;
3:24A5496929525A4A5496929525;
4:209165003209220812530329 9A;
5:24A549492B525A4A549492B525;
6:160A4D24A461437084D0CA0414;
7:24AD49692B525A4AD49692B525;
8:46C13209A28C606D17029A21C6;
9:24AD494929525A4AD494929525;
A:1088112E00504308A910644504;
B:24A5496929525A4A5496929525;
C:28D12D92221B828C16DA2228B0;
D:24A549492B525A4A549492B525;
E:334A45ADB443612484585B0636;
F:24AD49692B525A4AD49692B525;
END;
```

H.2.3 Matriz EP1 (bloque_ep1.vhd)

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;
USE ieee.std_logic_signed.all;

ENTITY bloque_ep1 IS
        PORT    (
                clk             : IN  STD_LOGIC;
                reset           : IN  STD_LOGIC;
                calcula         : IN  STD_LOGIC;
                dato_in_0       : IN  INTEGER RANGE 0 to 255;
                dato_in_1       : IN  INTEGER RANGE 0 to 255;
                dato_in_2       : IN  INTEGER RANGE 0 to 255;
                dato_in_3       : IN  INTEGER RANGE 0 to 255;
                dato_in_4       : IN  INTEGER RANGE 0 to 255;
                dato_in_5       : IN  INTEGER RANGE 0 to 255;
                dato_in_6       : IN  INTEGER RANGE 0 to 255;
                dato_in_7       : IN  INTEGER RANGE 0 to 255;
                dato_in_8       : IN  INTEGER RANGE 0 to 255;
                dato_in_9       : IN  INTEGER RANGE 0 to 255;
                dato_in_A       : IN  INTEGER RANGE 0 to 255;
                dato_in_B       : IN  INTEGER RANGE 0 to 255;
                dato_in_C       : IN  INTEGER RANGE 0 to 255;
                dato_in_D       : IN  INTEGER RANGE 0 to 255;
                dato_in_E       : IN  INTEGER RANGE 0 to 255;
                dato_in_F       : IN  INTEGER RANGE 0 to 255;
                bus_rom         : IN  STD_LOGIC_VECTOR(102 DOWNTO 0);
                sout_0          : OUT INTEGER RANGE -2040 TO 2040;
                sout_1          : OUT INTEGER RANGE -6120 TO 6120;
                sout_2          : OUT INTEGER RANGE -8160 TO 8160;
                sout_3          : OUT INTEGER RANGE -14280 TO 14280;
                hecho           : OUT STD_LOGIC);
```

```
END bloque_ep1;

ARCHITECTURE sistolico of bloque_ep1 is

-- Definición de los componentes
-- Paciencia

component ep1_nop1_0_pf
        GENERIC (
                V_RANGO_OUT_MIN_Y,V_RANGO_OUT_MAX_Y  : INTEGER;
                V_RANGO_MIN_X, V_RANGO_MAX_X         : INTEGER);
        PORT (
                clk     : IN  STD_LOGIC;
                reset   : IN  STD_LOGIC;
                calcula : IN  STD_LOGIC;
                xin     : IN  INTEGER RANGE (V_RANGO_MIN_X) to (V_RANGO_MAX_X);
                xout    : OUT INTEGER RANGE (V_RANGO_MIN_X) to (V_RANGO_MAX_X);
                sout    : OUT INTEGER RANGE (V_RANGO_OUT_MIN_Y) to (V_RANGO_OUT_MAX_Y);
                hecho   : OUT STD_LOGIC);
end component ep1_nop1_0_pf;

component ep1_nop1_0
        GENERIC (
                V_RANGO_IN_MIN_Y,V_RANGO_IN_MAX_Y    : INTEGER;
                V_RANGO_OUT_MIN_Y,V_RANGO_OUT_MAX_Y  : INTEGER;
                V_RANGO_MIN_X, V_RANGO_MAX_X         : INTEGER);
        PORT (
                clk     : IN  STD_LOGIC;
                reset   : IN  STD_LOGIC;
                calcula : IN  STD_LOGIC;
                xin     : IN  INTEGER RANGE (V_RANGO_MIN_X) to (V_RANGO_MAX_X);
                sin     : IN  INTEGER RANGE (V_RANGO_IN_MIN_Y) to (V_RANGO_IN_MAX_Y);
                xout    : OUT INTEGER RANGE (V_RANGO_MIN_X) to (V_RANGO_MAX_X);
                sout    : OUT INTEGER RANGE (V_RANGO_OUT_MIN_Y) to (V_RANGO_OUT_MAX_Y);
                hecho   : OUT STD_LOGIC);
end component ep1_nop1_0;

component ep1_nop1_2_pf
        GENERIC (
                V_RANGO_OUT_MIN_Y,V_RANGO_OUT_MAX_Y  : INTEGER;
                V_RANGO_MIN_X, V_RANGO_MAX_X         : INTEGER);
        PORT    (
                clk     : IN  STD_LOGIC;
                reset   : IN  STD_LOGIC;
                calcula : IN  STD_LOGIC;
                xin     : IN  INTEGER RANGE (V_RANGO_MIN_X) to (V_RANGO_MAX_X);
                xout    : OUT INTEGER RANGE (V_RANGO_MIN_X) to (V_RANGO_MAX_X);
                sout    : OUT INTEGER RANGE (V_RANGO_OUT_MIN_Y) to (V_RANGO_OUT_MAX_Y);
                hecho   : OUT STD_LOGIC);
end component ep1_nop1_2_pf;

component ep1_nop1_2
        GENERIC (
                V_RANGO_IN_MIN_Y,V_RANGO_IN_MAX_Y    : INTEGER;
                V_RANGO_OUT_MIN_Y,V_RANGO_OUT_MAX_Y  : INTEGER;
                V_RANGO_MIN_X, V_RANGO_MAX_X         : INTEGER);
        PORT    (
                clk     : IN  STD_LOGIC;
                reset   : IN  STD_LOGIC;
                calcula : IN  STD_LOGIC;
                xin     : IN  INTEGER RANGE (V_RANGO_MIN_X) to (V_RANGO_MAX_X);
                sin     : IN  INTEGER RANGE (V_RANGO_IN_MIN_Y) to (V_RANGO_IN_MAX_Y);
                xout    : OUT INTEGER RANGE (V_RANGO_MIN_X) to (V_RANGO_MAX_X);
                sout    : OUT INTEGER RANGE (V_RANGO_OUT_MIN_Y) to (V_RANGO_OUT_MAX_Y);
                hecho   : OUT STD_LOGIC);
end component ep1_nop1_2;

component ep1_nop2_2
        GENERIC (
                V_RANGO_IN_MIN_Y,V_RANGO_IN_MAX_Y    : INTEGER;
                V_RANGO_OUT_MIN_Y,V_RANGO_OUT_MAX_Y  : INTEGER;
                V_RANGO_MIN_X, V_RANGO_MAX_X         : INTEGER);
        PORT    (
                clk     : IN  STD_LOGIC;
                reset   : IN  STD_LOGIC;
                calcula : IN  STD_LOGIC;
                operacion      : IN  STD_LOGIC;
```

```
                xin     : IN  INTEGER RANGE (V_RANGO_MIN_X) to (V_RANGO_MAX_X);
                sin     : IN  INTEGER RANGE (V_RANGO_IN_MIN_Y) to (V_RANGO_IN_MAX_Y);
                xout    : OUT INTEGER RANGE (V_RANGO_MIN_X) to (V_RANGO_MAX_X);
                sout    : OUT INTEGER RANGE (V_RANGO_OUT_MIN_Y) to (V_RANGO_OUT_MAX_Y);
                hecho   : OUT STD_LOGIC);
end component ep1_nop2_2;

component ep1_nop3_1
        GENERIC (
                V_RANGO_IN_MIN_Y,V_RANGO_IN_MAX_Y    : INTEGER;
                V_RANGO_OUT_MIN_Y,V_RANGO_OUT_MAX_Y  : INTEGER;
                V_RANGO_MIN_X, V_RANGO_MAX_X         : INTEGER);
        PORT    (
                clk     : IN  STD_LOGIC;
                reset   : IN  STD_LOGIC;
                calcula : IN  STD_LOGIC;
                operacion      : IN  STD_LOGIC_VECTOR(1 DOWNTO 0);
                xin     : IN  INTEGER RANGE (V_RANGO_MIN_X) to (V_RANGO_MAX_X);
                sin     : IN  INTEGER RANGE (V_RANGO_IN_MIN_Y) to (V_RANGO_IN_MAX_Y);
                xout    : OUT INTEGER RANGE (V_RANGO_MIN_X) to (V_RANGO_MAX_X);
                sout    : OUT INTEGER RANGE (V_RANGO_OUT_MIN_Y) to (V_RANGO_OUT_MAX_Y);
                hecho   : OUT STD_LOGIC);
end component ep1_nop3_1;

component ep1_nop3_2
        GENERIC (
                V_RANGO_IN_MIN_Y,V_RANGO_IN_MAX_Y    : INTEGER;
                V_RANGO_OUT_MIN_Y,V_RANGO_OUT_MAX_Y  : INTEGER;
                V_RANGO_MIN_X, V_RANGO_MAX_X         : INTEGER);
        PORT    (
                clk     : IN  STD_LOGIC;
                reset   : IN  STD_LOGIC;
                calcula : IN  STD_LOGIC;
                operacion      : IN  STD_LOGIC_VECTOR(1 DOWNTO 0);
                xin     : IN  INTEGER RANGE (V_RANGO_MIN_X) to (V_RANGO_MAX_X);
                sin     : IN  INTEGER RANGE (V_RANGO_IN_MIN_Y) to (V_RANGO_IN_MAX_Y);
                xout    : OUT INTEGER RANGE (V_RANGO_MIN_X) to (V_RANGO_MAX_X);
                sout    : OUT INTEGER RANGE (V_RANGO_OUT_MIN_Y) to (V_RANGO_OUT_MAX_Y);
                hecho   : OUT STD_LOGIC);
end component ep1_nop3_2;

component ep1_nop5_4
        GENERIC (
                V_RANGO_IN_MIN_Y,V_RANGO_IN_MAX_Y    : INTEGER;
                V_RANGO_OUT_MIN_Y,V_RANGO_OUT_MAX_Y  : INTEGER;
                V_RANGO_MIN_X, V_RANGO_MAX_X         : INTEGER);
        PORT    (
                clk     : IN  STD_LOGIC;
                reset   : IN  STD_LOGIC;
                calcula : IN  STD_LOGIC;
                operacion      : IN  STD_LOGIC_VECTOR(2 DOWNTO 0);
                xin     : IN  INTEGER RANGE (V_RANGO_MIN_X) to (V_RANGO_MAX_X);
                sin     : IN  INTEGER RANGE (V_RANGO_IN_MIN_Y) to (V_RANGO_IN_MAX_Y);
                xout    : OUT INTEGER RANGE (V_RANGO_MIN_X) to (V_RANGO_MAX_X);
                sout    : OUT INTEGER RANGE (V_RANGO_OUT_MIN_Y) to (V_RANGO_OUT_MAX_Y);
                hecho   : OUT STD_LOGIC);
end component ep1_nop5_4;

-- Señales de interconexión de los componentes

SIGNAL hecho_0_0: STD_LOGIC; SIGNAL sout_0_0 :INTEGER RANGE 0 TO 0;
SIGNAL hecho_0_1: STD_LOGIC; SIGNAL sout_0_1 :INTEGER RANGE 0 TO 0;
SIGNAL hecho_0_2: STD_LOGIC; SIGNAL sout_0_2 :INTEGER RANGE 0 TO 0;
SIGNAL hecho_0_3: STD_LOGIC; SIGNAL sout_0_3 :INTEGER RANGE -510 TO 510;
SIGNAL hecho_1_0: STD_LOGIC; SIGNAL sout_1_0 :INTEGER RANGE -255 TO 255;
SIGNAL hecho_1_1: STD_LOGIC; SIGNAL sout_1_1 :INTEGER RANGE -510 TO 510;
SIGNAL hecho_1_2: STD_LOGIC; SIGNAL sout_1_2 :INTEGER RANGE -1020 TO 1020;
SIGNAL hecho_1_3: STD_LOGIC; SIGNAL sout_1_3 :INTEGER RANGE -1530 TO 1530;
SIGNAL hecho_2_0: STD_LOGIC; SIGNAL sout_2_0 :INTEGER RANGE -255 TO 255;
SIGNAL hecho_2_1: STD_LOGIC; SIGNAL sout_2_1 :INTEGER RANGE -1020 TO 1020;
SIGNAL hecho_2_2: STD_LOGIC; SIGNAL sout_2_2 :INTEGER RANGE -1020 TO 1020;
SIGNAL hecho_2_3: STD_LOGIC; SIGNAL sout_2_3 :INTEGER RANGE -2550 TO 2550;
SIGNAL hecho_3_0: STD_LOGIC; SIGNAL sout_3_0 :INTEGER RANGE -510 TO 510;
SIGNAL hecho_3_1: STD_LOGIC; SIGNAL sout_3_1 :INTEGER RANGE -1530 TO 1530;
SIGNAL hecho_3_2: STD_LOGIC; SIGNAL sout_3_2 :INTEGER RANGE -2040 TO 2040;
SIGNAL hecho_3_3: STD_LOGIC; SIGNAL sout_3_3 :INTEGER RANGE -3570 TO 3570;
SIGNAL hecho_4_0: STD_LOGIC; SIGNAL sout_4_0 :INTEGER RANGE -510 TO 510;
```

```
SIGNAL hecho_4_1: STD_LOGIC; SIGNAL sout_4_1 :INTEGER RANGE -1530 TO 1530;
SIGNAL hecho_4_2: STD_LOGIC; SIGNAL sout_4_2 :INTEGER RANGE -2040 TO 2040;
SIGNAL hecho_4_3: STD_LOGIC; SIGNAL sout_4_3 :INTEGER RANGE -4080 TO 4080;
SIGNAL hecho_5_0: STD_LOGIC; SIGNAL sout_5_0 :INTEGER RANGE -765 TO 765;
SIGNAL hecho_5_1: STD_LOGIC; SIGNAL sout_5_1 :INTEGER RANGE -2040 TO 2040;
SIGNAL hecho_5_2: STD_LOGIC; SIGNAL sout_5_2 :INTEGER RANGE -3060 TO 3060;
SIGNAL hecho_5_3: STD_LOGIC; SIGNAL sout_5_3 :INTEGER RANGE -5100 TO 5100;
SIGNAL hecho_6_0: STD_LOGIC; SIGNAL sout_6_0 :INTEGER RANGE -765 TO 765;
SIGNAL hecho_6_1: STD_LOGIC; SIGNAL sout_6_1 :INTEGER RANGE -2550 TO 2550;
SIGNAL hecho_6_2: STD_LOGIC; SIGNAL sout_6_2 :INTEGER RANGE -3060 TO 3060;
SIGNAL hecho_6_3: STD_LOGIC; SIGNAL sout_6_3 :INTEGER RANGE -6120 TO 6120;
SIGNAL hecho_7_0: STD_LOGIC; SIGNAL sout_7_0 :INTEGER RANGE -1020 TO 1020;
SIGNAL hecho_7_1: STD_LOGIC; SIGNAL sout_7_1 :INTEGER RANGE -3060 TO 3060;
SIGNAL hecho_7_2: STD_LOGIC; SIGNAL sout_7_2 :INTEGER RANGE -4080 TO 4080;
SIGNAL hecho_7_3: STD_LOGIC; SIGNAL sout_7_3 :INTEGER RANGE -7140 TO 7140;
SIGNAL hecho_8_0: STD_LOGIC; SIGNAL sout_8_0 :INTEGER RANGE -1020 TO 1020;
SIGNAL hecho_8_1: STD_LOGIC; SIGNAL sout_8_1 :INTEGER RANGE -3060 TO 3060;
SIGNAL hecho_8_2: STD_LOGIC; SIGNAL sout_8_2 :INTEGER RANGE -4080 TO 4080;
SIGNAL hecho_8_3: STD_LOGIC; SIGNAL sout_8_3 :INTEGER RANGE -7650 TO 7650;
SIGNAL hecho_9_0: STD_LOGIC; SIGNAL sout_9_0 :INTEGER RANGE -1275 TO 1275;
SIGNAL hecho_9_1: STD_LOGIC; SIGNAL sout_9_1 :INTEGER RANGE -3570 TO 3570;
SIGNAL hecho_9_2: STD_LOGIC; SIGNAL sout_9_2 :INTEGER RANGE -5100 TO 5100;
SIGNAL hecho_9_3: STD_LOGIC; SIGNAL sout_9_3 :INTEGER RANGE -8670 TO 8670;
SIGNAL hecho_10_0: STD_LOGIC; SIGNAL sout_10_0 :INTEGER RANGE -1275 TO 1275;
SIGNAL hecho_10_1: STD_LOGIC; SIGNAL sout_10_1 :INTEGER RANGE -4080 TO 4080;
SIGNAL hecho_10_2: STD_LOGIC; SIGNAL sout_10_2 :INTEGER RANGE -5100 TO 5100;
SIGNAL hecho_10_3: STD_LOGIC; SIGNAL sout_10_3 :INTEGER RANGE -9690 TO 9690;
SIGNAL hecho_11_0: STD_LOGIC; SIGNAL sout_11_0 :INTEGER RANGE -1530 TO 1530;
SIGNAL hecho_11_1: STD_LOGIC; SIGNAL sout_11_1 :INTEGER RANGE -4590 TO 4590;
SIGNAL hecho_11_2: STD_LOGIC; SIGNAL sout_11_2 :INTEGER RANGE -6120 TO 6120;
SIGNAL hecho_11_3: STD_LOGIC; SIGNAL sout_11_3 :INTEGER RANGE -10710 TO 10710;
SIGNAL hecho_12_0: STD_LOGIC; SIGNAL sout_12_0 :INTEGER RANGE -1530 TO 1530;
SIGNAL hecho_12_1: STD_LOGIC; SIGNAL sout_12_1 :INTEGER RANGE -4590 TO 4590;
SIGNAL hecho_12_2: STD_LOGIC; SIGNAL sout_12_2 :INTEGER RANGE -6120 TO 6120;
SIGNAL hecho_12_3: STD_LOGIC; SIGNAL sout_12_3 :INTEGER RANGE -11220 TO 11220;
SIGNAL hecho_13_0: STD_LOGIC; SIGNAL sout_13_0 :INTEGER RANGE -1785 TO 1785;
SIGNAL hecho_13_1: STD_LOGIC; SIGNAL sout_13_1 :INTEGER RANGE -5100 TO 5100;
SIGNAL hecho_13_2: STD_LOGIC; SIGNAL sout_13_2 :INTEGER RANGE -7140 TO 7140;
SIGNAL hecho_13_3: STD_LOGIC; SIGNAL sout_13_3 :INTEGER RANGE -12240 TO 12240;
SIGNAL hecho_14_0: STD_LOGIC; SIGNAL sout_14_0 :INTEGER RANGE -1785 TO 1785;
SIGNAL hecho_14_1: STD_LOGIC; SIGNAL sout_14_1 :INTEGER RANGE -5610 TO 5610;
SIGNAL hecho_14_2: STD_LOGIC; SIGNAL sout_14_2 :INTEGER RANGE -7140 TO 7140;
SIGNAL hecho_14_3: STD_LOGIC; SIGNAL sout_14_3 :INTEGER RANGE -13260 TO 13260;
SIGNAL hecho_15_0: STD_LOGIC; SIGNAL sout_15_0 :INTEGER RANGE -2040 TO 2040;
SIGNAL hecho_15_1: STD_LOGIC; SIGNAL sout_15_1 :INTEGER RANGE -6120 TO 6120;
SIGNAL hecho_15_2: STD_LOGIC; SIGNAL sout_15_2 :INTEGER RANGE -8160 TO 8160;
SIGNAL hecho_15_3: STD_LOGIC; SIGNAL sout_15_3 :INTEGER RANGE -14280 TO 14280;

SIGNAL x_in_0_0: INTEGER RANGE 0 TO 255;
SIGNAL x_in_0_1: INTEGER RANGE 0 TO 255;
SIGNAL x_in_0_2: INTEGER RANGE 0 TO 255;
SIGNAL x_in_1_0: INTEGER RANGE 0 TO 255;
SIGNAL x_in_1_1: INTEGER RANGE 0 TO 255;
SIGNAL x_in_1_2: INTEGER RANGE 0 TO 255;
SIGNAL x_in_2_0: INTEGER RANGE 0 TO 255;
SIGNAL x_in_2_1: INTEGER RANGE 0 TO 255;
SIGNAL x_in_2_2: INTEGER RANGE 0 TO 255;
SIGNAL x_in_3_0: INTEGER RANGE 0 TO 255;
SIGNAL x_in_3_1: INTEGER RANGE 0 TO 255;
SIGNAL x_in_3_2: INTEGER RANGE 0 TO 255;
SIGNAL x_in_4_0: INTEGER RANGE 0 TO 255;
SIGNAL x_in_4_1: INTEGER RANGE 0 TO 255;
SIGNAL x_in_4_2: INTEGER RANGE 0 TO 255;
SIGNAL x_in_5_0: INTEGER RANGE 0 TO 255;
SIGNAL x_in_5_1: INTEGER RANGE 0 TO 255;
SIGNAL x_in_5_2: INTEGER RANGE 0 TO 255;
SIGNAL x_in_6_0: INTEGER RANGE 0 TO 255;
SIGNAL x_in_6_1: INTEGER RANGE 0 TO 255;
SIGNAL x_in_6_2: INTEGER RANGE 0 TO 255;
SIGNAL x_in_7_0: INTEGER RANGE 0 TO 255;
SIGNAL x_in_7_1: INTEGER RANGE 0 TO 255;
SIGNAL x_in_7_2: INTEGER RANGE 0 TO 255;
SIGNAL x_in_8_0: INTEGER RANGE 0 TO 255;
SIGNAL x_in_8_1: INTEGER RANGE 0 TO 255;
SIGNAL x_in_8_2: INTEGER RANGE 0 TO 255;
SIGNAL x_in_9_0: INTEGER RANGE 0 TO 255;
SIGNAL x_in_9_1: INTEGER RANGE 0 TO 255;
```

```
SIGNAL x_in_9_2: INTEGER RANGE 0 TO 255;
SIGNAL x_in_10_0: INTEGER RANGE 0 TO 255;
SIGNAL x_in_10_1: INTEGER RANGE 0 TO 255;
SIGNAL x_in_10_2: INTEGER RANGE 0 TO 255;
SIGNAL x_in_11_0: INTEGER RANGE 0 TO 255;
SIGNAL x_in_11_1: INTEGER RANGE 0 TO 255;
SIGNAL x_in_11_2: INTEGER RANGE 0 TO 255;
SIGNAL x_in_12_0: INTEGER RANGE 0 TO 255;
SIGNAL x_in_12_1: INTEGER RANGE 0 TO 255;
SIGNAL x_in_12_2: INTEGER RANGE 0 TO 255;
SIGNAL x_in_13_0: INTEGER RANGE 0 TO 255;
SIGNAL x_in_13_1: INTEGER RANGE 0 TO 255;
SIGNAL x_in_13_2: INTEGER RANGE 0 TO 255;
SIGNAL x_in_14_0: INTEGER RANGE 0 TO 255;
SIGNAL x_in_14_1: INTEGER RANGE 0 TO 255;
SIGNAL x_in_14_2: INTEGER RANGE 0 TO 255;
SIGNAL x_in_15_0: INTEGER RANGE 0 TO 255;
SIGNAL x_in_15_1: INTEGER RANGE 0 TO 255;
SIGNAL x_in_15_2: INTEGER RANGE 0 TO 255;

BEGIN

-- Fila     0-------vv--------vv----
f_0_c_0: ep1_nop1_0_pf
        GENERIC MAP (V_RANGO_MIN_X => 0, V_RANGO_MAX_X => 255, V_RANGO_OUT_MIN_Y => 0,
V_RANGO_OUT_MAX_Y => 0)
        PORT MAP (clk => clk, reset => reset, calcula => calcula, xin => dato_in_0, xout =>
x_in_0_0, sout => sout_0_0, hecho=> hecho_0_0);

f_0_c_1: ep1_nop1_0_pf
        GENERIC MAP (V_RANGO_MIN_X => 0, V_RANGO_MAX_X => 255, V_RANGO_OUT_MIN_Y => 0,
V_RANGO_OUT_MAX_Y => 0)
        PORT MAP (clk => clk, reset => reset, calcula => calcula, xin => x_in_0_0, xout =>
x_in_0_1, sout => sout_0_1, hecho=> hecho_0_1);

f_0_c_2: ep1_nop1_0_pf
        GENERIC MAP (V_RANGO_MIN_X => 0, V_RANGO_MAX_X => 255, V_RANGO_OUT_MIN_Y => 0,
V_RANGO_OUT_MAX_Y => 0)
        PORT MAP (clk => clk, reset => reset, calcula => calcula, xin => x_in_0_1, xout =>
x_in_0_2, sout => sout_0_2, hecho=> hecho_0_2);

f_0_c_3: ep1_nop1_2_pf
        GENERIC MAP (V_RANGO_MIN_X => 0, V_RANGO_MAX_X => 255, V_RANGO_OUT_MIN_Y => -510,
V_RANGO_OUT_MAX_Y => 510)
       PORT MAP (clk => clk, reset => reset, calcula => calcula, xin => x_in_0_2, xout => OPEN,
sout => sout_0_3, hecho=> hecho_0_3);

-- Fila     1-------vv--------vv----
f_1_c_0: ep1_nop3_1
        GENERIC MAP (V_RANGO_MIN_X => 0, V_RANGO_MAX_X => 255, V_RANGO_IN_MIN_Y => 0,
V_RANGO_IN_MAX_Y => 0, V_RANGO_OUT_MIN_Y => -255, V_RANGO_OUT_MAX_Y => 255)
        PORT MAP (clk => clk, reset => reset, calcula => calcula, operacion => bus_rom(1 DOWNTO
0), xin => dato_in_1, xout => x_in_1_0, sin => sout_0_0, sout => sout_1_0, hecho=> hecho_1_0);

f_1_c_1: ep1_nop3_2
        GENERIC MAP (V_RANGO_MIN_X => 0, V_RANGO_MAX_X => 255, V_RANGO_IN_MIN_Y => 0,
V_RANGO_IN_MAX_Y => 0, V_RANGO_OUT_MIN_Y => -510, V_RANGO_OUT_MAX_Y => 510)
        PORT MAP (clk => clk, reset => reset, calcula => calcula, operacion => bus_rom(3 DOWNTO
2), xin => x_in_1_0, xout => x_in_1_1, sin => sout_0_1, sout => sout_1_1, hecho=> hecho_1_1);

f_1_c_2: ep1_nop5_4
        GENERIC MAP (V_RANGO_MIN_X => 0, V_RANGO_MAX_X => 255, V_RANGO_IN_MIN_Y => 0,
V_RANGO_IN_MAX_Y => 0, V_RANGO_OUT_MIN_Y => -1020, V_RANGO_OUT_MAX_Y => 1020)
        PORT MAP (clk => clk, reset => reset, calcula => calcula, operacion => bus_rom(6 DOWNTO
4), xin => x_in_1_1, xout => x_in_1_2, sin => sout_0_2, sout => sout_1_2, hecho=> hecho_1_2);

f_1_c_3: ep1_nop5_4
        GENERIC MAP (V_RANGO_MIN_X => 0, V_RANGO_MAX_X => 255, V_RANGO_IN_MIN_Y => -510,
V_RANGO_IN_MAX_Y => 510, V_RANGO_OUT_MIN_Y => -1530, V_RANGO_OUT_MAX_Y => 1530)
        PORT MAP (clk => clk, reset => reset, calcula => calcula, operacion => bus_rom(9 DOWNTO
7), xin => x_in_1_2, xout => OPEN, sin => sout_0_3, sout => sout_1_3, hecho=> hecho_1_3);

-- Fila     2-------vv--------vv----
f_2_c_0: ep1_nop1_0
        GENERIC MAP (V_RANGO_MIN_X => 0, V_RANGO_MAX_X => 255, V_RANGO_IN_MIN_Y => -255,
V_RANGO_IN_MAX_Y => 255, V_RANGO_OUT_MIN_Y => -255, V_RANGO_OUT_MAX_Y => 255)
```

```
        PORT MAP (clk => clk, reset => reset, calcula => calcula, xin => dato_in_2, xout =>
x_in_2_0, sin => sout_1_0, sout => sout_2_0, hecho=> hecho_2_0);

f_2_c_1: ep1_nop3_2
        GENERIC MAP (V_RANGO_MIN_X => 0, V_RANGO_MAX_X => 255, V_RANGO_IN_MIN_Y => -510,
V_RANGO_IN_MAX_Y => 510, V_RANGO_OUT_MIN_Y => -1020, V_RANGO_OUT_MAX_Y => 1020)
        PORT MAP (clk => clk, reset => reset, calcula => calcula, operacion => bus_rom(11
DOWNTO 10), xin => x_in_2_0, xout => x_in_2_1, sin => sout_1_1, sout => sout_2_1, hecho=>
hecho_2_1);

f_2_c_2: ep1_nop1_0
        GENERIC MAP (V_RANGO_MIN_X => 0, V_RANGO_MAX_X => 255, V_RANGO_IN_MIN_Y => -1020,
V_RANGO_IN_MAX_Y => 1020, V_RANGO_OUT_MIN_Y => -1020, V_RANGO_OUT_MAX_Y => 1020)
        PORT MAP (clk => clk, reset => reset, calcula => calcula, xin => x_in_2_1, xout =>
x_in_2_2, sin => sout_1_2, sout => sout_2_2, hecho=> hecho_2_2);

f_2_c_3: ep1_nop5_4
        GENERIC MAP (V_RANGO_MIN_X => 0, V_RANGO_MAX_X => 255, V_RANGO_IN_MIN_Y => -1530,
V_RANGO_IN_MAX_Y => 1530, V_RANGO_OUT_MIN_Y => -2550, V_RANGO_OUT_MAX_Y => 2550)
        PORT MAP (clk => clk, reset => reset, calcula => calcula, operacion => bus_rom(14
DOWNTO 12), xin => x_in_2_2, xout => OPEN, sin => sout_1_3, sout => sout_2_3, hecho=>
hecho_2_3);

-- Fila     3-------vv--------vv----
f_3_c_0: ep1_nop3_1
        GENERIC MAP (V_RANGO_MIN_X => 0, V_RANGO_MAX_X => 255, V_RANGO_IN_MIN_Y => -255,
V_RANGO_IN_MAX_Y => 255, V_RANGO_OUT_MIN_Y => -510, V_RANGO_OUT_MAX_Y => 510)
        PORT MAP (clk => clk, reset => reset, calcula => calcula, operacion => bus_rom(16
DOWNTO 15), xin => dato_in_3, xout => x_in_3_0, sin => sout_2_0, sout => sout_3_0, hecho=>
hecho_3_0);

f_3_c_1: ep1_nop3_2
        GENERIC MAP (V_RANGO_MIN_X => 0, V_RANGO_MAX_X => 255, V_RANGO_IN_MIN_Y => -1020,
V_RANGO_IN_MAX_Y => 1020, V_RANGO_OUT_MIN_Y => -1530, V_RANGO_OUT_MAX_Y => 1530)
        PORT MAP (clk => clk, reset => reset, calcula => calcula, operacion => bus_rom(18
DOWNTO 17), xin => x_in_3_0, xout => x_in_3_1, sin => sout_2_1, sout => sout_3_1, hecho=>
hecho_3_1);

f_3_c_2: ep1_nop5_4
        GENERIC MAP (V_RANGO_MIN_X => 0, V_RANGO_MAX_X => 255, V_RANGO_IN_MIN_Y => -1020,
V_RANGO_IN_MAX_Y => 1020, V_RANGO_OUT_MIN_Y => -2040, V_RANGO_OUT_MAX_Y => 2040)
        PORT MAP (clk => clk, reset => reset, calcula => calcula, operacion => bus_rom(21
DOWNTO 19), xin => x_in_3_1, xout => x_in_3_2, sin => sout_2_2, sout => sout_3_2, hecho=>
hecho_3_2);

f_3_c_3: ep1_nop5_4
        GENERIC MAP (V_RANGO_MIN_X => 0, V_RANGO_MAX_X => 255, V_RANGO_IN_MIN_Y => -2550,
V_RANGO_IN_MAX_Y => 2550, V_RANGO_OUT_MIN_Y => -3570, V_RANGO_OUT_MAX_Y => 3570)
        PORT MAP (clk => clk, reset => reset, calcula => calcula, operacion => bus_rom(24
DOWNTO 22), xin => x_in_3_2, xout => OPEN, sin => sout_2_3, sout => sout_3_3, hecho=>
hecho_3_3);

-- Fila     4-------vv--------vv----
f_4_c_0: ep1_nop1_0
        GENERIC MAP (V_RANGO_MIN_X => 0, V_RANGO_MAX_X => 255, V_RANGO_IN_MIN_Y => -510,
V_RANGO_IN_MAX_Y => 510, V_RANGO_OUT_MIN_Y => -510, V_RANGO_OUT_MAX_Y => 510)
        PORT MAP (clk => clk, reset => reset, calcula => calcula, xin => dato_in_4, xout =>
x_in_4_0, sin => sout_3_0, sout => sout_4_0, hecho=> hecho_4_0);

f_4_c_1: ep1_nop1_0
        GENERIC MAP (V_RANGO_MIN_X => 0, V_RANGO_MAX_X => 255, V_RANGO_IN_MIN_Y => -1530,
V_RANGO_IN_MAX_Y => 1530, V_RANGO_OUT_MIN_Y => -1530, V_RANGO_OUT_MAX_Y => 1530)
        PORT MAP (clk => clk, reset => reset, calcula => calcula, xin => x_in_4_0, xout =>
x_in_4_1, sin => sout_3_1, sout => sout_4_1, hecho=> hecho_4_1);

f_4_c_2: ep1_nop1_0
        GENERIC MAP (V_RANGO_MIN_X => 0, V_RANGO_MAX_X => 255, V_RANGO_IN_MIN_Y => -2040,
V_RANGO_IN_MAX_Y => 2040, V_RANGO_OUT_MIN_Y => -2040, V_RANGO_OUT_MAX_Y => 2040)
        PORT MAP (clk => clk, reset => reset, calcula => calcula, xin => x_in_4_1, xout =>
x_in_4_2, sin => sout_3_2, sout => sout_4_2, hecho=> hecho_4_2);

f_4_c_3: ep1_nop2_2
        GENERIC MAP (V_RANGO_MIN_X => 0, V_RANGO_MAX_X => 255, V_RANGO_IN_MIN_Y => -3570,
V_RANGO_IN_MAX_Y => 3570, V_RANGO_OUT_MIN_Y => -4080, V_RANGO_OUT_MAX_Y => 4080)
        PORT MAP (clk => clk, reset => reset, calcula => calcula, operacion => bus_rom(25), xin
=> x_in_4_2, xout => OPEN, sin => sout_3_3, sout => sout_4_3, hecho=> hecho_4_3);
```

```
-- Fila     5-------vv--------vv----
f_5_c_0: ep1_nop3_1
        GENERIC MAP (V_RANGO_MIN_X => 0, V_RANGO_MAX_X => 255, V_RANGO_IN_MIN_Y => -510,
V_RANGO_IN_MAX_Y => 510, V_RANGO_OUT_MIN_Y => -765, V_RANGO_OUT_MAX_Y => 765)
        PORT MAP (clk => clk, reset => reset, calcula => calcula, operacion => bus_rom(27
DOWNTO 26), xin => dato_in_5, xout => x_in_5_0, sin => sout_4_0, sout => sout_5_0, hecho=>
hecho_5_0);

f_5_c_1: ep1_nop3_2
        GENERIC MAP (V_RANGO_MIN_X => 0, V_RANGO_MAX_X => 255, V_RANGO_IN_MIN_Y => -1530,
V_RANGO_IN_MAX_Y => 1530, V_RANGO_OUT_MIN_Y => -2040, V_RANGO_OUT_MAX_Y => 2040)
        PORT MAP (clk => clk, reset => reset, calcula => calcula, operacion => bus_rom(29
DOWNTO 28), xin => x_in_5_0, xout => x_in_5_1, sin => sout_4_1, sout => sout_5_1, hecho=>
hecho_5_1);

f_5_c_2: ep1_nop5_4
        GENERIC MAP (V_RANGO_MIN_X => 0, V_RANGO_MAX_X => 255, V_RANGO_IN_MIN_Y => -2040,
V_RANGO_IN_MAX_Y => 2040, V_RANGO_OUT_MIN_Y => -3060, V_RANGO_OUT_MAX_Y => 3060)
        PORT MAP (clk => clk, reset => reset, calcula => calcula, operacion => bus_rom(32
DOWNTO 30), xin => x_in_5_1, xout => x_in_5_2, sin => sout_4_2, sout => sout_5_2, hecho=>
hecho_5_2);

f_5_c_3: ep1_nop5_4
        GENERIC MAP (V_RANGO_MIN_X => 0, V_RANGO_MAX_X => 255, V_RANGO_IN_MIN_Y => -4080,
V_RANGO_IN_MAX_Y => 4080, V_RANGO_OUT_MIN_Y => -5100, V_RANGO_OUT_MAX_Y => 5100)
        PORT MAP (clk => clk, reset => reset, calcula => calcula, operacion => bus_rom(35
DOWNTO 33), xin => x_in_5_2, xout => OPEN, sin => sout_4_3, sout => sout_5_3, hecho=>
hecho_5_3);

-- Fila     6-------vv--------vv----
f_6_c_0: ep1_nop1_0
        GENERIC MAP (V_RANGO_MIN_X => 0, V_RANGO_MAX_X => 255, V_RANGO_IN_MIN_Y => -765,
V_RANGO_IN_MAX_Y => 765, V_RANGO_OUT_MIN_Y => -765, V_RANGO_OUT_MAX_Y => 765)
        PORT MAP (clk => clk, reset => reset, calcula => calcula, xin => dato_in_6, xout =>
x_in_6_0, sin => sout_5_0, sout => sout_6_0, hecho=> hecho_6_0);

f_6_c_1: ep1_nop3_2
        GENERIC MAP (V_RANGO_MIN_X => 0, V_RANGO_MAX_X => 255, V_RANGO_IN_MIN_Y => -2040,
V_RANGO_IN_MAX_Y => 2040, V_RANGO_OUT_MIN_Y => -2550, V_RANGO_OUT_MAX_Y => 2550)
        PORT MAP (clk => clk, reset => reset, calcula => calcula, operacion => bus_rom(37
DOWNTO 36), xin => x_in_6_0, xout => x_in_6_1, sin => sout_5_1, sout => sout_6_1, hecho=>
hecho_6_1);

f_6_c_2: ep1_nop1_0
        GENERIC MAP (V_RANGO_MIN_X => 0, V_RANGO_MAX_X => 255, V_RANGO_IN_MIN_Y => -3060,
V_RANGO_IN_MAX_Y => 3060, V_RANGO_OUT_MIN_Y => -3060, V_RANGO_OUT_MAX_Y => 3060)
        PORT MAP (clk => clk, reset => reset, calcula => calcula, xin => x_in_6_1, xout =>
x_in_6_2, sin => sout_5_2, sout => sout_6_2, hecho=> hecho_6_2);

f_6_c_3: ep1_nop5_4
        GENERIC MAP (V_RANGO_MIN_X => 0, V_RANGO_MAX_X => 255, V_RANGO_IN_MIN_Y => -5100,
V_RANGO_IN_MAX_Y => 5100, V_RANGO_OUT_MIN_Y => -6120, V_RANGO_OUT_MAX_Y => 6120)
        PORT MAP (clk => clk, reset => reset, calcula => calcula, operacion => bus_rom(40
DOWNTO 38), xin => x_in_6_2, xout => OPEN, sin => sout_5_3, sout => sout_6_3, hecho=>
hecho_6_3);

-- Fila     7-------vv--------vv----
f_7_c_0: ep1_nop3_1
        GENERIC MAP (V_RANGO_MIN_X => 0, V_RANGO_MAX_X => 255, V_RANGO_IN_MIN_Y => -765,
V_RANGO_IN_MAX_Y => 765, V_RANGO_OUT_MIN_Y => -1020, V_RANGO_OUT_MAX_Y => 1020)
        PORT MAP (clk => clk, reset => reset, calcula => calcula, operacion => bus_rom(42
DOWNTO 41), xin => dato_in_7, xout => x_in_7_0, sin => sout_6_0, sout => sout_7_0, hecho=>
hecho_7_0);

f_7_c_1: ep1_nop3_2
        GENERIC MAP (V_RANGO_MIN_X => 0, V_RANGO_MAX_X => 255, V_RANGO_IN_MIN_Y => -2550,
V_RANGO_IN_MAX_Y => 2550, V_RANGO_OUT_MIN_Y => -3060, V_RANGO_OUT_MAX_Y => 3060)
        PORT MAP (clk => clk, reset => reset, calcula => calcula, operacion => bus_rom(44
DOWNTO 43), xin => x_in_7_0, xout => x_in_7_1, sin => sout_6_1, sout => sout_7_1, hecho=>
hecho_7_1);

f_7_c_2: ep1_nop5_4
        GENERIC MAP (V_RANGO_MIN_X => 0, V_RANGO_MAX_X => 255, V_RANGO_IN_MIN_Y => -3060,
V_RANGO_IN_MAX_Y => 3060, V_RANGO_OUT_MIN_Y => -4080, V_RANGO_OUT_MAX_Y => 4080)
        PORT MAP (clk => clk, reset => reset, calcula => calcula, operacion => bus_rom(47
DOWNTO 45), xin => x_in_7_1, xout => x_in_7_2, sin => sout_6_2, sout => sout_7_2, hecho=>
hecho_7_2);
```

```
f_7_c_3: ep1_nop5_4
        GENERIC MAP (V_RANGO_MIN_X => 0, V_RANGO_MAX_X => 255, V_RANGO_IN_MIN_Y => -6120,
V_RANGO_IN_MAX_Y => 6120, V_RANGO_OUT_MIN_Y => -7140, V_RANGO_OUT_MAX_Y => 7140)
        PORT MAP (clk => clk, reset => reset, calcula => calcula, operacion => bus_rom(50
DOWNTO 48), xin => x_in_7_2, xout => OPEN, sin => sout_6_3, sout => sout_7_3, hecho=>
hecho_7_3);

-- Fila     8-------vv--------vv----
f_8_c_0: ep1_nop1_0
        GENERIC MAP (V_RANGO_MIN_X => 0, V_RANGO_MAX_X => 255, V_RANGO_IN_MIN_Y => -1020,
V_RANGO_IN_MAX_Y => 1020, V_RANGO_OUT_MIN_Y => -1020, V_RANGO_OUT_MAX_Y => 1020)
        PORT MAP (clk => clk, reset => reset, calcula => calcula, xin => dato_in_8, xout =>
x_in_8_0, sin => sout_7_0, sout => sout_8_0, hecho=> hecho_8_0);

f_8_c_1: ep1_nop1_0
        GENERIC MAP (V_RANGO_MIN_X => 0, V_RANGO_MAX_X => 255, V_RANGO_IN_MIN_Y => -3060,
V_RANGO_IN_MAX_Y => 3060, V_RANGO_OUT_MIN_Y => -3060, V_RANGO_OUT_MAX_Y => 3060)
        PORT MAP (clk => clk, reset => reset, calcula => calcula, xin => x_in_8_0, xout =>
x_in_8_1, sin => sout_7_1, sout => sout_8_1, hecho=> hecho_8_1);

f_8_c_2: ep1_nop1_0
        GENERIC MAP (V_RANGO_MIN_X => 0, V_RANGO_MAX_X => 255, V_RANGO_IN_MIN_Y => -4080,
V_RANGO_IN_MAX_Y => 4080, V_RANGO_OUT_MIN_Y => -4080, V_RANGO_OUT_MAX_Y => 4080)
        PORT MAP (clk => clk, reset => reset, calcula => calcula, xin => x_in_8_1, xout =>
x_in_8_2, sin => sout_7_2, sout => sout_8_2, hecho=> hecho_8_2);

f_8_c_3: ep1_nop2_2
        GENERIC MAP (V_RANGO_MIN_X => 0, V_RANGO_MAX_X => 255, V_RANGO_IN_MIN_Y => -7140,
V_RANGO_IN_MAX_Y => 7140, V_RANGO_OUT_MIN_Y => -7650, V_RANGO_OUT_MAX_Y => 7650)
        PORT MAP (clk => clk, reset => reset, calcula => calcula, operacion => bus_rom(51), xin
=> x_in_8_2, xout => OPEN, sin => sout_7_3, sout => sout_8_3, hecho=> hecho_8_3);


-- Fila     9-------vv--------vv----
f_9_c_0: ep1_nop3_1
        GENERIC MAP (V_RANGO_MIN_X => 0, V_RANGO_MAX_X => 255, V_RANGO_IN_MIN_Y => -1020,
V_RANGO_IN_MAX_Y => 1020, V_RANGO_OUT_MIN_Y => -1275, V_RANGO_OUT_MAX_Y => 1275)
        PORT MAP (clk => clk, reset => reset, calcula => calcula, operacion => bus_rom(53
DOWNTO 52), xin => dato_in_9, xout => x_in_9_0, sin => sout_8_0, sout => sout_9_0, hecho=>
hecho_9_0);

f_9_c_1: ep1_nop3_2
        GENERIC MAP (V_RANGO_MIN_X => 0, V_RANGO_MAX_X => 255, V_RANGO_IN_MIN_Y => -3060,
V_RANGO_IN_MAX_Y => 3060, V_RANGO_OUT_MIN_Y => -3570, V_RANGO_OUT_MAX_Y => 3570)
        PORT MAP (clk => clk, reset => reset, calcula => calcula, operacion => bus_rom(55
DOWNTO 54), xin => x_in_9_0, xout => x_in_9_1, sin => sout_8_1, sout => sout_9_1, hecho=>
hecho_9_1);

f_9_c_2: ep1_nop5_4
        GENERIC MAP (V_RANGO_MIN_X => 0, V_RANGO_MAX_X => 255, V_RANGO_IN_MIN_Y => -4080,
V_RANGO_IN_MAX_Y => 4080, V_RANGO_OUT_MIN_Y => -5100, V_RANGO_OUT_MAX_Y => 5100)
        PORT MAP (clk => clk, reset => reset, calcula => calcula, operacion => bus_rom(58
DOWNTO 56), xin => x_in_9_1, xout => x_in_9_2, sin => sout_8_2, sout => sout_9_2, hecho=>
hecho_9_2);

f_9_c_3: ep1_nop5_4
        GENERIC MAP (V_RANGO_MIN_X => 0, V_RANGO_MAX_X => 255, V_RANGO_IN_MIN_Y => -7650,
V_RANGO_IN_MAX_Y => 7650, V_RANGO_OUT_MIN_Y => -8670, V_RANGO_OUT_MAX_Y => 8670)
        PORT MAP (clk => clk, reset => reset, calcula => calcula, operacion => bus_rom(61
DOWNTO 59), xin => x_in_9_2, xout => OPEN, sin => sout_8_3, sout => sout_9_3, hecho=>
hecho_9_3);

-- Fila    10-------vv--------vv----
f_10_c_0: ep1_nop1_0
        GENERIC MAP (V_RANGO_MIN_X => 0, V_RANGO_MAX_X => 255, V_RANGO_IN_MIN_Y => -1275,
V_RANGO_IN_MAX_Y => 1275, V_RANGO_OUT_MIN_Y => -1275, V_RANGO_OUT_MAX_Y => 1275)
        PORT MAP (clk => clk, reset => reset, calcula => calcula, xin => dato_in_A, xout =>
x_in_10_0, sin => sout_9_0, sout => sout_10_0, hecho=> hecho_10_0);

f_10_c_1: ep1_nop3_2
        GENERIC MAP (V_RANGO_MIN_X => 0, V_RANGO_MAX_X => 255, V_RANGO_IN_MIN_Y => -3570,
V_RANGO_IN_MAX_Y => 3570, V_RANGO_OUT_MIN_Y => -4080, V_RANGO_OUT_MAX_Y => 4080)
        PORT MAP (clk => clk, reset => reset, calcula => calcula, operacion => bus_rom(63
DOWNTO 62), xin => x_in_10_0, xout => x_in_10_1, sin => sout_9_1, sout => sout_10_1, hecho=>
hecho_10_1);
```

```
f_10_c_2: ep1_nop1_0
        GENERIC MAP (V_RANGO_MIN_X => 0, V_RANGO_MAX_X => 255, V_RANGO_IN_MIN_Y => -5100,
V_RANGO_IN_MAX_Y => 5100, V_RANGO_OUT_MIN_Y => -5100, V_RANGO_OUT_MAX_Y => 5100)
        PORT MAP (clk => clk, reset => reset, calcula => calcula, xin => x_in_10_1, xout =>
x_in_10_2, sin => sout_9_2, sout => sout_10_2, hecho=> hecho_10_2);

f_10_c_3: ep1_nop5_4
        GENERIC MAP (V_RANGO_MIN_X => 0, V_RANGO_MAX_X => 255, V_RANGO_IN_MIN_Y => -8670,
V_RANGO_IN_MAX_Y => 8670, V_RANGO_OUT_MIN_Y => -9690, V_RANGO_OUT_MAX_Y => 9690)
        PORT MAP (clk => clk, reset => reset, calcula => calcula, operacion => bus_rom(66
DOWNTO 64), xin => x_in_10_2, xout => OPEN, sin => sout_9_3, sout => sout_10_3, hecho=>
hecho_10_3);

-- Fila    11-------vv--------vv----
f_11_c_0: ep1_nop3_1
        GENERIC MAP (V_RANGO_MIN_X => 0, V_RANGO_MAX_X => 255, V_RANGO_IN_MIN_Y => -1275,
V_RANGO_IN_MAX_Y => 1275, V_RANGO_OUT_MIN_Y => -1530, V_RANGO_OUT_MAX_Y => 1530)
        PORT MAP (clk => clk, reset => reset, calcula => calcula, operacion => bus_rom(68
DOWNTO 67), xin => dato_in_B, xout => x_in_11_0, sin => sout_10_0, sout => sout_11_0, hecho=>
hecho_11_0);

f_11_c_1: ep1_nop3_2
        GENERIC MAP (V_RANGO_MIN_X => 0, V_RANGO_MAX_X => 255, V_RANGO_IN_MIN_Y => -4080,
V_RANGO_IN_MAX_Y => 4080, V_RANGO_OUT_MIN_Y => -4590, V_RANGO_OUT_MAX_Y => 4590)
        PORT MAP (clk => clk, reset => reset, calcula => calcula, operacion => bus_rom(70
DOWNTO 69), xin => x_in_11_0, xout => x_in_11_1, sin => sout_10_1, sout => sout_11_1, hecho=>
hecho_11_1);

f_11_c_2: ep1_nop5_4
        GENERIC MAP (V_RANGO_MIN_X => 0, V_RANGO_MAX_X => 255, V_RANGO_IN_MIN_Y => -5100,
V_RANGO_IN_MAX_Y => 5100, V_RANGO_OUT_MIN_Y => -6120, V_RANGO_OUT_MAX_Y => 6120)
        PORT MAP (clk => clk, reset => reset, calcula => calcula, operacion => bus_rom(73
DOWNTO 71), xin => x_in_11_1, xout => x_in_11_2, sin => sout_10_2, sout => sout_11_2, hecho=>
hecho_11_2);

f_11_c_3: ep1_nop5_4
        GENERIC MAP (V_RANGO_MIN_X => 0, V_RANGO_MAX_X => 255, V_RANGO_IN_MIN_Y => -9690,
V_RANGO_IN_MAX_Y => 9690, V_RANGO_OUT_MIN_Y => -10710, V_RANGO_OUT_MAX_Y => 10710)
        PORT MAP (clk => clk, reset => reset, calcula => calcula, operacion => bus_rom(76
DOWNTO 74), xin => x_in_11_2, xout => OPEN, sin => sout_10_3, sout => sout_11_3, hecho=>
hecho_11_3);

-- Fila    12-------vv--------vv----
f_12_c_0: ep1_nop1_0
        GENERIC MAP (V_RANGO_MIN_X => 0, V_RANGO_MAX_X => 255, V_RANGO_IN_MIN_Y => -1530,
V_RANGO_IN_MAX_Y => 1530, V_RANGO_OUT_MIN_Y => -1530, V_RANGO_OUT_MAX_Y => 1530)
        PORT MAP (clk => clk, reset => reset, calcula => calcula, xin => dato_in_C, xout =>
x_in_12_0, sin => sout_11_0, sout => sout_12_0, hecho=> hecho_12_0);

f_12_c_1: ep1_nop1_0
        GENERIC MAP (V_RANGO_MIN_X => 0, V_RANGO_MAX_X => 255, V_RANGO_IN_MIN_Y => -4590,
V_RANGO_IN_MAX_Y => 4590, V_RANGO_OUT_MIN_Y => -4590, V_RANGO_OUT_MAX_Y => 4590)
        PORT MAP (clk => clk, reset => reset, calcula => calcula, xin => x_in_12_0, xout =>
x_in_12_1, sin => sout_11_1, sout => sout_12_1, hecho=> hecho_12_1);

f_12_c_2: ep1_nop1_0
        GENERIC MAP (V_RANGO_MIN_X => 0, V_RANGO_MAX_X => 255, V_RANGO_IN_MIN_Y => -6120,
V_RANGO_IN_MAX_Y => 6120, V_RANGO_OUT_MIN_Y => -6120, V_RANGO_OUT_MAX_Y => 6120)
        PORT MAP (clk => clk, reset => reset, calcula => calcula, xin => x_in_12_1, xout =>
x_in_12_2, sin => sout_11_2, sout => sout_12_2, hecho=> hecho_12_2);

f_12_c_3: ep1_nop2_2
        GENERIC MAP (V_RANGO_MIN_X => 0, V_RANGO_MAX_X => 255, V_RANGO_IN_MIN_Y => -10710,
V_RANGO_IN_MAX_Y => 10710, V_RANGO_OUT_MIN_Y => -11220, V_RANGO_OUT_MAX_Y => 11220)
        PORT MAP (clk => clk, reset => reset, calcula => calcula, operacion => bus_rom(77), xin
=> x_in_12_2, xout => OPEN, sin => sout_11_3, sout => sout_12_3, hecho=> hecho_12_3);

-- Fila    13-------vv--------vv----
f_13_c_0: ep1_nop3_1
        GENERIC MAP (V_RANGO_MIN_X => 0, V_RANGO_MAX_X => 255, V_RANGO_IN_MIN_Y => -1530,
V_RANGO_IN_MAX_Y => 1530, V_RANGO_OUT_MIN_Y => -1785, V_RANGO_OUT_MAX_Y => 1785)
        PORT MAP (clk => clk, reset => reset, calcula => calcula, operacion => bus_rom(79
DOWNTO 78), xin => dato_in_D, xout => x_in_13_0, sin => sout_12_0, sout => sout_13_0, hecho=>
hecho_13_0);

f_13_c_1: ep1_nop3_2
```

```
        GENERIC MAP (V_RANGO_MIN_X => 0, V_RANGO_MAX_X => 255, V_RANGO_IN_MIN_Y => -4590,
V_RANGO_IN_MAX_Y => 4590, V_RANGO_OUT_MIN_Y => -5100, V_RANGO_OUT_MAX_Y => 5100)
        PORT MAP (clk => clk, reset => reset, calcula => calcula, operacion => bus_rom(81
DOWNTO 80), xin => x_in_13_0, xout => x_in_13_1, sin => sout_12_1, sout => sout_13_1, hecho=>
hecho_13_1);

f_13_c_2: ep1_nop5_4
        GENERIC MAP (V_RANGO_MIN_X => 0, V_RANGO_MAX_X => 255, V_RANGO_IN_MIN_Y => -6120,
V_RANGO_IN_MAX_Y => 6120, V_RANGO_OUT_MIN_Y => -7140, V_RANGO_OUT_MAX_Y => 7140)
        PORT MAP (clk => clk, reset => reset, calcula => calcula, operacion => bus_rom(84
DOWNTO 82), xin => x_in_13_1, xout => x_in_13_2, sin => sout_12_2, sout => sout_13_2, hecho=>
hecho_13_2);

f_13_c_3: ep1_nop5_4
        GENERIC MAP (V_RANGO_MIN_X => 0, V_RANGO_MAX_X => 255, V_RANGO_IN_MIN_Y => -11220,
V_RANGO_IN_MAX_Y => 11220, V_RANGO_OUT_MIN_Y => -12240, V_RANGO_OUT_MAX_Y => 12240)
        PORT MAP (clk => clk, reset => reset, calcula => calcula, operacion => bus_rom(87
DOWNTO 85), xin => x_in_13_2, xout => OPEN, sin => sout_12_3, sout => sout_13_3, hecho=>
hecho_13_3);

-- Fila    14-------vv--------vv----
f_14_c_0: ep1_nop1_0
        GENERIC MAP (V_RANGO_MIN_X => 0, V_RANGO_MAX_X => 255, V_RANGO_IN_MIN_Y => -1785,
V_RANGO_IN_MAX_Y => 1785, V_RANGO_OUT_MIN_Y => -1785, V_RANGO_OUT_MAX_Y => 1785)
        PORT MAP (clk => clk, reset => reset, calcula => calcula, xin => dato_in_E, xout =>
x_in_14_0, sin => sout_13_0, sout => sout_14_0, hecho=> hecho_14_0);

f_14_c_1: ep1_nop3_2
        GENERIC MAP (V_RANGO_MIN_X => 0, V_RANGO_MAX_X => 255, V_RANGO_IN_MIN_Y => -5100,
V_RANGO_IN_MAX_Y => 5100, V_RANGO_OUT_MIN_Y => -5610, V_RANGO_OUT_MAX_Y => 5610)
        PORT MAP (clk => clk, reset => reset, calcula => calcula, operacion => bus_rom(89
DOWNTO 88), xin => x_in_14_0, xout => x_in_14_1, sin => sout_13_1, sout => sout_14_1, hecho=>
hecho_14_1);

f_14_c_2: ep1_nop1_0
        GENERIC MAP (V_RANGO_MIN_X => 0, V_RANGO_MAX_X => 255, V_RANGO_IN_MIN_Y => -7140,
V_RANGO_IN_MAX_Y => 7140, V_RANGO_OUT_MIN_Y => -7140, V_RANGO_OUT_MAX_Y => 7140)
        PORT MAP (clk => clk, reset => reset, calcula => calcula, xin => x_in_14_1, xout =>
x_in_14_2, sin => sout_13_2, sout => sout_14_2, hecho=> hecho_14_2);

f_14_c_3: ep1_nop5_4
        GENERIC MAP (V_RANGO_MIN_X => 0, V_RANGO_MAX_X => 255, V_RANGO_IN_MIN_Y => -12240,
V_RANGO_IN_MAX_Y => 12240, V_RANGO_OUT_MIN_Y => -13260, V_RANGO_OUT_MAX_Y => 13260)
        PORT MAP (clk => clk, reset => reset, calcula => calcula, operacion => bus_rom(92
DOWNTO 90), xin => x_in_14_2, xout => OPEN, sin => sout_13_3, sout => sout_14_3, hecho=>
hecho_14_3);

-- Fila    15-------vv--------vv----
f_15_c_0: ep1_nop3_1
        GENERIC MAP (V_RANGO_MIN_X => 0, V_RANGO_MAX_X => 255, V_RANGO_IN_MIN_Y => -1785,
V_RANGO_IN_MAX_Y => 1785, V_RANGO_OUT_MIN_Y => -2040, V_RANGO_OUT_MAX_Y => 2040)
        PORT MAP (clk => clk, reset => reset, calcula => calcula, operacion => bus_rom(94
DOWNTO 93), xin => dato_in_F, xout => x_in_15_0, sin => sout_14_0, sout => sout_15_0, hecho=>
hecho_15_0);

f_15_c_1: ep1_nop3_2
        GENERIC MAP (V_RANGO_MIN_X => 0, V_RANGO_MAX_X => 255, V_RANGO_IN_MIN_Y => -5610,
V_RANGO_IN_MAX_Y => 5610, V_RANGO_OUT_MIN_Y => -6120, V_RANGO_OUT_MAX_Y => 6120)
        PORT MAP (clk => clk, reset => reset, calcula => calcula, operacion => bus_rom(96
DOWNTO 95), xin => x_in_15_0, xout => x_in_15_1, sin => sout_14_1, sout => sout_15_1, hecho=>
hecho_15_1);

f_15_c_2: ep1_nop5_4
        GENERIC MAP (V_RANGO_MIN_X => 0, V_RANGO_MAX_X => 255, V_RANGO_IN_MIN_Y => -7140,
V_RANGO_IN_MAX_Y => 7140, V_RANGO_OUT_MIN_Y => -8160, V_RANGO_OUT_MAX_Y => 8160)
        PORT MAP (clk => clk, reset => reset, calcula => calcula, operacion => bus_rom(99
DOWNTO 97), xin => x_in_15_1, xout => x_in_15_2, sin => sout_14_2, sout => sout_15_2, hecho=>
hecho_15_2);

f_15_c_3: ep1_nop5_4
        GENERIC MAP (V_RANGO_MIN_X => 0, V_RANGO_MAX_X => 255, V_RANGO_IN_MIN_Y => -13260,
V_RANGO_IN_MAX_Y => 13260, V_RANGO_OUT_MIN_Y => -14280, V_RANGO_OUT_MAX_Y => 14280)
        PORT MAP (clk => clk, reset => reset, calcula => calcula, operacion => bus_rom(102
DOWNTO 100), xin => x_in_15_2, xout => OPEN, sin => sout_14_3, sout => sout_15_3, hecho=>
hecho_15_3);

-- Señal de hecho global ---
```

```
-- Por el gran retraso que suponía, se decidió eliminar esta parte de código
--
-- hecho <=    hecho_0_0 AND hecho_0_1 AND hecho_0_2 AND hecho_0_3 AND
--                     hecho_1_0 AND hecho_1_1 AND hecho_1_2 AND hecho_1_3 AND
--                     hecho_2_0 AND hecho_2_1 AND hecho_2_2 AND hecho_2_3 AND
--                     hecho_3_0 AND hecho_3_1 AND hecho_3_2 AND hecho_3_3 AND
--                     hecho_4_0 AND hecho_4_1 AND hecho_4_2 AND hecho_4_3 AND
--                     hecho_5_0 AND hecho_5_1 AND hecho_5_2 AND hecho_5_3 AND
--                     hecho_6_0 AND hecho_6_1 AND hecho_6_2 AND hecho_6_3 AND
--                     hecho_7_0 AND hecho_7_1 AND hecho_7_2 AND hecho_7_3 AND
--                     hecho_8_0 AND hecho_8_1 AND hecho_8_2 AND hecho_8_3 AND
--                     hecho_9_0 AND hecho_9_1 AND hecho_9_2 AND hecho_9_3 AND
--                     hecho_10_0 AND hecho_10_1 AND hecho_10_2 AND hecho_10_3 AND
--                     hecho_11_0 AND hecho_11_1 AND hecho_11_2 AND hecho_11_3 AND
--                     hecho_12_0 AND hecho_12_1 AND hecho_12_2 AND hecho_12_3 AND
--                     hecho_13_0 AND hecho_13_1 AND hecho_13_2 AND hecho_13_3 AND
--                     hecho_14_0 AND hecho_14_1 AND hecho_14_2 AND hecho_14_3 AND
--                     hecho_15_0 AND hecho_15_1 AND hecho_15_2 AND hecho_15_3;

hecho <= hecho_13_3;

sout_0 <= sout_15_0; sout_1 <= sout_15_1; sout_2 <= sout_15_2; sout_3 <= sout_15_3;

END sistolico;
```

ep1_nop1_0.vhd

```
-- Elementos de proceso ep1 que solo hacen la
-- operación s_out<=s_in; x_out<=x_in;
--
-- Como sólo tiene que hacer una cosa, no necesita ninguna
-- entrada que le indique lo que hacer, es decir, no tiene puerto "coef"

LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;
USE ieee.std_logic_signed.all;

--  Entity Declaration

ENTITY ep1_nop1_0 IS
        -- {{ALTERA_IO_BEGIN}} DO NOT REMOVE THIS LINE!
        GENERIC (
                V_RANGO_IN_MIN_Y,V_RANGO_IN_MAX_Y    : INTEGER := 0;
                V_RANGO_OUT_MIN_Y,V_RANGO_OUT_MAX_Y  : INTEGER := 0;
                V_RANGO_MIN_X, V_RANGO_MAX_X         : INTEGER := 0
        );
        PORT
        (
                clk     : IN  STD_LOGIC;
                reset   : IN  STD_LOGIC;
                calcula : IN  STD_LOGIC;
                xin     : IN  INTEGER RANGE (V_RANGO_MIN_X) to (V_RANGO_MAX_X);
                sin     : IN  INTEGER RANGE (V_RANGO_IN_MIN_Y) to (V_RANGO_IN_MAX_Y);
                xout    : OUT INTEGER RANGE (V_RANGO_MIN_X) to (V_RANGO_MAX_X);
                sout    : OUT INTEGER RANGE (V_RANGO_OUT_MIN_Y) to (V_RANGO_OUT_MAX_Y);
                hecho   : OUT STD_LOGIC
        );
        -- {{ALTERA_IO_END}} DO NOT REMOVE THIS LINE!
END ep1_nop1_0;

ARCHITECTURE ep1_architecture OF ep1_nop1_0 IS
        SIGNAL estado :integer range 0 to 2;
BEGIN

proceso_ep1: PROCESS(clk,reset)

BEGIN
        IF (reset='1') THEN
                estado<=0;
                hecho<='0';
        ELSIF (clk'event AND clk='1') THEN
                if (estado=0) then
```

```
                        xout<=0; sout<=0;
                        estado<=1;
                elsif (estado=1) then
                        hecho<='1';
                        if (calcula='1') then
                                estado<=2;
                                hecho<='0';
                        elsif (calcula='0') then
                                estado<=1;
                        end if;
                elsif (estado=2) then
                        sout<=sin;
                        xout<=xin;
                        estado<=1;
                end if;
        END IF;

END PROCESS proceso_ep1;

END ep1_architecture;
```

ep1_nop1_0_pf.vhd

```
-- Elementos de proceso ep1 que solo hacen la
-- operación s_out<=s_in; x_out<=x_in;
--
-- Como sólo tiene que hacer una cosa, no necesita ninguna
-- entrada que le indique lo que hacer, es decir, no tiene puerto "coef"

LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;
USE ieee.std_logic_signed.all;


--  Entity Declaration

ENTITY ep1_nop1_0_pf IS
        -- {{ALTERA_IO_BEGIN}} DO NOT REMOVE THIS LINE!
        GENERIC (
                V_RANGO_OUT_MIN_Y,V_RANGO_OUT_MAX_Y  : INTEGER := 0;
                V_RANGO_MIN_X                        : INTEGER := 0;
                V_RANGO_MAX_X                        : INTEGER := 255
        );
        PORT
        (
                clk     : IN  STD_LOGIC;
                reset   : IN  STD_LOGIC;
                calcula : IN  STD_LOGIC;
                xin     : IN  INTEGER RANGE (V_RANGO_MIN_X) to (V_RANGO_MAX_X);
                xout    : OUT INTEGER RANGE (V_RANGO_MIN_X) to (V_RANGO_MAX_X);
                sout    : OUT INTEGER RANGE (V_RANGO_OUT_MIN_Y) to (V_RANGO_OUT_MAX_Y);
                hecho   : OUT STD_LOGIC
        );
        -- {{ALTERA_IO_END}} DO NOT REMOVE THIS LINE!
END ep1_nop1_0_pf;

ARCHITECTURE ep1_architecture OF ep1_nop1_0_pf IS
SIGNAL estado :integer range 0 to 2;
BEGIN

sout<=0;

proceso_ep1: PROCESS(clk,reset)
BEGIN
        IF (reset='1') THEN
                estado<=0;
                hecho<='0';
        ELSIF (clk'event AND clk='1') THEN
                if (estado=0) then
                        xout<=0;
                        estado<=1;
                elsif (estado=1) then
                        hecho<='1';
```

```
                        if (calcula='1') then
                                estado<=2;
                                hecho<='0';
                        elsif (calcula='0') then
                                estado<=1;
                        end if;
                elsif (estado=2) then
                        xout<=xin;
                        estado<=1;
                end if;
        END IF;
END PROCESS proceso_ep1;
END ep1_architecture;
```

ep1_nop1_2_pf.vhd

```
-- Elementos de proceso ep1 que solo hacen la
-- operación s_out<=s_in+2*x_in; x_out<=x_in;
--
-- Como sólo tiene que hacer una cosa, no necesita ninguna
-- entrada que le indique lo que hacer, es decir, no tiene puerto "coef"

LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;
USE ieee.std_logic_signed.all;

--  Entity Declaration

ENTITY ep1_nop1_2_pf IS
        -- {{ALTERA_IO_BEGIN}} DO NOT REMOVE THIS LINE!
        GENERIC (
                V_RANGO_OUT_MIN_Y,V_RANGO_OUT_MAX_Y  : INTEGER := 0;
                V_RANGO_MIN_X, V_RANGO_MAX_X         : INTEGER := 0
        );
        PORT
        (
                clk     : IN  STD_LOGIC;
                reset   : IN  STD_LOGIC;
                calcula : IN  STD_LOGIC;
                xin     : IN  INTEGER RANGE (V_RANGO_MIN_X) to (V_RANGO_MAX_X);
                xout    : OUT INTEGER RANGE (V_RANGO_MIN_X) to (V_RANGO_MAX_X);
                sout    : OUT INTEGER RANGE (V_RANGO_OUT_MIN_Y) to (V_RANGO_OUT_MAX_Y);
                hecho   : OUT STD_LOGIC
        );
        -- {{ALTERA_IO_END}} DO NOT REMOVE THIS LINE!
END ep1_nop1_2_pf;

ARCHITECTURE ep1_architecture OF ep1_nop1_2_pf IS
        SIGNAL estado :integer range 0 to 2;
BEGIN


proceso_ep1: PROCESS(clk,reset)
BEGIN
        IF (reset='1') THEN
                estado<=0;
                hecho<='0';
        ELSIF (clk'event AND clk='1') THEN
                if (estado=0) then
                        xout<=0; sout<=0;
                        estado<=1;
                elsif (estado=1) then
                        hecho<='1';
                        if (calcula='1') then
                                estado<=2;
                                hecho<='0';
                        elsif (calcula='0') then
                                estado<=1;
                        end if;
                elsif (estado=2) then
                        sout<=xin*2;
                        xout<=xin;
```

```
                        estado<=1;
                end if;
        END IF;
END PROCESS proceso_ep1;

END ep1_architecture;
```

ep1_nop2_2.vhd

```
-- Elementos de proceso ep1 que hacen dos tipos
-- de operaciones:
--      operacion = 0 => s_out=s_in-x_in*2;
--      operacion = 1 => s_out=s_in+x_in*2;
--      x_out = x_in;
--

LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;
USE ieee.std_logic_signed.all;

--  Entity Declaration

ENTITY ep1_nop2_2 IS
        -- {{ALTERA_IO_BEGIN}} DO NOT REMOVE THIS LINE!
        GENERIC (
                V_RANGO_IN_MIN_Y,V_RANGO_IN_MAX_Y     : INTEGER := 0;
                V_RANGO_OUT_MIN_Y,V_RANGO_OUT_MAX_Y   : INTEGER := 0;
                V_RANGO_MIN_X, V_RANGO_MAX_X          : INTEGER := 0
        );
        PORT
        (
                clk     : IN  STD_LOGIC;
                reset   : IN  STD_LOGIC;
                calcula : IN  STD_LOGIC;
                operacion      : IN  STD_LOGIC;
                xin     : IN  INTEGER RANGE (V_RANGO_MIN_X) to (V_RANGO_MAX_X);
                sin     : IN  INTEGER RANGE (V_RANGO_IN_MIN_Y) to (V_RANGO_IN_MAX_Y);
                xout    : OUT INTEGER RANGE (V_RANGO_MIN_X) to (V_RANGO_MAX_X);
                sout    : OUT INTEGER RANGE (V_RANGO_OUT_MIN_Y) to (V_RANGO_OUT_MAX_Y);
                hecho   : OUT STD_LOGIC
        );
        -- {{ALTERA_IO_END}} DO NOT REMOVE THIS LINE!
END ep1_nop2_2;

ARCHITECTURE ep1_architecture OF ep1_nop2_2 IS
        SIGNAL estado :integer range 0 to 2;
BEGIN

proceso_ep1: PROCESS(clk,reset)

BEGIN
        IF (reset='1') THEN
                estado<=0;
                hecho<='0';
        ELSIF (clk'event AND clk='1') THEN
                if (estado=0) then
                        xout<=0; sout<=0;
                        estado<=1;
                elsif (estado=1) then
                        hecho<='1';
                        if (calcula='1') then
                                estado<=2;
                                hecho<='0';
                        elsif (calcula='0') then
                                estado<=1;
                        end if;
                elsif (estado=2) then
                        if (operacion='0') then
                                sout<=sin-(xin*2);
                        else
                                sout<=sin+(xin*2);
                        end if;
                        xout<=xin;
```

```
                        estado<=1;
                end if;
        END IF;
END PROCESS proceso_ep1;

END ep1_architecture;
```

ep1_nop3_1.vhd

```
-- Elementos de proceso ep1 que hacen tres tipos
-- de operaciones:
--      operacion = 00 => s_out=s_in-x_in;
--      operacion = 01 => s_out=s_in;
--      operacion = 10 => s_out=s_in+x_in;
--      x_out = x_in;
--

LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;
USE ieee.std_logic_signed.all;

--  Entity Declaration

ENTITY ep1_nop3_1 IS
        -- {{ALTERA_IO_BEGIN}} DO NOT REMOVE THIS LINE!
        GENERIC (
                V_RANGO_IN_MIN_Y,V_RANGO_IN_MAX_Y    : INTEGER := 0;
                V_RANGO_OUT_MIN_Y,V_RANGO_OUT_MAX_Y  : INTEGER := 0;
                V_RANGO_MIN_X, V_RANGO_MAX_X         : INTEGER := 0
        );
        PORT
        (
                clk     : IN  STD_LOGIC;
                reset   : IN  STD_LOGIC;
                calcula : IN  STD_LOGIC;
                operacion      : IN  STD_LOGIC_VECTOR(1 DOWNTO 0);
                xin     : IN  INTEGER RANGE (V_RANGO_MIN_X) to (V_RANGO_MAX_X);
                sin     : IN  INTEGER RANGE (V_RANGO_IN_MIN_Y) to (V_RANGO_IN_MAX_Y);
                xout    : OUT INTEGER RANGE (V_RANGO_MIN_X) to (V_RANGO_MAX_X);
                sout    : OUT INTEGER RANGE (V_RANGO_OUT_MIN_Y) to (V_RANGO_OUT_MAX_Y);
                hecho   : OUT STD_LOGIC
        );
        -- {{ALTERA_IO_END}} DO NOT REMOVE THIS LINE!
END ep1_nop3_1;

ARCHITECTURE ep1_architecture OF ep1_nop3_1 IS
        SIGNAL estado :integer range 0 to 2;
BEGIN

proceso_ep1: PROCESS(clk,reset)

BEGIN
        IF (reset='1') THEN
                estado<=0;
                hecho<='0';
        ELSIF (clk'event AND clk='1') THEN
                if (estado=0) then
                        xout<=0; sout<=0;
                        estado<=1;
                elsif (estado=1) then
                        hecho<='1';
                        if (calcula='1') then
                                estado<=2;
                                hecho<='0';
                        elsif (calcula='0') then
                                estado<=1;
                        end if;
                elsif (estado=2) then
                        CASE operacion IS
                                WHEN "00" => sout<=sin-xin;
                                WHEN "01" => sout<=sin;
                                WHEN OTHERS => sout<=sin+xin; -- "10"
                        END CASE;
```

```
                        xout<=xin;
                        estado<=1;
                end if;
        END IF;
END PROCESS proceso_ep1;

END ep1_architecture;
```

ep1_nop3_2.vhd

```
-- Elementos de proceso ep1 que hacen tres tipos

-- de operaciones:
--      operacion = 00 => s_out=s_in-2*x_in;
--      operacion = 01 => s_out=s_in;
--      operacion = 10 => s_out=s_in+2*x_in;
--      x_out = x_in;
--
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;
USE ieee.std_logic_signed.all;

--  Entity Declaration

ENTITY ep1_nop3_2 IS
        -- {{ALTERA_IO_BEGIN}} DO NOT REMOVE THIS LINE!
        GENERIC (
                V_RANGO_IN_MIN_Y,V_RANGO_IN_MAX_Y     : INTEGER := 0;
                V_RANGO_OUT_MIN_Y,V_RANGO_OUT_MAX_Y   : INTEGER := 0;
                V_RANGO_MIN_X, V_RANGO_MAX_X          : INTEGER := 0
        );
        PORT
        (
                clk     : IN  STD_LOGIC;
                reset   : IN  STD_LOGIC;
                calcula : IN  STD_LOGIC;
                operacion      : IN  STD_LOGIC_VECTOR(1 DOWNTO 0);
                xin     : IN  INTEGER RANGE (V_RANGO_MIN_X) to (V_RANGO_MAX_X);
                sin     : IN  INTEGER RANGE (V_RANGO_IN_MIN_Y) to (V_RANGO_IN_MAX_Y);
                xout    : OUT INTEGER RANGE (V_RANGO_MIN_X) to (V_RANGO_MAX_X);
                sout    : OUT INTEGER RANGE (V_RANGO_OUT_MIN_Y) to (V_RANGO_OUT_MAX_Y);
                hecho   : OUT STD_LOGIC
        );
        -- {{ALTERA_IO_END}} DO NOT REMOVE THIS LINE!
END ep1_nop3_2;

ARCHITECTURE ep1_architecture OF ep1_nop3_2 IS
        SIGNAL estado :integer range 0 to 2;
BEGIN

proceso_ep1: PROCESS(clk,reset)

BEGIN
        IF (reset='1') THEN
                estado<=0;
                hecho<='0';
        ELSIF (clk'event AND clk='1') THEN
                if (estado=0) then
                        xout<=0; sout<=0;
                        estado<=1;
                elsif (estado=1) then
                        hecho<='1';
                        if (calcula='1') then
                                estado<=2;
                                hecho<='0';
                        elsif (calcula='0') then
                                estado<=1;
                        end if;
                elsif (estado=2) then
                        CASE operacion IS
                                WHEN "00" => sout<=sin-(xin*2);
                                WHEN "01" => sout<=sin;
                                WHEN OTHERS => sout<=sin+(xin*2); -- "10"
```

```
                        END CASE;
                        xout<=xin;
                        estado<=1;
                end end if;
        END IF;
END PROCESS proceso_ep1;

END ep1_architecture;
```

ep1_nop5_4.vhd

```
-- Elementos de proceso ep1 que hacen cinco tipos
-- de operaciones:
--      operacion = 000 => s_out=s_in-4*x_in;
--      operacion = 001 => s_out=s_in-2*x_in;
--      operacion = 010 => s_out=s_in;
--      operacion = 011 => s_out=s_in+2*x_in;
--      operacion = 100 => s_out=s_in+4*x_in;
--      x_out = x_in;
--
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;
USE ieee.std_logic_signed.all;

--  Entity Declaration

ENTITY ep1_nop5_4 IS
        -- {{ALTERA_IO_BEGIN}} DO NOT REMOVE THIS LINE!
        GENERIC (
                V_RANGO_IN_MIN_Y,V_RANGO_IN_MAX_Y     : INTEGER := 0;
                V_RANGO_OUT_MIN_Y,V_RANGO_OUT_MAX_Y   : INTEGER := 0;
                V_RANGO_MIN_X, V_RANGO_MAX_X          : INTEGER := 0
        );
        PORT
        (
                clk     : IN  STD_LOGIC;
                reset   : IN  STD_LOGIC;
                calcula : IN  STD_LOGIC;
                operacion       : IN  STD_LOGIC_VECTOR(2 DOWNTO 0);
                xin     : IN  INTEGER RANGE (V_RANGO_MIN_X) to (V_RANGO_MAX_X);
                sin     : IN  INTEGER RANGE (V_RANGO_IN_MIN_Y) to (V_RANGO_IN_MAX_Y);
                xout    : OUT INTEGER RANGE (V_RANGO_MIN_X) to (V_RANGO_MAX_X);
                sout    : OUT INTEGER RANGE (V_RANGO_OUT_MIN_Y) to (V_RANGO_OUT_MAX_Y);
                hecho   : OUT STD_LOGIC
        );
        -- {{ALTERA_IO_END}} DO NOT REMOVE THIS LINE!
END ep1_nop5_4;

ARCHITECTURE ep1_architecture OF ep1_nop5_4 IS
        SIGNAL estado :integer range 0 to 2;
BEGIN

proceso_ep1: PROCESS(clk,reset)

BEGIN
        IF (reset='1') THEN
                estado<=0;
                hecho<='0';
        ELSIF (clk'event AND clk='1') THEN
                if (estado=0) then
                        xout<=0; sout<=0;
                        estado<=1;
                elsif (estado=1) then
                        hecho<='1';
                        if (calcula='1') then
                                estado<=2;
                                hecho<='0';
                        elsif (calcula='0') then
                                estado<=1;
                        end if;
                elsif (estado=2) then
                        CASE operacion IS
                                WHEN "000" => sout<=sin-(xin*4);
```

```
                                WHEN "001" => sout<=sin-(xin*2);
                                WHEN "010" => sout<=sin;
                                WHEN "011" => sout<=sin+(xin*2);
                                WHEN OTHERS => sout<=sin+(xin*4); -- "100"
                        END CASE;
                        xout<=xin;
                        estado<=1;
                end if;
        END IF;
END PROCESS proceso_ep1;

END ep1_architecture;
```

H.2.4 Otros módulos

pulsador.vhd

```
library ieee;


use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;

ENTITY pulsador IS
PORT ( senal: IN  std_logic;
       Clk  : IN  std_logic;
       reset: IN  std_logic;
       pulso: OUT std_logic);
END pulsador;

ARCHITECTURE uno OF pulsador IS
signal salida: std_logic;
BEGIN

proceso1: PROCESS (clk,reset,senal)
variable estado: integer;
begin
        if (reset='1') then
                estado := 0;
        elsif (clk'event and clk='1') then
                if (estado=0) then
                        if (senal='1') then
                                salida<='1';
                                estado:=1;
                        else
                                salida<='0';
                                estado:=0;
                        end if;
                elsif (estado=1) then
                        salida<='0';
                        if (senal='0') then
                                estado:=0;
                        else
                                estado:=1;
                        end if;
                end if;
        end if;
end process;

pulso <= salida;

END uno;
```

subebaja.vhd

```
library ieee;

use IEEE.std_logic_1164.all;
```

```
ENTITY subebaja IS
PORT ( arriba: IN  std_logic;
       abajo : IN  std_logic;
       salida: OUT std_logic);
END subebaja;

ARCHITECTURE uno OF subebaja IS
BEGIN
proceso1: PROCESS (arriba,abajo)
begin
       if (abajo='1') then
              salida <= '0';
       elsif (arriba'event and arriba='1') then
              salida <= '1';
       end if;
end process;

END uno;
```

I El software del Subsistema Externo

El software del SSE se encarga de actuar como interfaz de comandos utilizando como canal de entrada y salida el puerto serie y de realizar las operaciones de Transformación Discreta de Hartley, tanto de vectores de 16 componentes así como de matrices de 256 componentes. La interacción con esta interfaz viene detallada en el apartado 4.8 (pág. 52 de la memoria).

En este anexo se ha incluido únicamente el código fuente principal (principal.c) del programa contenido en el SSE, puesto que el resto son prefabricados por Altera. De todos modos, se puede consultar la totalidad del código en el CD-ROM que acompaña a la memoria.

I.1 principal.c

```
#include "stdio.h"
#include "pio_lcd16207.h"
#include "nios.h"


#define TAMANYO_VECTOR 16
#define TAMANYO_MATRIZ 256
//#define CICLOS_ESPERA  34
#define CICLOS_ESPERA  17
#define DESFASE        18
//#define USE_BREAKPOINT 0

// variable used to trigger a data breakpoint
//unsigned int finished;

np_pio *pio = na_led_pio;



const char info[]="\nSistema Kiseki/Excalibur 1.7\nCarles Rellan Martinez\nMiguel Angel
Sanchez Lopez\nPFC DEE ETSEIB UPC\n";
const int referencia[TAMANYO_VECTOR]={ 32,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 };

static unsigned char filtro[8];
static int v16[16];
static int sm1[16];
static int m256[256];
static int pulmon[256];
static int min_pulmon[16],sum_pulmon[16];
static unsigned char mf[256];


static unsigned char indice_e, indice_l; //posiciones del vector de entrada y de salida.

void inicializar(void) {
int i;
        for (i=0; i<8; i++) filtro[i]=0xFF;
        for (i=0; i<16; i++) {
                v16[i]=0;
                min_pulmon[i]=0;
                sum_pulmon[i]=0;
        }
        for (i=0; i<256; i++) {
                m256[i]=0;
                mf[i]=1;
                pulmon[i]=0;
        }
        indice_e=0;
        indice_l=0;
        pio->np_piodirection = 3;
        pio->np_piodata = 1;
        nr_pio_lcdinit(na_lcd_pio);
                                /*0123456789ABCDEF0123456789ABCDEF*/
        nr_pio_lcdwritescreen("Kiseki 1.7------(c)  MASL,CRM'04");
}
```

```
int tdh_el(unsigned char c) {
unsigned char i;
        *na_DHT=(int)(c);
        for (i=0; i<CICLOS_ESPERA; i++); //tiempo de espera;
        indice_e++; indice_e %= TAMANYO_VECTOR;
        indice_l++; indice_l %= TAMANYO_VECTOR;
        return(*na_DHT);
}

void tdhv_stdin(int *v) {
unsigned char i;
        //j=indice_e;
        nr_pio_lcdwritescreen("Recibiendo...    vector 16 bytes");
        for (i=0; i<16; i++) {
                nr_pio_showhex(i);
                tdh_el((unsigned char)getchar());
        }
        //                      0123456789abcdef0123456789abcdef
        nr_pio_lcdwritescreen("Calculo TDH16 OKEsperando orden.");
        for (i=0; i<2;  i++) tdh_el(0);
        for (i=0; i<16; i++) v[i]=tdh_el(0);
}

void recibir_filtro(void) {
unsigned char i,j;
        nr_pio_lcdwritescreen("Recibiendo...     filtro 8 bytes");
        for (i=0; i<8; i++) {
                nr_pio_showhex(i);
                filtro[i]=(unsigned char)getchar();
        }
        for (i=0; i<8; i++) {
                for (j=0; j<8; j++) {
                        if (filtro[i] & (1<<(7-j))) {
                                mf[i*16+j]=mf[i*16+15-j]=mf[(15-i)*16+j]=mf[(15-i)*16+15-j]=1;
                        } else {
                                mf[i*16+j]=mf[i*16+15-j]=mf[(15-i)*16+j]=mf[(15-i)*16+15-j]=0;
                        }
                }
        }
        nr_pio_lcdwritescreen("Filtro OK       Esperando orden.");
}

void recibir_vector_suma(void) {
unsigned char i,j,k,l;
        nr_pio_lcdwritescreen("Recibiendo...      vector sumas.");
        for (i=0; i<16; i++) {
                nr_pio_showhex(i);
                k=(unsigned char)getchar();
                l=(unsigned char)getchar();
                sml[i]=(l << 8)|k;
        }
        nr_pio_lcdwritescreen("Vector sumas    Esperando orden.");
}


/*
  A esta función le pasaremos como parámetro una matriz origen x
  de 16x16 números enteros 0<=x[i]<=255, a la que se le aplicará
  la transformación discreta de hartley en dos dimensiones y cuyo
  resultado se almacenará en la matriz apuntada por x_tdh

  sentido=0 => comprimir
  sentido=1 => descomprimir
*/

void tdhm_stdin(char sentido) {
int k,i,c,d,t,p;

for (i=0; i<16; i++) {min_pulmon[i]=0; sum_pulmon[i]=0; }
for (i=0; i<256; i++) pulmon[i]=0;
nr_pio_lcdwritescreen("Recibiendo...   matriz 256 bytes");
for (i=0; i<(512+16+DESFASE); i++) {
        // LECTURA
        if (i==256) nr_pio_lcdwritescreen("Recepcion valida   calculando...");
        if (i<256) {
                //c=tdh_el(x[i]);
                nr_pio_showhex(i);
```

```
                c=tdh_el((unsigned char)getchar());
        } else
        if ((i>=(256+16)) && (i<(512+16))) {
                k=i-256-16;
                p=(k%16)*16+(k/16);
                t=pulmon[p]-min_pulmon[k/16];
                c=tdh_el((unsigned char)(t));
        } else c=tdh_el(0);
        // ESCRITURA
        if ((i>=DESFASE) && (i<(256+DESFASE))) {
            if (!sentido) { /*comprimir */
                  c=c/32;
         } else {         /*descomprimir - aquí se necesita el vector de sumas*/
              if ((i-DESFASE)%16) {c = c/2;} else {c=sm1[(i-DESFASE)/16]/2;}
         }

                pulmon[i-DESFASE]=c;
                k=(i-DESFASE)%16;
                if ((i-DESFASE)<16) {
                        min_pulmon[k]=c;
                } else {
                        if (c < min_pulmon[k]) {
                                min_pulmon[k]=c;
                        }
                }
                sum_pulmon[k] += c;
        } else
        if ((i>=(256+16+DESFASE)) && (i<(512+16+DESFASE))) {
                t = i-(256+16+DESFASE);
                p = (t%16)*16+(t/16);
                if (!(t%16)) /*c=sum_pulmon[t/16];*/c=(sum_pulmon[t/16]<<1);
                //c = c/32;
                m256[p]=c;
        }
}
//                      0123456789abcdef0123456789abcdef
nr_pio_lcdwritescreen("Calculo TDH^2 OKEsperando orden.");
}

void cazar_cero(void) {
unsigned int i,j;
    i=0; j=0;
    while (i<TAMANYO_VECTOR) {
        if (referencia[i]==tdh_el(1)) {
                i++;
        } else {
                i=0;
        }
        j++;
    };
        for (i=0; i<TAMANYO_VECTOR-(DESFASE-TAMANYO_VECTOR); i++)
        tdh_el(1);
        indice_e=0; indice_l=2;
}

void purga(void) {
        while(indice_e) tdh_el(0);
}

void mostrar_estado(void) {
unsigned int i;
unsigned char c;
        nr_pio_lcdwritescreen("Ofreciendo info de estado...    ");
        i=0;
        while (c=info[i++]) nr_uart_txchar(c,0);
        printf("\nfiltro:\n");
        for (i=0; i<8; i++) {
                printf("%02X",filtro[i]);
                if (i!=7) printf(",");
        }
        printf("\n");
        for (i=0; i<256; i++) {
                if (mf[i]) { printf("X"); } else { printf("_"); }
                if ((i%16)==15) printf("\n");
        }
        printf("\nVector:\n");
```

```
        for (i=0; i<16; i++) {
                printf("%5i",v16[i]);
                if ((i%16)!=15)        {printf(",");}
        }
        printf("\nSumas:\n");

        for (i=0; i<16; i++) {
                printf("%5i",sm1[i]);
                if ((i%16)!=15)        {printf(",");}
        }
        printf("\nPulmon:\n");
        for (i=0; i<256; i++) {
                printf("%5i",pulmon[i]);//nr_uart_txhex(pulmon[i]);
                if ((i%16)==15) nr_uart_txchar('\n',0); else
                {printf(",");}
        }
        printf("\nMatriz:\n");
        for (i=0; i<256; i++) {
                printf("%5i",m256[i]);//nr_uart_txhex(pulmon[i]);
                if ((i%16)==15) nr_uart_txchar('\n',0); else
                {printf(",");}
        }
        nr_pio_lcdwritescreen("              OKEsperando orden.");
}

void mostrar_v16(void) {
unsigned char i;
        nr_pio_lcdwritescreen("Enviando vector         TDH16...");
        for (i=0; i<16; i++) {
                nr_uart_txchar((unsigned char)(v16[i] & 0x00FF),0);
                nr_uart_txchar((unsigned char)((v16[i] & 0xFF00)>>8),0);
        }
        nr_pio_lcdwritescreen("              OKEsperando orden.");
}

void mostrar_m256(void) {
unsigned int i;
        nr_pio_lcdwritescreen("Enviando matriz        TDH^2...");
        for (i=0; i<256; i++) {
                if (mf[i]) {
                        nr_uart_txchar((unsigned char)(m256[i] & 0x00FF),0);
                        nr_uart_txchar((unsigned char)((m256[i] & 0xFF00)>>8),0);
                }
        }
        nr_pio_lcdwritescreen("              OKEsperando orden.");
}


void main(void) {
int x,y,z;
char c,comando[100];

        inicializar();
        cazar_cero();
        while (1) { //Si, vamos a hacer una espera infinita ;)
                nr_uart_txchar('#',0); //señal de espera;
                nr_pio_showhex(0);
                c=getchar(); //espera recibir una instrucción
                pio->np_piodata ^= 3;
                if (c=='?') { //Información del sistema
                        mostrar_estado();
                } else
                if (c=='f') { //filtro 2D
                        recibir_filtro();
                } else
                if (c=='v') { //vector de 16 bytes
                        tdhv_stdin(v16);
                        purga();
                } else
                if (c=='V') {
                        mostrar_v16();
                } else
                if ((c=='m')||(c=='c')) { //matriz de 16x16 - compresión
                        tdhm_stdin(0);
                        purga();
                } else
                if (c=='d') { //matriz de 16x16 - descompresión
```

```
                        tdhm_stdin(1);
                        purga();
                } else
                if ((c=='M')||(c=='C')||(c=='D')) {
                        mostrar_m256();
                } else
                if (c=='s') {
                        recibir_vector_suma();
                } else { //comando no especificado

                }
        }
}
```

J El software del Subsistema Interno

J.1 Introducción

En este anexo se mostrará el código que ha sido desarrollado para implementar el Subsistema Interno, desarrollado a través de Visual Basic 6.0. A parte de los formularios de la aplicación, que ya han sido descritos en el anexo referido a la aplicación del sistema y en los que existe código de respuesta a eventos que suceden en el propio formulario; el software consta de una serie de módulos que son los que implementan las funciones pertinentes respondiendo a estos eventos.

En este sentido se ha dividido el software en dos partes distintas: por un lado, los que hacen referencia a la configuración de la captura de la imagen y, por otro lado, los que se centran en el propio proceso de seguimiento.

J.2 Módulos de Configuración

A continuación, en la Tabla J.1, se describen los módulos de configuración utilizados, así como la función que desarrollan cada uno de ellos dentro del subsistema interno. También se describe el código de respuesta a eventos de cada uno de los formularios.

Nombre del módulo	**Función**
`Calibracion.bas`	Módulo que se encarga de responder a los eventos del formulario de calibración y almacenar la información necesaria para proceso de seguimiento.
`Configuracionimagen.bas`	Módulo en el que se determinan las características de la imagen que va a ser capturada en función de los criterios predeterminados en el formulario de la aplicación a este respecto.
Nombre del formulario	**Función**
`formCalibracion.frm`	Parte de código que responde a los eventos sucedidos en el formulario de calibración.
`formConfiguracionImag.frm`	Parte de código que responde a los eventos sucedidos en el formulario de Configuración imagen..
`formConffiltro.frm`	Se ha considerado no necesario elaborar un módulo independiente para la configuración del filtro, así que se ha incluido como código de formulario.

Tabla J.1 Listados de los módulos de configuración

J.2.1 Calibracion.bas

```
Attribute VB_Name = "Calibracion"

Public cord1 As coordenadas
Public cord2 As coordenadas
Public cordI1 As coordenadas
Public cordI2 As coordenadas

'Captura de imagen desde el dispositivo
Public Sub Captura_Imagen(simage As imgdes)
```

```
Dim srect As RECT
Dim showUI As Long
Dim rcode As Long

'Definicion de la dimension imagen a capturar
srect.left = 0
srect.top = 0
srect.right = 576
srect.bottom = 480

'No mostrar cuadro de dialogo captura imagen
showUI = 0
pageno = 0

'Uso de funcion de librería Victor para captura imagen
rcode = TWscanimageex(hwnd, simage, srect, showUI)
End Sub

'Visualizar Imagen en el control PictureBox
Public Sub Visualizar_Imagen(simage As imgdes)

formIniCalibra.PicCalibra = image_to_picturebox(simage)
formIniCalibra.PicCalibra.Refresh

End Sub

'Procedimiento que envia la imagen capturada al cuadro Picture Box.
Public Function image_to_picturebox(srcimg As imgdes) As Picture
Dim retval As Long
Dim Pic As PicBmp

Dim IPic As IPicture
Dim IID_IDispatch As Guid
Dim tempimage As imgdes

    If (srcimg.hBitmap = 0) Then
        'Con el paquete dib, se crea una sección tipo dib de la imagen

        rcode = dibtoimage(srcimg.bmh, tempimage)
            If rcode = NO_ERROR Then
                ' Reemplazamos la imagen previa
                freeimage srcimg
                copyimgdes tempimage, srcimg
            End If
    End If

    ' Se rellena con la IDispatch Interface ID.
    With IID_IDispatch
        .Data1 = &H20400
        .Data4(0) = &HC0
        .Data4(7) = &H46
    End With

    ' Rellenamos Pic con sus partes necesarias
    With Pic
        .Size = Len(Pic)           ' Longitud de la estructura
        .Type = vbPicTypeBitmap    ' Tipo de Picture.
        .hBmp = srcimg.hBitmap     ' Acceso al Bitmap
        .hPal = 0                  ' acceso a la paleta
    End With

    ' Se crea un objeto Picture Box
    retval = OleCreatePictureIndirect(Pic, IID_IDispatch, 1, IPic)

    If (retval = 0) Then
        Set image_to_picturebox = IPic
    End If

End Function

Public Sub Calcular_CoordenadasMon(c1 As coordenadas, c2 As coordenadas)

'Estimacion de las coordenadas del monitor

c1.X = Abs((mcord.cord(1).X + mcord.cord(2).X) / 2)
c1.Y = Abs((mcord.cord(1).Y + mcord.cord(3).Y) / 2)
```

```
c2.X = Abs((mcord.cord(3).X + mcord.cord(4).X) / 2)
c2.Y = Abs((mcord.cord(2).Y + mcord.cord(4).Y) / 2)

End Sub

Public Sub actualizar(c1 As coordenadas, c2 As coordenadas, c3 As coordenadas, c4 As
coordenadas)

cord1 = c1
cord2 = c2
cordI1 = c3
cordI2 = c4

End Sub
```

J.2.2 Configuracionimagen.bas

```
Attribute VB_Name = "Configuracionimagen"

'Módulo en que se definen las características que se envían a la cámara según se 'haya
decidido en el formulario

Public Sub Configurar_Caract_Imagen()

'Definimos un tipo de imagen que luego visualizaremos para ver cambios.
Dim simage As imgdes

Call Configurar_Brillo
Call Configurar_Contraste
Call Configurar_TipoPixel
Call Configurar_Unidades
Call Configurar_DimensionImagen
Call Captura_Imagen_Conf(simage)

End Sub

'Esta es la subrutina que configura brillo
Public Sub Configurar_Brillo()

Dim dato As TWAIN_CAP_DATA
Dim rdato As Long

Call Iniciar_Brillo(dato)
rdato = TWsetbrightness(hwnd, dato)

End Sub

'Se inicia en un tipo TWAIN_CAP_DATA los datos y rangos definidos en el formulario
Public Sub Iniciar_Brillo(dato As TWAIN_CAP_DATA)

dato.conType = 6
dato.range.min = -1000
dato.range.max = 1000
dato.range.stepSize = 2000
dato.range.currentVal = formImageCaract.txtBrillo.Text
dato.range.defaultVal = formImageCaract.txtBrillo.Text

End Sub

'Esta es la subrutina que configura contraste
Public Sub Configurar_Contraste()

Dim dato As TWAIN_CAP_DATA
Dim rdato As Long

Call Iniciar_Contraste(dato)
rdato = TWsetcontrast(hwnd, dato)

End Sub

'Se inicia en un tipo TWAIN_CAP_DATA los datos y rangos definidos en el formulario
Public Sub Iniciar_Contraste(dato As TWAIN_CAP_DATA)
```

```
dato.conType = 6
dato.range.min = -1000
dato.range.max = 1000
dato.range.stepSize = 2000
dato.range.currentVal = formImageCaract.txtContraste.Text
'ato.range.defaultVal = formImageCaract.txtContraste.Text

End Sub

'Se configuran aqui las unidades con las que se va a medir la imagen

Public Sub Configurar_Unidades()

Dim dato As TWAIN_CAP_DATA
Dim rdato As Long

dato.conType = 5

Select Case formImageCaract.cmbunidades.Text

    Case "Inches"
        dato.oneVal = TWUN_INCHES
        radato = TWsetmeasureunit(hwnd, dato)

    Case "Centimeters"
        dato.oneVal = TWUN_CENTIMETERS
        radato = TWsetmeasureunit(hwnd, dato)

    Case "Twips"
        dato.oneVal = TWUN_TWIPS
        radato = TWsetmeasureunit(hwnd, dato)

    Case "Pixels"
        dato.oneVal = TWUN_PIXELS
        radato = TWsetmeasureunit(hwnd, dato)

End Select

End Sub

'Se configura aquí la dimensión o el tipo de píxel que se va a utilizar. En este caso escala
de grises

Public Sub Configurar_TipoPixel()

Dim dato As TWAIN_CAP_DATA
Dim rdato As Long

dato.conType = 5

Select Case formImageCaract.cmbunidades.Text

    Case "Black and White"
        dato.oneVal = TWPT_BW
        radato = TWsetpixeltype(hwnd, dato)

    Case "GrayScale"
        dato.oneVal = TWPT_GRAY
        radato = TWsetpixeltype(hwnd, dato)

    Case "RGB"
        dato.oneVal = TWPT_RGB
        radato = TWsetpixeltype(hwnd, dato)

    Case "Palette Color"
        dato.oneVal = TWPT_PALETTE
        radato = TWsetpixeltype(hwnd, dato)

    Case "CMY"
        dato.oneVal = TWPT_CMY
        radato = TWsetpixeltype(hwnd, dato)
    Case "YUV"

        dato.oneVal = TWPT_YUV
        radato = TWsetpixeltype(hwnd, dato)

End Select
```

```
End Sub

'Se implementa la función que sirve para capturar la imagen desde el dispositivo

Public Sub Captura_Imagen_Conf(simage As imgdes)

Dim srect As RECT
Dim showUI As Long
Dim rcode As Long

'Se marca la dimensión de la imagen que se quiere capturar
'En este caso al tratarse de una cámara digital la dimensión está en píxeles

srect.left = 0
srect.top = 0
srect.right = 100
srect.bottom = 100

'En principio no se va a mostrar el cuadro de diálogo de captura
showUI = 0
pageno = 0

'Capturamos la imagen a partir de la función definida en la librería

rcode = TWscanimageex(hwnd, simage, srect, showUI)
Call Visualizar_Imagen_conf(simage)

End Sub

Public Sub Visualizar_Imagen_conf(simage As imgdes)

'Aquí lo que se hace es dada la imagen se visualiza en el picture box

formImageCaract.piccaract = image_to_picturebox(simage)
formImageCaract.piccaract.Refresh

End Sub

Public Function image_to_picturebox(srcimg As imgdes) As Picture

Dim retval As Long
Dim Pic As PicBmp

Dim IPic As IPicture
Dim IID_IDispatch As Guid
Dim tempimage As imgdes

    If (srcimg.hBitmap = 0) Then
        'Con el paquete dib, se crea una sección tipo dib de la imagen

        rcode = dibtoimage(srcimg.bmh, tempimage)
            If rcode = NO_ERROR Then
                ' Reemplazamos la imagen previa
                freeimage srcimg
                copyimgdes tempimage, srcimg
            End If
    End If

    ' Se rellena con la IDispatch Interface ID.
    With IID_IDispatch
        .Data1 = &H20400
        .Data4(0) = &HC0
        .Data4(7) = &H46
    End With

    ' Rellenamos Pic con sus partes necesarias
    With Pic
        .Size = Len(Pic)           ' Longitud de la estructura
        .Type = vbPicTypeBitmap    ' Tipo de Picture.
        .hBmp = srcimg.hBitmap     ' Acceso al Bitmap
        .hPal = 0                  ' acceso a la paleta
    End With

    ' Se crea un objeto Picture Box
    retval = OleCreatePictureIndirect(Pic, IID_IDispatch, 1, IPic)

    If (retval = 0) Then
```

```
        Set image_to_picturebox = IPic
    End If

End Function

'Se describe  a continuación los tipos de datos que han sido definidos para determinar
'las características de la imagen que queremos capturar

Type TWAIN_CAP_DATA
   conType As Integer           ' Container type, TWON_ONEVALUE, TWON_ENUMERATION, or
TWON_RANGE,
   oneValue As TWAIN_ONEVALUE    ' Data if using ONEVALUE-type container
   enumType As TWAIN_ENUMTYPE    ' Data if using ENUM-type container
   range As TWAIN_RANGE          ' Data if using RANGE-type container
End Type

Type TWAIN_RANGE
   min As Integer           ' Starting value in the range
   max As Integer           ' Final value in the range
   stepSize As Integer      ' Increment from min to max
   currentVal As Integer    ' The value that is currently in effect
   defaultVal As Integer    ' Power-up value
End Type
```

J.2.3 formCalibracion.frm

```
Attribute VB_Name = "formCalibracion"

'La calibración servirá para minimizar la cantidad de datos a tratar

'Se capturar una imagen con dimensión definida y ubicarla en PictureBox

Private Sub cmdCapturaImag_Click()
    Dim im As imgdes
    'Proceso que captura una imagen con los parámetros establecidos
    Call Captura_Imagen(im)
    'Procedimeinto que sitúa imagen capturada en PictureBox
    Call Visualizar_Imagen(im)

End Sub

'Visualizan coordenada pulsada con mouse en cuadros de esquina pantalla
Private Sub cmdID_Click()

    mcord.cord(4).X = lblcordx.Caption
    mcord.cord(4).Y = lblcordy.Caption
    lbidx.Caption = mcord.cord(4).X
    lbidy.Caption = mcord.cord(4).Y

End Sub
'Visualizan coordenada pulsada con mouse en cuadros de esquina pantalla
Private Sub cmdII_Click()

    mcord.cord(3).X = lblcordx.Caption
    mcord.cord(3).Y = lblcordy.Caption
    lbliix.Caption = mcord.cord(3).X
    lbliiy.Caption = mcord.cord(3).Y

End Sub
'Visualizan coordenada pulsada con mouse en cuadros de esquina pantalla
Private Sub cmdSD_Click()

    mcord.cord(2).X = lblcordx.Caption
    mcord.cord(2).Y = lblcordy.Caption
    lblsdx.Caption = mcord.cord(2).X
    lblsdy.Caption = mcord.cord(2).Y

End Sub
'Visualizan coordenada pulsada con mouse en cuadros de esquina pantalla
Private Sub cmdSI_Click()

    mcord.cord(1).X = lblcordx.Caption
    mcord.cord(1).Y = lblcordy.Caption
    lblsix.Caption = mcord.cord(1).X
```

```
    lblsiy.Caption = mcord.cord(1).Y
End Sub
'Lo que hace es resetear los puntos calculados al considerarlos no óptimos
Private Sub cmdnuevospuntos_Click()
    Dim i As Integer

    lbidx.Caption = ""
    lbidy.Caption = ""
    lbliix.Caption = ""
    lbliiy.Caption = ""
    lblsdx.Caption = ""
    lblsdy.Caption = ""
    lblsix.Caption = ""
    lblsiy.Caption = ""
    txtx1.Text = ""
    txty1.Text = ""
    txtx2.Text = ""
    txty2.Text = ""

'Se borra también el contenido de las variables obtenidas para calcular factor escala
    For i = 1 To 4
        mcord.cord(i).X = 0
        mcord.cord(i).Y = 0
    Next i
End Sub

'Funcion que visualiza en referncia Picture las coordenadas del mouse y las visualiza

Private Sub PicCalibra_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
    lblcordx.Caption = X
    lblcordy.Caption = Y
End Sub

'Una vez se dan por buenos los datos capturados se calculan la posicion de los pixeles que se
analizarán de la imagen total capturada
Public Sub cmdcalibracionok_Click()

Dim ca As coordenadas
Dim cb As coordenadas
Dim caux1 As coordenadas
Dim caux2 As coordenadas


Dim fehoriz, fevert As Variant

Call Calcular_CoordenadasMon(ca, cb)

formIniCalibra.txtx1.Text = ca.X
formIniCalibra.txty1.Text = ca.Y
formIniCalibra.txtx2.Text = cb.X
formIniCalibra.txty2.Text = cb.Y

'Se debe calcular el factor de escala existente entre el PB y la imagen

'Para ello se divide el valor de la dimensión imagen capturada con el del PBox

fehoriz = Abs(576 / formIniCalibra.PicCalibra.ScaleWidth)
feverti = Abs(485 / formIniCalibra.PicCalibra.ScaleHeight)

'Una vez se conocen las coordenadas definitivas en referencia PB se calculan en referencia
imagen

caux1.X = fehoriz * ca.X
caux1.Y = fevert * ca.Y
caux2.X = fehoriz * cb.X
caux2.Y = fevert * cb.Y

Call actualizar(ca, cb, caux1, caux2)

End Sub

Private Sub Calibrar_Escala(ca As coordenadas, cb As coordenadas)

'Si se llega a este punto es que se han considerado las coordenadas como válidas y lo
'que se hace  ahora es calcular un factor de escala para la visualización del movimiento,
'teniendo en cuenta la dimensión en píxeles de este monitor visualizado y las dimensiones
'del control usado para visualizar el recorrido del tracking.
```

```
'Además ésta área servirá como para situar el monitor dentro de la imágen global y
'será útil para calcular la ROI y la situación inicial del puntero del ratón.

Dim fehoriz, fevert As Variant

Call Calcular_CoordenadasMon(ca, cb)

formIniCalibra.txtx1.Text = ca.X
formIniCalibra.txty1.Text = ca.Y
formIniCalibra.txtx2.Text = cb.X
formIniCalibra.txty2.Text = cb.Y

'Se debe calcular el factor de escala existente entre el PB y la imagen

'Para ello se divide el valor de la dimensión imagen capturada con el del PBox

fehoriz = Abs(576 / formIniCalibra.PicCalibra.ScaleWidth)
feverti = Abs(485 / formIniCalibra.PicCalibra.ScaleHeight)

'Una vez se conocen las coordenadas definitivas en referencia PB se calculan en referencia
imagen

cordI1.X = fehoriz * cord1.X
cordI1.Y = fevert * cord1.Y
cordI2.X = fehoriz * cord2.X
cordI2.Y = fevert * cord2.Y

'El valor de cordI serán las coordenadas imagen que se deberán capturar para el seguimiento.

End Sub
```

J.2.4 formConfiguracionImag

```
Attribute VB_Name = "formConfiguracionImag"

'En este código de formulario se definen las características que se van a poder elegir en cada
menu

Public brillo As Variant
Public contraste As Variant


Private Sub Form_Load()

'Se actualizan los diferentes campos de los controles de diálogo correspondientes

'Se actualiza el tipo de pixel
cmbpixelType.AddItem "Black and White"
cmbpixelType.AddItem "GrayScale"
cmbpixelType.AddItem "RGB"
cmbpixelType.AddItem "Palette Color"
cmbpixelType.AddItem "CMY"
cmbpixelType.AddItem "YUV"

'Se actualiza las unidades con que se capturara la imagen
cmbunidades.AddItem "Inches"
cmbunidades.AddItem "Centimeters"
cmbunidades.AddItem "Twips"
cmbunidades.AddItem "Pixels"

End Sub

'El brillo se muestra su valor mediante un ScrollBar
Private Sub hscrBrillo_Change()
txtBrillo.Text = hscrBrillo.value
End Sub

'El contraste se muestra su valor mediante un Scrollbar
Private Sub hscrcontraste_Change()
txtContraste.Text = hscrcontraste.value
End Sub
```

```
'Una vez validados los parámetros mediante botón de comando serán enviados a cámara.
Private Sub cmdConfigurar_Click()
Call Configurar_Caract_Imagen
End Sub
```

J.2.5 formConffiltro.frm

```
Private Sub scrFiltro_Change()
Call Evaluar
End Sub

'Lo que hace esta función es el % de imagen original que elegimos para ser tratada
Public Sub Evaluar()
lblaux.Caption = (scrFiltro.value / 800) * 100
End Sub

Public Sub EnviarFiltro(typefilt)
'Se envia el filtro elegido al SSE a través del puerto serie
Dim comando As Variant
Dim i As Integer

comando = "f"

'Lo primero que se hace es abrir el puerto según especificaciones
MSComm1.CommPort = 1
MSComm1.Settings = "256000,N,8,1"
MSComm1.PortOpen = True

'Enviamos ahora el comando que indica el filtro y los datos

MSComm1.Output = comando

'Se envían los 8 bytes que simbolizan el filtro
For i = 1 To 8
    MSComm1.Output = typefilt(i)
Next i

End Sub
```

J.3 Módulo de seguimiento

Este anexo comprende todos aquellos módulos que componen el proceso de seguimiento de la imagen, así como el proceso que sigue el sistema hasta la detección del puntero. Tanto los que se refieren al proceso de captura de la imagen, al envío y recepción de ésta del SSE, el seguimiento de los datos obtenidos y el proceso de actualización del objeto analizado.

A continuación, en la Tabla J.2, se describen las funciones y la nomenclatura de los módulos más significativos que se han utilizado para este bloque de código.

Nombre del código	Función
formSeguimiento.frm	Código asociado al formulario de seguimiento donde se presenta el código que responde a eventos en el mismo.
Seguimiento.bas	Módulo en el que se describe el proceso global de seguimiento que se va desglosando en diferentes niveles.
Segui_DetectarPuntero.bas	Módulo en el que se describe el proceso y funciones que sirven para detectar el puntero que es el que marcará el inicio del proceso de seguimiento.
Segui_DetectarPuntero_aux.bas	Módulo donde se describen las funciones auxiliares que complementan al módulo de detección del puntero.
Seguimiento_Cambio.bas	Módulo en el que se analiza si ha sucedido algún cambio en la porción de imagen de 48x48 que se quiere analizar.
Tracking.bas	Módulo en el que una vez ya se ha detectado la presencia del puntero se inicia el proceso de seguimiento
Tracking_aux.bas	Módulo en el que se ubican funciones y procedimientos auxiliares del módulo Tracking.bas
CapturarImagen.bas	Módulo auxiliar en el que se especifica el modo de capturar una imagen desde dispositivo de captura compatible con TWAIN.

Tabla J.2 Listados de los módulos de seguimiento.

J.3.1 formSeguimiento.frm

```
Attribute VB_Name = "formSeguimiento"

Private Sub cmddeletetrayectoria_Click()
'En este caso lo que se hará es parar el proceso
Parar_proceso = True
End Sub

Private Sub cmdinicaptura_Click()
'Con esta acción se inicia el proceso de seguimiento
Call Seguimiento
End Sub
```

```
Public Sub Seguimiento()
'Lo primero que se hace es detectar presencia de puntero que activará el seguimiento
Call Detectar_Puntero
End Sub
```

J.3.2 Seguimiento.bas

```
Attribute VB_Name = "Seguimiento"

Public Sub Detectar_Puntero()
    Dim presencia As Boolean
    Dim im1, im2 As imgdes
    Dim imaux1, imaux2 As image48
    Dim kiseki1, kiseki2 As datokiseki
'Con esto se tiene las coordenadas del rectángulo en referencia formulario seguimiento
    cordpic.X = formSeguimiento.shaperef.left
    cordpic.Y = formSeguimiento.shaperef.top
'Se inicializa el valor de la variable que indicará si el puntero aparece
    presencia = False
'Se captura la imagen que nos va a servir como referencia
    Call Captura_Imreferencia(im1)
    Do While presencia = False
    Call Calibracion.Captura_Imagen(im2)
'Ahora se envían las dos imágenes a Kiseki para que éste las devuelva transformadas
    Call ComKiseki(im1, im2, kiseki1, kiseki2)
'Miramos ahora si realmente se ha producido algún cambio en las dos imágenes
    If Existe_cambio(kiseki1, kiseki2) Then
        Call Calcular_Posición(cordgen, kiseki1, kiseki2)
        presencia = True
    End If
Loop
End Sub
```

J.3.3 Segui_DetectarPuntero.bas

```
Attribute VB_Name = "Segui_DetectarPuntero"

Public Sub Cogerim48(ByRef c As coordenadas, ByRef imaux As image48, ByRef im As imgdes)

Dim pixel As Byte

For i = 1 To 48
    For j = 1 To 48

    'Se captura de la imagen según la librería la imagen de 48x48 que deseamos
    imaux.pixel(i, j) = CalcularPixel(i, j, im, c)
    Next j
Next i

End Sub

Public Sub ComunicaKiseki(im As image48, kis As datokiseki)
'Se envia la imagen de 48x48 y recibir la imagen tratada desde Kiseki

Dim im16 As image16
Dim i, j, r, s As Integer
Dim cont As Integer
Dim b As Boolean

b = False


cont = 0
```

```
For r = 1 To 3
    For s = 1 To 3
        For i = 1 To 16
            For j = 1 To 16

                im16.pixel(i, j) = im.pixel(r * i, s * j)

                'Calculamos cuando se tiene una imagen de 16x16 y se envia
                cont = j * i * s * r
                b = (cont Mod 256 = 0)

                If b = True Then
                'Lo que se debe hacer es enviar la imagen y recepcionar según filtro
                'Pasando r i s se sabe en que cuadrante nos hallamos para almacenar datos

                Call enviar_Imagen(im16, kis, r, s)

                b = False

                End If
            Next j
        Next i
    Next s
Next r
End Sub


Public Sub ComKiseki(im1 As imgdes, im2 As imgdes, kis1 As datokiseki, kis2 As datokiseki)

    Dim imaux1, imaux2 As image48

    Call Cogerim48(cordpic, imaux1, im1)
    Call Cogerim48(cordpic, imaux2, im2)

    Call ComunicaKiseki(imaux1, kis1)
    Call ComunicaKiseki(imaux2, kis2)

End Sub
```

J.3.4 Segui_DetectarPuntero_aux.bas

```
Attribute VB_Name = "Segui_DetectarPuntero_aux"

'Aqui se va a implementar el tercer nivel en el análisis de funciones

'Funcion que dada una posición de la matriz 48x48 y las coordenadas globales devuelve el valor
del pixel

Public Function CalcularPixel(i As Integer, j As Integer, im As imgdes, caux As coordenadas)
As Long

'La función devuelve el valor de un píxel determinado
Dim pix As Long
Dim a, b As Long

'Se calcula la posición a partir de las coordenadas del rectángulo en imagen y las posiciones
que se avancen
a = c.X + i
b = c.Y + j

'Funcion que te calcula el valor del pixel de la imagen tipo Vic pasando las coordenadas
pix = getpixelcolor(im, a, b)

'Se devuelve el valor del pixel que se desea
CalcularPixel = pix

End Function

'Funcion que envia al sistema kiseki las imagenes capturadas y que recibe de éste la imagen
'transformada según especificaciones del filtro.

Public Sub enviar_Imagen(im16 As image16, kis As datokiseki, r As Integer, s As Integer)
```

```
Call enviar_16(im16)
Call recibir_16(kis, r, s)

End Sub

Public Sub enviar_16(im16 As image16)

    Dim i, j As Integer

    'Hay que ver el número de puerto serie que corresponde al PC que utilicemos
    formSeguimiento.MSComm1.CommPort = 2
    'Hay que ver los standards en velocidad de transmisión del puerto serie
    formSeguimiento.MSComm1.Settings = "256000,N,8,1"
    'Abrimos el puerto
    formSeguimiento.MSComm1.PortOpen = True
    'Se le indica que quiere enviarse Matriz de datos
    formSeguimiento.MSComm1.Output = "m"

    'Se envia datos de imagen byte a byte en el orden especificado

    For i = 1 To 16
        For j = 1 To 16
            formSeguimiento.MSComm1.Output = im16(i, j)
        Next j
    Next i

    'Una vez finalizado se cierra el puerto
    formSeguimiento.MSComm1.PortOpen = False

End Sub


'En este procedimiento se reciben los datos del sistema a partir de el filtro definido

Public Sub recibir_16(kis As datokiseki, r As Integer, s As Integer)
    Dim i As Integer

    'Esta es la función que captura datos que nos envia el sistema kiseki

    'Hay que ver el número de puerto serie que corresponde al PC que utilicemos
    formSeguimiento.MSComm1.CommPort = 2
    'Hay que ver los standards en velocidad de transmisión del puerto serie
    formSeguimiento.MSComm1.Settings = "256000,N,8,1"
    'Se abre el puerto
    formSeguimiento.MSComm1.PortOpen = True
    'Se le indica que quiere enviarse Matriz de datos
    formSeguimiento.MSComm1.Output = "M"
    'Ahora en principio se recibirán datos según sea el filtro ideado

    i = (FilterType) * (FilterType) * 4

    ReDim kis.cuadrante(r * s).datos(i)
    'Recibimos los datos a través del puerto serie en forma de paquete.
    kis.cuadrante(r * s).datos(i) = formSeguimiento.MSComm1.Input

End Sub
```

J.3.5 Seguimiento_Cambio.bas

```
Attribute VB_Name = "Seguimiento_Cambio"

'Aquí se va a determinar si ha existido cambio en la matriz de 48x48 y si ha habido donde ha
sucedido

Public Function Existe_cambio(kis1 As datokiseki, kis2 As datokiseki)

Dim b As Boolean
Dim i, j As Integer
Dim aux As Integer
Dim dato As Variant
```

```
b = False

For i = 1 To 9

'Se calcula el número de datos que van a tratarse en cada matriz de 16x16 según sea el filtro
    aux = (FilterType) * (FilterType) * 4
    For j = 1 To aux
        'Se calcula la diferencia entre ambos cuadrantes
      dato = kis1.cuadrante(i).datos(j) - kis2.cuadrante(i).datos(j)

        'Damos un valor para demostrar la existencia de cambio.

        If (dato > kis1.cuadrante(i).datos(j) / 10) Then

            b = True

        End If

        If b = True Then
            Exit For
        End If
    Next j

    If b = True Then
        Exit For
    End If

Next i

cuadrante = i
Existe_cambio = b
End Function

'Lo que hará este procedimiento es calcular en que punto de la región de interés se va a
iniciar el proceso de tracking

Public Sub Calcular_Posición(cord As coordenadas)
'Se sabe cuales son las coordenadas del rectángulo de referencia en referencia imagen
Select Case cuadrante
Case 1
    cord.X = cordpic.X
    cord.Y = cordpic.Y
Case 2
    cord.X = cordpic.X + 16
    cord.Y = cordpic.Y
Case 3
    cord.X = cordpic.X + 32
    cord.Y = cordpic.Y
Case 4
    cord.X = cordpic.X
    cord.Y = cordpic.Y + 16
Case 5
    cord.X = cordpic.X + 16
    cord.Y = cordpic.Y + 16
Case 6
    cord.X = cordpic.X + 32
    cord.Y = cordpic.Y + 16
Case 7
    cord.X = cordpic.X
    cord.Y = cordpic.Y + 32
Case 8
    cord.X = cordpic.X + 16
    cord.Y = cordpic.Y + 32
Case 9
    cord.X = cordpic.X + 32
    cord.Y = cordpic.Y + 32

End Select
End Sub
```

J.3.6 Tracking.bas

```
Attribute VB_Name = "Tracking"
'En este módulo se va a implementar el proceso de tracking
```

```
'Se parte de la posición dentro de la imagen de la esquina superior del
'cuadrante de 16x16 en el que se ha detectado movimiento.

Public Sub Seguir_Objeto()

Dim im1, im2, imaux As imgdes
Dim im48_1, im48_2, ima48_aux As image48
Dim kis1, kis2, kisaux As datokiseki
Dim movimiento As Boolean


'Se  inicia capturando dos imagenes de la secuencia de las mismas
Call Calibracion.Captura_Imagen(im1)
Call Calibracion.Captura_Imagen(im2)

'Una vez capturadas se selecciona la región de 48x48 que quiere analizarse
Call Cogerim48_Track(cordgen, im48_1, im1)
Call Cogerim48_Track(cordgen, im48_2, im2)

'Una vez capturada estas regiones se envian al SSE y se captura imágenes transformadas
Call Segui_DetectarPuntero.ComunicaKiseki(im48_1, kis1)
Call Segui_DetectarPuntero.ComunicaKiseki(im48_2, kis2)

'Se inicia propiamente el proceso de Tracking mientras no se pare

While Not (Parar_proceso)
'Debemos ver si ha existido cambio en las imagenes

    While Not (Existe_cambio(kis1, kis2))

    'Mientras no existe cambio lo que se hace es volver a comparar

    kisaux = kis2
    kis1 = kisaux

    Call Calibracion.Captura_Imagen(im2)
    Call Cogerim48_Track(cordgen, im48_2, im2)
    Call Segui_DetectarPuntero.ComunicaKiseki(im48_2, kis2)

    Wend

    'Si salimos de este bucle es que ha habido variación de cuadrante

    'Por un lado se debe calcular la nueva posición del cuadrante
    Call Seguimiento_Cambio.Calcular_Posición(cordgen)
    'Se visualizará el movimiento en la pantalla
    Call Ubicar_Posicion
    'Se vuelve a iniciar la captura de imágenes pero con nuevas posiciones
    'Aqui quizas se podría mejorar algo

    Call Calibracion.Captura_Imagen(im1)
    Call Calibracion.Captura_Imagen(im2)
    Call Cogerim48_Track(cordgen, im48_1, im1)
    Call Cogerim48_Track(cordgen, im48_2, im2)
    Call Segui_DetectarPuntero.ComunicaKiseki(im48_1, kis1)
    Call Segui_DetectarPuntero.ComunicaKiseki(im48_2, kis2)

Wend

End Sub
```

J.3.7 Tracking_aux

```
Attribute VB_Name = "Tracking_aux"

Public Function CalcularPixelTrack(i As Integer, j As Integer, im As imgdes, caux As
coordenadas) As Long
Dim pix As Long
Dim a, b As Long

'Lo que se hace es considerar que ya que el cuadrante de 16x16 estará en el centro de la nueva
región de búsqueda se desplaza el inicio de la esquina de ROI a buscar
```

```
a = c.X + i - 16
b = c.Y + j + 16

pix = getpixelcolor(im, a, b)
CalcularPixel = pix

End Function

Public Sub Cogerim48_Track(ByRef c As coordenadas, ByRef imaux As image48, ByRef im As imgdes)

Dim pixel As Byte

For i = 1 To 48
    For j = 1 To 48

    'Se captura de la imagen según la librería la imagen de 48x48 que deseamos
    imaux.pixel(i, j) = CalcularPixelTrack(i, j, im, c)
    Next j
Next i

End Sub

'Esta función lo que va a hacer es ubicar de forma aproximada la posición del puntero dentro
del recuadro de recorrido

Public Sub Ubicar_Posicion()
'Partiremos de las coordenadas generales que indicarán la nueva posición del objeto

'Lo primero que se debe hacer es pasar estas coordenadas generales en referncia imagen a
referencia formulario.

Dim cord, caux As coordenadas
Dim fehoriz, fevertical As Variant

caux.X = formSeguimiento.shapemov.left
caux.Y = formSeguimiento.shapemov.top


fehoriz = Abs((cordI1.X - cordI2.X) / formSeguimiento.width)
fevertical = Abs((cordI1.Y - cordI2.Y) / formSeguimiento.height)

cord.X = Abs(cordgen.X / fehoriz)
cord.Y = Abs(cordgen.Y / fevertical)

'Con esto se mueve el rectángulo a la nueva posición que se desea del puntero
Call formSeguimiento.shapemov.Move(cord.X, cord.Y, 16, 16)

'Se dibujará una línea de la trayectoria para visulizar recorrido

formSeguimiento.picseguimiento.DrawStyle = vbDot
formSeguimiento.picseguimiento.Line Step(cuax.X, caux.Y)-Step(cord.X, cord.Y)

End Sub
```

J.3.8 CapturarImagen.bas

```
Attribute VB_Name = "CapturarImagen"

'En este módulo lo que se hace es una vez configurados todos los parámetros que indican
'como debe realizarse el envío de la imagen que ya se ha configurado en los respectivos
'formularios, capturamos la imagen.

Public Sub Capturar_Imagen(ByRef dato As configenvio)

Dim aux_dato As configenvio
Dim rcode, rcode1, rcode2 As Long
Dim myimage As imgdes
Dim showUI As Long
Dim srect As RECT

aux_dato = dato
showUI = 0

Call Actualizar_ROI(srect)
```

```
    'rcode1 = TWopen(hwnd)
    rcode = TWscanimageex(hwnd, myimage, srect, showUI)

    If (rcode = NO_ERROR) Then

    Select Case aux_dato.Formato

    Case "BMP"
        rcode = savebmp("test.bmp", myimage, 0)
        formIniCalibra.PicCalibra.ScaleHeight = myimage.sty - myimage.endy
        formIniCalibra.PicCalibra.ScaleWidth = myimage.stx - myimage.endx


        formIniCalibra.PicCalibra = image_to_picturebox(myimage)
        formIniCalibra.PicCalibra.Refresh
       freeimage myimage

    Case "TIF"
        rcode = savetif("test.tif", myimage, 0)
        freeimage myimage
    Case "GIF"
        rcode = savetif("test.gif", myimage, 0)
        freeimage myimage
    Case "JPG"
        rcode = savetif("test.jpg", myimage, 0)
        freeimage myimage

    End Select

    End If

        ' Handle any errors
    If rcode <> NO_ERROR Then
        MsgBox "Ha habido un error en al captura de datos"
    End If



End Sub

Public Sub Actualizar_ROI(srect As RECT)

'Esto despúes se actualizará con el parámetro que enviemos.

srect.left = 1000
srect.top = 1000
srect.right = srect.left + 3000 - 1
srect.bottom = srect.top + 2000 - 1

End Sub
```

J.4 Tipos definidos

A parte de los módulos propiamente funcionales, han sido utilizados otros para la definición de tipos propios que han sido utilizados a lo largo del sistema.

J.4.1 Otrostipos.bas

```
Attribute VB_Name = "TiposPropios"
'Vamos a definir algunos tipos que nos vana ser muy útiles
Public typefilt(1 To 8) As Integer
Public FilterType As Integer

Type configenvio
        Formato As Variant
        noimag As Variant
End Type

Public dato As configenvio
```

```
Public Type coordenadas
    X As Variant
    Y As Variant
End Type


Public Type cordmonitor
    Cord(1 To 4) As coordenadas
End Type

Public mcord As cordmonitor
'Estas serán las coordendas que definirán la pantalla
Public cord1, cord2 As coordenadas
'Definimos como se quiere almacenar la información en imagenes
Public Type image48
    pixel(1 To 48, 1 To 48) As Bytes
End Type

Public Type image16
    pixel(1 To 16, 1 To 16) As Bytes
End Type

Public cordpic As coordenadas
Public cordobject As coordenadas

'El dato kiseki es un array dinámica para almacenar los datos que vienen del sistema kiseki

Public Type datokiseki
'puede recibir números con sigo por ello se asigna el tipo de datos Variant
cuadrante(1 To 9) As kiseki
End Type

Public Type kiseki
datos() As Bytes
End Type
```

K Software auxiliar

En este apartado se presentará el código fuente de algunas aplicaciones que han sido utilizadas de forma puntual durante el diseño o para la elaboración de algunos experimentos.

Nómbre del código	Función
`Genera_vhdl.c`	Este programa para plataforma PC se creó como ayuda para crear el código fuente VHDL que define la matriz sistólica. El resultado con algunas modificaciones manuales se puede ver en H.2.3.
`d_mayer_fft.c`	Conjunto de rutinas para el cálculo de la Transformada Rápida de Fourier y de Hartley desarrolladas por Ron Mayer (mayer@acuson.com). Estas rutinas fueron introducidas en el SSE para realizar los experimentos de comparación de velocidades mostrados en 4.7 (pág. 49 vol. memoria).

Tabla K.1 Listado de programas utilizado de forma puntual.

Por otro lado, para realizar el experimento cuyos resultados están mostrados en el capítulo 6 se desarrolló una aplicación para windows compuesto por el siguiente código.

Nómbre del código	Función
`main.c`	Código principal de la aplicación, encargado de comprimir y/o descomprimir una imagen en formato BMP de 8 bits por pixel en tonos de gris, de un tamaño fijo de 256x256 píxeles.
`bmp.c` `bmp.h`	Rutinas relacionadas con el tratamiento de imágenes en formato BMP.
`serie.c` `serie.h`	Rutinas relacionadas con el manejo del puerto de serie bajo plataformas Win32.

Tabla K.2 Listado del código utilizado en las pruebas del capítulo 6.

Por último también se mostrará el código del software utilizado para hacer el experimento del apartado A.5 del anexo A (Pág. 9).

Nómbre del código	Función
`main.c`	Código principal de la aplicación, encargado de comprimir y/o descomprimir una imagen en formato BMP de 8 bits por pixel en tonos de gris, de un tamaño fijo de 256x256 píxeles.
`bmp.c` `bmp.h`	Rutinas relacionadas con el tratamiento de imágenes en formato BMP.
`dht.c` `dht.h`	Rutinas relacionadas con el cálculo de la Transformada Discreta de Hartley mediante el método de los enteros algebraicos.
`polinomio.c` `polinomio.h`	Rutinas orientadas al tratamiento de polinomios.

Tabla K.3 Listado de código utilizado en las pruebas del Anexo A.

En las siguientes páginas se mostrará el código de los archivos listados hasta ahora según su orden de aparición.

K.1 Genera_vhdl.c

```
/* Genera_vhdl.c:
   Este programa se utilizará para generar un archivo VHDL que describa
   el bloque principal del sistema sistólico.
*/

#include <stdio.h>

#define N_FILS 16
#define N_COLS  4
#define N_TIEM 16

/*
   -------------------------------------
   Esto de a continuación es una cutrada. No lo repitáis en casa sin
   la supervisión de un adúltero X'DDDD.
   -------------------------------------
*/
#define M_m4 marcador[0]
#define M_m2 marcador[1]
#define M_m1 marcador[2]
#define M_0  marcador[3]
#define M_1  marcador[4]
#define M_2  marcador[5]
#define M_4  marcador[6]
/* ------------ Hasta aqui -------------*/


/* Tabla de coeficientes de la funcion cas */
const int P[N_FILS][N_COLS] = {      /* P[filas][columnas] */
        { 2, 0, 0, 0},         /* 2*cas( 0*PI/16) */
        { 0,-2, 0, 1},         /* 2*cas( 2*PI/16) */
        {-4, 0, 2, 0},         /* 2*cas( 4*PI/16) */
        { 0,-2, 0, 1},         /* 2*cas( 6*PI/16) */
        { 2, 0, 0, 0},         /* 2*cas( 8*PI/16) */
        { 0, 4, 0,-1},         /* 2*cas(10*PI/16) */
        { 0, 0, 0, 0},         /* 2*cas(12*PI/16) */
        { 0,-4, 0, 1},         /* 2*cas(14*PI/16) */
        {-2, 0, 0, 0},         /* 2*cas(16*PI/16) */
        { 0, 2, 0,-1},         /* 2*cas(18*PI/16) */
        { 4, 0,-2, 0},         /* 2*cas(20*PI/16) */
        { 0, 2, 0,-1},         /* 2*cas(22*PI/16) */
        {-2, 0, 0, 0},         /* 2*cas(24*PI/16) */
        { 0,-4, 0, 1},         /* 2*cas(26*PI/16) */
        { 0, 0, 0, 0},         /* 2*cas(28*PI/16) */
        { 0, 4, 0,-1}          /* 2*cas(30*PI/16) */
};

/* Información necesaria para cada elemento de proceso EP1 */

typedef struct {
        int a[N_TIEM];         /* Coeficiente en funcion del tiempo */
        int numero_operaciones; /* Número de operaciones distintas */
        int coef_max;          /* Máximo multiplicador */
        int rex_max,rex_min;   /* Rango máximo y mínimo de entrada X */
        int rsx_max,rsx_min;   /* Rango máximo y mínimo de salida X */
        int rey_max,rey_min;   /* Rango máximo y mínimo de entrada Y */
        int rsy_max,rsy_min;   /* Rango máximo y mínimo de salida Y */
} tipo_ep1;

/*
   Empezamos. Se ha optado por no hacer éste programa muy modular dada la
   pequeña extensión del mismo
*/

main () {
        unsigned int i,j,k,t;
        int indice_P, indice_ep1_a;
        int coef_max,num_coef;
        int marcador[7];


        tipo_ep1 ep1[N_COLS][N_FILS]; /* Matriz de ep1 */
```

```
/* CALCULO_EP1: INICIO ---------------------------------------- */

        /* Empezaremos rellenando las tablas 'a' para cada ep1 */
        /* RELLENO: INICIO ---------------------------------- */
        for (i=0; i<N_COLS; i++) {
                for (j=0; j<N_FILS; j++) {
                        /* Inicializamos a cero */
                        for (t=0; t<N_TIEM; t++) ep1[i][j].a[t]=0;
                }
        }


        /*for (k=0; k<N_TIEM; k++) {*/
         k=0;
         while (k<N_TIEM) {
                for (j=0; j<N_FILS; j++) {
                        indice_P=(k*j) % N_TIEM;
                        for (i=0; i<N_COLS; i++) {
                                indice_ep1_a=(k+i+j)%16;
                                /*if ((indice_ep1_a>=0) && (indice_ep1_a<=15))*/
                                ep1[i][j+1].a[indice_ep1_a]=P[indice_P][3-i];
                                printf("estoy aqui %i - %i %i \n",k,j,i);
                        };
                        printf("---- for J ---\n");
                };
                printf("-------- FOR T: %i -------\n",k);
                  k++;
        }
        /* RELLENO: FIN ---------------------------------------- */

        /* Ahora calcularemos la cantidad de operaciones distintas*/
        /* que ve cada ep1 así como el máximo coeficiente por el  */
        /* que hay que multiplicar                                */

        /* ANALISIS_COEFS: INICIO ------------------------------ */
        for (j=0; j<N_FILS; j++) {
                for (i=0; i<N_COLS; i++) {

                        coef_max = 0;
                        num_coef = 0;
                        M_m4 = 0;
                        M_m2 = 0;
                        M_m1 = 0;
                        M_0 = 0;
                        M_1 = 0;
                        M_2 = 0;
                        M_4 = 0;

                        for (t=0; t<N_TIEM; t++) {
                                coef_max=(ep1[i][j].a[t]>coef_max)?ep1[i][j].a[t]:coef_max;

                                switch (ep1[i][j].a[t]) {
                                        case -4: M_m4++; break;
                                        case -2: M_m2++; break;
                                        case -1: M_m1++; break;
                                        case  0: M_0++; break;
                                        case  1: M_1++; break;
                                        case  2: M_2++; break;
                                        case  4: M_4++; break;
                                }

                        }

                        for (k=0; k<7; k++) num_coef=num_coef+((marcador[k])?1:0);

                        ep1[i][j].numero_operaciones=num_coef;
                        ep1[i][j].coef_max=coef_max;
                }
        }
        /* ANALISIS_COEFS: FIN ---------------------------------- */

        /* Lo que nos queda por hacer ahora es calcular los rangos de entrada */
        /* y salida para cada uno de los ep1, a partir de los elementos previos */
        /* CALCULO_RANGOS: INICIO ------------------------------ */
        for (j=0;j<N_FILS; j++) {
                for (i=0; i<N_COLS; i++) {
                        ep1[i][j].rex_max=255; ep1[i][j].rex_min=0;
```

```
                        ep1[i][j].rsx_max=255; ep1[i][j].rsx_min=0;
                        if (!j) {
                                ep1[i][j].rey_max=ep1[i][j].rey_min=0;
                        } else {
                                ep1[i][j].rey_max=ep1[i][j-1].rsy_max;
                                ep1[i][j].rey_min=ep1[i][j-1].rsy_min;
                        }
                       ep1[i][j].rsy_max=ep1[i][j].rex_max*ep1[i][j].coef_max+ep1[i][j].rey_max;
                        ep1[i][j].rsy_min=-
ep1[i][j].rex_max*ep1[i][j].coef_max+ep1[i][j].rey_min;
                }
        }
        /* CALCULO_RANGOS: FIN ---------------------------------- */
/* CALCULO_EP1: FIN ----------------------------------------- */

/* Provisional: Para comparar con los resultados obtenidos con anterioridad */

for (j=0; j<N_FILS; j++) {
        for (i=0; i<N_COLS; i++) {
                printf("ep1(%i,%i): [",i,j);
                for (t=0; t<N_TIEM; t++) printf("%i,",ep1[i][j].a[t]);
                printf("] %i %i\n",ep1[i][j].numero_operaciones,ep1[i][j].coef_max);
        }
        printf("\n");
}

/* Hasta aquí */
return 0;
};
```

K.2 d_mayer_fft.c

```
/*
** FFT and FHT routines
**  Copyright 1988, 1993; Ron Mayer
**
**  mayer_fht(fz,n);
**      Does a hartley transform of "n" points in the array "fz".
**  mayer_fft(n,real,imag)
**      Does a fourier transform of "n" points of the "real" and
**      "imag" arrays.
**  mayer_ifft(n,real,imag)
**      Does an inverse fourier transform of "n" points of the "real"
**      and "imag" arrays.
**  mayer_realfft(n,real)
**      Does a real-valued fourier transform of "n" points of the
**      "real" array.  The real part of the transform ends
**      up in the first half of the array and the imaginary part of the
**      transform ends up in the second half of the array.
**  mayer_realifft(n,real)
**      The inverse of the realfft() routine above.
**
**
** NOTE: This routine uses at least 2 patented algorithms, and may be
**       under the restrictions of a bunch of different organizations.
**       Although I wrote it completely myself, it is kind of a derivative
**       of a routine I once authored and released under the GPL, so it
**       may fall under the free software foundation's restrictions;
**       it was worked on as a Stanford Univ project, so they claim
**       some rights to it; it was further optimized at work here, so
**       I think this company claims parts of it.  The patents are
**       held by R. Bracewell (the FHT algorithm) and O. Buneman (the
**       trig generator), both at Stanford Univ.
**       If it were up to me, I'd say go do whatever you want with it;
**       but it would be polite to give credit to the following people
**       if you use this anywhere:
**           Euler     - probable inventor of the fourier transform.
**           Gauss     - probable inventor of the FFT.
**           Hartley   - probable inventor of the hartley transform.
**           Buneman   - for a really cool trig generator
**           Mayer(me) - for authoring this particular version and
**                       including all the optimizations in one package.
**       Thanks,
**       Ron Mayer; mayer@acuson.com
```

```
**
*/

/* This is a slightly modified version of Mayer's contribution; write
* msp@ucsd.edu for the original code.  Kudos to Mayer for a fine piece
* of work.  -msp
*/

#ifdef MSW
#pragma warning( disable : 4305 )  /* uncast const double to float */
#pragma warning( disable : 4244 )  /* uncast double to float */
#pragma warning( disable : 4101 )  /* unused local variables */
#endif

#define REAL float
#define GOOD_TRIG

#ifdef GOOD_TRIG
#else
#define FAST_TRIG
#endif

#if defined(GOOD_TRIG)
#define FHT_SWAP(a,b,t) {(t)=(a);(a)=(b);(b)=(t);}
#define TRIG_VARS                                                \
      int t_lam=0;
#define TRIG_INIT(k,c,s)                                         \
     {                                                           \
      int i;                                                     \
      for (i=2 ; i<=k ; i++)                                     \
          {coswrk[i]=costab[i];sinwrk[i]=sintab[i];}                 \
      t_lam = 0;                                                 \
      c = 1;                                                     \
      s = 0;                                                     \
     }
#define TRIG_NEXT(k,c,s)                                         \
     {                                                           \
        int i,j;                                                     \
         (t_lam)++;                                              \
         for (i=0 ; !((1<<i)&t_lam) ; i++);                      \
         i = k-i;                                                \
         s = sinwrk[i];                                                \
         c = coswrk[i];                                                \
         if (i>1)                                                \
            {                                                    \
             for (j=k-i+2 ; (1<<j)&t_lam ; j++);                 \
             j        = k - j;                                          \
             sinwrk[i] = halsec[i] * (sinwrk[i-1] + sinwrk[j]);  \
             coswrk[i] = halsec[i] * (coswrk[i-1] + coswrk[j]);  \
            }                                                          \
     }
#define TRIG_RESET(k,c,s)
#endif

#if defined(FAST_TRIG)
#define TRIG_VARS                                   \
      REAL t_c,t_s;
#define TRIG_INIT(k,c,s)                            \
    {                                               \
     t_c  = costab[k];                               \
     t_s  = sintab[k];                               \
     c    = 1;                                       \
     s    = 0;                                       \
    }
#define TRIG_NEXT(k,c,s)                            \
    {                                                   \
     REAL t = c;                                        \
     c   = t*t_c - s*t_s;                           \
     s   = t*t_s + s*t_c;                           \
    }
#define TRIG_RESET(k,c,s)
#endif

static REAL halsec[20]=
    {
     0,
     0,
```

```
     .541196100146196984399723205366389420061072063378 01,
     .509795579104159168941939803987843913682618491908 93,
     .502419286188155705511670119280120922478593371939 63,
     .500602998235196301345504106766382396117586325995 91,
     .500150636020650988214771012710976584959749130103 40,
     .500037651915547722967781390779054928475031653983 45,
     .500009412535887756765128704691865335385231337579 83,
     .500002353106286080514012671712044089393262973764 26,
     .500000588274841178798685267309168049257806372761 81,
     .500000147068602148754637982838711982061791180932 51,
     .500000036767143778073158644006430203151034908839 72,
     .500000009191785522073665603488534553339391125693 80,
     .500000002297946354115628877679068685589919223489 20,
     .500000000574486586878733022351472724588122634013 72
    };
static REAL costab[20]=
    {
     .000000000000000000000000000000000000000000000000 00,
     .707106781186547524400844362104849039284835937688 47,
     .923879532511286756128183189396788286822416625863 64,
     .980785280403230449126182236134239036973933730893 33,
     .995184726672196886244836953109479921575474868729 85,
     .998795456205172392714771604759100694443203614704 61,
     .999698818696204220115765649666172196850061081257 72,
     .999924701839144540921646491196383224350606468802 21,
     .999981175282601142656990437728567716173917250944 33,
     .999995293809576171511580125700119899552987633622 18,
     .999998823451701909929025710171526019048267922889 76,
     .999999705862882219160228217738765677116263899349 30,
     .999999926465717851144731480707387856948201155688 92,
     .999999981616429293808346915402909714505076051242 78,
     .999999995404107312890971933139606148958894303189 45,
     .999999998851026827562673307794554108400537416194 28
    };
static REAL sintab[20]=
    {
     1.000000000000000000000000000000000000000000000000 00,
     .707106781186547524400844362104849039284835937688 46,
     .382683432365089771728459984030398866761344562485 61,
     .195090322016128267848284868477022240927691617751 95,
     .098017140329560601994195563888641845861136673167 49,
     .049067674327418014254954976942682658314745363025 74,
     .024541228522912288031734529459282925065466119239 44,
     .012271538285719926079408261951003212140372319591 76,
     .006135884649154475359640234590372580917057886317 38,
     .003067956762965976270145365490919842518944610213 44,
     .001533980186284765612303697150264079079954864575 22,
     .000766990318742704526938568357948576643140919452 05,
     .000383495187571395589072461681181381263395026034 95,
     .000191747597310703307439909561989000933468874033 85,
     .000095873799095977345870517210976476351187065612 84,
     .000047936899603066884549003990494658872746866687 68
    };
static REAL coswrk[20]=
    {
     .000000000000000000000000000000000000000000000000 00,
     .707106781186547524400844362104849039284835937688 47,
     .923879532511286756128183189396788286822416625863 64,
     .980785280403230449126182236134239036973933730893 33,
     .995184726672196886244836953109479921575474868729 85,
     .998795456205172392714771604759100694443203614704 61,
     .999698818696204220115765649666172196850061081257 72,
     .999924701839144540921646491196383224350606468802 21,
     .999981175282601142656990437728567716173917250944 33,
     .999995293809576171511580125700119899552987633622 18,
     .999998823451701909929025710171526019048267922889 76,
     .999999705862882219160228217738765677116263899349 30,
     .999999926465717851144731480707387856948201155688 92,
     .999999981616429293808346915402909714505076051242 78,
     .999999995404107312890971933139606148958894303189 45,
     .999999998851026827562673307794554108400537416194 28
    };
static REAL sinwrk[20]=
    {
     1.000000000000000000000000000000000000000000000000 00,
     .707106781186547524400844362104849039284835937688 46,
     .382683432365089771728459984030398866761344562485 61,
```

```
     .195090322016128267848284868477022240927691617751 95,
     .098017140329560601994195563888641845861136673167 49,
     .049067674327418014254954976942682658314745363025 74,
     .024541228522912288031734529459282925065466119239 44,
     .012271538285719926079408261951003212140372319591 76,
     .006135884649154475359640234590372580917057886317 38,
     .003067956762965976270145365490919842518944610213 44,
     .001533980186284765612303697150264079079954864575 22,
     .000766990318742704526938568357948576643140919452 05,
     .000383495187571395589072461681181381263395026034 95,
     .000191747597310703307439909561989000933468874033 85,
     .000095873799095977345870517210976476351187065612 84,
     .000047936899603066884549003990494658872746866687 68
    };


#define SQRT2_2   0.707106781186547524400844362104 84
#define SQRT2   2*0.707106781186547524400844362104 84

void mayer_fht(REAL *fz, int n)
{
/*  REAL a,b;
REAL c1,s1,s2,c2,s3,c3,s4,c4;
 REAL f0,g0,f1,g1,f2,g2,f3,g3; */
 int  k,k1,k2,k3,k4,kx;
 REAL *fi,*fn,*gi;
 TRIG_VARS;

 for (k1=1,k2=0;k1<n;k1++)
    {
     REAL aa;
     for (k=n>>1; (!((k2^=k)&k)); k>>=1);
     if (k1>k2)
        {
             aa=fz[k1];fz[k1]=fz[k2];fz[k2]=aa;
        }
    }
 for ( k=0 ; (1<<k)<n ; k++ );
 k  &= 1;
 if (k==0)
    {
         for (fi=fz,fn=fz+n;fi<fn;fi+=4)
            {
             REAL f0,f1,f2,f3;
             f1     = fi[0 ]-fi[1 ];
             f0     = fi[0 ]+fi[1 ];
             f3     = fi[2 ]-fi[3 ];
             f2     = fi[2 ]+fi[3 ];
             fi[2 ] = (f0-f2);
             fi[0 ] = (f0+f2);
             fi[3 ] = (f1-f3);
             fi[1 ] = (f1+f3);
            }
    }
 else
    {
         for (fi=fz,fn=fz+n,gi=fi+1;fi<fn;fi+=8,gi+=8)
            {
             REAL bs1,bc1,bs2,bc2,bs3,bc3,bs4,bc4,
               bg0,bf0,bf1,bg1,bf2,bg2,bf3,bg3;
             bc1     = fi[0 ] - gi[0 ];
             bs1     = fi[0 ] + gi[0 ];
             bc2     = fi[2 ] - gi[2 ];
             bs2     = fi[2 ] + gi[2 ];
             bc3     = fi[4 ] - gi[4 ];
             bs3     = fi[4 ] + gi[4 ];
             bc4     = fi[6 ] - gi[6 ];
             bs4     = fi[6 ] + gi[6 ];
             bf1     = (bs1 - bs2);
             bf0     = (bs1 + bs2);
             bg1     = (bc1 - bc2);
             bg0     = (bc1 + bc2);
             bf3     = (bs3 - bs4);
             bf2     = (bs3 + bs4);
             bg3     = SQRT2*bc4;
             bg2     = SQRT2*bc3;
             fi[4 ] = bf0 - bf2;
```

```
            fi[0 ] = bf0 + bf2;
            fi[6 ] = bf1 - bf3;
            fi[2 ] = bf1 + bf3;
            gi[4 ] = bg0 - bg2;
            gi[0 ] = bg0 + bg2;
            gi[6 ] = bg1 - bg3;
            gi[2 ] = bg1 + bg3;
           }
    }
 if (n<16) return;

 do
    {
     REAL s1,c1;
     int ii;
     k  += 2;
     k1  = 1  << k;
     k2  = k1 << 1;
     k4  = k2 << 1;
     k3  = k2 + k1;
     kx  = k1 >> 1;
         fi  = fz;
         gi  = fi + kx;
         fn  = fz + n;
         do
            {
             REAL g0,f0,f1,g1,f2,g2,f3,g3;
             f1      = fi[0 ] - fi[k1];
             f0      = fi[0 ] + fi[k1];
             f3      = fi[k2] - fi[k3];
             f2      = fi[k2] + fi[k3];
             fi[k2]  = f0       - f2;
             fi[0 ]  = f0       + f2;
             fi[k3]  = f1       - f3;
             fi[k1]  = f1       + f3;
             g1      = gi[0 ] - gi[k1];
             g0      = gi[0 ] + gi[k1];
             g3      = SQRT2  * gi[k3];
             g2      = SQRT2  * gi[k2];
             gi[k2]  = g0       - g2;
             gi[0 ]  = g0       + g2;
             gi[k3]  = g1       - g3;
             gi[k1]  = g1       + g3;
             gi     += k4;
             fi     += k4;
            } while (fi<fn);
     TRIG_INIT(k,c1,s1);
     for (ii=1;ii<kx;ii++)
        {
         REAL c2,s2;
          TRIG_NEXT(k,c1,s1);
          c2 = c1*c1 - s1*s1;
          s2 = 2*(c1*s1);
             fn = fz + n;
             fi = fz +ii;
             gi = fz +k1-ii;
             do
                {
                 REAL a,b,g0,f0,f1,g1,f2,g2,f3,g3;
                 b       = s2*fi[k1] - c2*gi[k1];
                 a       = c2*fi[k1] + s2*gi[k1];
                 f1      = fi[0 ]    - a;
                 f0      = fi[0 ]    + a;
                 g1      = gi[0 ]    - b;
                 g0      = gi[0 ]    + b;
                 b       = s2*fi[k3] - c2*gi[k3];
                 a       = c2*fi[k3] + s2*gi[k3];
                 f3      = fi[k2]    - a;
                 f2      = fi[k2]    + a;
                 g3      = gi[k2]    - b;
                 g2      = gi[k2]    + b;
                 b       = s1*f2     - c1*g3;
                 a       = c1*f2     + s1*g3;
                 fi[k2]  = f0        - a;
                 fi[0 ]  = f0        + a;
                 gi[k3]  = g1        - b;
                 gi[k1]  = g1        + b;
```

```
                b       = c1*g2     - s1*f3;
                a       = s1*g2     + c1*f3;
                gi[k2]  = g0        - a;
                gi[0 ]  = g0        + a;
                fi[k3]  = f1        - b;
                fi[k1]  = f1        + b;
                gi     += k4;
                fi     += k4;
               } while (fi<fn);
        }
     TRIG_RESET(k,c1,s1);
    } while (k4<n);
}

void mayer_fft(int n, REAL *real, REAL *imag)
{
 REAL a,b,c,d;
 REAL q,r,s,t;
 int i,j,k;
 for (i=1,j=n-1,k=n/2;i<k;i++,j--) {
  a = real[i]; b = real[j];  q=a+b; r=a-b;
  c = imag[i]; d = imag[j];  s=c+d; t=c-d;
  real[i] = (q+t)*.5; real[j] = (q-t)*.5;
  imag[i] = (s-r)*.5; imag[j] = (s+r)*.5;
 }
 mayer_fht(real,n);
 mayer_fht(imag,n);
}

void mayer_ifft(int n, REAL *real, REAL *imag)
{
 REAL a,b,c,d;
 REAL q,r,s,t;
 int i,j,k;
 mayer_fht(real,n);
 mayer_fht(imag,n);
 for (i=1,j=n-1,k=n/2;i<k;i++,j--) {
  a = real[i]; b = real[j];  q=a+b; r=a-b;
  c = imag[i]; d = imag[j];  s=c+d; t=c-d;
  imag[i] = (s+r)*0.5;  imag[j] = (s-r)*0.5;
  real[i] = (q-t)*0.5;  real[j] = (q+t)*0.5;
 }
}

void mayer_realfft(int n, REAL *real)
{
 REAL a,b,c,d;
 int i,j,k;
 mayer_fht(real,n);
 for (i=1,j=n-1,k=n/2;i<k;i++,j--) {
  a = real[i];
  b = real[j];
  real[j] = (a-b)*0.5;
  real[i] = (a+b)*0.5;
 }
}

void mayer_realifft(int n, REAL *real)
{
 REAL a,b,c,d;
 int i,j,k;
 for (i=1,j=n-1,k=n/2;i<k;i++,j--) {
  a = real[i];
  b = real[j];
  real[j] = (a-b);
  real[i] = (a+b);
 }
 mayer_fht(real,n);
}
```

K.3 Programa de test del SSE

K.3.1 main.c

```
#include <stdio.h>
#include <stdlib.h>
#include "serie.h"
#include "bmp.h"



void envia_buffer(char *buffer, int tamanyo) {
char *c,r[2];
int i,e,l;
for (i=0; i<tamanyo; i++) {
    c=buffer+i;
    WriteFile(hCom,c,1,&e,NULL);
    ReadFile(hCom,r,1,&l,NULL);
    Sleep(0);
}
}

void envia_sumas(int *sumas) {
int i;
unsigned char c[2];
envia_buffer("s",1);
for (i=0; i<16; i++) {
    c[0]=(unsigned char)(sumas[i]&0x00FF);
    c[1]=(unsigned char)((sumas[i]&0xFF00)>>8);
    envia_buffer(c,2);
}
}

int recibir_buffer_centinela(char *buffer) {
char r[2];
int l,j;
  ReadFile(hCom,r,1,&l,NULL);
  j=0;
  while (r[0]!='#') {
    buffer[j]=r[0];j++;
    ReadFile(hCom,r,1,&l,NULL);
  }
  buffer[j]=r[0]; j++;
  return(j);
}

int recibir_buffer_tamanyo(char *buffer, int tamanyo) {
char r[2];
int l,j;
  j=0;
  while (j<tamanyo) {
    FlushFileBuffers(hCom);
    ReadFile(hCom,r,1,&l,NULL);
    buffer[j]=r[0];j++;
  }
  return(j);

/*    ReadFile(hCom,buffer,tamanyo,&l,NULL);
    return(l);*/
}


void comprimir_region(unsigned char *origen, char *destino) {
int i,j,k;
char c;
int a_arreglar[256];
unsigned char buffer_entrada[514];
/*      offset_region(origen,20);*/
      envia_buffer("c",1);
      envia_buffer(origen,256);
      j=recibir_buffer_centinela(buffer_entrada);
      envia_buffer("C",1);
      j=recibir_buffer_tamanyo(buffer_entrada,513);
      for (i=0; i<256; i++) {
```

```
            if (buffer_entrada[i*2+1]&0x80) {
                        k=0xFFFF0000;
            } else k=0;
            k= k | (buffer_entrada[i*2+1]<<8) | (buffer_entrada[i*2]);
            a_arreglar[i]=k/32;
            if (a_arreglar[i]>128) {a_arreglar[i]=128;} else
            if (a_arreglar[i]<-127) {a_arreglar[i]=-127;}
            destino[i]=(char)a_arreglar[i];
      }
}

void descomprimir_region(char *origen, unsigned char *destino) {
int i,j,k;
int a_arreglar[256];
unsigned char buffer_entrada[514],arreglado[256];
int sumas[16];
      arreglar_region_envio(origen,arreglado,sumas);
      envia_sumas(sumas);
      j=recibir_buffer_centinela(buffer_entrada);
      envia_buffer("d",1);
      envia_buffer(arreglado,256);
      j=recibir_buffer_centinela(buffer_entrada);
      envia_buffer("D",1);
      j=recibir_buffer_tamanyo(buffer_entrada,513);
      for (i=0; i<256; i++) {
            if (buffer_entrada[i*2+1]&0x80) {
                        k=0xFFFF0000;
            } else k=0;
            k= k | (buffer_entrada[i*2+1]<<8) | (buffer_entrada[i*2]);

            a_arreglar[i]=k/2;
            if (a_arreglar[i]>255) {a_arreglar[i]=255;} else
            if (a_arreglar[i]<0) {a_arreglar[i]=0;}
            destino[i]=(unsigned char)a_arreglar[i];
      }
}

void escala(unsigned char *bitmap, float fescala) {
int i;
float j;
    for (i=0; i<TAM_IMAGEN; i++) {
        bitmap[i]=(unsigned char)((float)(bitmap[i])*fescala);
    }
}

int main(int argc, char *argv[])
{
  FILE *origen,*destino;
  int x,y,i,j;
  unsigned char b_entrada[4096];
  unsigned char cabecera_bmp[TAM_CABECERA],paleta_bmp[TAM_PALETA];
  unsigned char bitmap_origen[TAM_IMAGEN], bitmap_destino[TAM_IMAGEN];
  char cs_region[256], cs_salida[256];
  unsigned char ss_region[256], ss_salida[256], aux[256];

  if (argc>1) {
    if (!(origen=fopen(argv[1],"rb"))) {
        printf("No puedo abrir el fichero %s\n",argv[1]);
    }
  } else {
    printf("Indica el fichero bmp que quieres pasar como parámetro\n");
  }

#define DESCOMPRIME

  /* Suponemos que es un bmp en tonos de gris de 256x256 pixeles, 8bpp */

  fread(cabecera_bmp,1,TAM_CABECERA,origen);
  fread(paleta_bmp,1,TAM_PALETA,origen);
  fread(bitmap_origen,1,TAM_IMAGEN,origen);

  #ifdef COMPRIME
    escala(bitmap_origen,0.25);
  #endif

  fclose(origen);
```

```
  inicializar_serie();

  envia_buffer("f\xFF\xFF\xFF\xFF\xFF\xFF\xFF\xFF",9);
  recibir_buffer_centinela(b_entrada);

  for (y=0; y<16; y++) {
    for (x=0; x<16; x++) {



    #ifdef DESCOMPRIME
        leer_region(x,y,bitmap_origen,cs_region);
        filtrar_region(cs_region);
        descomprimir_region(cs_region,ss_salida);
        printf("D ---> %02i,%02i<-----\n",x,y);
        escribir_region(x,y,bitmap_destino,ss_salida);
    #else
        leer_region(x,y,bitmap_origen,ss_region);
        comprimir_region(ss_region,cs_salida);
        printf("C ---> %02i,%02i<-----\n",x,y);
        escribir_region(x,y,bitmap_destino,cs_salida);
    #endif
    }
    printf("\n");
  }

  #ifdef DESCOMPRIME
    escala(bitmap_destino,4);
  #endif


  destino=fopen("salida.bmp","wb");
  fwrite(cabecera_bmp,1,TAM_CABECERA,destino);
  fwrite(paleta_bmp,1,TAM_PALETA,destino);
  fwrite(bitmap_destino,1,TAM_IMAGEN,destino);
  fclose(destino);

  envia_buffer("?",1);
  j=recibir_buffer_centinela(b_entrada);
  for (i=0; i<j; i++) {
    if (b_entrada[i]==0x0D) { putchar('\n'); } else
    putchar(b_entrada[i]);
  }


/*  envia_buffer("v\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00",17);
  envia_buffer("v\x02\x02\x02\x02\x02\x02\x02\x02\x02\x02\x02\x02\x02\x02\x02\x02",17);
  j=recibir_buffer_centinela(buffer_entrada);*/

  finalizar_serie();

/*  system("PAUSE");  */
  return 0;
}
```

K.3.2 bmp.h

```
#ifndef BMP_H
#define BMP_H

#define TAM_CABECERA 54
#define TAM_PALETA 1024
#define TAM_IMAGEN 65536
#define ANCHO_IMAGEN 256
#define ALTO_IMAGEN ANCHO_IMAGEN

#define BIT_COUNT 28
#define BIT_COMPR 30

void leer_region(int x, int y, char *origen, char *buffer);
void escribir_region(int x, int y, char *destino, char *buffer);
void arreglar_region_envio(char *region, unsigned char *resultado, int *sumas);
void filtrar_region(char *buffer);
void traspuesta(char *ro, char *rd);
```

```
void offset_region(unsigned char *region, unsigned char ofs);

#endif
```

K.3.3 bmp.c

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

#include "bmp.h"

#ifndef FISTRO
#define FISTRO
const char fistro[256] = {
1,0,0,0, 0,0,0,0, 0,0,0,0, 0,0,0,1,
0,0,0,0, 0,0,0,0, 0,0,0,0, 0,0,0,0,
0,0,0,0, 0,0,0,0, 0,0,0,0, 0,0,0,0,
0,0,0,0, 0,0,0,0, 0,0,0,0, 0,0,0,0,

0,0,0,0, 0,0,0,0, 0,0,0,0, 0,0,0,0,
0,0,0,0, 0,0,0,0, 0,0,0,0, 0,0,0,0,
0,0,0,0, 0,0,0,0, 0,0,0,0, 0,0,0,0,
0,0,0,0, 0,0,0,0, 0,0,0,0, 0,0,0,0,

0,0,0,0, 0,0,0,0, 0,0,0,0, 0,0,0,0,
0,0,0,0, 0,0,0,0, 0,0,0,0, 0,0,0,0,
0,0,0,0, 0,0,0,0, 0,0,0,0, 0,0,0,0,
0,0,0,0, 0,0,0,0, 0,0,0,0, 0,0,0,0,

0,0,0,0, 0,0,0,0, 0,0,0,0, 0,0,0,0,
0,0,0,0, 0,0,0,0, 0,0,0,0, 0,0,0,0,
0,0,0,0, 0,0,0,0, 0,0,0,0, 0,0,0,0,
1,0,0,0, 0,0,0,0, 0,0,0,0, 0,0,0,1};
#endif

void leer_region(int x, int y, char *origen, char *buffer) {
int i,j;
    for (i=0; i<16; i++) {
        for (j=0; j<16; j++) {
                buffer[16*i+j]=origen[((16-y)*16-i)*256+(16*x+j)];
        }
    }
};

void escribir_region(int x, int y, char *destino, char *buffer) {
int i,j;
    for (i=0; i<16; i++) {
        for (j=0; j<16; j++) {
                destino[((16-y)*16-i)*256+(16*x+j)]=buffer[16*i+j];
        }
    }
};

void arreglar_region_envio(char *region, unsigned char *resultado, int *sumas) {
int i,j,m,s;
    for (j=0; j<16; j++) {
        s=0; m=(int)region[j*16];
        for (i=0; i<16; i++) {
                if ((int)region[j*16+i]<m) m=(int)region[j*16+i];
                s+=(int)region[j*16+i];
        }
        sumas[j]=s*2;
        for (i=0; i<16; i++) {
                resultado[j*16+i]=(unsigned char)((int)region[j*16+i]-m);
        }
    }
}

void offset_region(unsigned char *region, unsigned char ofs) {
int i;
    for (i=0; i<255; i++) {
        if (region[i]<ofs) region[i]=0; else region[i]-=ofs;
```

```
    }
}


void traspuesta(char *ro, char *rd) {
int i,j;
for (j=0; j<16; j++) {
    for (i=0; i<16; i++) {
         rd[i*16+j]=ro[j*16+i];
    }
}

}

void filtrar_region(char *buffer) {
int i,j;
    for (i=0; i<255; i++) {
        if (!fistro[i]) {buffer[i]=0;}
    }
}
```

K.3.4 serie.h

```
/* Estas serán las rutinas básicas para el manejo del puerto
   de serie
   Miguel Ángel Sánchez López, 11/06/2004 */

#ifndef SERIE_H
#define SERIE_H
#include <windows.h>

DCB dcb;
HANDLE hCom;

void inicializar_serie(void);
void finalizar_serie(void);
#endif
```

K.3.5 serie.c

```
#include <windows.h>
#include "serie.h"

//
//Estas variables globales de aquí debajo son una estructura
//para configurar el puerto de serie (dcb) y un manipulador de
//ficheros de windows.
//
//Más informacón en
//http://msdn.microsoft.com/library/default.asp?url=/library/en-us/devio/base/dcb_str.asp
//

void inicializar_serie(void) {
    //Creamos el manipulador del fichero "COM2"
    //Más información en
    //http://msdn.microsoft.com/library/default.asp?url=/library/en-
us/fileio/base/createfile.asp
    hCom = CreateFile(
        "COM1",
        GENERIC_READ | GENERIC_WRITE,
        0,
        NULL,
        OPEN_EXISTING,
        FILE_FLAG_WRITE_THROUGH | FILE_FLAG_NO_BUFFERING,
        NULL);

    //Obtengo las propiedades actuales del puerto de serie
    GetCommState(hCom, &dcb);

    //Configuro los parámetros iniciales de la impresora
```

```
    dcb.BaudRate      = CBR_115200;
    dcb.ByteSize      = 8;
    dcb.Parity        = NOPARITY;
    dcb.StopBits      = TWOSTOPBITS;
    /*dcb.StopBits      = ONESTOPBIT;*/

    //Y finalmente guardamos la confirguración para el manipulador hCom.
    SetCommState(hCom, &dcb);
}


void finalizar_serie(void) {
    CloseHandle(hCom);
}
```

K.4 Software utilizado en las pruebas del Anexo A

K.4.1 main.c

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

#include "bmp.h"
#include "dht.h"

int main(int argc, char *argv[]) {

/*unsigned*/ char cabecera[TAM_CABECERA], paleta[TAM_PALETA], imagen[TAM_IMAGEN],
imagen2[TAM_IMAGEN];
int base[TAM_IMAGEN], base2[TAM_IMAGEN]; /*matrices de enteros auxiliares*/

/*char vector[256];
char vector2[32];*/

int vector[256];
int vector2[32];

int f,i,j,k;
int direccion, tamanyo, ab;
        if (argc>4) {
/* LECTURA */

                if ((argv[2][0]!='0') && (argv[2][0]!='1'))
                {
                        printf("El parámetro de dirección no es correcto\n\n");
                        goto INSTRUCCIONES;
                } else
                {
                   if (argv[2][0]=='0') { direccion=0; } else { direccion=1; }
                }

                if ((argv[3][0]!='0') && (argv[3][0]!='1'))
                {
                        printf("El formato de tamaño no es correcto\n\n");
                        goto INSTRUCCIONES;
                } else
                {
                   if (argv[3][0]=='0') { tamanyo=0; } else { tamanyo=1; }
                }


                sscanf(argv[4],"%i",&ab);
                if ((tamanyo && (ab>16)) || (!(tamanyo) && (ab>8))) {
                        printf("El ancho de banda debe ser menor o igual que la mitad del
tamaño\n\n");
                        goto INSTRUCCIONES;
                }
```

```
                if (f=open(argv[1],O_RDONLY)) {
                        k=carga_bmp(f,cabecera,paleta,imagen);
                        close(f);
                        if (!k) { printf("El archivo bmp no tiene un formato válido\n");
return(0); }
                }
                else
                {
                        printf("No se puede abrir el fichero %s\n",argv[1]);
                        return(0);
                }

/* FIN LECTURA */

/* TRATAMIENTO */

#define TV (tamanyo?32:16)
#define NUMSEG (tamanyo?8:16)

#define E1 1
#define E2 1
#define E3 1
#define E4 1


/* Esto lo anyado hoy 29/7/04 */
        if (direccion) {
                presaturacion((unsigned char *)imagen,239,20)
                escala_ch2i(imagen,base,E1);
        } else {
                escala_ch2i(imagen,base,E2);
        }
/* Hasta aqui */

                for (i=0; i<256; i++) {
                        dame_fila(i,base,vector);
                        for (j=0; j<NUMSEG; j++) {
                                for (k=0; k<TV;k++) { vector2[k]=vector[j*TV+k]; }
                                if (tamanyo) { dht32(vector2,direccion); } else
{ dht(vector2,direccion); }
                                /* FILTRO PASABAJOS */
                                if (ab) {
                                        for (k=ab; k<TV-ab; k++) { vector2[k]=0; }
                                }
                                /* HASTA AQUI */
                                for (k=0; k<TV;k++) { vector[j*TV+k]=vector2[k]; }
                        }
                        toma_fila(i,base2,vector);
                }

                for (i=0; i<256; i++) {
                        dame_columna(i,base2,vector);
                        for (j=0; j<NUMSEG; j++) {
                                for (k=0; k<TV;k++) { vector2[k]=vector[j*TV+k]; }
                                if (tamanyo) { dht32(vector2,direccion); } else
{ dht(vector2,direccion); }
                                /* FILTRO PASABAJOS */
                                if (ab) {
                                        for (k=ab; k<TV-ab; k++) { vector2[k]=0; }
                                }
                                /* HASTA AQUI */
                                for (k=0; k<TV;k++) { vector[j*TV+k]=vector2[k]; }
                        }
                        toma_columna(i,base2,vector);
                }

/* Esto lo anyado hoy 29/7/04 */
        if (direccion) {
                 escala_i2ch(imagen2,base2,E3);
        } else {
                 escala_i2ch(imagen2,base2,E4);
        }
/* Hasta aqui */

#undef NUMSEG
#undef TV
```

```
/* FIN TRATAMIENTO */

/* ALMACENAMIENTO */

                if (f=open("kk.bmp",O_CREAT|O_WRONLY)) {
                        k=guarda_bmp(f,cabecera,paleta,imagen2);
                        close(f);
                        printf("hecho! --> %i %i\n",(tamanyo?32:16),direccion);
                } else
                printf("No puedo crear el archivo\n");

/* FIN ALMACENAMIENTO */

        } else {

INSTRUCCIONES:

                printf("Modo de empleo:\n\n\t%s nombre.bmp direccion tamanyo
ab\n\nnombre.bmp\t:\tnombre del fichero bmp\ndireccion\t:\t0 si inversa, 1 si
directa\ntamanyo\t\t:\t0 si 16, 1 si 32\nab\t\t:\tancho de banda del filtro pasabajos
ideal.\n\t\t\tEl valor maximo es LA MITAD DEL TAMAÑO de muestra.\n\t\t\tSi vale 0, no habrá
filtro.\n",argv[0]);
                return(0);
        }

        return (1);
}
```

K.4.2 bmp.h

```
#ifndef BMP_H
#define BMP_H

/* bmp.h : Aquí se intentarán definir unos cuantos tipos de datos
           y los prototipos de algunas rutinas útiles para el tratamiento
           de ficheros bmp de ocho bits no comprimidos.

           Lo único que pretendo es volcar el dibujo en un array y vice-
           versa. A parte quiero poder modificar la paleta de colores
           a mi gusto.

           Voy a definir una paleta estandar que sea azul-verde-rojo, según
           el valor que tenga el pixel en cuestión.
*/

#define TAM_CABECERA 54
#define TAM_PALETA 1024
#define TAM_IMAGEN 65536
#define ANCHO_IMAGEN 256
#define ALTO_IMAGEN ANCHO_IMAGEN

#define BIT_COUNT 28
#define BIT_COMPR 30

short int  carga_bmp(int f, char *cabecera, char *paleta, char *imagen);
short int guarda_bmp(int f, char *cabecera, char *paleta, char *imagen);

void escala_ch2i(char *mch, int *mint, float escala);
void escala_i2ch(char *mch, int *mint, float escala);
void presaturacion(unsigned char *matriz, unsigned char maximo, unsigned char minimo);

void dame_fila(int fila, int *imagen, int *vector);
void dame_columna(int columna, int *imagen, int *vector);
void toma_fila(int fila, int *imagen, int *vector);
void toma_columna(int columna, int *imagen, int *vector);


#endif
```

K.4.3 bmp.c

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

#include "bmp.h"


/*
  Esto se supone que debe cargar un archivo bmp en estos tres trozos. Se supone que el archivo
  está previamente abierto. Si el bmp no es de 8 bits o si está comprimido, entonces devolverá
  error.
*/
short int carga_bmp(int f, char *cabecera, char *paleta, char *imagen) {
        if (read(f,cabecera,TAM_CABECERA)==TAM_CABECERA) {
                if ((cabecera[BIT_COUNT+1]=!8) || (cabecera[BIT_COMPR+3]!=0))
                        { return(0); }
        }
        else { return(0); }

        if (read(f,paleta,TAM_PALETA)!=TAM_PALETA) { return(0); }

        if (read(f,imagen,TAM_IMAGEN)!=TAM_IMAGEN) { return(0); }

        return(1);
};

/*
  Esto de a continuación es el complementario del anterior. No se realiza ninguna comprobación
respecto
  al formato.
*/
short int guarda_bmp(int f, char *cabecera, char *paleta, char *imagen) {
        write(f,cabecera,TAM_CABECERA);
        write(f,paleta,TAM_PALETA);
        write(f,imagen,TAM_IMAGEN);
        return(1);
}

void dame_fila(int fila, int *imagen, int *vector) {
int i;
int offset=ANCHO_IMAGEN*fila;
        for (i=0; i<ANCHO_IMAGEN; i++) { vector[i]=imagen[offset+i]; }
}

void dame_columna(int columna, int *imagen, int *vector) {
int i;
        for (i=0; i<ALTO_IMAGEN; i++) { vector[i]=imagen[columna+i*ANCHO_IMAGEN]; }
}

void toma_fila(int fila, int *imagen, int *vector) {
int i;
int offset=ANCHO_IMAGEN*fila;
        for (i=0; i<ANCHO_IMAGEN; i++) { imagen[offset+i]=vector[i]; }
}

void toma_columna(int columna, int *imagen, int *vector) {
int i;
        for (i=0; i<ALTO_IMAGEN; i++) { imagen[columna+i*ANCHO_IMAGEN]=vector[i]; }
}

void escala_ch2i(char *mch, int *mint, float escala) {
int i;

        for (i=0; i<TAM_IMAGEN; i++) { mint[i]=(int)(((float)mch[i])*escala);}
}

void escala_i2ch(char *mch, int *mint, float escala) {
int i;
float j;
        for (i=0; i<TAM_IMAGEN; i++) {
                j=(float)(mint[i])*escala;
                /*if (j<-127.0) { j=-127.0; } else
```

```
                if (j>128.0) { j=128.0; }*/
                mch[i]=(char)j;
        }
}

void presaturacion(unsigned char *matriz, unsigned char maximo, unsigned char minimo) {
int i;
        for (i=0; i<TAM_IMAGEN; i++) {
                if (matriz[i]>maximo) {matriz[i]=maximo;}
                if (matriz[i]<minimo) {matriz[i]=minimo;}
        }
}
```

K.4.4 dht.h

```
#ifndef DHT_H
#define DHT_H

#define TAM 16
#define TAM2 32
#define Z 1.84765625
#define Z2 1.9609375

#endif
```

K.4.5 dht.c

```
#include <stdio.h>
#include "dht.h"
#include "polinomio.h"

const int cas_16[TAM][MAX_DIM_POLINOMIO] = {
        { 2, 0, 0, 0 },        /* 2cas( 0 * PI / 16) */
        { 0,-2, 0, 1 },        /* 2cas( 2 * PI / 16) */
        {-4, 0, 2, 0 },        /* 2cas( 4 * PI / 16) */
        { 0,-2, 0, 1 },        /* 2cas( 6 * PI / 16) */
        { 2, 0, 0, 0 },        /* 2cas( 8 * PI / 16) */
        { 0, 4, 0,-1 },        /* 2cas(10 * PI / 16) */
        { 0, 0, 0, 0 },        /* 2cas(12 * PI / 16) */
        { 0,-4, 0, 1 },        /* 2cas(14 * PI / 16) */
        {-2, 0, 0, 0 },        /* 2cas(16 * PI / 16) */
        { 0, 2, 0,-1 },        /* 2cas(18 * PI / 16) */
        { 4, 0,-2, 0 },        /* 2cas(20 * PI / 16) */
        { 0, 2, 0,-1 },        /* 2cas(22 * PI / 16) */
        {-2, 0, 0, 0 },        /* 2cas(24 * PI / 16) */
        { 0,-4, 0, 1 },        /* 2cas(26 * PI / 16) */
        { 0, 0, 0, 0 },        /* 2cas(28 * PI / 16) */
        { 0, 4, 0,-1 } /* 2cas(30 * PI / 16) */
};

const int cas_32[TAM2][MAX_DIM_POLINOMIO] = {
        { 2, 0, 0, 0 }, /* 2cas( 0 * PI / 32) */
        {-4, 4,-4, 2 }, /* 2cas( 2 * PI / 32) */
        { 0,-2, 0, 1 }, /* 2cas( 4 * PI / 32) */
        {-7, 3,-8, 5 }, /* 2cas( 6 * PI / 32) */
        {-4, 0, 2, 0 }, /* 2cas( 8 * PI / 32) */
        {-7, 3,-8, 5 }, /* 2cas(10 * PI / 32) */
        { 0,-2, 0, 1 }, /* 2cas(12 * PI / 32) */
        {-4, 4,-4, 2 },        /* 2cas(14 * PI / 32) */
        { 2, 0, 0, 0 },        /* 2cas(16 * PI / 32) */
        {-1,-6, 4, 0 },        /* 2cas(18 * PI / 32) */
        { 0,-4, 0, 1 },        /* 2cas(18 * PI / 32) */
        {-7,-7, 6, 0 },        /* 2cas(20 * PI / 32) */
        { 0, 0, 0, 0 },        /* 2cas(22 * PI / 32) */
        {-7,-7, 6, 0 },        /* 2cas(24 * PI / 32) */
        { 0,-4, 0, 1 },        /* 2cas(26 * PI / 32) */
        {-1,-6, 4, 0 },        /* 2cas(28 * PI / 32) */
        {-2, 0, 0, 0 },        /* 2cas(30 * PI / 32) */
        { 4,-4, 4,-2 },        /* 2cas(32 * PI / 32) */
```

```
        { 0, 2, 0,-1 },        /* 2cas(34 * PI / 32) */
        { 7,-3, 8,-5 },        /* 2cas(36 * PI / 32) */
        { 4, 0,-2, 0 },        /* 2cas(38 * PI / 32) */
        { 7,-3, 8,-5 },        /* 2cas(40 * PI / 32) */
        { 0, 2, 0,-1 },        /* 2cas(42 * PI / 32) */
        { 4,-4, 4,-2 },        /* 2cas(44 * PI / 32) */
        {-2, 0, 0, 0 },        /* 2cas(46 * PI / 32) */
        { 1, 6,-4, 0 },        /* 2cas(48 * PI / 32) */
        { 0, 4, 0,-1 },        /* 2cas(50 * PI / 32) */
        { 7, 7,-6, 0 },        /* 2cas(52 * PI / 32) */
        { 0, 0, 0, 0 },        /* 2cas(54 * PI / 32) */
        { 7, 7,-6, 0 },        /* 2cas(56 * PI / 32) */
        { 0, 4, 0,-1 },        /* 2cas(58 * PI / 32) */
        { 1, 6,-4, 0 } /* 2cas(60 * PI / 32) */
};

/*
  Vamos a hacer una cosa:
  voy a intentar implementar la transformada discreta de Hartley haciendo uso de la
  representación de las funciones seno, coseno y caseno en forma de polinomio.

  Estas transformaciones sólo se dedicarán de momento a hacer espectros de vectores de
  16 muestras solamente. Mi objetivo es conseguir que la imagen resultante sea como una
  matriz de espectros de 16*16... A ver qué tal resulta la idea.

  Si sentido = 1 => transformada directa. Si sentido=0 => transformada inversa.
*/

/* Aquí el vector tiene que ser de tamaño = 16 */

/*
void dht(char *vector, int sentido) {
*/
void dht(int *vector, int sentido) {
int i,j;
int eba[TAM][MAX_DIM_POLINOMIO];
int pol[MAX_DIM_POLINOMIO];
int pol_0[MAX_DIM_POLINOMIO]={0, 0, 0, 0};
int escala;
float resultado;

        escala=sentido?2*TAM:2;

        for (i=0; i<TAM; i++) {
                pol_copia(eba[i],pol_0);
                for (j=0; j<TAM; j++) {
                        pol_copia(pol,cas_16[(i * j) %  TAM]);
                        pol_producto_escalar(pol,(int)(vector[j]));
                        pol_suma(eba[i],pol);
                }
        }

        for (i=0; i<TAM; i++) {
                resultado=pol_evalua(eba[i],Z,escala);
/* Esto implica saturación ====================================
                if (resultado<-128.0) { resultado=-128.0; }
                if (resultado>127.0) { resultado=127.0; }
                vector[i]=(char)(resultado);
========================================== hasta aquí */
                vector[i]=(int)(resultado); /* <- trunca pero no satura */
        }
}

/*void dht32(char *vector, int sentido) {*/
void dht32(int *vector, int sentido) {
int i,j;
int eba[TAM2][MAX_DIM_POLINOMIO];
int pol[MAX_DIM_POLINOMIO];
int pol_0[MAX_DIM_POLINOMIO]={0, 0, 0, 0};
int escala;
float resultado;

        escala=sentido?2*TAM2:2;

        for (i=0; i<TAM2; i++) {
                pol_copia(eba[i],pol_0);
                for (j=0; j<TAM2; j++) {
```

```
                        pol_copia(pol,cas_32[(i * j) %  TAM2]);
                        pol_producto_escalar(pol,(int)(vector[j]));
                        pol_suma(eba[i],pol);
                }
        }

        for (i=0; i<TAM2; i++) {
                resultado=pol_evalua(eba[i],Z,escala);
/* Igual que antes
                if (resultado<-128.0) { resultado=-128.0; }
                if (resultado>127.0) { resultado=127.0; }
                vector[i]=(char)(resultado);
=====*/
                vector[i]=(int)(resultado);
        }
}
```

K.4.6 polinomio.h

```
#ifndef POLINOMIO_H
#define POLINOMIO_H

/* Aquí trataré de definir algunos tipos bastante útiles, a ver que tal resulta la cosa */
#define MAX_DIM_POLINOMIO 4

/* pol1 <= pol2 */
int pol_copia(int *pol1, int *pol2);

/* pol_suma => pol1 = pol1 + pol2; */
int pol_suma(int *pol1, int *pol2);

/* pol_producto_escalar => pol = pol*x; */
int pol_producto_escalar(int *pol, int x);

/* pol_evalua = pol(x); */
float pol_evalua(int *pol, float x, float escala);

#endif
```

K.4.7 polinomio.c

```
#include <stdio.h>
#include "polinomio.h"

int pol_copia(int *pol1, int *pol2) {
int i;
        for (i=0; i<MAX_DIM_POLINOMIO; i++) {
                pol1[i]=pol2[i];
        }
        return(1);
}

int pol_suma(int *pol1, int *pol2) {
int i;
        for (i=0; i<MAX_DIM_POLINOMIO; i++) {
                pol1[i]=pol1[i]+pol2[i];
        }
         return(1);
}

int pol_producto_escalar(int *pol, int x) {
int i;
         for (i=0; i<MAX_DIM_POLINOMIO; i++) {
                 pol[i]=pol[i]*x;
         }
         return(1);
}

float pol_evalua(int *pol, float x, float escala) {
```

```
int i;
float resultado;
        resultado=((((((float)(pol[3])/escala)*x+((float)(pol[2])/escala))*x+((float)(pol[1])/e
scala))*x+((float)(pol[0])/escala)));
         return ( resultado );
}
```

L Presupuesto

Este proyecto se ha desarrollado dentro del Departamento de Ingeniería Electrónica de la Escuela Técnica Superior de Ingeniería Industrial de Barcelona.

L.1 Costes

A continuación se dividen los costes de ejecución material del Proyecto en costes de equipamiento (o costes materiales) y costes de mano de obra.

L.1.1 Costes de Material

El Departamento de Ingeniería Electrónica de la ETSEIB cuenta ya con una serie de equipos e instalaciones que han sido de gran ayuda para el desarrollo del Proyecto, tanto en lo que a desarrollo como a las diferentes pruebas de funcionamiento efectuadas se refiere.

Para la valoración de este equipamiento utilizado se han considerado diferentes porcentajes según la utilización del equipamiento en funciones correspondientes al desarrollo de este proyecto en concreto. Estos porcentajes han sido incluidos en el presupuesto de este proyecto para calcular la influencia del mismo en la inversión total realizada. También se han incluido como costes generales el suministro eléctrico, el material fungible y el gasto en comunicaciones.

En la Tabla L.1 se desglosa el coste por dispositivos utilizados en el sistema.

Material	Coste (euros)
Placa de desarrollo EPXA 10	830
Aplicaciones desarrollo Excalibur	240
Cámara digital	60
Total	**1130**

Tabla L.1 Coste por dispositivos utilizados.

A continuación, en la Tabla L.2, se muestra el total de materiales utilizados en el proyecto.

Material	Coste (€)	Uso (%)	Coste Total (€)
2 PC Intel Pentium 2,4 GHz	2000	10	200
Cables conexionado	30	10	3
Dispositivos Sistema (Placa, Software, Cámara)	1130	30	339
Osciloscopio	450	10	45
Total			**587**

Tabla L.2 Total de materiales utilizados en el proyecto.

L.1.2 Costes Mano de Obra

Para la realización del Proyecto, debe considerarse que se han debido realizar diferentes tareas. Para el desarrollo de la parte técnica han sido desarrolladas tareas propias de un Ingeniero, así como trabajos asociados a análisis de datos y programación informática. Por otro lado también existen otras tareas de búsqueda de información y elaboración de la memoria escrita que corresponden más bien a tareas de ámbito administrativo.

En la estimación del tiempo invertido para cada una de las tareas llevadas a cabo, se ha tenido en cuenta un factor corrector de 0,8 con el objetivo de compensar sobre los precios de referencia establecidos, el hecho que el desarrollo del proyecto no ha sido llevado a cabo por personal profesional. Además se debe tener en cuenta el tiempo que se ha dedicado al aprendizaje de las diferentes herramientas utilizadas, tanto a nivel de software como hardware.

El tiempo estimado que se ha invertido en el desarrollo del proyecto es de 40 semanas durante las cuales se ha considerado una media de trabajo diario de 6 horas, resultando un total de **1680 horas**, teniendo en cuenta ya en el cómputo de horas que el trabajo ha sido desarrollado por dos proyectistas. Aplicando el factor corrector antes considerado, queda un total de horas dedicadas de **1344 horas**.

Respecto al desglose en que se dividen las diferentes tareas en cuanto al tipo de trabajo llevado a cabo, decir que aproximadamente un 60% del tiempo ha sido dedicado a tareas de Ingeniería (recopilación información, comparación alternativas, concepción sistema, diseño arquitectura sistema, diseño de comunicación etc.), un 25 % a tareas de programación (desarrollo software de control, software SOPC, software desarrollo) y un 15% en lo que se refiere a redacción de la memoria y anexos.

En la Tabla L.3 se describen las tareas realizadas junto al coste de cada una de ellas así como el tipo de recurso utilizado.

Tarea	Recurso	Duración	Coste/hora (€)	Coste Total (€)
Estudio Previo Software/Hardware Altera	Ingeniería	30	42,07	1262,1
Estudio Sistema a Implementar	Ingeniería	40	42,07	1682,8
Estudio Previo Tratamiento Digital Imágenes	Ingeniería	30	42,07	1262,1
Estudio métodos tratamiento frecuencial Imágenes	Ingeniería	30	42,07	1262,1
Concepción Sistema	Ingeniería	50	42,07	2103,5
Desarrollo sistema	Ingeniería	150	42,07	6310,5
Desarrollo aplicaciones específicas	Programación	160	30,05	4808
Simulaciones	Programación	36	30,05	1081,8
Redacción memoria	Redacción	150	15,02	2253
Total				**22025,9**

Tabla L.3 Tareas realizadas.

L.1.3 Gastos Generales

Bajo este concepto se incluyen gastos derivados directamente del uso de los diferentes equipos que se han citado anteriormente. Respecto a los equipos informáticos gastos como consumo eléctrico, mantenimiento, licencias de software utilizado, uso de impresoras, fotocopias, material de oficina, material de laboratorio, amortizaciones de equipos, etc.

Se considerará que el valor de esta partida corresponde a un 15% del valor que posea el presupuesto de ejecución material, que comprende las partidas dedicadas a gasto de Material y gasto por mano de obra.

L.2 Presupuesto Total

De este modo obtenemos un presupuesto global tal como se muestra en la Tabla L.4.

Concepto	Coste (euros)
Coste Material	587,00
Coste mano de obra	22025,90
Presupuesto Ejecución	**22612,90**
Gastos Generales	3390,14
Presupuesto Global	**26003,00**
IVA (16%)	**4160,48**
Total	**30163,48**

Tabla L.4 Presupuesto total.

El presupuesto global alcanza un total de **treinta mil ciento sesenta y tres euros con cuarenta y ocho céntimos.**

M Legislación del proyecto

M.1 Introducción

Dado el carácter experimental del proyecto resulta difícil evaluar a qué tipo de normativa debería hacerse referencia para este tipo de proyectos. Es por ello, que se va a desarrollar este apartado atendiendo a los siguientes criterios que podrían vincularse en lo que al desarrollo de un Proyecto experimental en el campo del diseño electrónico digital se refiere.

Por un lado y dada la creciente preocupación por la sensibilidad medioambiental se vana a hacer referencia a las principales directivas que a éste respecto afectan a los componentes y dispositivos electrónicos utilizados para este proyecto.

Por orto lado, se tendrán en cuenta todas aquellas normativas que hagan referencia a la fabricación de componentes electrónicos, en este caso placas de desarrollo, así como aquellas normativas que en cuanto a seguridad deban cumplir.

Por último y dado que se trata de un proyecto experimental, se abordaría de forma escueta todo lo que hace referencia a patentes de nuevos procesos o tecnologías desarrolladas.

M.2 Impacto medioambiental

La política medioambiental de la Unión Europea, cuyo objetivo primordial es la conservación, protección y mejora de la calidad del medioambiente; promovió en el año 2003 la emisión de **la Directiva 2002/96/CE** del Parlamento Europeo y del Consejo de 27 de Enero de 2003 sobre residuos de aparatos eléctricos y electrónicos (DOUE núm. L37, de 13 de Febrero de 2003) (véase el CD-ROM adjunto en la memoria).

Esta directiva pretende ofrecer una herramienta de regulación para los residuos de los aparatos o dispositivos electrónicos (RAEE), sobre los cuales hasta el momento no se ejercía ningún control específico. El rápido incremento de la cantidad de este tipo de residuos que se han generado en la Unión Europea, fue considerado un problema importante en la fase de gestión y reciclado de residuos debido a la peligrosidad de algunos de los componentes que conforman estos dispositivos. Todo ello motivó la aparición de esta directiva, como respuesta a esta necesidad.

El ámbito de aplicación de la presente Directiva debe comprender todos los aparatos eléctricos y electrónicos, tanto los de consumo como los de uso profesional.

Otras normativas vinculadas a esta normativa y con las que guarda estrecha relación serían **la Directiva 75/442/CEE**; en la que se describen qué tipo de dispositivos pueden considerarse englobados dentro de la nomenclatura RAEE (residuos de aparatos eléctricos y electrónicos), así como otras normativas vinculadas a la gestión de residuos como la **Directiva 91/157/CEE**.

M.3 Legislación de patentes

En caso de que el procedimiento implementado o cualquier aplicación o desarrollo posterior sobre el mismo, fuera susceptible de ser considerado como una innovación tecnológica asociada a la petición de patentes, cabría tener en cuenta este tipo de normativas relacionadas.

Es criterio unánime en todos los países industrializados, que la legislación en materia de patentes influye decisivamente en la organización de la economía, al constituir un elemento fundamental para impulsar la innovación tecnológica.

Por otra parte, una Ley de Patentes, que proteja eficazmente los resultados de una investigación, constituye un elemento necesario dentro de la política de un país para el fomento de la investigación y el desarrollo tecnológico.

A nivel Europeo se conoce la existencia de un derecho de patentes constituido por el **Convenio de Munich de 5 de octubre de 1973 sobre la Patente Europea**, y el Convenio de Luxemburgo sobre la Patente Comunitaria de 15 de diciembre de 1975, derecho que ha sido recogido en la casi totalidad de las legislaciones de patentes europeas.

En este sentido y con le objetivo de actuar en consonancia con las normativas de la Unión Europea, se plasmó **la ley 11/1986, de 20 de Marzo, de Patentes**, mediante la cual se pretende proteger todo lo que esté relacionado con la aparición de innovaciones e invenciones en diferentes sectores de la economía, entre ellos obviamente el sector tecnológico.

Tal procedimiento regula todo el proceso de consideración o no de patentes así como el proceso que debe llevarse a cabo para regularizar esta situación.

N Contenido del CD-ROM

A continuación se muestra un resumen del contenido del CD-ROM que se adjunta con la memoria de este proyecto.

- Directorio "Artículos": en este directorio se pueden encontrar, en formato PDF, algunos de los artículos que han sido incluidos como referencia.

- Directorio "Códigos Fuente": en este directorio se pueden encontrar los códigos fuente de todo lo que se ha desarrollado en este proyecto, incluyendo aquellos que han sido listados en los anexos H, I, J y K. Para respetar la estructura que utiliza el software Quartus II, el software del SSE se encuentra en la ruta "Códigos Fuente\SSE\standard_16\cpu_sdk\src\".

- Directorio "Documentación de Altera": este directorio contiene toda la documentación proporcionada con el paquete de software Quartus II de Altera, tanto *datasheets* como notas de aplicación, esquemas y guías de usuario.

- Directorio "Otra documentación": en este directorio se incluyen otros documentos, en formato PDF, que se han consultado, parte de los cuales han sido incluidos como referencia.

- Directorio "Software": este directorio incluye el siguiente software para plataformas Windows:

    - Scanning Probe Image Processor (SPIP) 3.3019 (Versión de demostración)
    - Adobe Acrobat Reader, versión 5.05 en Español
    - Dev-C++ 4.9.8.0

- La memoria y el presente volumen de anexos que componen este proyecto, en formato PDF.