PONTIFICIA **UNIVERSIDAD CATÓLICA** DEL PERÚ

PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

FACULTAD DE CIENCIAS E INGENIERÍA

DESARROLLO DE UNA ARQUITECTURA PARA LA OBTENCIÓN DE LA FASE DE UNA COMPONENTE AM-FM DE UNA IMAGEN DIGITAL EN UN FPGA

Tesis para Optar el Título de:

INGENIERO ELECTRÓNICO

Presentado por:

VÍCTOR MANUEL MURRAY HERRERA

Lima – Perú

2004

RESUMEN

La demodulación AM-FM modela los datos de entrada (en general, cualquier señal de N-dimensiones) como una combinación lineal de funciones de amplitud positiva multiplicadas por sinusoides con funciones de fase no-lineal [1], y tanto la amplitud como la fase pueden variar continuamente sobre el espacio de N-dimensiones. Esta propiedad permite a la demodulación AM-FM modelar imágenes no estacionarias o videos. En este sentido, la eficacia de los modelos AM-FM para la estimación de la frecuencia instantánea y la caracterización de imágenes y señales no estacionarias han sido fuertemente estudiados desde cerca del año 1995 [2]. Debido a que los algoritmos para la demodulación AM-FM requieren procesar gran cantidad de datos a altas velocidades es necesaria su implementación en procesadores de aplicación específica (*hardware*). Este trabajo presenta el desarrollo de una arquitectura flexible que realiza las operaciones necesarias para la obtención de la fase relacionada a una imagen, en escala de grises, analizada en una componente canalizada AM-FM aprovechando el paralelismo brindado por los Arreglos de Puertas Programables en Campo (FPGAs). El diseño se basa en una arquitectura *pipeline* que permite separar el proceso de una imagen en varias etapas de procesamiento, la cual se implementó usando el Lenguaje de Descripción de Hardware de Circuitos Integrados de alta velocidad (VHDL). Se realiza un análisis del rendimiento de sus diferentes partes (máxima frecuencia de reloj posible y consumo de celdas lógicas), según el tamaño de las imágenes para diferentes familias de FPGAs. Estos análisis fueron realizados en el entorno Quartus II considerando las diferentes opciones de síntesis lógicas.

Los resultados de las pruebas experimentales realizadas demuestran una alta confiabilidad en los resultados y buenas velocidades de procesamiento, pudiendo procesar imágenes en escala de grises de 256 tonos con un tamaño de 512x512 píxeles, utilizando siete coeficientes en los filtros, a una velocidad máxima de 167.17MHz con una ocupación del 90.30% del FPGA STRATIX EP1S25F672C6 [3].

A Dios, a Jesús, al Espíritu Santo, a la Virgen María, a San Marcelino Champagnat, a mi ángel de la guarda, y a San José.

A mis padres, por todo lo que me han dado, no hay espacio para escribir todo ni palabras suficientes para agradecerles. Los amo.

A mi familia, por todo su apoyo y cariño.

A Andrés Flores, por su amistad y por toda la confianza depositada en mi, desde que fui su alumno, hasta que fui su compañero de trabajo y asesorado, además por haber sacrificado su tiempo en leer este tomo.

A Paul Rodríguez, por su amistad, consejos, enseñanzas y apoyo.

Al Ing. Gerard Santillán, por haber confiado en mi para la realización de este tema y por haberme orientado hacia la investigación.

A mis profesores, ahora amigos y compañeros de trabajo, por todos sus consejos y aliento.

César Carranza, Manuel Morales, Javier Chang

Al Grupo de Procesamiento Digital de Señales e Imágenes –GPDSI-, por el apoyo académico y amistad brindada, además de todo lo vivido y compartido.

Jordán, Akio, Jorge, Carlos, Danielux, Alonzo, Daniel Llamocca, Úrsula, Pedro, Sergio

A Adita, por su apoyo, aliento, ayuda, cariño y compañía.

A mis amigos que me apoyaron durante el desarrollo de la tesis, incluyendo a los integrantes del Grupo de Microelectrónica –GuE-

Orlando, Johanna Yáñez, Eduardo, Chris, César Saldaña, Carlos Silva

A Eduardo Ísmodes por su apoyo incondicional a la investigación, y al Ing. Bernasconi por su apoyo económico para la realización de esta tesis.

A todos, muchas gracias.

Escrito de forma anónima en una cadena de correo electrónico, justo luego de un robo y con eso la culminación de una mala etapa en mi vida:

“Ante una caída en la vida, hay 2 tipos de personas: los que se quedan en el piso lamentándose y, los que aprovechan la caída para tomar un gran impulso”

Yo decidí salir adelante con más fuerza en cada caída.

ÍNDICE

ANEXOS (ver CD-ROM adjunto)

ANEXO A: CÓDIGO DE DESCRIPCIÓN EN HARDWARE DE LOS BLOQUES DEL SISTEMA

ANEXO B: SIMULACIONES

ANEXO C: CÓDIGO DESARROLLADO EN MATLAB Y EN C

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

La demodulación AM-FM modela los datos de entrada (en general, cualquier señal de N-dimensiones) como una combinación lineal de funciones de amplitud positiva multiplicadas por sinusoides con funciones de fase no-lineal [1], y tanto la amplitud como la fase pueden variar continuamente sobre el espacio de N-dimensiones. Esta propiedad permite a la demodulación AM-FM modelar imágenes no estacionarias o videos. En este sentido, la eficacia de los modelos AM-FM para la estimación de la frecuencia instantánea y la caracterización de imágenes y señales no estacionarias han sido fuertemente estudiados desde cerca del año 1995 [2]. Entre las principales aplicaciones de la demodulación AM-FM se encuentra el proceso de imágenes de video en modo de movimiento (*Motion mode videos* ó *M-mode videos*) observando la evolución de un píxel de la imagen del video en el tiempo, y esto es utilizado para aplicaciones militares en los sonares, aplicaciones médicas, como en el caso de la ecocardiografía [1], y en reconocimiento de huellas digitales [4].

La demodulación AM-FM puede ser un método computacional muy intenso, y en general, la bibliografía referida a AM-FM no muestra aplicaciones en tiempo real. Dicha intensidad en el procesamiento computacional se debe a los pasos a realizar, los cuales involucran la obtención de la señal analítica [5, 6], el análisis de ella en las distintas bandas, y luego la obtención de su fase, frecuencia, y amplitud. Los filtros pasa banda utilizados tienen la característica de tener coeficientes con componentes reales e imaginarios [7] debido al análisis de la imagen analítica, lo que produce un mayor procesamiento de

datos. Actualmente, se está realizando como tesis doctoral en *The University of New Mexico*, USA, una rápida implementación que toma ventaja de la tecnología de los procesadores modernos para realizar dicho procesamiento en tiempo real [1]. Dicha implementación se basa en instrucciones singulares de datos múltiples (SIMD) que tienen la ventaja de operar simultáneamente sobre bloques similares de datos.

La propuesta de este trabajo de tesis es desarrollar la arquitectura física (*hardware*) para realizar dicha demodulación AM-FM de imágenes digitales en escala de grises de 8 bits, obteniendo en tiempo real la fase relacionada a la imagen analizada en una componente canalizada AM-FM, es decir, obviar el uso de un programa computacional (*software)* de manera que se realicen procesos en forma paralela. Con dicha fase es posible realizar el análisis en los casos mencionados en los párrafos anteriores [1].

La propuesta se basa en arquitecturas *pipeline* [8, 9] para la implementación de los diversos procesos requeridos en la obtención de la fase de una componente canalizada de la demodulación AM-FM. Dicha implementación se ha realizado utilizando el Lenguaje de Descripción de Hardware de Circuitos Integrados de alta velocidad –VHDL- y, si bien es flexible a las diversas familias de Arreglos de Puertas Programables en Campo (FPGAs), se utilizó un STRATIX EP1S25 [3] para aprovechar los multiplicadores embebidos, optimizados para el procesamiento de señales, y los *PLLs* que posee, los cuales no ocupan Elementos Lógicos, con lo que la opción de un mayor paralelismo es posible [3]. La arquitectura propuesta, conjuntamente con una breve descripción

matemática para la demodulación AM-FM, es presentada en el Capítulo 1. El diseño de la arquitectura propuesta, desarrollada por bloques independientes, y un estudio del rendimiento de la arquitectura y los bloques independientes en distintas clases de FPGAs de las familias de los STRATIX [3], STRATIX II [10] y ACEX1K [11], ambos de Altera, ante imágenes con un tamaño máximo de 512x512 píxeles y 256 tonalidades de grises, es presentado en el Capítulo 2. Las pruebas y sus respectivos análisis de los resultados experimentales, obtenidos luego de la implementación en un STRATIX EP1S25F672C6 de la tarjeta de desarrollo MJL Stratix Development Kit [12], respecto al tiempo de procesamiento y error obtenido respecto al mismo algoritmo implementado en Matlab, además de una comparación respecto al tiempo de procesamiento del algoritmo utilizando SIMD, son mostrados en el Capítulo 3. Las conclusiones alcanzadas luego del desarrollo del trabajo se muestran en el Capítulo 4 y, finalmente se muestran las recomendaciones para una mejora del sistema, además de las referencias y anexos correspondientes al desarrollo del trabajo.

1. REPRESENTACIÓN AM-FM DE UNA IMAGEN Y ARQUITECTURA PROPUESTA

1.1 Introducción

Este capítulo presentará una breve descripción de la demodulación AM-FM (sección 1.2), además de exponer luego la arquitectura propuesta para la obtención de la fase relacionada a una imagen analizada en una componente canalizada AM-FM en un FPGA (sección 1.4).

La demodulación AM-FM modela los datos de entrada (en general, cualquier señal de N-dimensiones) como una combinación lineal de funciones de amplitud positiva multiplicadas por sinusoides con funciones de fase no-lineal [1]. Dos métodos relacionados serán descritos: el Análisis de la Componente Dominante –DCA- y el Análisis del Componente Canalizado –CCA-, los cuales están basados en la transformada de Hilbert [6, 13].

La demodulación AM-FM ha mostrado una gran capacidad para segmentar imágenes digitales [14, 15, 16]. Un ejemplo de la técnica para segmentar la imagen basándose solamente en la información de la fase proveniente de la demodulación AM-FM será descrito en la sección 1.3. Este procedimiento tiene su origen en la observación de algunas interesantes propiedades mostradas en la reconstrucción FM de la imagen original cuando es hecho con múltiples armónicos usando un único componente canalizado AM-FM [1].

1.2 Demodulación AM-FM: Teoría

Sea $X(n_1,n_2)$ la representación de una imagen digital, donde n_1 es usado para indicar las filas y n_2 para indicar las columnas, el Análisis de la Componente AM-FM Dominante –DCA- describe una imagen digital en términos de:

$$\hat{X}(n_1,n_2)=\sum_{k=1}^{D} I_{[m(a(n_1,n_2))=k]}a_k(n_1,n_2)\cos(\varphi_k(n_1,n_2)) \quad (1)$$

donde:

$\hat{X}(n_1,n_2)$ es la versión reconstruida de la imagen de entrada utilizando D diferentes filtros pasa banda.

$$a(n_1,n_2)=máximo\{a_k(n_1,n_2):k\in[1,D]\} \quad (2)$$

$$m(a(n_1,n_2))=\{k:a_k(n_1,n_2)=a(n_1,n_2)\}$$

$I_{[\bullet]}$ es la función Indicador.

El algoritmo de demodulación AM-FM opera sobre la versión analítica [5, 6] de la imagen de entrada:

$$X_A(n_1,n_2)=X(n_1,n_2)+jH\{X(n_1,n_2)\} \quad (3)$$

donde $H\{.\}$ es la Transformada Discreta de Hilbert que opera sobre las filas ó sobre las columnas de $X(n_1,n_2)$ [15, 17]. X_A (ver ecuación 3), es llamada la "Imagen Analítica" [5, 6], la cual es necesaria para evitar ambigüedad cuando se estima la amplitud instantánea y la fase (o frecuencia) (ver ecuaciones 4 a 6) [18, 19]. La amplitud $a_k(n_1,n_2)$ (ver ecuación 6) y la fase $\varphi_k(n_1,n_2)$ (ver ecuación 5) son estimadas de la salida de los D filtros complejos pasa banda $h_1,h_2,...,h_D$.

$$X_{A_k}=X_A*h_k \quad (4)$$

$$\varphi_k(n_1,n_2)=\arctan\left(\frac{imag\left(X_{A_k}(n_1,n_2)\right)}{real\left(X_{A_k}(n_1,n_2)\right)}\right) \quad (5)$$

$$a_k(n_1,n_2)\approx\left|\frac{X_{A_k}(n_1,n_2)}{\mathrm{H}_k(\nabla(\varphi_k(n_1,n_2)))}\right| \quad (6)$$

$$\nabla(\varphi_k(n_1,n_2))\approx real\left(\frac{\nabla\left(X_{A_k}(n_1,n_2)\right)}{jX_{A_k}(n_1,n_2)}\right) \quad (7)$$

donde:

$\nabla(\varphi_k(n_1,n_2))$ es la frecuencia instantánea

H_k es la respuesta en frecuencia del filtro h_k

En el algoritmo del Análisis de la Componente Dominante –DCA- (ecuaciones de 1 a 7), de la señal estimada en cada filtro pasa banda se selecciona la señal estimada con la máxima amplitud estimada:

$a(n_1,n_2) = máximo\{a_k(n_1,n_2): k \in [1,D]\}$. El funcionamiento del algoritmo de demodulación AM-FM en una y dos dimensiones ha sido estudiado en [20, 21, 4], y un algoritmo en espacio discreto es descrito en [20, 21]. El diagrama de bloques del algoritmo DCA es mostrado en la figura 1.1.

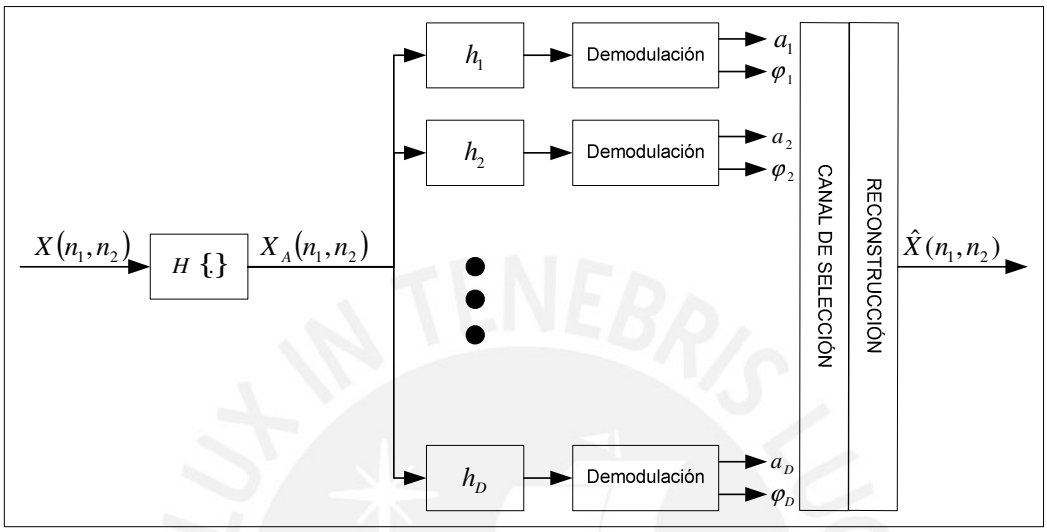


Figura 1.1. Diagrama de Bloques del Algoritmo DCA.

El Análisis del Componente Canalizado –CCA- es distinto del DCA debido a que cada canal es usado para reconstruir una imagen. Es asumido que cada filtro pasa banda (canal) aísla los componentes en un canal. Este procedimiento proporciona una fuerte descripción caracterizada no sólo en la estructura de la imagen dominante, sino también estructuras particulares de cada pasa banda, por lo que, este será el algoritmo a implementar. El diagrama de bloques del algoritmo CCA es mostrado en la figura 1.2.

El espectro de Fourier de la "Imagen Analítica" $X_A(n_1,n_2)$ (ecuación 3 [13]) está definida sólo en dos cuadrantes del plano de frecuencias (en donde el espectro de la imagen original $X(n_1,n_2)$ ocupa cuatro cuadrantes). Esto implica que hay por lo menos dos posibles caminos para calcular $X_A(n_1,n_2)$ dependiendo de la dirección en que la transformada de Hilbert es aplicada (filas o columnas). En general, el resultado de aplicar la transformada de Hilbert en dirección de las filas o de las columnas no será el mismo y, por lo tanto, diferentes reconstrucciones serán obtenidas. Si la imagen de entrada es continua, su

transformada de Fourier puede ser calculada como se indica en la ecuación 8 (que podría ser aplicada también en la dirección de las columnas, dependiendo de la dirección escogida). Sin embargo, la imagen de entrada será una imagen digital y la ecuación 8 no podrá ser aplicada. En este caso, la transformada de Hilbert puede ser hallada utilizando la ecuación 9, basada en la transformada directa e inversa de Fourier, ó utilizando la ecuación 10, la cual se basa en utilizar la ventana de Káiser para hallar los coeficientes de un filtro FIR de Hilbert de orden M que puede ser aplicado a la imagen original en la dirección de filas o columnas. Los métodos mencionados son resumidos en las ecuaciones 8 a 11.

$$H_{n_1}\{X(n_1,n_2)\}=\frac{1}{\pi}\int_{\Re} X(n_1-\xi,n_2)\frac{d\xi}{\xi} \tag{8}$$

$$H_{n_1}\{X(n_1,n_2)\}=FFT_{2D}\{X(n_1,n_2)\}+conj(FFT_{2D}\{X(n_1,n_2)\}) \tag{9}$$

$$H_{n_1}\{X(n_1,n_2)\}=X(n_1,n_2)*g_H(n_1) \tag{10}$$

$$g_H(n_1)=(1-I_{[n_1=0]})\frac{2}{n\pi}\sin^2\left(\frac{n_1\pi}{2}\right) \quad n_1\in[-M,M] \tag{11}$$

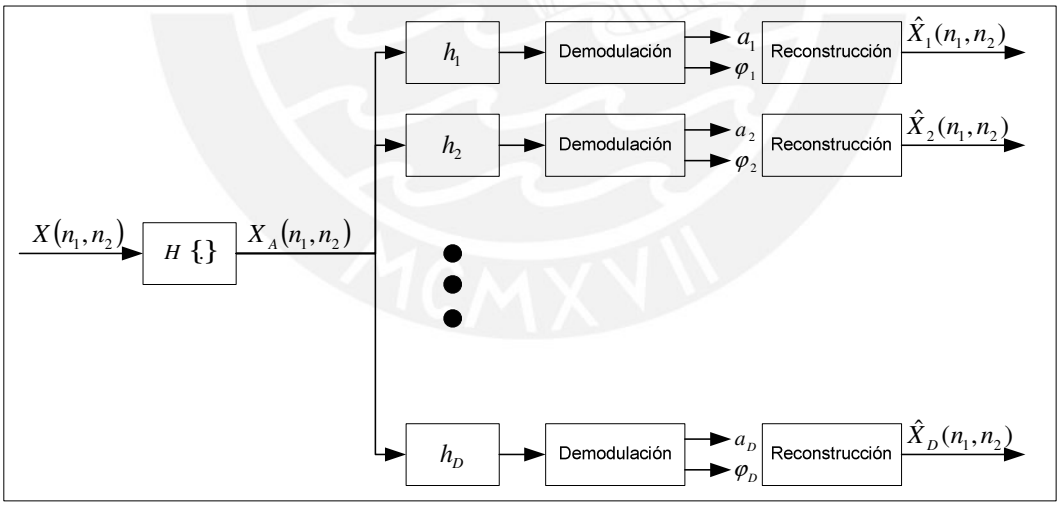


Figura 1.2. Diagrama de Bloques del Algoritmo CCA.

Un proceso popular en la literatura de AM-FM [1, 2, 4, 13, 20, 21] ha utilizado un banco de filtros 2D Gabor [22] usados para el funcionamiento de la demodulación de las múltiples componentes AM-FM. Este banco de filtros produce salida de filtros similares a una descomposición Wavelet usando funciones de Gabor para los filtros Wavelet. En este trabajo, se considera el

uso de un banco de filtros basados en un arreglo de filtros 1D FIR separables [7] diseñados en el dominio de la frecuencia debido a su flexibilidad, facilidad de diseño y sus propiedades prácticas de implementación. Dichos filtros poseen coeficientes con parte real y parte imaginaria. Además, cabe resaltar que la optimización indicada usando filtros Gabor en la demodulación AM-FM continua se pierde en el caso discreto [1].

1.3 Ejemplo de demodulación con múltiples armónicos de CCA

Para ilustrar la aplicación de la demodulación AM-FM de la teoría descrita en la sección anterior, una demodulación AM-FM de la imagen de Lena usando el algoritmo CCA (usando sólo el canal de la frecuencia más baja) y su reconstrucción utilizando múltiples armónicos usando sólo la información de la fase (ver figura 1.3) es descrito utilizando la ecuación 12. Se observa que mientras mayor es el número de armónicos utilizados, más oscura se obtiene la imagen.

$$\hat{X}(n_1,n_2)=\sum_{k=1}^{L}\cos\left(k\varphi(n_1,n_2)\right) \tag{12}$$

donde:

$\hat{X}(n_1,n_2)$ es la imagen reconstruida sólo con la fase.

L es el número de armónicos considerados

$\varphi(n_1,n_2)$ es calculada en la ecuación 5.

1.4 Arquitectura propuesta

Con la teoría descrita en las secciones previas y habiendo explicado el algoritmo CCA, el cual será utilizado, la idea es conocer el problema desde una manera global y general para luego mostrar la mejor alternativa para el desarrollo teniendo como base un FPGA.

Los pasos a seguir desde el tener la imagen original de manera digital, en una computadora personal (PC) por ejemplo, hasta la obtención de la fase relacionada a la imagen analizada en una componente canalizada AM-FM se resumen en:

a. Enviar la imagen digital original desde la PC hacia el FPGA.

b. Obtener la fase relacionada a la imagen original analizada en una componente canalizada AM-FM.

c. Enviar la fase desde el FPGA hacia la PC.

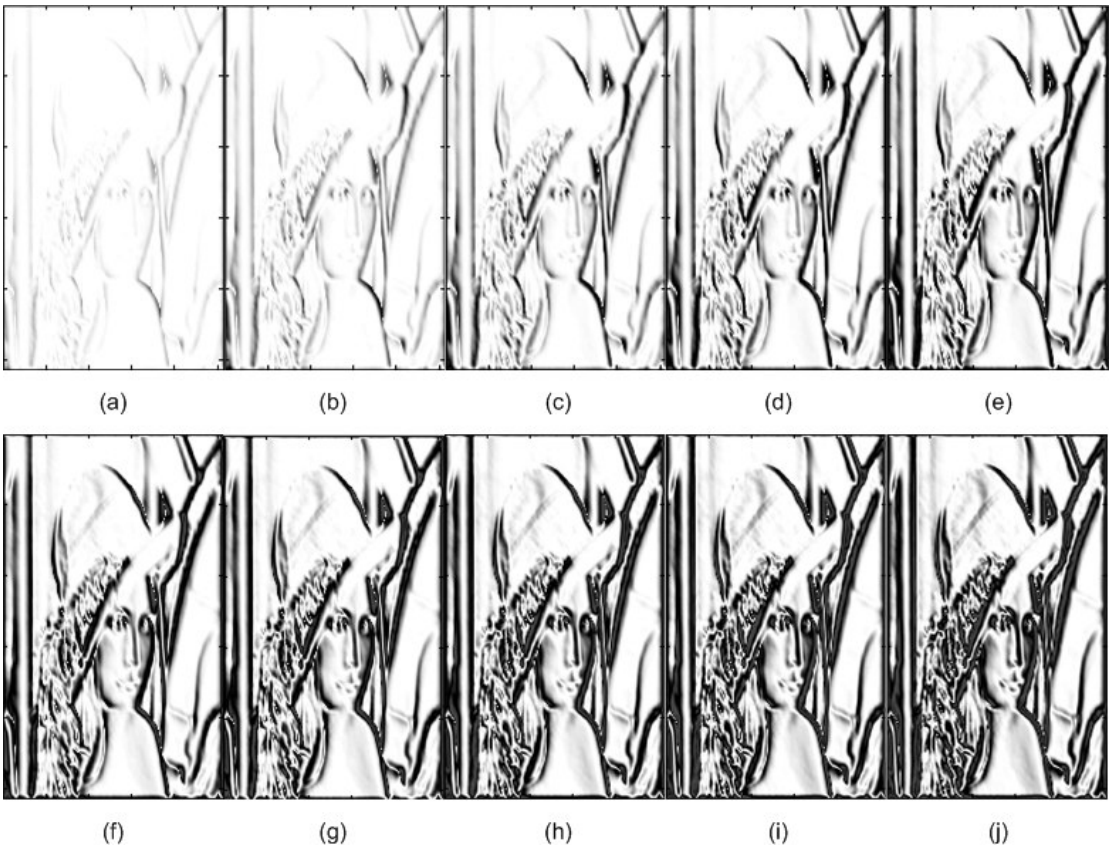


Figura 1.3. Ejemplo de demodulación usando el algoritmo CCA utilizando 1,2, 3, 4, 5, 6, 7, 8, 9, y 10 armónicos (figuras de “a” a “j”, respectivamente). Las imágenes mostradas fueron halladas usando Matlab con el algoritmo utilizado en la presente tesis.

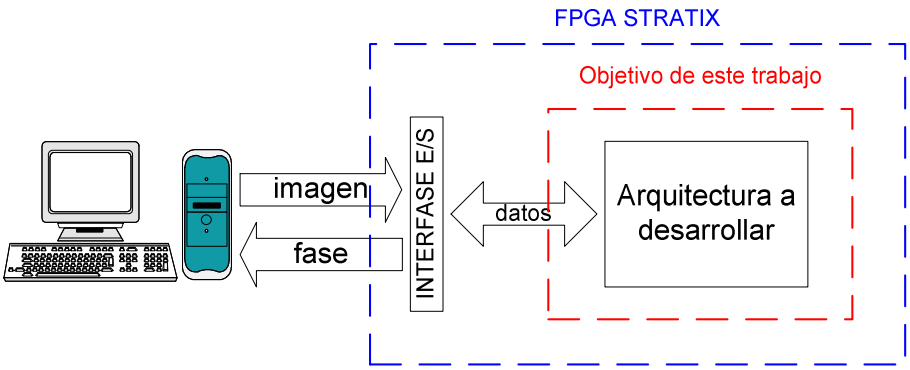


Figura 1.4. Proceso a seguir en la obtención de la fase relacionada a la imagen analizada en una componente canalizada AM-FM.

Teniendo claros los pasos a seguir, la figura 1.4 muestra en forma gráfica las partes involucradas. Ahora, dichos pasos a seguir de manera general, es necesario ir descomponiéndolos en forma de procesos, de modo que los procesos a seguir son los siguientes:

a. Enviar la imagen digital original desde la PC hacia el FPGA.
b. Hallar la transformada de Hilbert de la imagen original.
c. Hallar la Imagen Analítica de la imagen original.
d. Filtrar la imagen analítica en el canal deseado.
e. Obtener la fase relacionada a cada píxel mediante la función ARCO TANGENTE (ARCTAN).
f. Enviar la fase desde FPGA hacia la PC.

Teniendo los procesos que involucran la obtención de la fase relacionada a la imagen analizada en una componente canalizada AM-FM, es necesario definir el objetivo del trabajo conjuntamente con un primer punto a tomar en cuenta: El objetivo es desarrollar la arquitectura en un FPGA para la obtención de la fase relacionada a la imagen analizada en una componente canalizada AM-FM, es decir, los procesos desde la letra “b” hasta la “e”, cuyo tiempo de procesamiento a medir, del proceso total, será el que demore en realizar dichos pasos (ver figura 1.4) No se tiene como objetivo realizar una interfase E/S (de entrada y salida) rápida, pero sí es necesario usar una implementada en el FPGA para la realización del proceso. Tampoco se tiene como objetivo

comparar si la demodulación AM-FM presenta mejores prestaciones que otras técnicas como podría ser el caso de la transformada de Wavelet.

Para realizar dichos procesos se decidió utilizar un FPGA STRATIX EP1S25 debido a las características que tiene este STRATIX que se resumen en:

- 25660 Elementos Lógicos.
- 1944576 bits de memoria RAM interna.
- 40 multiplicadores embebidos de 18x18 bits.
- 6 PLLs.

En el momento de realizar la adquisición de dicho FPGA (noviembre del 2003), la única tarjeta de desarrollo de Altera que contaba con eso, y que sea económicamente adquirible era la MJL Stratix Development Kit [12], ya que otras tarjetas con ese FPGA superaban los 3 mil dólares (actualmente, julio del 2004, por menos de mil dólares es posible adquirir una tarjeta con dicho FPGA y muy buenas prestaciones externas). Luego de adquirir la tarjeta, si bien se tuvo el FPGA requerido, como segundo punto a tomar en cuenta, se tomó como meta no realizar gastos de adquisición adicionales, por lo que la arquitectura a desarrollar debe ser capaz de funcionar con los componentes que se tienen, entre los cuales, y utilizables para la tesis, tenemos:

- 1 Stratix EP1S25.
- 4 MB de memoria FLASH.
- módulos de 64KBx16 bits de SRAM configurados como una SRAM única de 64KBx32 bits.
- 1 módulo de 2Mx32 bits de SDRAM.
- 1 conector RS-232.
- 1 controlador USB SL811HS.
- 1 controlador para conexión Ethernet.
- 1 Oscilador externo de 33.333MHz.

De esta manera, un funcionamiento óptimo (ver figura 1.5) estaría limitado a usar la conexión USB o Ethernet, pues funcionaría a alta velocidad, pero debido a que la tarjeta adquirida no contenía ningún controlador para ninguna de sus componentes externos, realizar el controlador para controlar en un

FPGA la conexión USB o Ethernet es muy complicada y escapa a los objetivos de esta tesis, por lo que se usará una interfase serial usando el protocolo RS-232 y la imagen original será almacenada en una memoria externa al FPGA pero interna de la tarjeta, de modo que el tiempo de procesamiento no considera el retraso por transmisión y recepción de forma serial (ver figura 1.6). Es por ese motivo que la idea mostrada en la figura 1.6 es la arquitectura fue la utilizada y con la cual se trabajará de aquí en adelante.

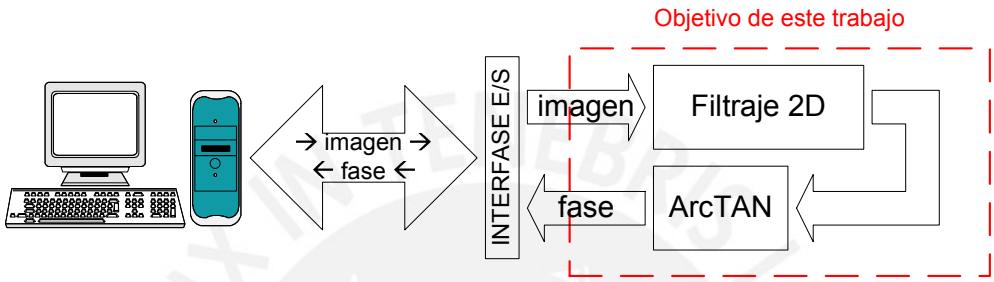


Figura 1.5. Arquitectura con funcionamiento óptimo con una interfase E/S rápida.

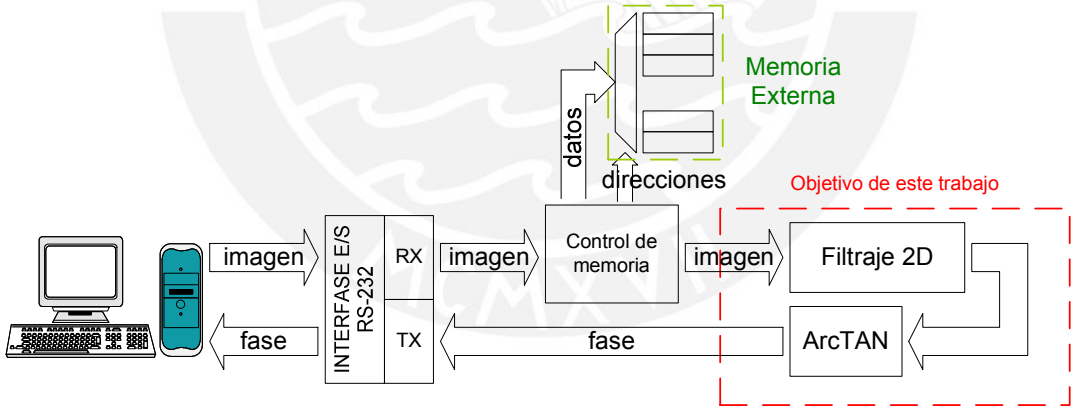


Figura 1.6. Arquitectura usando RS-232 y almacenamiento temporal en una memoria externa.

Ahora, antes de mostrar en detalle la propuesta para lograr la arquitectura mostrada (sección 1.4.2), es importante mencionar que una de las grandes ventajas de usar FPGAs es la característica de poder realizar bloques que funcionen usando *pipeline,* dicha característica será explicada en las siguientes líneas (sección 1.4.1).

1.4.1 Arquitecturas *Pipeline* [23]

Debido a que el análisis de imágenes requiere el manejo y proceso de una gran cantidad de datos a altas velocidades de procesamiento, es necesario procesar la información en paralelo y de manera síncrona. Algunas décadas atrás se empezaron a diseñar procesadores basados en arquitecturas síncronas para procesar datos en paralelo, algunos ejemplos son los procesadores vectoriales, *SIMD* y de arreglos sistólicos [8]; una arquitectura *pipeline* es un tipo de arreglo sistólico. Los procesadores basados en arquitecturas *pipeline* son comúnmente empleados para realizar operaciones sobre vecindades en imágenes de manera eficiente, tales como convolución, correlación y operaciones morfológicas [8, 23].

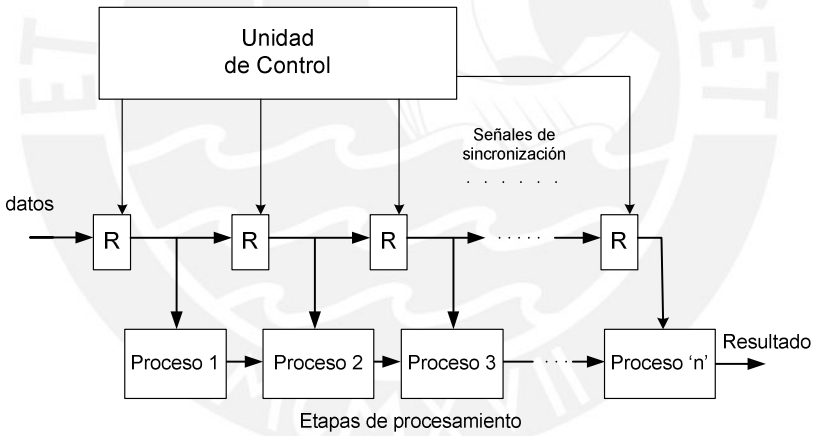


Figura 1.7. Cadena de registros en una arquitectura *pipeline.*

Una arquitectura *pipeline* contiene un arreglo de etapas de procesamiento, el cual consiste en una cadena de registros de almacenamiento cuyos datos son desplazados continuamente en una sola dirección (ver figura 1.7). La información de cada registro es parte de una etapa y se deriva a un bloque de procesamiento que puede realizar una operación particular, luego esta información es entregada al registro de la siguiente etapa junto con un resultado parcial. Una ventaja de esta arquitectura radica en el hecho que la información a analizar no necesita ser almacenada en arreglos complicados,

sólo se requieren simples interconexiones; además, su funcionamiento sólo requiere de señales de sincronización que son entregadas por una unidad de control.

1.4.2 Arquitectura Propuesta en detalle

Definida la ventaja de la realización de arquitecturas *pipeline* para explotar así las ventajas ofrecidas por los FPGAs, un objetivo es que todos los bloques propuestos cumplan con este tipo de arquitectura.

Un proceso crítico tanto desde el punto de vista de diseño como del consumo de recursos del FPGA es la parte del filtraje en 2 dimensiones. Un filtro en 2 dimensiones posee r filas y s columnas, es decir, posee $r \times s$ posiciones, lo que es igual a la misma cantidad de multiplicadores necesarios. Además si consideramos que $r = s$, que a partir de la imagen analítica los datos tendrán parte real y parte imaginaria, y que los filtros indicados en la sección 1.2 poseen coeficientes con parte real y parte imaginaria [7], tenemos que para realizar una multiplicación en un píxel, requerida para la convolución necesaria, se necesitan cuatro multiplicadores (ver ecuación 13), y para realizar el filtraje total, basado en la convolución, se necesita realizar $4 \times r^2$ multiplicaciones lo que es muy complejo de desarrollar y se necesita una mayor cantidad de recursos.

$$z(n_1, m_2) = X(n_1, m_2) \times f(r_1, s_2) = (a + jb) \times (c + jd) = (ac - bd) + j(ad + bc) \qquad (13)$$

donde:

$X(n_1, m_2) = a + jb$ píxel de la imagen con parte real a y parte imaginaria b localizado en la fila n y columna m.

$f(r_1, s_2) = c + jd$ componente del filtro con parte real c y parte imaginaria d localizado en la fila r y columna s.

$z(n_1, m_2)$ producto obtenido.

De esta forma, se prefiere utilizar los filtros separables descritos en la sección 1.2 de manera que almacenando en la memoria externa los datos de la imagen

de forma secuencial en la dirección de las columnas, fila por fila, se realiza el filtrado en una primera etapa de cada fila independientemente, fila por fila, y luego el filtrado de cada columna independiente, columna por columna, de la imagen resultante de la primera etapa (ver figura 1.8).

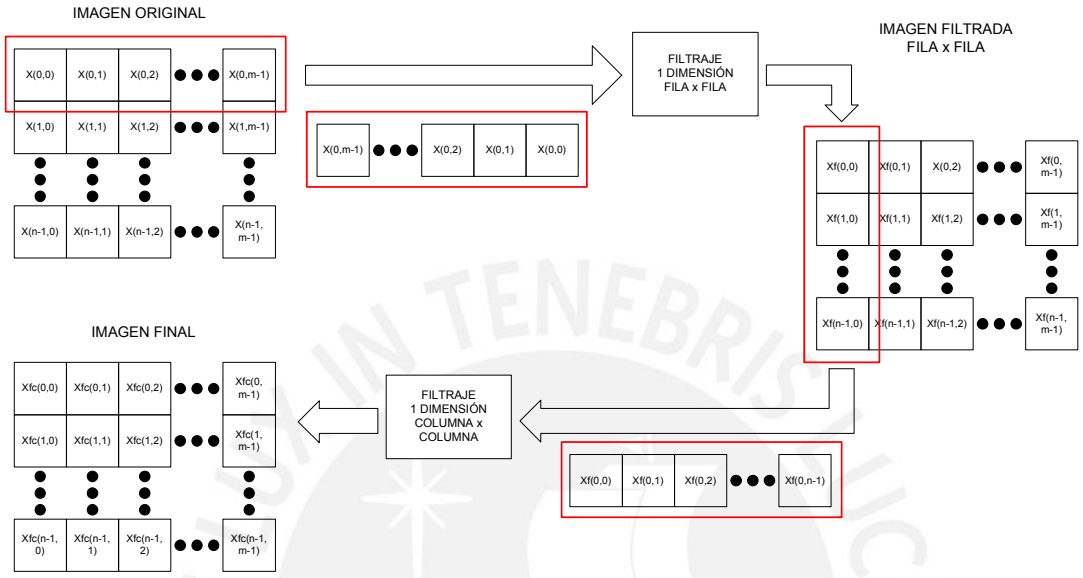


Figura 1.8. Filtraje de una imagen utilizando filtros separables de una dimensión.

Teniendo en claro el proceso que debe realizar el bloque de filtraje, es necesario volver a mencionar que el filtraje en una dimensión requiere el uso de varios recursos del FPGA, por lo que se optó desarrollar un solo bloque de filtrado y utilizar éste en las distintas etapas que requiera filtraje, cambiando sólo sus coeficientes. Esto implica que es necesario almacenar:

a. La imagen original.
b. La imagen analítica.
c. La imagen analítica filtrada fila por fila.
d. La fase que se obtiene luego de tener la imagen obtenida en “c” filtrada columna por columna.

Es decir, se almacena información cuatro veces utilizando las memorias disponibles. En una primera propuesta se pensó en utilizar la memoria externa SRAM [24] para almacenar la imagen original y para el resto (“b”, “c” y “d”) usar

la memoria externa SDRAM [25], pero la diferencia de velocidades de almacenamiento entre una y otra (si bien la SRAM es asíncrona, se puede trabajar como síncrona a una frecuencia de 33MHz, y la SDRAM trabaja a 133MHz) hace que la mejor manera sea usar para todo el proceso la memoria SDRAM.

En ese sentido, debemos considerar que el filtraje de una señal de p datos con un filtro de q coeficientes utilizando la convolución produce una señal de $p+q$ datos, por lo que si se aplica esa teoría a una imagen de tamaño $n \times m$, de n filas y m columnas con un filtro separable de t coeficientes en una dimensión (en dos dimensiones: t filas y t columnas) se obtendrá una imagen de tamaño $(n+t)\times(m+t)$ con lo cual hay que utilizar sólo los datos correctos de modo que se obtenga una imagen del mismo tamaño que la original, es decir, de n filas y m columnas (ver figura 1.9).

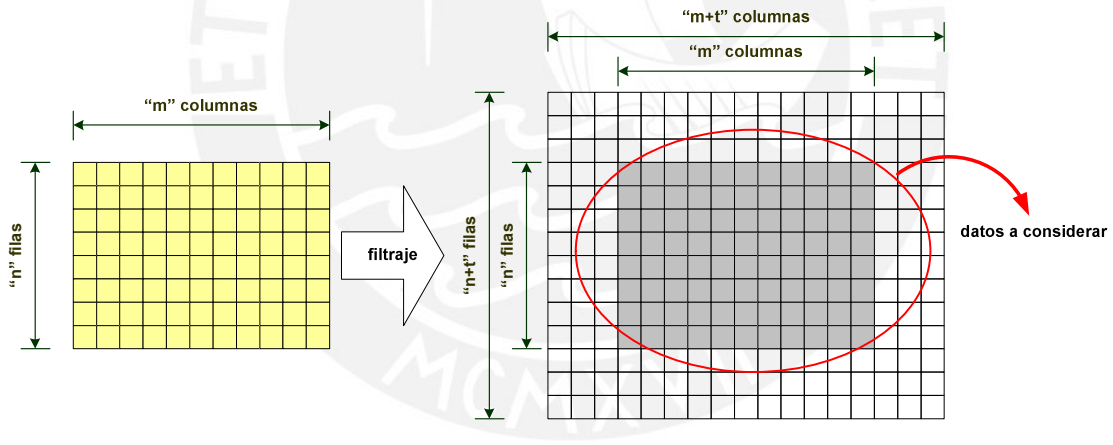


Figura 1.9. Tamaño del resultado del filtraje de una imagen de tamaño $n \times m$.

Es por ese motivo que se pensó contar con un Control Global del sistema, de modo que sea éste quien decida almacenar o no un dato obtenido luego del filtraje (ver figura 1.10 en donde se observa el uso de un solo bloque para el filtraje, un bloque que halla la función ARCTAN, y el uso de multiplexores y demultiplexores para decidir cuando se utiliza o no el ARCTAN). Es necesario controlar la SDRAM de manera independiente, ya que esta memoria (y todas las memorias disponibles en la tarjeta utilizada) tienen un solo bus de datos, el

cual trabaja de manera bidireccional, además de controlar su inicio y configuración.

Sin embargo, realizar una unidad de Control Global del sistema resulta muy complejo, ya que si bien su realización es completamente posible, el uso de contadores involucrados para determinar no sólo los datos a considerar del filtraje de la imagen, sino también el control de las direcciones de la SDRAM, además de la retención de datos en la obtención de la Imagen Analítica (ya que la parte real de dicha imagen es la señal original), hacen que la frecuencia de trabajo sea muy baja, la cual sería menor a los 133MHz con que trabaja la SDRAM disponible, con lo que un funcionamiento en modo *pipeline* no sería aprovechado. Es por ese motivo que la arquitectura desarrollada se basa en dicho esquema (ver figura 1.10), pero con la diferencia que la unidad de Control Global del sistema es reemplazado por ocho unidades independientes, de menor tamaño, y una unidad de control general que tiene la función de controlar, mediante el uso de habilitadores, esas ocho unidades independientes, los multiplexores y demultiplexores a requerirse. Cada una de las ocho unidades de control independiente servirán para:

i. Recibir los coeficientes de usuario para el filtro, el tamaño total de la imagen original, el número de filas, número de columnas, la imagen original y escribirla en la SDRAM (bloque CARGADATOSSDRAM en figura 1.11).
ii. Leer imagen original (bloque LEEORIGINAL en figura 1.11).
iii. Escribir la imagen analítica (bloque ESCRIBEHILBERT en figura 1.11).
iv. Leer la imagen analítica (bloque LEEHILBERT en figura 1.11).
v. Escribir la imagen analítica filtrada en filas (bloque ESCRIBEFILAS en figura 1.11).
vi. Leer el resultado anterior (bloque LEEFILAS en figura 1.11).
vii. Escribir la fase luego de filtrar el resultado anterior en columnas (bloque ESCRIBEFASE en figura 1.11).
viii. Leer fase obtenida y enviarla a la PC (bloque LEEFASE en figura 1.11).

De esta forma, hay ciertas unidades independientes que deben trabajar en conjunto, al mismo tiempo, con lo que los procesos a seguir quedan definidos así:

a. Recepcionar los coeficientes de usuario, los datos de la imagen original (tamaño, número de filas y columnas), la imagen original y escribirla en la SDRAM.
b. Leer la imagen original y escribir la imagen analítica.
c. Leer la imagen analítica y escribir la imagen filtrada en filas.
d. Leer la imagen filtrada en filas, filtrarla en columnas, obtener fase y escribirla.
e. Enviar fase a la PC.

Se incluye un bloque cuya función es llevar la cuenta de los ciclos que toma el proceso para poder luego conocer el tiempo que se tomó en realizar el proceso completo. Además, debido a que en el proceso "b" se utilizan los coeficientes para la transformada de Hilbert y luego (procesos "c" y "d"), los coeficientes del usuario, fue añadido un bloque encargado de enviar los coeficientes requeridos al bloque encargado de realizar el filtraje en una dimensión. La figura 1.11 muestra los bloques descritos considerando las señales de control necesarias para el sincronismo del sistema y el uso de multiplexores y demultiplexores necesarios para un correcto control del sistema.

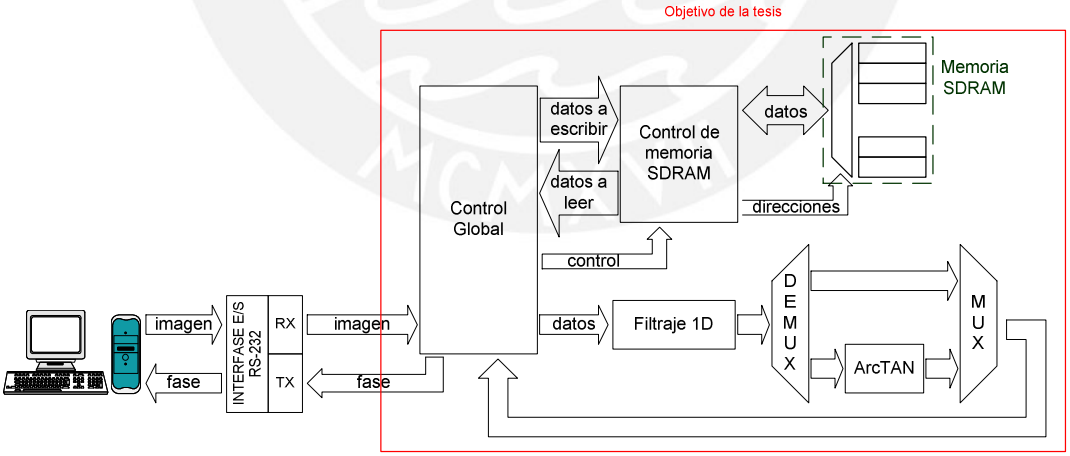


Figura 1.10. Arquitectura propuesta usando una memoria SDRAM y una unidad de Control Global.

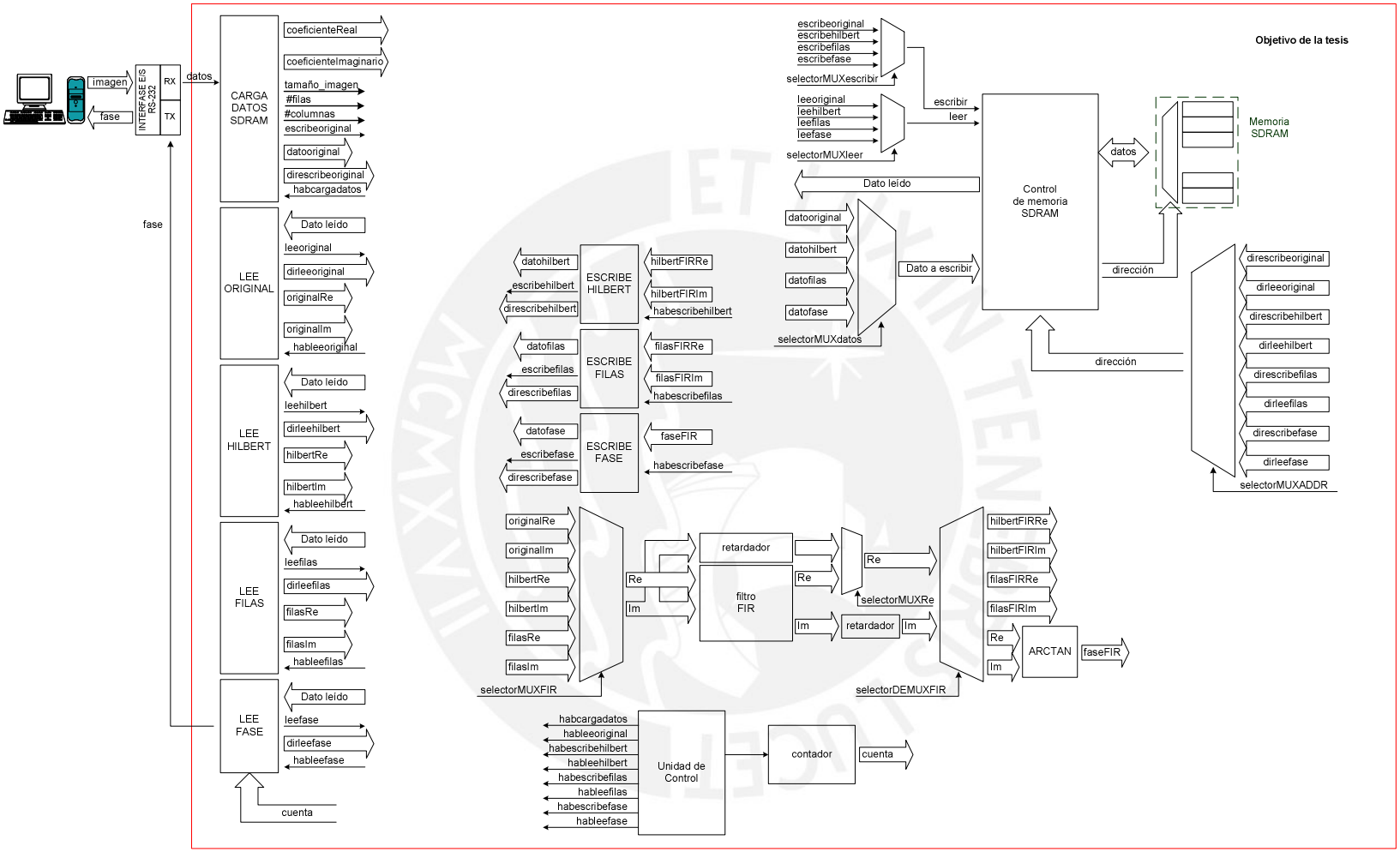


Figura 1.11. Arquitectura propuesta a desarrollar. Se observa las ocho unidades independientes, la unidad de control, el uso de multiplexores y demultiplexores, además de bloques RETARDADORES (registros) para asegurar el sincronismo.

2. ARQUITECTURA DESARROLLADA

2.1 Introducción

En la sección 1.4.2 se describió la arquitectura propuesta para el desarrollo del presente trabajo, por lo que en esta sección se muestra el desarrollo de la arquitectura propuesta para que funcione de manera correcta y eficiente en un FPGA.

Al momento de desarrollar la arquitectura, se agregó un bloque encargado de cargar los coeficientes requeridos en el proceso de filtraje, dicho bloque es controlado por la Unidad de Control (ver figura 2.1). Este bloque, conjuntamente con los descritos en la sección anterior, fue desarrollado y optimizados para su funcionamiento en un FPGA utilizando las siguientes consideraciones:

- Cada bloque debe ser independiente. Esto es, debe ser capaz de funcionar en este sistema como en cualquier otro en que se desee usar.
- Cada bloque debe ser lo más amigable posible para el usuario. Es decir, su uso debe ser de forma sencilla de manera que no sea necesario un alto conocimiento del tema para utilizarlo.
- Cada bloque debe tener una señal que le indique que realice su trabajo y una señal que indique que el trabajo ha sido realizado. Dichas señales sirven como indicadores de datos correctos tanto a la entrada como a la salida.
- Debido a que la memoria SDRAM externa trabaja con una frecuencia máxima de 143MHz [25] (la frecuencia óptima es de 133MHz), y que la frecuencia del reloj externo es de 33.333MHz, todos los bloques deben funcionar como mínimo a 135MHz, siendo lo ideal una frecuencia de 143MHz. Se tomó el valor de 135MHz debido no sólo a los 2MHz de diferencia con la frecuencia de la memoria SDRAM, sino también que ante la experiencia en el desarrollo de los trabajos utilizando FPGAs se ha deducido que la frecuencia máxima de trabajo de un sistema en conjunto como mínimo será la del bloque más lento y, no necesariamente la frecuencia del bloque de manera individual es la misma que la mínima del sistema, ya que por el uso de constantes en las señales internas la frecuencia máxima puede aumentar. La frecuencia de 133.33MHz se logra con el uso de uno de los *PLLs* internos del FPGA STRATIX.

- Relacionado al punto anterior, cada bloque debe ser capaz de procesar un dato diferente por cada ciclo de reloj.
- Cada bloque debe registrar los valores tanto de entrada como salida, de modo que se obtenga un mejor sincronismo de procesamiento, además de obtener una mayor frecuencia máxima de trabajo.

Con las consideraciones mencionadas y el objetivo de que sea una arquitectura *pipeline* indicado en la sección 1.4, el proceso a seguir en el proceso total de la imagen está definido por:

i. Recepcionar los coeficientes de usuario, los datos de la imagen original (tamaño, número de filas y columnas), la imagen original y escribirla en la SDRAM.
ii. Cargar los coeficientes necesarios para la transformada de Hilbert.
iii. Leer la imagen original y escribir la imagen analítica, al mismo tiempo que se inicia el contador que lleva la cuenta del tiempo del proceso total.
iv. Cargar los coeficientes necesarios para el filtrado de la imagen en el canal deseado (coeficientes definidos por el usuario).
v. Leer la imagen analítica y escribir la imagen filtrada en filas.
vi. Leer la imagen filtrada en filas, filtrarla en columnas, obtener fase y escribirla en la memoria SDRAM.
vii. Detener el contador del proceso total.
viii. Enviar el tiempo empleado y la fase obtenida a la PC.

De esta forma, los bloques desarrollados para obtener la fase son:

a. Receptor serial
b. Bloque de convolución
c. Controlador de la memoria SDRAM
d. Bloque arcotangente
e. Unidad de control
f. Escribe coeficientes
g. Bloque cargar datos y coeficientes del usuario
h. Bloque leer imagen original
i. Bloque escribir imagen analítica
j. Bloque leer imagen analítica

k. Bloque escribir imagen filtrada en filas
l. Bloque leer imagen filtrada en filas
m. Bloque escribir fase obtenida
n. Bloque leer fase obtenida y contador
o. Demultiplexor y multiplexores
p. Bloques retardadores.
q. Transmisor serial.
r. Contador del proceso.

Siendo los bloques del “g” al “n” unidades de control que son controladas por una unidad de control del sistema desarrollado (ver sección 2.6).

Todos los bloques fueron descritos utilizando VHDL en el programa Quartus II 4.0, y se consideró como límite imágenes en escala de grises de 256 tonalidades y con dimensiones de máximo 512 filas y 512 columnas, siendo una condición del sistema que el número de columnas sea múltiplo de ocho. Todos los bloques cumplen con las consideraciones antes mencionadas y las siguientes líneas describirán el comportamiento funcional de los bloques, así como su desarrollo, resultados independientes y consideraciones tomadas en cada uno de ellos.

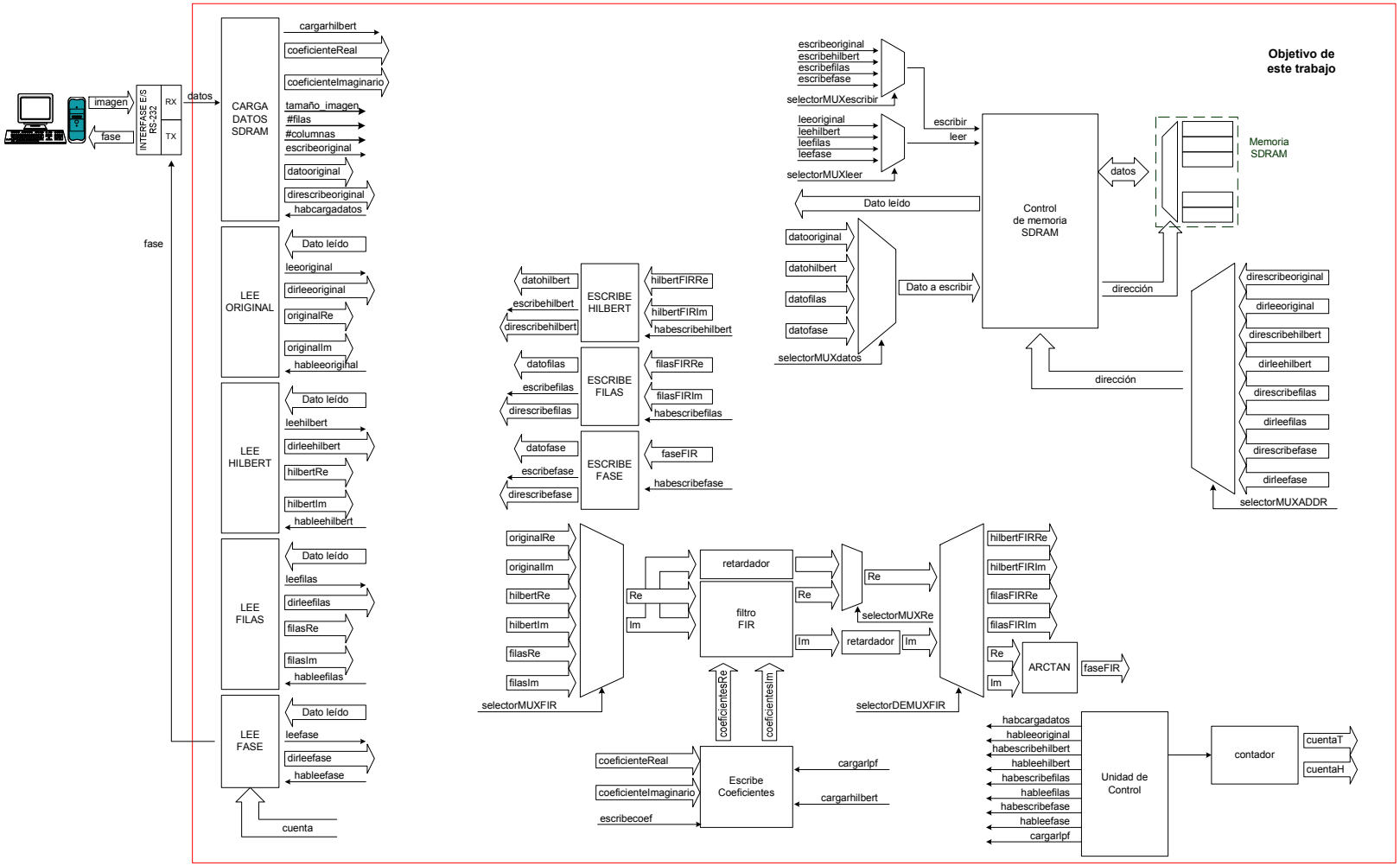


Figura 2.1. Arquitectura desarrollada.

2.2 Receptor serial

2.2.1 Comportamiento funcional

El bloque encargado de la recepción serial de los datos desde una PC debe ser parametrizable en cuanto a la velocidad de transferencia de datos, además debe contar con una entrada de un solo bit y una entrada de reinicio asíncrono. Como salida debe entregar la palabra recibida con una señal que indique el resultado entregado es el correcto. Es decir, recibe datos de manera serial desde la PC y los entrega al sistema en forma paralela.

Se decidió utilizar el formato de envío con 8 bits de datos, sin paridad, y con un solo bit de parada para la recepción. Esta decisión se argumentó en la utilización de palabras de 8 bits, además que no era necesario realizar un control de recepción de datos, ya que se tuvo como meta recepcionar la información para almacenarla en la memoria externa de forma simple debido a que, como se indicó en el capítulo 1, el objetivo de la tesis no involucra el diseño de una interfase de comunicación rápida.

2.2.2 Desarrollo

Para el desarrollo del bloque se decidió utilizar como principal componente una máquina de estados en donde cada estado representa a un bit de transmisión. El bloque desarrollado presenta 3 entradas, de 1 bit cada una, y 2 salidas, 1 de 1 bit y la otra de 8 bits con la palabra recibida (ver tabla 2.1). Además, se tiene como parámetro de ingreso el divisor requerido para la velocidad de transmisión deseada. Dicho divisor se puede elegir de una tabla indicada en el código de la descripción realizada (ver RXSERIAL.VHD en sección Anexos).

Para una segura captura de los datos, se generaron 3 señales de reloj, cuyos valores son definidos por el parámetro *divisor* antes mencionado: *rm*, *r57600*, y *rcap*. La señal *rm* controla a las otras dos señales y tiene una frecuencia igual a la velocidad de transmisión deseada dividido entre dos, y se obtiene mediante la ecuación 14.

$$divisor = entero\left(\frac{frecuencia\ del\ oscilador\ externo}{\left(velocidad\ de\ transmisión\ deseada / 2\right)}\right) \quad (14)$$

Parámetro	**Función**	
divisor	Controla la velocidad de transferencia de datos.	
Señal de entrada	**Función**	**Número de bits**
reloj133	Señal de sincronismo del bloque.	1
reset_ext	Señal de reinicio del bloque.	1
RX	Señal de entrada de datos de forma serial.	1
Señal de salida	**Función**	**Número de bits**
datoRX	Bus de datos de la palabra recibida.	8
listoRX	Señal que indica que el dato entregado es correcto.	1

Tabla 2.1. Parámetro y señales de entrada, y salida, del bloque Receptor Serial.

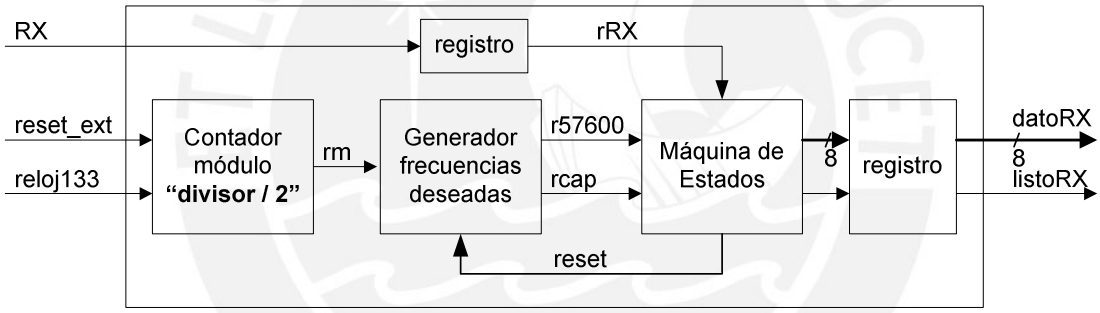


Figura 2.2. Sub-bloques utilizados en el desarrollo del bloque Receptor Serial.

El desarrollo del bloque se realizó mediante tres sub-bloques y dos registros internos (ver figura 2.2.). Los sub-bloques interactúan entre ellos de manera síncrona y con esto se logra un mejor desempeño en el procesamiento. El bloque *Contador módulo "divisor / 2"* utiliza el parámetro divisor y es el encargado de generar la señal de reloj *rm* con la frecuencia antes indicada para controlar el bloque *Generador frecuencias deseadas*, en donde se generan las otras dos señales mencionadas. Las señales *r57600* y *rcap* son de la misma frecuencia, con la diferencia que mientras la primera tiene un desfase de 0 grados y sirve para controlar los cambios de estado, la segunda tiene un desfase de 180 grados y sirve para capturar la señal *rRX*, la cual es el dato del registro que almacenó la señal de entrada *RX* (ver figura 2.3).

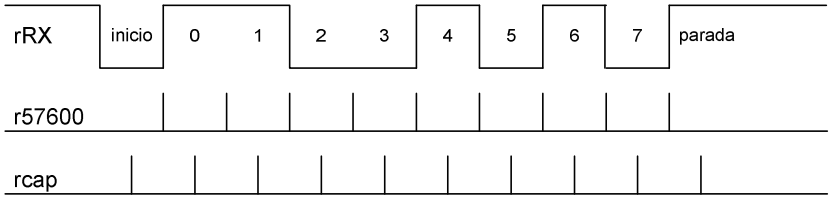


Figura 2.3. Señales *r57600* y *rcap* para su utilización con la señal *rRX*.

El sub-bloque *Máquina de estados* tiene diez estados: *espera, inicio, rec0, rec1, rec2, rec3, rec4, rec5, rec6, rec7* y *parada*; estos estados son controlados por las señales *rRX, rcap* y *r57600*, y son los encargados de capturar el dato correcto, incluyendo el reconocer si se trata del bit de inicio o de parada (ver figura 2.4). Además, en la máquina de estados se genera la señal referida a la correcta recepción del dato que, conjuntamente con el dato capturado, pasan por un registro y generan las señales *datoRX* y *listoRX*.

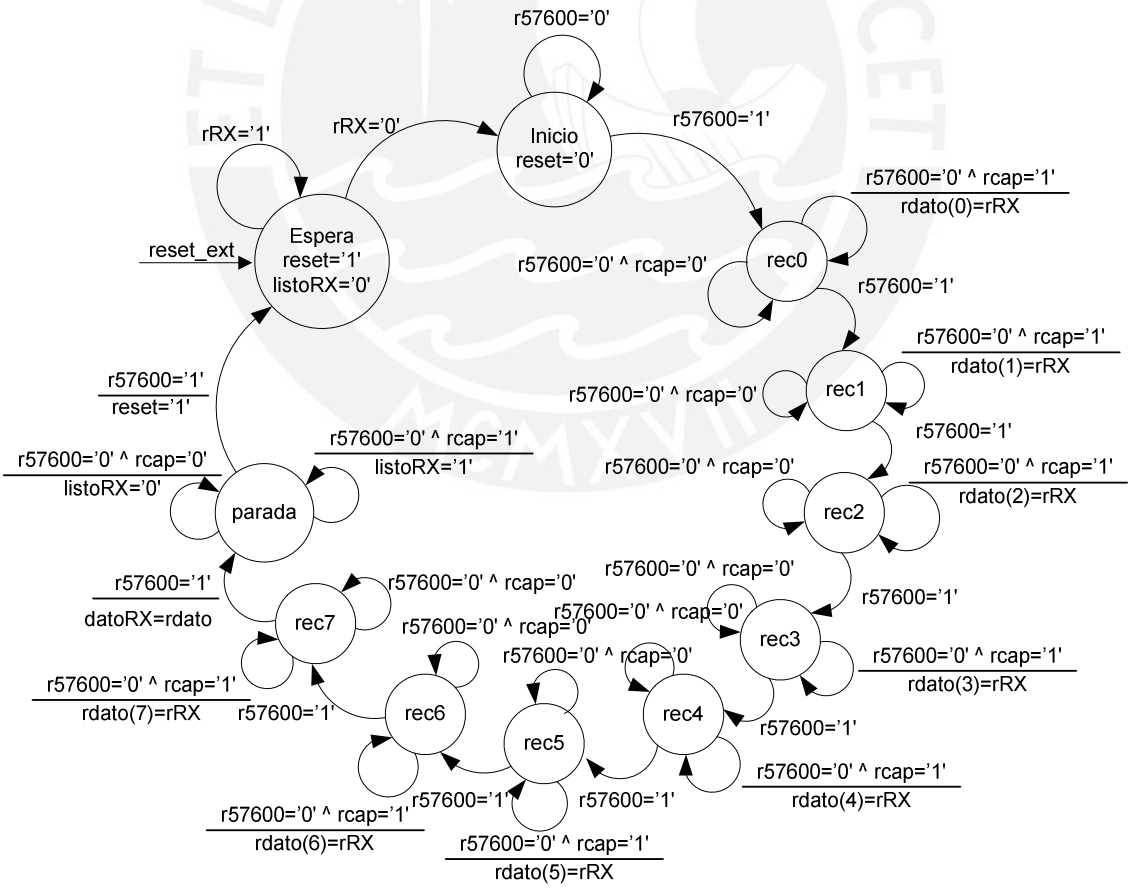


Figura 2.4. Máquina de estados del bloque Receptor Serial.

2.2.3 Resultados independientes

El bloque fue compilado utilizando un *divisor* igual a 1157 para generar una frecuencia de transmisión de 57600bps. El bloque diseñado ocupa solamente 84 Elementos Lógicos –LEs- en el FPGA STRATIX EP1S25F672C6 utilizado, alcanzando una frecuencia máxima de trabajo de 190.62MHz con una compilación optimizada en velocidad, mientras que 77 LEs y 137.65MHz con una optimización para el área a utilizar. La tabla 2.2 muestra los resultados obtenidos ante diferentes tipos de STRATIX, STRATIX II y de ACEX1K. La simulación del bloque, a la frecuencia de transmisión indicada, se muestra en la sección Anexos.

	Optimización	Área			Velocidad		
	FPGA	LEs	%LEs	fmax (MHz)	LEs	%LEs	fmax (MHz)
STRATIX	EP1S10F672C6	77	0.73	137.65	84	0.79	190.62
	EP1S10F672C7	77	0.73	132.56	84	0.79	120.96
	EP1S25F672C6	77	0.30	137.65	84	0.33	190.62
	EP1S25F672C7	77	0.30	132.56	84	0.33	120.96
	EP1S60F1508C6	77	0.13	137.65	84	0.15	190.62
	EP1S60F1508C7	77	0.13	132.56	84	0.15	120.96
STRATIX II	EP2S15F484-3	58	0.37	287.36	60	0.38	311.33
	EP2S15F484-4	58	0.37	266.31	60	0.38	293.00
	EP2S15F484-5	58	0.37	239.06	60	0.38	268.24
	EP2S30F672-3	58	0.17	287.36	60	0.18	311.33
	EP2S30F672-4	58	0.17	266.31	60	0.18	293.00
	EP2S30F672-5	58	0.17	239.06	60	0.18	268.24
ACEX	EP1K10QC208-1	78	13.54	161.81	87	15.10	165.29
	EP1K10QC208-2	78	13.54	141.24	87	15.10	143.88
	EP1K10QC208-3	78	13.54	108.01	87	15.10	110.50
	EP1K50FC484-1	78	2.71	173.01	87	3.02	173.91
	EP1K50FC484-2	78	2.71	137.36	87	3.02	137.93
	EP1K50FC484-3	78	2.71	108.93	87	3.02	109.29

Tabla 2.2. Resultados obtenidos usando distintos FPGAs para el bloque receptor serial.

2.3 Bloque de convolución

2.3.1 Objetivo

Desarrollar un bloque para el filtraje de señales, de tipo complejo, usando filtros, del tipo complejo también, en tiempo real y de forma paralela, de alta velocidad, y sin el uso de memorias o acumuladores internos, es decir, un bloque del tipo *pipeline*. Este bloque es el bloque principal de toda la

arquitectura desarrollada debido a que el trabajo se basa, en casi todo el proceso, en realizar filtraje de filas o columnas.

2.3.2 Comportamiento funcional

El bloque recibe como entrada un dato del tipo complejo (parte real y parte imaginaria), este dato es capturado mediante una señal encargada de cargarlo, luego este dato es convolucionado con los filtros complejos almacenados, conjuntamente con los demás datos antes ingresados, para obtener la convolución final, el dato obtenido se entrega conjuntamente con una señal que informa que el dato está listo. Además, el bloque debe permitir al usuario ingresar el número de bits del dato de entrada y de los coeficientes utilizados. El tamaño de la palabra resultante es mayor en 5 bits que la suma del tamaño del dato de entrada más el tamaño del coeficiente usado.

2.3.3 Desarrollo

Para lograr que el bloque desarrollado sea completamente parametrizable, se desarrolló el código del bloque (ver CONVOLUCION.VHD en sección Anexos) de modo que tanto el número de ciclos de sus multiplicaciones y sumas involucradas, como el número de coeficientes utilizados son parámetros de entrada.

Para lograr el objetivo de que se realice el filtrado en tiempo real fue necesario hacer que todos los bloques funcionen en tiempo real, además que tengan la ventaja de *pipeline* individualmente. De esta manera, el sistema desarrollado tiene los bloques mostrados en la figura 2.5 (ver tabla 2.3 para los nombres de los parámetros y de las señales de entrada y salida del bloque). Cabe señalar que el diagrama de bloques indicado considera que cada registro está almacenando un número complejo, por lo que los multiplicadores mostrados son de números complejos, mientras que el sumador final, viene a ser 2 sumadores: uno para la parte real y otro para la parte imaginaria. De esta forma, las siguientes líneas explicarán los dos componentes principales del sistema desarrollado, es decir, el multiplicador y el sumador final para que trabajen de modo parametrizable.

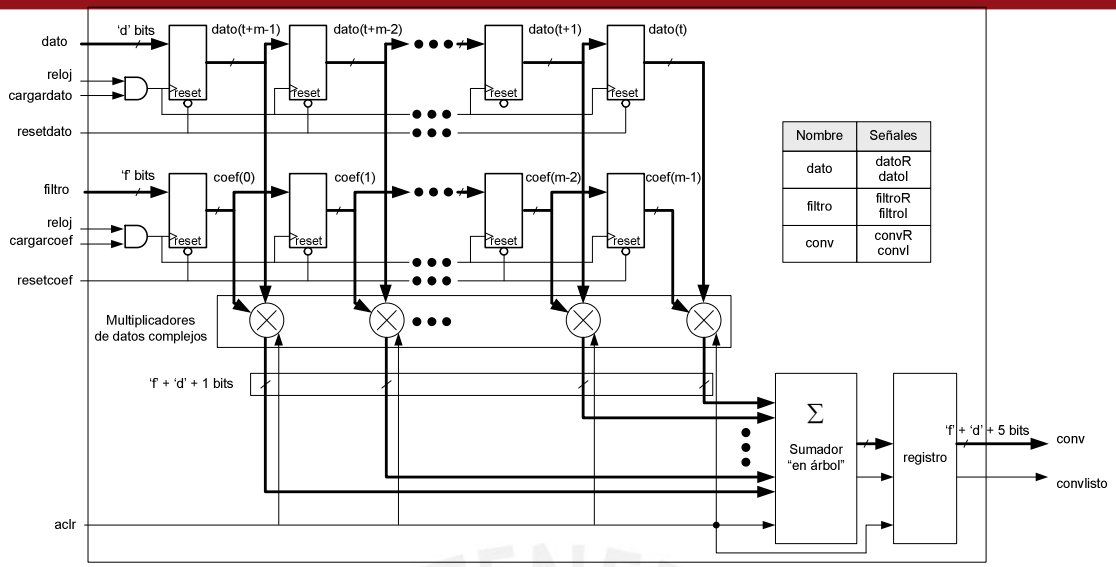


Figura 2.5. Diagrama de bloques del bloque CONVOLUCIÓN.

Parámetro	**Función**	
tam_ab	Indica el tamaño en bits del dato de entrada.	
tam_cd	Indica el tamaño en bits del coeficiente.	
pipeline_mul	Indica el número de ciclos que se demora el sistema en cada una de las sumas que realiza.	
pipeline_sum	Indica el número de ciclos que se demora el sistema en cada una de las multiplicaciones que realiza.	
taps	Indica el número de coeficientes a usar.	
Señal de entrada	**Función**	**Número de bits**
reloj133	Señal de sincronismo del bloque.	1
reset	Señal de reinicio del bloque.	1
datoR	Parte real del dato de entrada.	tam_ab
datoI	Parte imaginaria del dato de entrada.	tam_ab
filtroR	Parte real del coeficiente a cargar.	tam_cd
filtroI	Parte imaginaria del coeficiente a cargar.	tam_cd
cargarcoef	Señal que sirve para cargar los coeficientes a usar.	1
cargardato	Señal que sirve para cargar el dato a usar.	1
Señal de salida	**Función**	**Número de bits**
convR	Parte real de la convolución obtenida.	tam_ab + tam_cd + 5
convI	Parte imaginaria de la convolución obtenida.	tam_ab + tam_cd + 5
convlisto	Señal que indica que el dato obtenido está listo.	1
coeflistos	Señal que indica que los coeficientes han sido almacenados.	1

Tabla 2.3. Parámetros y señales de entrada, y salida, del bloque Convolución.

2.3.3.1 Multiplicador complejo

Basándose en la ecuación 13, las entradas del multiplicador complejo son 2 números de ese tipo, de esta manera se realizó la arquitectura mostrada en la figura 2.6.

Dicho multiplicador es común a cualquier otro realizado en VHDL en cualquier sistema (ver MULT_COMP.VHD en sección Anexos), incluyendo el parámetro del número de ciclos en obtención de la respuesta dominado por la señal de reloj entrante. Ahora, la diferencia y principal ventaja para el sistema de filtraje desarrollado es que se aprovecha las características de la arquitectura de los FPGAs de la familia STRATIX [3], ya que esta familia presenta multiplicadores-sumadores que no ocupan elementos lógicos. Es decir, según el STRATIX utilizado y el número de bits de la palabra entrante, se tendrá el número total de multiplicadores que se recomienda tener de manera de no utilizar LEs para este fin, así sólo se utilizan los LEs para los registros necesarios.

De esta forma, por cada multiplicación compleja que se realiza, se utiliza cuatro multiplicadores y dos sumadores, con lo que según el tamaño de los datos de entrada y el número de coeficientes a usarse, se sabrá cuántos multiplicadores complejos se pueden configurar según la capacidad del STRATIX.

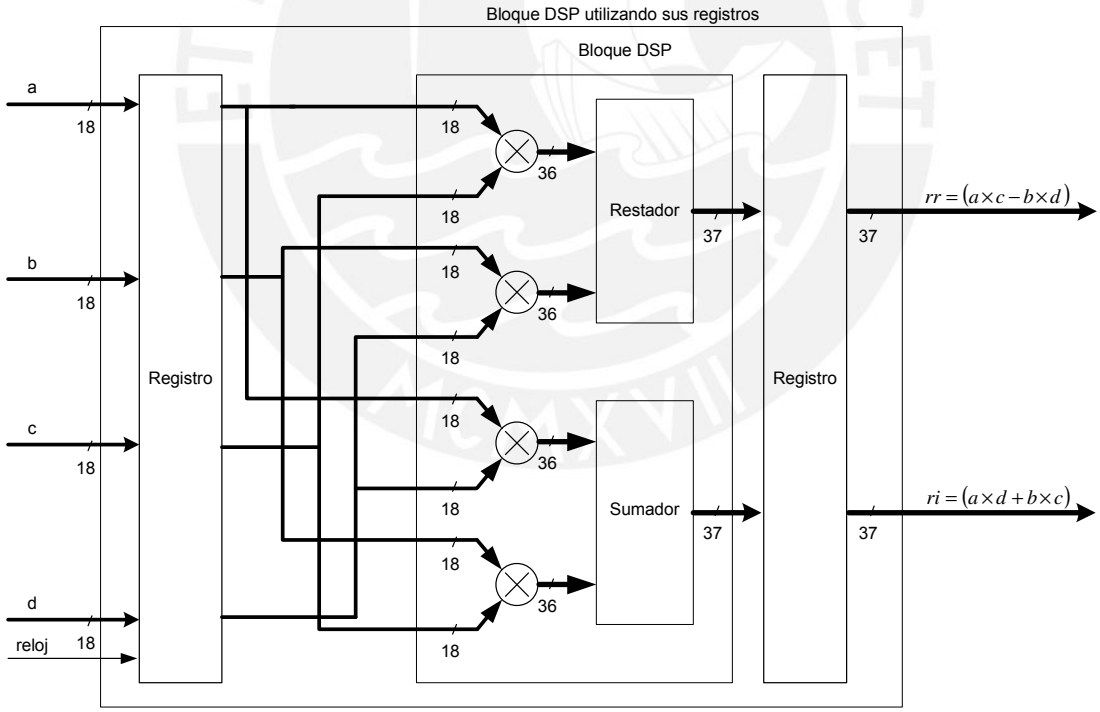


Figura 2.6. Multiplicador de datos complejos.

2.3.3.2 Sumador Complejo

Como se indicó al principio de esta sección, el sumador complejo en la última etapa del diseño, es en realidad dos sumadores: uno para la parte real y otro

para la parte imaginaria. Ahora, desde que el objetivo del trabajo es realizar un sistema *pipeline*, fue necesario evitar el uso de memorias internas o acumuladores, ya que de usarse éstos producen un retardo en ciclos de reloj, además que produce que el bloque de multiplicación no se use durante la suma acumulativa, ya que de obtener un nuevo valor, éste entraría al sumador y no permitiría el correcto funcionamiento del sumador-acumulador.

De esta manera, se desarrolló un sumador en forma de árbol, donde el número de ramas a configurarse en el FPGA depende directa y exclusivamente del número de coeficientes a utilizarse. Dicho número de coeficientes, depende de la capacidad de multiplicadores-sumadores que tenga el STRATIX a utilizar.

Debido a que la implementación realizada se utilizó un Stratix EP1S25F672C6, se desarrolló el código en VHDL para tener como máximo 16 coeficientes en el filtro, ya que se puede tener 10 multiplicadores complejos y el código se basa en potencias de 2 (ver figura 2.7).

En esta etapa, la descripción VHDL conjuntamente con el parámetro número de coeficientes, cumple un papel fundamental en el diseño ya que si el número de coeficientes es pequeño, sería un desperdicio la creación de ramas innecesarias para la suma. Así, si el número de coeficientes determina el número de ramas a tener (ver tabla 2.4).

Número de Coeficientes (M)	Número de ramas
8 < M <= 16	4
4 < M <= 8	3
2 < M <= 4	2
M <= 2	1

Tabla 2.4. Número de ramas según el número de coeficientes.

De esta forma, una vez creada la arquitectura con el número de ramas correspondientes, los datos que no estén presentes serán conectados internamente a tierra (“0” voltios) con lo que el sumador correspondiente no es creado por el compilador. Además, debido a que según como va incrementando el número de ramas, el número de bits aumenta en 1, la

descripción duplica el bit más significativo del dato para conservar el signo del número en complemento a 2.

Finalmente, la interacción entre los multiplicadores y sumadores, nos permite ingresar un dato por cada ciclo de reloj para que luego de un número de ciclos obtener una respuesta por cada ciclo de reloj. Cabe resaltar que de esta manera, todos los bloques del sistema son utilizados siempre, con lo que se logra el *pipeline* deseado.

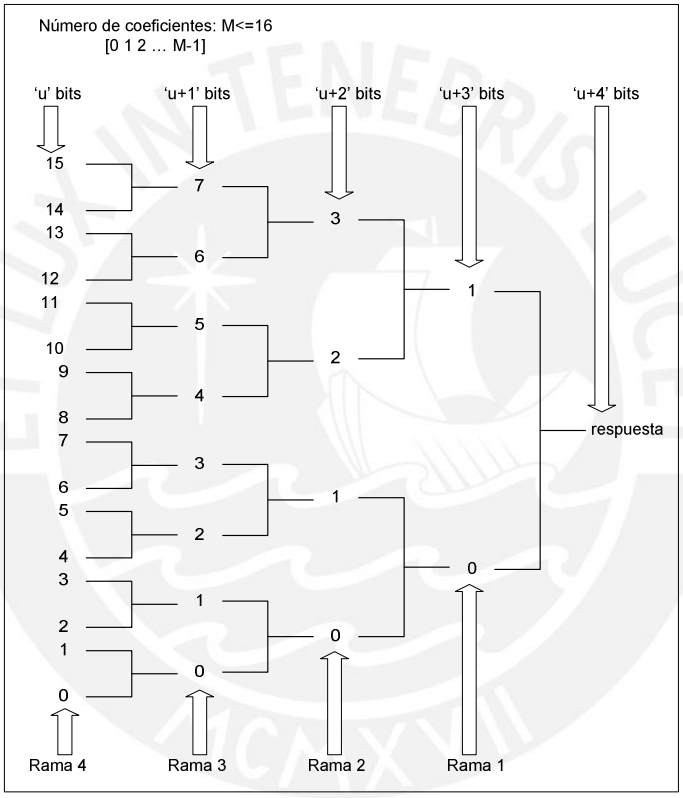


Figura 2.7. Sumador en árbol.

2.3.4 Resultados independientes

El bloque fue compilado utilizando los siguientes valores de los parámetros: pipeline_mul = 3, pipeline_sum = 3, tam_ab = 18, tam_cd = 18.

El bloque diseñado, con 5 coeficientes, ocupa 5784 LEs en el FPGA STRATIX EP1S25F672C6 utilizado, alcanzando una frecuencia máxima de trabajo de 138.37MHz con una compilación optimizada en velocidad, mientras que 5784 LEs y 138.37MHz con una optimización para el área a utilizar. El resultado de la convolución se obtiene 20 ciclos de reloj luego de haber cargado el dato, es

decir, 144.54ns luego de haber ingresado el primer dato (ver simulación en sección Anexos). La tabla 2.5 muestra los resultados obtenidos ante diferentes tipos de STRATIX y STRATIX II.

			ÁREA				
	FPGA	Coef.	LEs	%LEs	%DSPs	fmax (MHz)	Tiempo Teórico (us)
Stratix	EP1S10F672-6	5	5784	54.72	83.33	138.37	0.14454
	EP1S10F672-7	5	5784	54.72	83.33	129.62	0.15430
	EP1S25F672-6	5	5784	22.54	50.00	138.37	0.14454
	EP1S25F672-7	5	5784	22.54	50.00	129.62	0.15430
	EP1S60F1508-6	5	5784	10.13	27.78	138.37	0.14454
	EP1S60F1508-7	5	5784	10.13	27.78	129.62	0.15430
	EP1S10F672-6	7	9466	89.56	100.00	49.44	0.40453
	EP1S10F672-7	7	9466	89.56	100.00	46.22	0.43271
	EP1S25F672-6	7	7410	28.88	70.00	138.37	0.14454
	EP1S25F672-7	7	7410	28.88	70.00	129.62	0.15430
	EP1S60F1508-6	7	7410	12.97	38.89	138.37	0.14454
	EP1S60F1508-7	7	7410	12.97	38.89	129.62	0.15430
Stratix II	EP2S15F484-3	5	5001	32.06	41.67	117.19	0.17066
	EP2S15F484-4	5	5001	32.06	41.67	113.01	0.17698
	EP2S15F484-5	5	5001	32.06	41.67	106.93	0.18704
	EP2S30F672-3	5	5001	14.76	31.25	117.19	0.17066
	EP2S30F672-4	5	5001	14.76	31.25	113.01	0.17698
	EP2S30F672-5	5	5001	14.76	31.25	106.93	0.18704
	EP2S15F484-3	7	6335	40.61	58.33	117.19	0.17066
	EP2S15F484-4	7	6335	40.61	58.33	113.01	0.17698
	EP2S15F484-5	7	6335	40.61	58.33	106.93	0.18704
	EP2S30F672-3	7	6335	18.70	43.75	117.19	0.17066
	EP2S30F672-4	7	6335	18.70	43.75	113.01	0.17698
	EP2S30F672-5	7	6335	18.70	43.75	106.93	0.18704
			VELOCIDAD				
Stratix	EP1S10F672-6	5	5784	54.72	83.33	138.37	0.14454
	EP1S10F672-7	5	5784	54.72	83.33	129.62	0.15430
	EP1S25F672-6	5	5784	22.54	50.00	138.37	0.14454
	EP1S25F672-7	5	5784	22.54	50.00	129.62	0.15430
	EP1S60F1508-6	5	5784	10.13	27.78	138.37	0.14454
	EP1S60F1508-7	5	5784	10.13	27.78	129.62	0.15430
	EP1S10F672-6	7	9466	89.56	100.00	49.44	0.40453
	EP1S10F672-7	7	9466	89.56	100.00	46.22	0.43271
	EP1S25F672-6	7	7410	28.88	70.00	138.37	0.14454
	EP1S25F672-7	7	7410	28.88	70.00	129.62	0.15430
	EP1S60F1508-6	7	7410	12.97	38.89	138.37	0.14454
	EP1S60F1508-7	7	7410	12.97	38.89	129.62	0.15430
Stratix II	EP2S15F484-3	5	5001	32.06	41.67	117.19	0.17066
	EP2S15F484-4	5	5001	32.06	41.67	113.01	0.17698
	EP2S15F484-5	5	5001	32.06	41.67	106.93	0.18704
	EP2S30F672-3	5	5001	14.76	31.25	117.19	0.17066
	EP2S30F672-4	5	5001	14.76	31.25	113.01	0.17698
	EP2S30F672-5	5	5001	14.76	31.25	106.93	0.18704

	EP2S15F484-3	7	5001	32.06	58.33	117.19	0.17066
	EP2S15F484-4	7	5001	32.06	58.33	113.01	0.17698
	EP2S15F484-5	7	5001	32.06	58.33	106.93	0.18704
	EP2S30F672-3	7	5001	14.76	43.75	117.19	0.17066
	EP2S30F672-4	7	5001	14.76	43.75	113.01	0.17698
	EP2S30F672-5	7	5001	14.76	43.75	106.93	0.18704

Tabla 2.5. Resultados obtenidos usando distintos FPGAs para el bloque convolución.

2.4 Controlador de la memoria SDRAM

2.4.1 Introducción

Controlar una memoria externa SDRAM involucra la generación de las señales de control necesarias para su inicio, configuración y control. Las tarjetas de evaluación y desarrollo basadas en FPGAs que se venden actualmente incorporan controladores diseñados por el fabricante de la tarjeta o de la misma memoria. Debido a que la tarjeta adquirida no contaba con un controlador para su memoria, se contactó a los fabricantes y ellos recomendaron usar el controlador de la compañía Altera. El controlador mencionado no era compatible con la memoria SDRAM usada, además de tener una forma de usar no amigable y compleja, motivo por el cual, se alteró el controlador para que sea compatible con la memoria SDRAM usada y, se diseñó un controlador para el controlador mencionado. Estos dos bloques en conjunto generan el controlador completo de manera que sea fácil de usar. Las siguientes líneas explicarán el desarrollo de ambos controladores para luego explicar el funcionamiento del controlador completo.

2.4.2 Controlador de memoria SDRAM de Altera

El controlador de memoria SDRAM de Altera (ver SDR_SDRAM.VHD en sección Anexos) es accesible vía la página web de la compañía: www.altera.com. Debido a que diseñar el controlador de la memoria SDRAM fue una necesidad extra por la falta de un controlador por la compañía MJL, las siguientes líneas explicarán de forma breve el controlador original de Altera para luego indicar los cambios realizados para que sea compatible con la memoria utilizada: SDRAM de 64Mbits de Micron (MT48LC2M32B2TG-7).

Las características del controlador de Altera son, entre otras:

- Selección de tamaño de *burst* de 1, 2, 4 o 8 palabras.
- Selección de 2 o 3 ciclos de *CAS*.
- Contador de 16 bits para el refresco de la memoria.
- Dos selectores de *chip* de la SDRAM.
- Soporta las instrucciones: NOP, READA, WRITEA, AUTO_REFRESH, PRECHARGE, ACTIVATE, BURST_STOP.
- Selección de tamaño de datos de 16, 32 o 64 bits.

Para lo cual cuenta con las entradas y salidas (ver figura 2.8) indicadas en la tabla 2.6.

El controlador de memoria SDRAM utiliza comandos para generar los comandos enviados a la memoria externa. Estos comandos son:

- NOP: No realizar ninguna operación.
- READA: Realizar la lectura según la configuración (*burst* o página completa).
- WRITEA: Realizar la lectura según la configuración (*burst* o página completa).
- REFRESH: Realizar un refresco de la memoria (sólo en modo página completa).
- PRECHARGE: Comando de PRECARGA para los bancos de la memoria SDRAM.
- LOAD_MODE: Comando para cargar el registro referente al tipo de lectura/escritura de la memoria SDRAM.
- LOAD_REG1: Comando para cargar el registro del controlador referente al tipo de lectura/escritura de la memoria SDRAM.
- LOAD_REG2: Comando para cargar el registro del contador utilizado para el refresco de la memoria SDRAM.

Parámetro	Función	
asize	Indica el tamaño en bits de la dirección de entrada.	
dsize	Indica el tamaño en bits de los datos.	
rowsize	Indica el tamaño en bits de la fila indicada en la dirección.	
colsize	Indica el tamaño en bits de la columna indicada en la dirección.	
banksize	Indica el tamaño en bits de los bancos disponibles en la memoria SDRAM.	
rowstart	Indica la posición de inicio de la fila deseada en la dirección de ingreso.	
colstart	Indica la posición de inicio de la columna deseada en la dirección de ingreso.	
bankstart	Indica la posición de inicio del banco deseada en la dirección de ingreso.	
Señal de entrada	**Función**	**Número de bits**
CLK	Señal de sincronismo del bloque.	1
RESET_N	Señal de reinicio del bloque.	1
ADDR	Dirección del dato a utilizar	asize
CMD	Comando a utilizar.	3
DATAIN	Dato de entrada para escribir en la memoria SDRAM.	dsize
DM	Máscara para la selección del *byte* a utilizar del dato.	dsize/4
Señal bidireccional	**Función**	**Número de bits**
DQ	Bus de datos para la escritura o lectura de la memoria SDRAM.	dsize
Señal de salida	**Función**	**Número de bits**
CMDACK	Señal que indica que el comando ingresado ha sido reconocido.	3
DATAOUT	Parte imaginaria de la convolución obtenida.	dsize
SA	Bus de direcciones que indica la posición de la fila y de la columna a utilizar.	12
BA	Banco de la memoria SDRAM a utilizar.	2
CS_N	Selector de *chip.*	2
CKE	Habilitador de la señal de reloj.	1
RAS_N	Selector de filas.	1
CAS_N	Selector de columnas.	1
WE_N	Habilitador de escritura.	1
DQM	Máscara para la selección del *byte* a utilizar del dato.	dsize/4

Tabla 2.6. Señales de entrada y salida del controlador de memoria SDRAM de Altera.

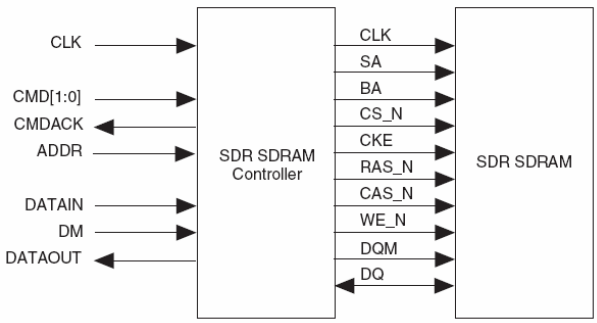


Figura 2.8. Bloque de control de memoria SDRAM de Altera (obtenida de [26]).

La memoria utilizada, como se ha indicado, no era compatible con este controlador debido a que:

- La memoria MT48LC2M32B2TG-7 sólo tiene un selector de *chip*, mientras que el controlador tiene dos.
- La memoria MT48LC2M32B2TG-7 tiene un bus de direcciones de 11 bits, mientras que el controlador usa 12 bits.
- No existía relación entre los registros del controlador y el registro a cargar de la memoria externa SDRAM.

Los problemas indicados, además que el controlador no era amigable en su forma de utilizar, fueron solucionados alterando la descripción de modo que tenga como salida sólo un selector de *chip*. Este cambio tuvo un efecto directo en el bus de direcciones para enviar a la memoria SDRAM, ya que originalmente se basaba en los dos selectores de *chip* para generar dicha dirección. Además, al tener el registro relacionado a la instrucción LOAD_REG1 las especificaciones del tipo de lectura/escritura, fue necesario describir que al momento de cargar el registro de la memoria externa, esta información sea enviada. De esta forma, este controlador está descrito para interactuar con la memoria externa SDRAM en cuanto a los comandos y direcciones necesarias, cumpliendo también con el tamaño en bits necesarios para su control.

2.4.3 Controlador del controlador de memoria SDRAM de Altera

Este bloque fue creado con el objetivo que conjuntamente con el controlador de memoria SDRAM de Altera, descrito anteriormente, se obtenga un controlador amigable y capaz de generar las secuencias necesarias para, en un principio, la inicialización de la memoria SDRAM, y luego poder escribir o leer en ella.

Considerando la secuencia de comandos necesarios para almacenar los registros del controlar de memoria SDRAM de Altera, y al mismo tiempo de la memoria externa SDRAM, se creó un bloque que es capaz de iniciar, mediante una señal, la memoria SDRAM con un ciclo *burst* definido por el usuario, además de una señal de escritura y otra para escritura. Además, el bloque

entrega el dato leído de la memoria SDRAM tanto en forma paralela como serial, indicando mediante otras señales que el dato está listo y que la memoria se puede usar, es decir, que no se encuentra en un proceso de refresco o de lectura o escritura (ver CONTROL_CTRLSDRAM.VHD en sección Anexos). De esta manera, el bloque tiene los parámetros, entradas y salidas definidas en la tabla 2.7.

El diseño del bloque se realizó usando sub-bloques (ver figura 2.9) basándose en una máquina de estados para el control del bloque. De esta manera, el usuario para usar la memoria SDRAM externa debe realizar los siguientes pasos:

- Iniciar la memoria SDRAM activando por un periodo de reloj la entrada *iniciar* indicando al mismo tiempo, mediante la entrada *burst,* el número de ciclos para la lectura o escritura burst.
- Esperar a que la salida *iniciado* esté activada. Esto indica que la memoria SDRAM y el controlador de memoria SDRAM han sido iniciados correctamente.
- Una vez inicializada la memoria SDRAM, el usuario puede leer o escribir en ella considerando que:
    - El usuario leerá o escribirá un número de datos continuos igual al número de ciclos burst configurado. Por ejemplo: Si se configuró un número de ciclos de burst igual a ocho, cuando se lea se leerán 8 datos seguidos y cuando se escriba se debe escribir 8 datos seguidos, siendo estos no necesariamente uno por ciclo, lo cual representa una ventaja ante el controlador de memoria SDRAM descrito en 2.4.2.
    - Cuando se lee, la señal *leidoOK* y *listoSERIAL* indicarán que los datos desde *datoout_0* hasta *datoout_7* o la salida *salidaSERIAL,* respectivamente, están listos.
    - Antes de volver a realizar una lectura o escritura, el usuario debe esperar a que la señal *usar* esté activa, caso contrario significará que una lectura o escritura está en proceso.

Parámetro	**Función**	
cont_REFRESH	Indica el valor del registro de refresco de la memoria (ver ecuación 15). Por ejemplo, la memoria SDRAM utilizada requiere 4096 ciclos de reloj cada 64ms, y utilizando una frecuencia externa de 133MHz, el valor es 2083.	
CAS_Latency	Indica el número de ciclos luego del comando READ o WRITE en activar el dato.	
RAS_CAS_delay	Indica el número de ciclos entre el comando ACTIVE y el comando READ o WRITE (ver ecuación 16)	
REFR_RAS_delay	Indica el número de ciclos entre el refresco y el comando ACTIVE (ver ecuación 17)	
asize	Indica el tamaño en bits de la dirección de entrada.	
dsize	Indica el tamaño en bits de los datos.	
Señal de entrada	**Función**	**Número de bits**
reloj133p333	Señal de sincronismo del bloque con una frecuencia de 133MHz.	1
reset	Señal de reinicio del bloque.	1
CMDACK	Señal que indica que el comando enviado al controlador de memoria SDRAM ha sido reconocido.	3
DATAIN	Dato recibido del controlador de memoria SDRAM.	Dsize
datoin	Dato de entrada para ser enviado al controlador de memoria SDRAM.	Dsize
escribir	Señal que indica que se desea escribir un dato en la memoria SDRAM.	1
iniciar	Señal que indica que se desea iniciar la memoria SDRAM.	1
leer	Señal que indica que se desea leer datos de la memoria SDRAM.	1
burst	Bus de datos que indica el número de ciclo *burst* a utilizar.	4
direcc	Dirección de entrada para escribir o leer en la memoria SDRAM.	Asize
Señal de salida	**Función**	**Número de bits**
CMD	Señal que envía el comando necesario al controlador de memoria SDRAM.	1
ADDR	Dirección enviada al controlador de memoria SDRAM.	asize
DM	Máscara para la selección del *byte* a utilizar del dato.	dsize/4
DATAOUT	Dato para enviar al controlador de memoria SDRAM.	dsize
datoout_0	Dato leído en la primera lectura burst.	dsize
datoout_1	Dato leído en la segunda lectura burst.	dsize
datoout_2	Dato leído en la tercera lectura burst.	dsize
datoout_3	Dato leído en la cuarta lectura burst.	dsize
datoout_4	Dato leído en la quinta lectura burst.	dsize
datoout_5	Dato leído en la sexta lectura burst.	dsize
datoout_6	Dato leído en la séptima lectura burst.	dsize
datoout_7	Dato leído en la octava lectura burst.	dsize
iniciado	Señal que indica que la memoria SDRAM ha sido inicializada.	1
usar	Señal que indica que la memoria SDRAM ha sido inicializada y que no hay ninguna escritura o lectura en proceso.	1
leidoOK	Señal que indica que el dato leído desde datoout_0 hasta datoout_7 está listo.	1
salidaSERIAL	Datos leídos de la memoria SDRAM entregados uno por uno.	dsize
listoSERIAL	Señal que indica que el dato recibido de manera serial está listo.	1

Tabla 2.7. Parámetros, señales de entrada y de salida del controlador del controlador de memoria SDRAM.

$$cont\_REFRESH = entero\left(\frac{periodo\ de\ refresco}{\left(periodo\ de\ reloj \Big/ periodo\ del\ oscilador\ usado\right)}\right) \quad (15)$$

$$RAS\_CAS\_delay = entero\left(\frac{tiempo\ de\ retardo\ entre\ RAS\ y\ CAS}{periodo\ de\ reloj}\right) \quad (16)$$

$$REFR\_RAS\_delay = entero\left(\frac{tiempo\ de\ retardo\ entre\ refresco\ y\ RAS}{periodo\ de\ reloj}\right) \quad (17)$$

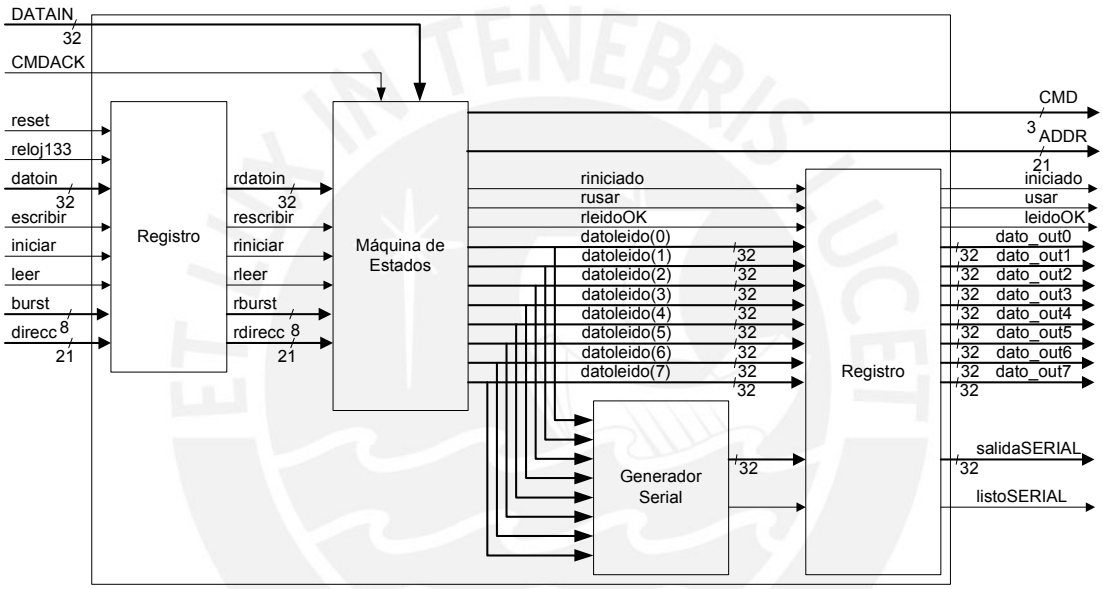


Figura 2.9. Controlador del controlador de memoria SDRAM. Bloques y señales de entrada y salida.

La máquina de estados que controla este bloque consta de 35 estados: *espera, precarga, reg2, esperareg2, reg1, esperareg1, mode, memlista, capdatos, escribedatos1, escribedatos2, leer1, leer2, leer3, rcap1, rcap2, rcap3, rcap4, rcap5, rcap6, rcap7, wcap1, wcap2, wcap3, wcap4, wcap5, wcap6, wcap7, ecap1, ecap2, ecap3, ecap4, ecap5, ecap6, y ecap7*. Debido a la complejidad de la máquina de estados y de la cantidad de estados, podemos separarlos en tres grupos: estados de inicio, estados de lectura y estados de escritura (ver figuras 2.10, 2.11 y 2.12). Es en esta máquina de estados que se genera las salidas (ver tabla 2.7) del bloque controlador del controlador de memoria

SDRAM. El sub-bloque *Generador Serial* es un multiplexor de ocho entradas y una sola salida, el cual se activa con la señal *rleidoOK* se activa.

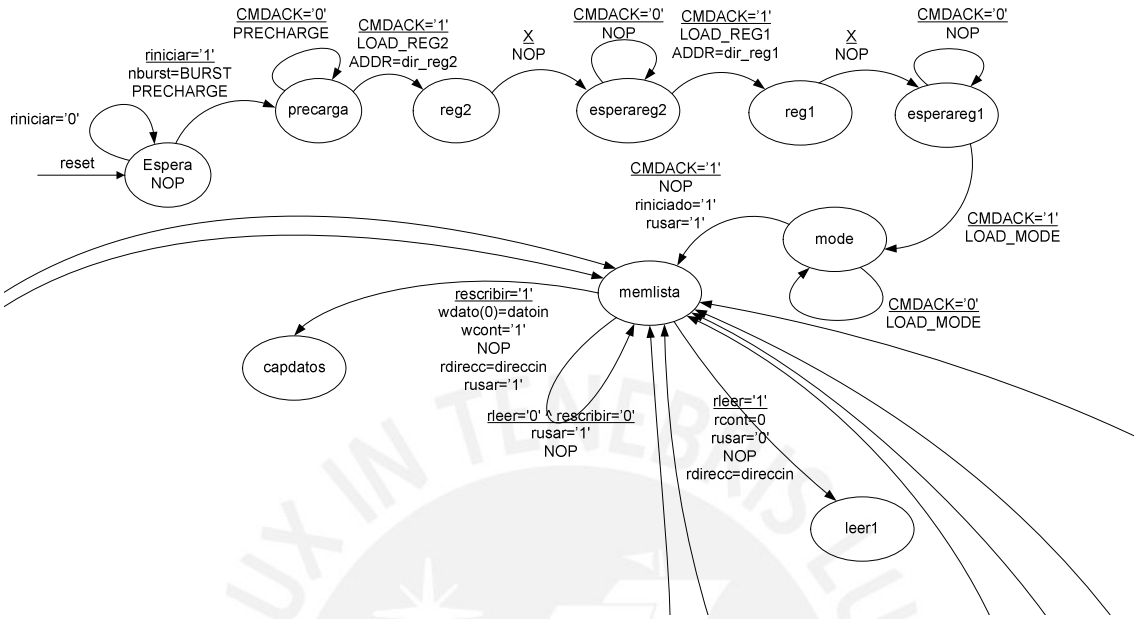


Figura 2.10. Máquina de estados primera parte. Se muestra los estados de inicio: desde el estado *espera* hasta *memlista*.

2.4.4 Resultados independientes del controlador del controlador de memoria SDRAM de Altera

El bloque diseñado ocupa solamente 1401 LEs en el FPGA STRATIX EP1S25F672C6 utilizado, alcanzando una frecuencia máxima de trabajo de 134.19MHz con una compilación optimizada en velocidad, mientras que 1397 LEs y 107MHz con una optimización para el área a utilizar. La tabla 2.8 muestra los resultados obtenidos ante diferentes tipos de STRATIX, STRATIX II y de ACEX1K. Como se observa, este bloque representa un bloque crítico a la hora de la implementación de la arquitectura completa debido a su máxima frecuencia de operación la cual es muy cercana a la frecuencia de trabajo de la memoria SDRAM.

		ÁREA			VELOCIDAD		
	FPGA	LEs	%LEs	fmax (MHz)	LEs	%LEs	fmax (MHz)
Stratix	EP1S10F672-6	1397	13.22	107.00	1401	13.25	134.19
	EP1S10F672-7	1397	13.22	103.17	1401	13.25	130.29
	EP1S25F672-6	1397	5.44	107.00	1401	5.46	134.19
	EP1S25F672-7	1397	5.44	107.00	1401	5.46	130.29

	EP1S60F1508-6	1397	2.45	103.17	1401	2.45	134.19
	EP1S60F1508-7	1397	2.45	107.00	1401	2.45	130.29
Stratix II	EP2S15F484-3	1015	6.51	224.22	1015	6.51	221.83
	EP2S15F484-4	1015	6.51	216.59	1015	6.51	214.00
	EP2S15F484-5	1015	6.51	205.47	1015	6.51	202.84
	EP2S30F672-3	1015	3.00	224.22	1015	3.00	221.83
	EP2S30F672-4	1015	3.00	216.59	1015	3.00	214.00
	EP2S30F672-5	1015	3.00	205.47	1015	3.00	202.84
Acex	EP1K10QC208-1	1430	248.26	99.30	1436	249.31	145.99
	EP1K10QC208-2	1430	248.26	90.33	1436	249.31	129.20
	EP1K10QC208-3	1430	248.26	69.11	1436	249.31	101.63
	EP1K50FC484-1	1430	49.65	109.05	1436	49.86	148.15
	EP1K50FC484-2	1430	49.65	88.73	1436	49.86	124.38
	EP1K50FC484-3	1430	49.65	70.57	1436	49.86	100.60

Tabla 2.8. Resultados obtenidos usando distintos FPGAs para el controlador del controlador de memoria SDRAM de Altera.

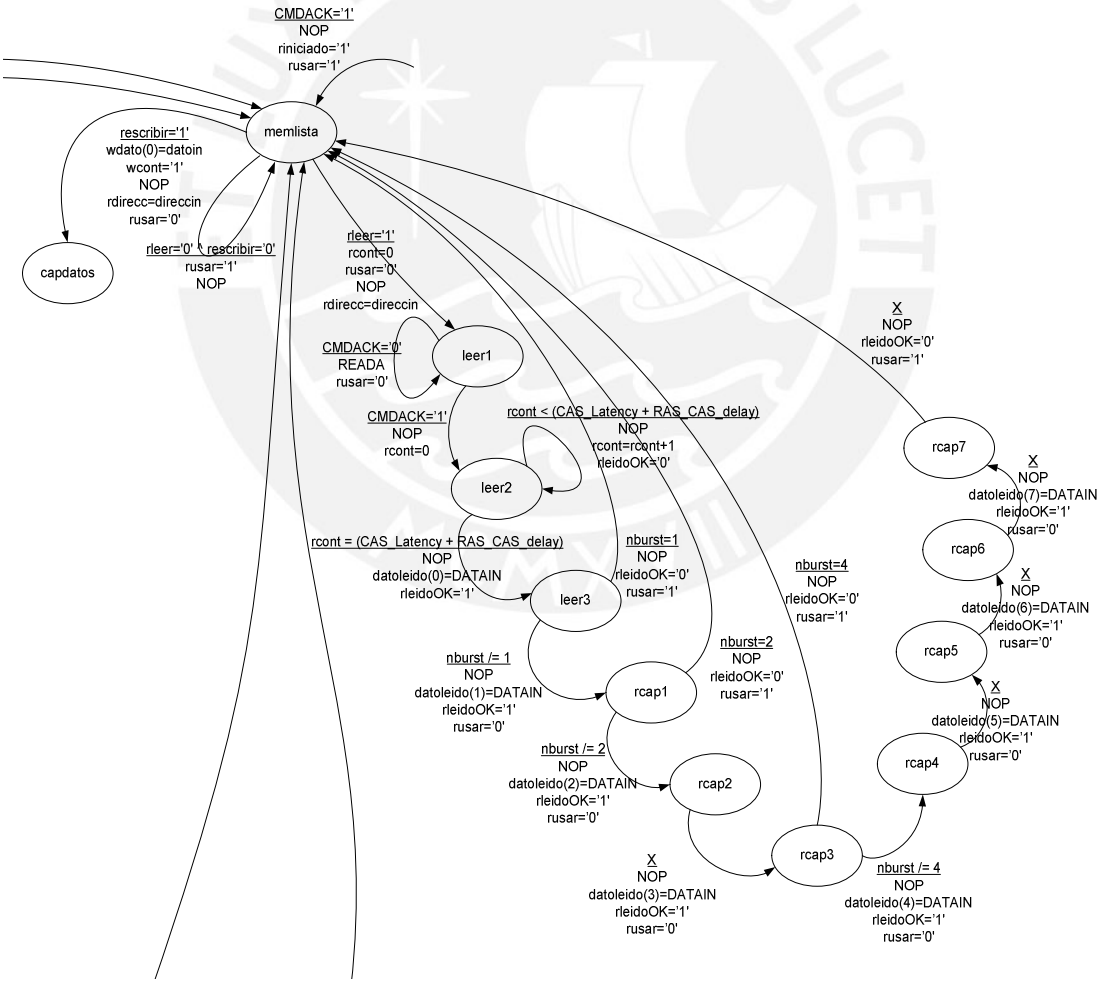


Figura 2.11. Máquina de estados segunda parte. Se muestra los estados de lectura: desde el estado *memlista* hasta *rcap7*.

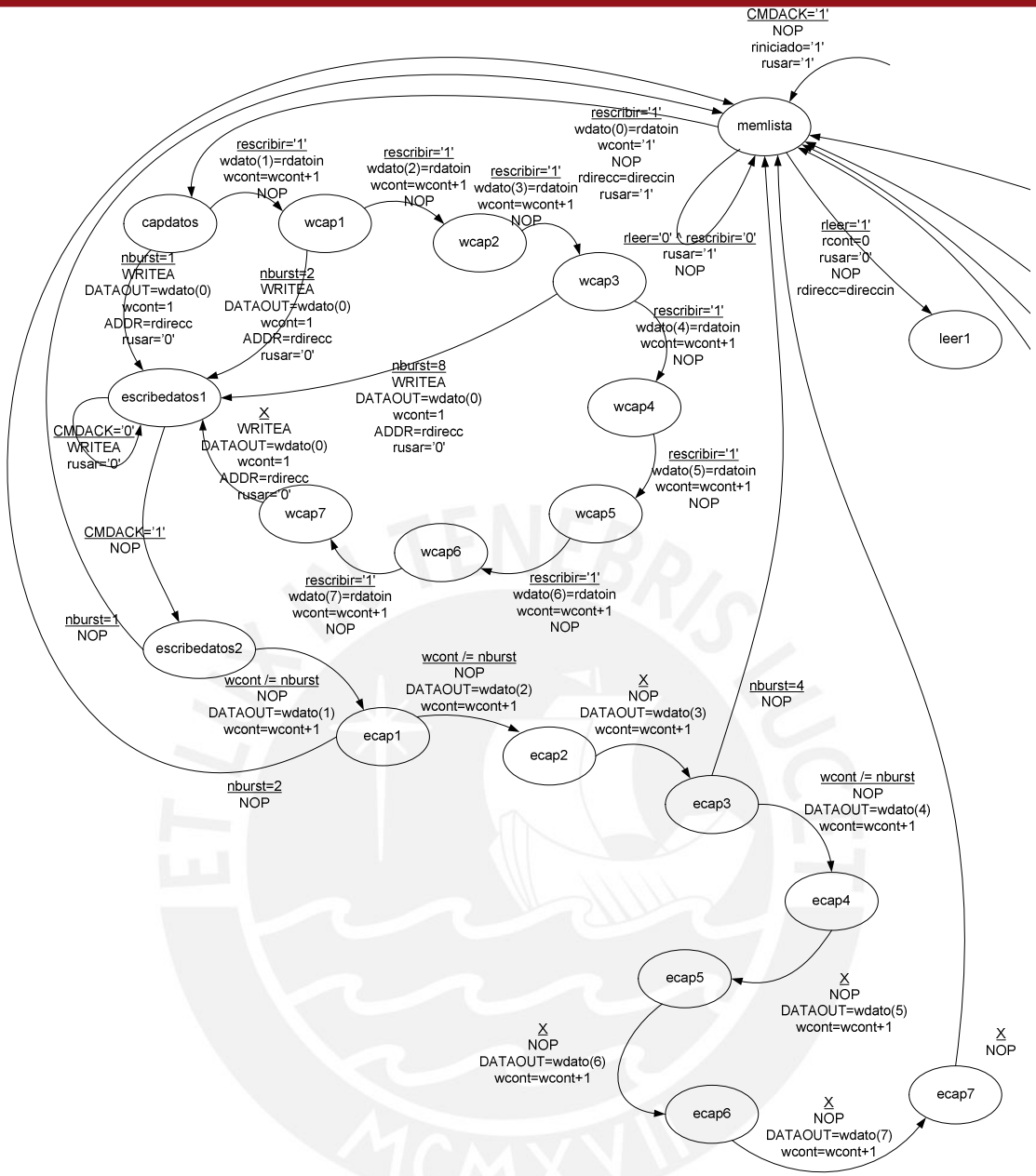


Figura 2.12. Máquina de estados tercera parte. Se muestra los estados de escritura: desde el estado *memlista* hasta *ecap7*.

2.5 Bloque arcotangente

2.5.1 Introducción

Para la realización de funciones trigonométricas e hiperbólicas en *hardware* y en programación de bajo nivel se utiliza el algoritmo CORDIC (*Coordinate Rotation Digital Computer*), el cual se basa en sucesivas restas, sumas, desplazamientos y multiplicaciones para la obtención de lo deseado.

Arquitecturas diseñadas para la obtención de estas funciones utilizando el CORDIC han sido temas de tesis de maestría en universidades extranjeras, por lo que el diseño de dicho algoritmo escapa a los objetivos de este trabajo. Es por ese motivo que se utilizó, como base, un *core* de libre distribución diseñado por *Richard Herveille* y que se encuentra en libre distribución en: www.opencores.org. Si bien dicho diseño funcionaba correctamente a frecuencias menores a 40MHz, para frecuencias mayores no se conseguía el *pipeline* requerido debido a que el diseño no aprovechaba propiedades de los FPGAs en su diseño. Es por ese motivo que el trabajo realizado aquí fue el de lograr que este *core* trabaje a una velocidad mayor a los 135MHz deseados, que tenga una señal que indique que el dato se encuentra listo, además que sea parametrizable según los requerimientos del tamaño del dato de entrada y del dato de salida.

Las siguientes líneas explican, de forma breve, algunos detalles a tomar en cuenta cuando se utiliza el algoritmo CORDIC y luego se presenta los parámetros, las señales de entrada y de salida del bloque, además de los resultados obtenidos de forma independiente.

2.5.2 Algoritmo CORDIC y bloque original

El algoritmo CORDIC tiene como base la rotación de coordenadas para obtener la función requerida. Una aplicación del *core* utilizado es la transformación de coordenadas rectangular a coordenadas polares, lo que implica realizar la función raíz cuadrada para la obtención de la amplitud, y la función arcotangente para la obtención de la fase involucrada.

Para la obtención de la fase, el algoritmo CORDIC se basa en una tabla que tiene almacenado el valor del arcotangente de potencias de "2" según el número de bits utilizados para la respuesta, dichos valores son referenciados al $Z(n)$ utilizado. Por ejemplo, el diseño realizado utilizado tiene un tamaño de palabra para la fase igual a $p = 31$ bits (el más significativo usado para el signo, los tres siguientes indican la parte entera y el resto la parte decimal). De esta forma, se tiene que la equivalencia de un radian está definida por la ecuación 18, y cada dato de la tabla con los arcotangentes está definido por la ecuación

19. El número de datos en la tabla depende del número de bits a utilizarse como resolución.

$$1\ radián = \frac{2^p}{2\times\pi} \tag{18}$$

$$dato(n) = \arctan\left(\frac{1}{2^n}\right)\times 1\ radián \tag{19}$$

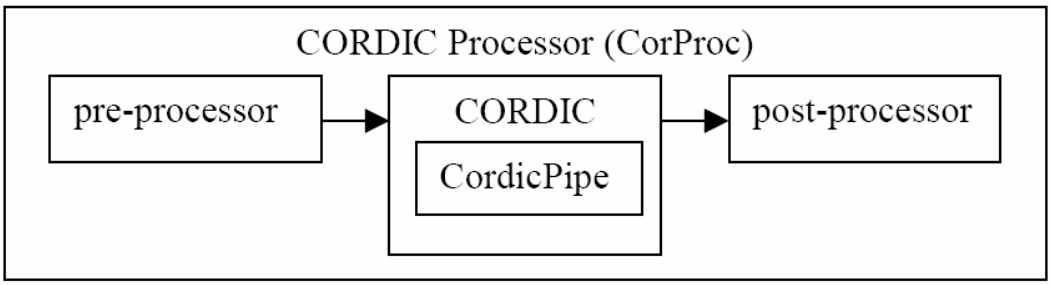


Figura 2.13. Bloque original de conversión de coordenadas rectangulares a coordenadas polares (obtenida de [27]).

El bloque original se basa en cuatro sub-bloques (ver figura 2.13): *pre-processor*, *cordic*, *cordicpipe* y *post-processor*. La función de *pre-processor* es entregar el valor absoluto de las coordenadas rectangulares de entrada *X* e *Y*, entregando como *Xi* el mayor valor absoluto y como *Yi* el menor. Además, indica mediante tres señales la información respecto a si el valor de cada coordenada original era positivo o negativo, además si se hizo un cambio de posicionamiento de *X* por *Y* al momento de entregar *Xi* e *Yi*. La función del sub-bloque *cordic* es generar la secuencia de desplazamientos, sumas y restas, según el número de bits de tamaño de palabra utilizado, para hallar la fase. Esta tarea la realiza generando tantos sub-bloques *cordicpipe* como número de bits se tenga y debido a que los valores entregados del primer sub-bloque son positivos, la fase hallada será referente al primer cuadrante. El resultado obtenido pasa al sub-bloque *post-processor* que se encarga de hallar la fase real utilizando la información proveniente del primer bloque respecto a si los valores de coordenadas originales eran positivos o negativos o si se realizó un cambio de *X* por *Y*. Esta tarea la realiza por medio de comparaciones y restas con los límites de π, $\frac{\pi}{2}$ o 2π. Además, este bloque realiza el factor de

corrección indicado en [27] para el valor de la magnitud de la coordenada polar, pero este procedimiento no es necesario para los fines de este trabajo.

2.5.3 Bloque modificado

El bloque modificado continúa usando los cuatro sub-bloques: *pre-processor* (ver R2P_PRE.VHD en sección Anexos), *cordic* (ver R2P_CORDIC.VHD en sección Anexos), *cordicpipe* (ver R2P_CORDICPIPE.VHD en sección Anexos) y *post-processor* (ver R2P_POST.VHD en sección Anexos). Estos bloques son integrados para formar el bloque completo (ver R2P_CORPROC.VHD en sección Anexos).

Los comparadores del sub-bloque *pre-processor* fueron reemplazados por comparadores basados en Megafunciones de Altera, los cuales nos permiten controlar el número de ciclos en que se obtiene el dato correcto, además, se utilizó la señal *ena* como señal para cargar los datos (ver tabla 2.9) y de esta forma generar una señal que indica que el proceso del bloque ha sido realizado. El sub-bloque *cordic,* que controla la generación de sub-bloques *cordicpipe*, fue alterado de modo que cargue los datos cuando el sub-bloque *pre-processor* informe que los el dato está listo, y que además, genere una señal que indique que el proceso ha sido realizado y con esto, el dato está listo. El sub-bloque *cordicpipe* fue alterado en cuanto a la tabla de valores arcotangente que posee, además que la señal entregada por el sub-bloque *pre-processor* referente a los signos de las coordenadas originales, y si se realizó cambio entre *X* e *Y,* sean desplazadas mediante registros de modo que el dato obtenido al final de este sub-bloque, que genera también una señal que indica que el dato está listo, sea entregado junto con la información original del primer sub-bloque. La tabla con valores de arcotangente se halla utilizando el programa TABLAARCTAN.M (ver Anexos), el cual relaciona el número de datos deseados con el número de bits de resolución y el número de bits representando a la parte decimal que se desea. Es de esta manera que en esta versión del bloque, el dato obtenido en el sub-bloque *cordic* llega conjuntamente con la información del primer sub-bloque, lo que da la opción de *pipeline* al sistema (ver figura 2.14). Las alteraciones realizadas en el último sub-bloque (bloque *post-processor*) son similares a las del primer bloque, es

decir, se utiliza Megafunciones que permiten controlar el número de ciclos en que se obtienen las comparaciones, sumas o restas.

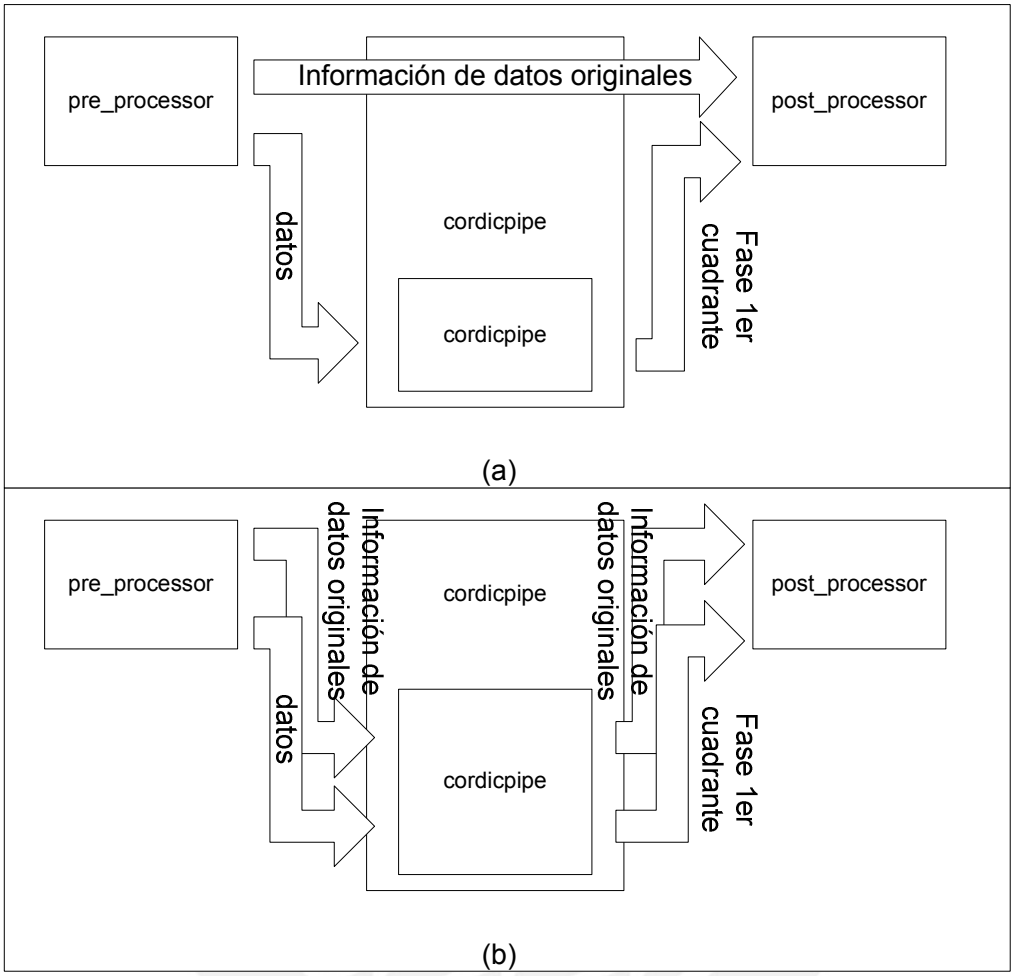


Figura 2.14. Flujo de información de datos originales. (a) Antes de la modificación. (b) Después de la modificación.

Al final de los cambios, se obtuvo un bloque que halla el arcotangente con una alta frecuencia de trabajo (mayor a 140MHz), capaz de trabajar un dato por ciclo. Además, cuenta con una señal *listoA* que indica que el dato se encuentra listo (ver tabla 2.9).

Parámetro	**Función**
dsize	Tamaño en bits de las coordenadas rectangulares.
tamano_data	Tamaño de la tabla de arcotangente del sub-bloque *cordicpipe*.
Pipesum	Número de ciclos en efectuar las sumas y restas involucradas.
tamanoZ	Tamaño en bits de la fase de salida.
PIPELINE	Número de sub-bloques *cordicpipe* que se generan.
EXT_PRECISION	Número de bits de precisión que se utilizó en la tabla del sub-bloque *cordicpipe.*

Señal de entrada	Función	Número de bits
clk	Señal de sincronismo del bloque.	1
reset	Señal de reinicio del bloque.	1
ena	Señal de habilitación. Sirve para cargar el dato al bloque.	1
Xin	Coordenada *X* del dato de entrada. En este trabajo: parte real del dato.	dsize
Yin	Coordenada *Y* del dato de entrada. En este trabajo: parte imaginaria del dato.	dsize
Señal de salida	**Función**	**Número de bits**
Aout	Bus de datos con la fase obtenida	tamanoZ
listoA	Señal que indica que el dato entregado es correcto.	1

Tabla 2.9. Parámetros, señales de entrada y salida del bloque Arcotangente.

2.5.4 Resultados independientes

El bloque fue compilado utilizando los siguientes valores de los parámetros: dsize = 24, tamano_data = 31, pipesum = 3, tamanoZ = 28, PIPELINE = 15 y EXT_PRECISION = 4. El bloque diseñado ocupa 8548 LEs en el FPGA STRATIX EP1S25F672C6 utilizado, alcanzando una frecuencia máxima de trabajo de 151.79MHz con una compilación optimizada en velocidad o en área a utilizar. El resultado de la convolución se obtiene 110 ciclos de reloj luego de haber cargado el dato, es decir 724.69ns luego de haber ingresado el dato (ver simulación en sección ANEXOS). La tabla 2.10 muestra los resultados obtenidos ante diferentes tipos de STRATIX y STRATIX II.

		ÁREA				VELOCIDAD			
	FPGA	LEs	%LEs	fmax (MHz)	Tiempo Teórico (us)	LEs	%LEs	fmax (MHz)	Tiempo Teórico (us)
Stratix	EP1S10F672-6	8548	80.87	151.79	0.72469	8548	80.87	151.79	0.72469
	EP1S10F672-7	8548	80.87	144.18	0.76294	8548	80.87	144.18	0.76294
	EP1S25F672-6	8548	33.31	151.79	0.72469	8548	33.31	151.79	0.72469
	EP1S25F672-7	8548	33.31	144.18	0.76294	8548	33.31	144.18	0.76294
	EP1S60F1508-6	8548	14.96	151.79	0.72469	8548	14.96	151.79	0.72469
	EP1S60F1508-7	8548	14.96	144.18	0.76294	8548	14.96	144.18	0.76294
Stratix II	EP2S15F484-3	6746	43.24	82.65	1.33091	6746	43.24	82.65	1.33091
	EP2S15F484-4	6746	43.24	80.03	1.37448	6746	43.24	80.03	1.37448
	EP2S15F484-5	6746	43.24	76.20	1.44357	6746	43.24	76.20	1.44357
	EP2S30F672-3	6746	19.91	82.65	1.33091	6746	19.91	82.65	1.33091
	EP2S30F672-4	6746	19.91	80.03	1.37448	6746	19.91	80.03	1.37448
	EP2S30F672-5	6746	19.91	76.20	1.44357	6746	19.91	76.20	1.44357

Tabla 2.10. Resultados obtenidos usando distintos FPGAs para el bloque arcotangente.

2.6 Unidad de control

2.6.1 Comportamiento funcional

Si bien el bloque convolución (ver sección 2.3) es el más importante en cuanto al procesamiento de la imagen, es el bloque Unidad de Control el encargado de controlar los demás bloques para seleccionar qué bloque trabaja en qué momento, además de controlar los multiplexores y demultiplexores involucrados en la arquitectura desarrollada (ver sección 2.1). De esta forma, se seleccionan los datos a cargar en el bloque convolucionador así como definir su destino en el sistema.

Todos los multiplexores y el demultiplexor son controlados por este bloque, además, los bloques que se controlan son:

i. Bloque cargar datos y coeficientes del usuario
ii. Bloque leer imagen original
iii. Bloque escribir imagen analítica
iv. Bloque leer imagen analítica
v. Bloque escribir imagen filtrada en filas
vi. Bloque leer imagen filtrada en filas
vii. Bloque escribir fase obtenida
viii. Bloque leer fase obtenida y contador

De esta manera, el bloque es el encargado de lograr el procedimiento indicado en las secciones 1.4.2 y 2.1, el cual es:

i. Recepcionar los coeficientes de usuario, los datos de la imagen original (tamaño, número de filas y columnas), la imagen original y escribirla en la SDRAM.
ii. Cargar los coeficientes necesarios para la transformada de Hilbert.
iii. Leer la imagen original y escribir la imagen analítica, al mismo tiempo que se inicia el contador que lleva la cuenta del tiempo del proceso total.
iv. Cargar los coeficientes necesarios para el filtrado de la imagen en el canal deseado (coeficientes definidos por el usuario).

v. Leer la imagen analítica y escribir la imagen filtrada en filas.
vi. Leer la imagen filtrada en filas, filtrarla en columnas, obtener fase y escribirla en la memoria SDRAM.
vii. Detener el contador del proceso total.
viii. Enviar el tiempo empleado y la fase obtenida a la PC.

2.6.2 Desarrollo

El bloque Unidad de Control recibe las señales provenientes de los bloques de lectura y escritura de datos (ver secciones 2.8 a 2.15) y del bloque Escribe Coeficientes (ver sección 2.7, y tabla 2.11). Estas son almacenadas en registros internos (ver figura 2.15) y luego procesadas en una máquina de estados, de manera que genera las señales de habilitación de bloques y selectores de multiplexores y demultiplexores necesarias para el sistema (ver UNIDADDECONTROL.VHD en sección Anexos).

Señal de entrada	Función	Número de bits
reloj	Señal de sincronismo del bloque.	1
reset	Señal de reinicio del bloque.	1
acabeescribiroriginal	Señal que indica que el bloque cargar datos y coeficientes del usuario terminó de escribir los datos originales y cargar los coeficientes de Hilbert en el bloque convolución (ver sección 2.8)	1
envie8original	Señal que indica que el bloque leer imagen original leyó ocho datos de la imagen original guardada en la memoria SDRAM (ver sección 2.9).	1
escribi8hilbert	Señal que indica que el bloque escribir imagen analítica escribió ocho datos de la imagen analítica en la memoria SDRAM (ver sección 2.10).	1
acabeescribirhilbert	Señal que indica que el bloque escribir imagen analítica terminó de escribir toda la imagen analítica en la memoria SDRAM (ver sección 2.10).	1
listolpf	Señal que indica que los coeficientes del filtro de la banda deseada han sido cargados en el bloque convolución (ver sección 2.7).	1
envie8hilbert	Señal que indica que el bloque leer imagen analítica leyó ocho datos de la imagen analítica guardada en la memoria SDRAM (ver sección 2.11).	1
escribi8filas	Señal que indica que el bloque escribir imagen filtrada en filas escribió ocho datos de la imagen filtrada en filas en la memoria SDRAM (ver sección 2.12).	1
acabeescribirfilas	Señal que indica que el bloque escribir imagen filtrada en filas terminó de escribir toda la imagen filtrada en filas en la memoria SDRAM (ver sección 2.12).	1
envie1fila	Señal que indica que el bloque leer imagen filtrada en filas leyó un dato de la imagen filtrada en filas guardada en la memoria SDRAM (ver sección 2.13).	1

escribi1fase	Señal que indica que el bloque escribir fase obtenida escribió un dato de la fase obtenida en la memoria SDRAM (ver sección 2.14).	1
acabeescribirfase	Señal que indica que el bloque escribir fase obtenida terminó de escribir toda la fase obtenida en la memoria SDRAM (ver sección 2.14).	1
acabeleerfase	Señal que indica que el bloque leer fase obtenida y contador terminó de enviar la fase obtenida y el contador a la PC (ver sección 2.15).	1
Señal de salida	**Función**	**Número de bits**
habescribeoriginal	Señal que habilita el bloque cargar datos y coeficientes (ver sección 2.8).	1
hableeoriginal	Señal que habilita el bloque leer imagen original (ver sección 2.9).	1
habescribehilbert	Señal que habilita el bloque escribir imagen analítica (ver sección 2.10).	1
cargarlpf	Señal que indica que los coeficientes del filtro de la banda deseada deben ser cargados en el bloque convolución (ver sección 2.7).	1
hableehilbert	Señal que habilita el leer imagen analítica (ver sección 2.11).	1
habescribefilas	Señal que habilita el escribir imagen filtrada en filas (ver sección 2.12).	1
hableefilas	Señal que habilita el bloque leer imagen filtrada en filas (ver sección 2.13).	1
habescribefase	Señal que habilita el bloque escribir fase obtenida (ver sección 2.14).	1
hableefase	Señal que habilita el bloque leer fase obtenida y contador (ver sección 2.15).	1
MUXADDR	Selector del multiplexor de direcciones de la memoria SDRAM (ver sección 2.16).	3
MUXdatos	Selector del multiplexor de datos de entrada de la memoria SDRAM (ver sección 2.16).	2
MUXleer	Selector del multiplexor de la señal de lectura de la memoria SDRAM (ver sección 2.16).	2
MUXescribir	Selector del multiplexor de la señal de escritura de la memoria SDRAM (ver sección 2.16).	2
MUXRe	Selector del multiplexor de la parte real del filtraje obtenido (ver sección 2.16).	1
MUXFIR	Selector del multiplexor de los datos de entrada al bloque convolución (ver sección 2.16).	2
DEMUXFIR	Selector del demultiplexor de los datos de salida del bloque convolución (ver sección 2.16).	2

Tabla 2.11. Parámetros, señales de entrada y salida del bloque Unidad de Control.

La máquina de estados desarrollada cuenta con diez estados (ver figura 2.16): *inicio, leeoriginal, escribehilbert, cargandolpf, leehilbert, escribefilas, leefilas, escribefase, leefase* y *fin*. El bloque funciona de manera que habilita de manera continua los bloques: *cargar datos y coeficientes* y, *leer fase obtenida y contador*, mientras que a los demás bloques de lectura y escritura los habilita

de manera que actúen juntos. Es decir, cuando se encuentra en el procedimiento de leer la imagen original y escribir la imagen analítica, habilita por un ciclo *burst* la lectura (en este trabajo, una lectura *burst* implica leer ocho datos), y una vez que los datos han pasado por el bloque convolución, habilita la recepción y escritura de estos, de esta manera, se leen ocho datos y se escriben ocho datos. En el siguiente procedimiento, leer la imagen analítica y escribir la imagen filtrada en filas, la combinación de habilitación de bloques es similar a la descrita anteriormente. En el penúltimo procedimiento, leer la imagen filtrada en filas, filtrarla en columnas, obtener fase y escribirla, la combinación es similar, con la diferencia que si bien se lee y escribe ocho datos, sólo se utiliza un dato (ver secciones 2.13 y 2.14). En todos estos procedimientos, el uso de los multiplexores cumple un papel fundamental pues son los que seleccionarán las señales correctas al controlador de la memoria SDRAM (ver sección 2.4).

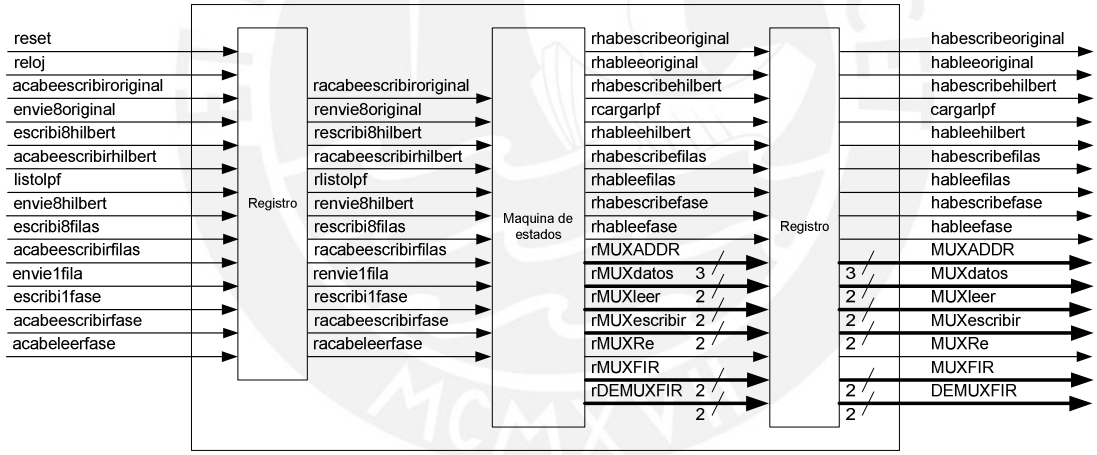


Figura 2.15. Bloque Unidad de Control.

2.6.3 Resultados independientes

El bloque diseñado ocupa solamente 97 LEs en el FPGA STRATIX EP1S25F672C6 utilizado, alcanzando una frecuencia máxima de trabajo de 203.79MHz con una compilación optimizada en velocidad, mientras que 94 LEs y 141.28MHz con una optimización para el área a utilizar. La tabla 2.12 muestra los resultados obtenidos ante diferentes tipos de STRATIX, STRATIX II y de ACEX1K. La simulación del bloque se puede observar en la sección Anexos.

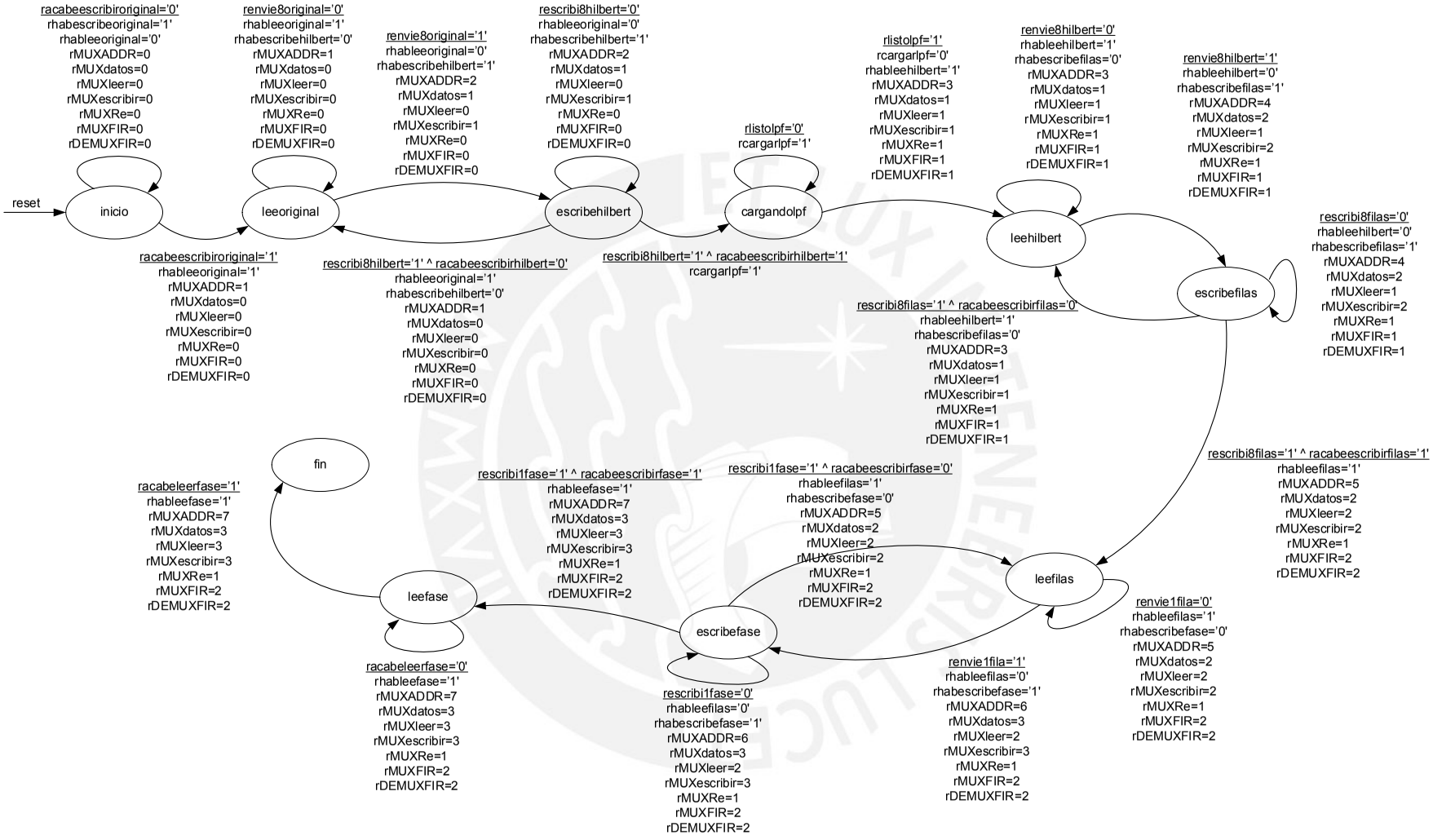


Figura 2.16. Máquina de estados del bloque Unidad de Control.

		ÁREA			VELOCIDAD		
	FPGA	LEs	%LEs	fmax (MHz)	LEs	%LEs	fmax (MHz)
Stratix	EP1S10F672-6	94	0.89	141.28	97	0.92	203.79
	EP1S10F672-7	94	0.89	136.28	97	0.92	193.91
	EP1S25F672-6	94	0.37	141.28	97	0.38	203.79
	EP1S25F672-7	94	0.37	136.28	97	0.38	193.91
	EP1S60F1508-6	94	0.16	141.28	97	0.17	203.79
	EP1S60F1508-7	94	0.16	136.28	97	0.17	193.91
Stratix II	EP2S15F484-3	58	0.37	415.45	58	0.37	422.12
	EP2S15F484-4	58	0.37	380.95	58	0.37	422.12
	EP2S15F484-5	58	0.37	337.27	58	0.37	390.02
	EP2S30F672-3	58	0.17	415.45	58	0.17	422.12
	EP2S30F672-4	58	0.17	380.95	58	0.17	422.12
	EP2S30F672-5	58	0.17	337.27	58	0.17	390.02
Acex	EP1K10QC208-1	92	15.97	163.13	99	17.19	212.77
	EP1K10QC208-2	92	15.97	144.30	99	17.19	181.82
	EP1K10QC208-3	92	15.97	107.18	99	17.19	140.85
	EP1K50FC484-1	92	3.19	163.13	99	3.44	212.77
	EP1K50FC484-2	92	3.19	144.30	99	3.44	181.82
	EP1K50FC484-3	92	3.19	107.18	99	3.44	140.85

Tabla 2.12. Resultados obtenidos usando distintos FPGAs para el bloque Unidad de Control.

2.7 Bloque escribe coeficientes

2.7.1 Comportamiento funcional

Este bloque se encarga de almacenar los coeficientes que serán cargados al bloque convolución. Debido a que en el proceso se utilizan dos tipos de filtros (el primero para hallar la imagen analítica y el segundo respecto a la banda a utilizar), el bloque almacena de forma continua usando una memoria ROM interna al FPGA los coeficientes referidos a la imagen analítica, y los coeficientes referidos a la banda deseada, que son ingresados por usuario usando una señal de entrada, en memorias SRAM de doble bus de datos (uno para la escritura y otro para la escritura). El envío de los coeficientes deseados se realiza activando una u otra señal de carga de coeficientes (ver tabla 2.13), y luego de cargar los coeficientes, el bloque activa una señal que indica que los coeficientes deseados han sido cargados correctamente (ver figura 2.17).

2.7.2 Desarrollo

El bloque desarrollado (ver ESCRIBECOEF.VHD en sección Anexos) se basa en el uso de contadores de módulo "*taps*" (ver tabla 2.13), dichos contadores son utilizados para controlar las direcciones de lectura en las memorias SRAM y ROM, además de la escritura en las memorias SRAM. Las señales de entrada son almacenadas en una primera etapa mediante el uso de registros para luego ser enviadas a las memorias. Las señales *chilbert* y *clpf* sirven como señales de control para el modo lectura en las memorias internas utilizadas, y luego, al pasar por registros internos, sirven como seleccionador de los multiplexores de salida para enviar los coeficientes correctos y deseados. Al mismo tiempo que cada coeficiente es entregado al bloque convolución, se activa la señal *escribeFIR* para activar la carga de los coeficientes en dicho bloque (ver sección 2.3).

Parámetro	**Función**	
taps	Indica el número de coeficientes utilizados.	
coef	Indica el tamaño en bits de los coeficientes.	
Señal de entrada	**Función**	**Número de bits**
reloj	Señal de sincronismo del bloque.	1
reset	Señal de reinicio del bloque.	1
chilbert	Señal que indica que se debe cargar los coeficientes para hallar la imagen analítica.	1
clpf	Señal que indica que se debe cargar los coeficientes referidos al canal deseado.	1
escribecoef	Señal que indica que se desea almacenar los coeficientes ingresados por el usuario.	1
RXcoefRe	Parte real de los coeficientes del canal deseado ingresados por el usuario.	coef
RXcoefIm	Parte imaginaria de los coeficientes del canal deseado ingresados por el usuario.	coef
Señal de salida	**Función**	**Número de bits**
coeficientesRe	Parte real de los coeficientes a cargar.	coef
coeficientesIm	Parte imaginaria de los coeficientes a cargar.	coef
listohilbert	Señal que indica que los coeficientes referidos la obtención de la imagen analítica han sido cargados.	1
listolpf	Señal que indica que los coeficientes referidos al canal deseado han sido cargados.	1
escribeFIR	Señal que indica le indica al bloque convolución que debe almacenar los coeficientes.	1

Tabla 2.13. Parámetros, señales de entrada y salida del bloque escribe coeficientes.

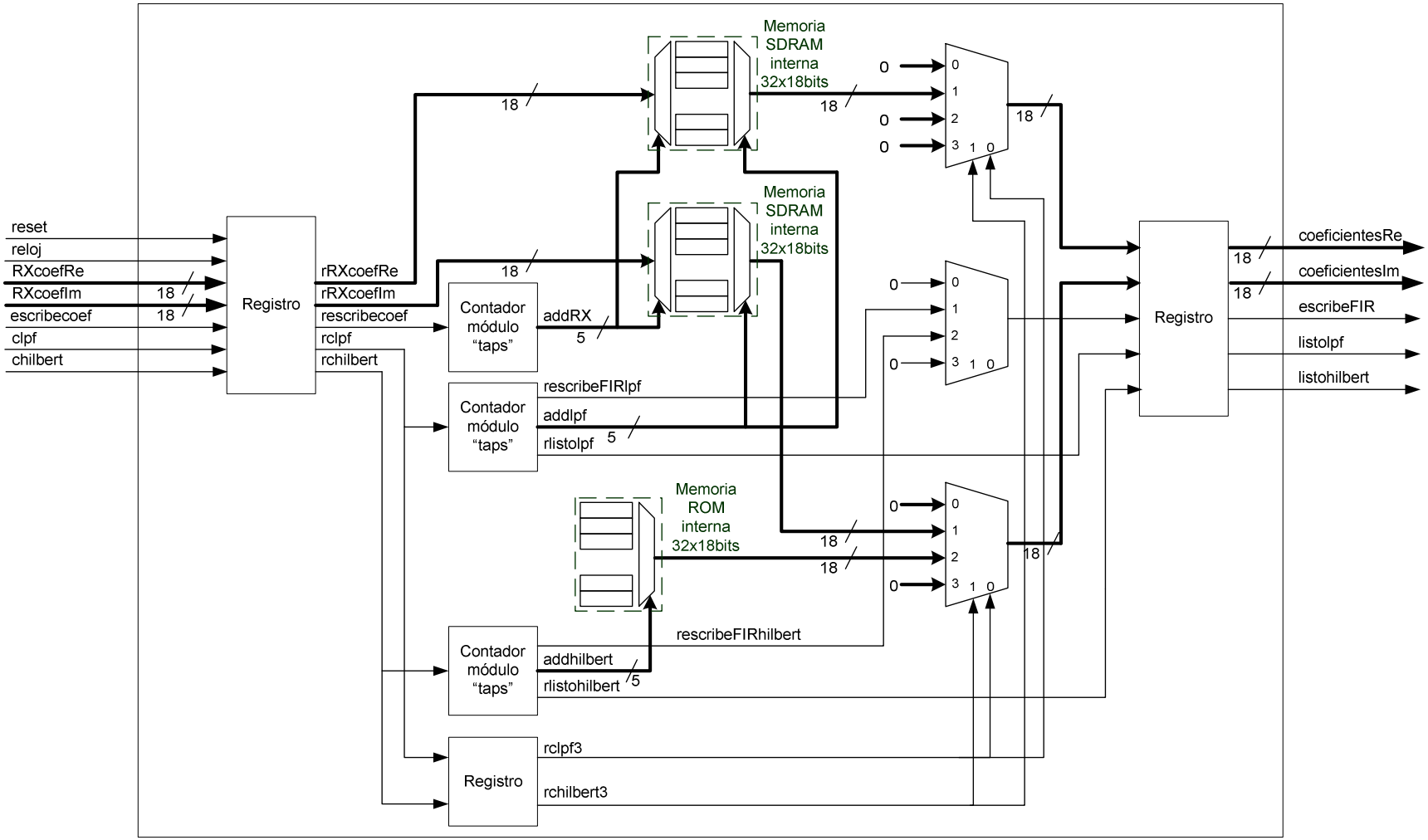


Figura 2.17. Bloque escribe coeficientes.

2.7.3 Resultados independientes

El bloque diseñado ocupa solamente 109 LEs en el FPGA STRATIX EP1S25F672C6 utilizado (se utiliza el 0.09% de los bits totales), alcanzando una frecuencia máxima de trabajo de 171.64MHz con una compilación optimizada en velocidad, mientras que 107 LEs y 175.47MHz con una optimización para el área a utilizar. La tabla 2.14 muestra los resultados obtenidos ante diferentes tipos de STRATIX, STRATIX II y de ACEX1K.

		ÁREA				VELOCIDAD			
	FPGA	LEs	%LEs	fmax (MHz)	%bits	LEs	%LEs	fmax (MHz)	%bits
Stratix	EP1S10F672-6	107	1.01	175.47	0.19	109	1.03	171.64	0.19
	EP1S10F672-7	107	1.01	167.79	0.19	109	1.03	163.77	0.19
	EP1S25F672-6	107	0.42	175.47	0.09	109	0.42	171.64	0.09
	EP1S25F672-7	107	0.42	167.79	0.09	109	0.42	163.77	0.09
	EP1S60F1508-6	107	0.19	175.47	0.03	109	0.19	171.64	0.03
	EP1S60F1508-7	107	0.19	167.79	0.03	109	0.19	163.77	0.03
Stratix II	EP2S15F484-3	101	0.65	288.93	0.41	101	0.65	311.92	0.41
	EP2S15F484-4	101	0.65	269.11	0.41	101	0.65	296.12	0.41
	EP2S15F484-5	101	0.65	243.19	0.41	101	0.65	266.88	0.41
	EP2S30F672-3	101	0.30	288.93	0.13	101	0.30	311.92	0.13
	EP2S30F672-4	101	0.30	269.11	0.13	101	0.30	296.12	0.13
	EP2S30F672-5	101	0.30	243.19	0.13	101	0.30	266.88	0.13
Acex	EP1K10QC208-1	107	18.58	191.94	14.06	107	18.58	191.94	14.06
	EP1K10QC208-2	107	18.58	172.12	14.06	107	18.58	172.12	14.06
	EP1K10QC208-3	107	18.58	138.70	14.06	107	18.58	138.70	14.06
	EP1K50FC484-1	107	3.72	195.69	4.22	107	3.72	195.69	4.22
	EP1K50FC484-2	107	3.72	166.39	4.22	107	3.72	166.39	4.22
	EP1K50FC484-3	107	3.72	136.80	4.22	107	3.72	136.80	4.22

Tabla 2.14. Resultados obtenidos usando distintos FPGAs para el bloque Escribe Coeficientes.

2.8 Bloque cargar datos y coeficientes del usuario

2.8.1 Comportamiento funcional

La función de este bloque es recepcionar los datos provenientes del bloque receptor serial (ver sección 2.2) y obtener con ellos: los coeficientes que el usuario desea usar para analizar el canal deseado, el tamaño en bytes de la imagen original, el número de columnas que tiene, su número de filas y la imagen original. Mientras recibe la imagen original, la escribe en la memoria

externa SDRAM para que una vez terminado este proceso, cargue los coeficientes necesarios en el bloque convolución (ver sección 2.3) para hallar la imagen analítica.

2.8.2 Desarrollo

El bloque desarrollado (ver CARGARDATOS.VHD en sección Anexos) se basa en una máquina de estados que es controlada por el tamaño de la imagen que se está enviando, y por el parámetro *taps* (ver tabla 2.15). El tamaño de los coeficientes, definido por el parámetro *csize*, debe ser entre 17 y 23 bits, esta condición se basa en que la precisión utilizada para el número de bits de los coeficientes fue de 18. Además, debido a que el tamaño máximo de la imagen a trabajar es de 512 filas por 512 columnas, el tamaño de la imagen, al igual que el número de filas y columnas, se trabaja con 20 bits. La secuencia de envío de datos que realiza el usuario desde la PC debe ser:

i. Envío de la parte real del primer coeficiente: Se envía en tres partes, desde del byte menos significativo al *byte* más significativo.
ii. Envío de la parte imaginaria del primer coeficiente: Se envía en tres partes, desde del byte menos significativo al *byte* más significativo.
iii. Los procesos i y ii son repetidos hasta que se envie el total de coeficientes configurados.
iv. Envío del tamaño de la imagen: Se envía en tres partes, desde del *byte* menos significativo al byte más significativo.
v. Envío del número de columnas de la imagen: Se envía en tres partes, desde del *byte* menos significativo al *byte* más significativo.
vi. Envío del número de filas de la imagen: Se envía en tres partes, desde del byte menos significativo al byte más significativo.
vii. Envío de los datos de la imagen píxel por píxel, que es igual a byte por byte. La forma de envío debe ser fila por fila, desde la primera hasta la última columna (ver sección 1.4.2).

Luego del envío de los datos, el bloque ordena la carga de los coeficientes para la obtención de la imagen analítica e informa que ya terminó su trabajo una vez que los coeficientes han sido cargados.

El bloque usa registros de los datos de entrada y de los datos de salida (ver figura 2.18) para con ellos identificar los datos recibidos usando una máquina

de estados (ver figura 2.19) que consta de veintidós estados: *inicio, iniciadaSDRAM, esperaSDRAM, coefRebyte0, coefRebyte1, coefRebyte2, coefImbyte0, coefImbyte1, coefImbyte2, cargatamano0, cargatamano1, cargatamano2, cargacol0, cargacol1, cargacol2, cargafil0, cargafil1, cargafil2, escribiendoSDRAM, cargahilbertcoef, esperandofin* y *finoriginal*. Esta máquina de estados interactúa con contadores para la cuenta de coeficientes y del número de datos de la imagen recepcionados, al mismo tiempo, controla la generación de direcciones de la memoria externa SDRAM, utilizando el rango de direcciones desde 000000H hasta 07FFFFH.

Al tener la memoria externa SDRAM utilizada un tamaño de palabra de 32 bits, utiliza los 2 bytes más significativos para la parte real de la imagen, y los 2 bytes menos significativos para la parte imaginaria, que en este caso es cero. Siendo la imagen proveniente de la PC una imagen de 8 bits de datos sin signo, la parte real de la imagen será escrita colocando en el bit más significativo el valor de '0', continuando con la información recibida para colocar en el resto de los bits, los menos significativos, el valor de '0' también.

Parámetro	**Función**	
tsize	Indica el tamaño en bits del tamaño, número de columnas y número de filas.	
dinsize	Indica el tamaño en bits del dato de entrada.	
doutsize	Indica el tamaño en bits del dato de salida.	
csize	Indica el tamaño en bits de los coeficientes.	
asize	Indica el tamaño en bits de la dirección.	
taps	Indica el número de coeficientes utilizados.	
burst	Indica el número de ciclos *burst* utilizados.	
Señal de entrada	**Función**	**Número de bits**
reloj	Señal de sincronismo del bloque.	1
reset	Señal de reinicio del bloque.	1
hab	Señal que habilita el bloque.	1
RX	Señal que indica que el dato de entrada *datoin* debe ser cargado.	1
datoin	Dato recepcionado por el bloque Receptor Serial.	dinsize
listohilbert	Señal que indica que los coeficientes para la obtención de la imagen analítica han sido cargados.	1
Señal de salida	**Función**	**Número de bits**
acabeescribiroriginal	Señal que indica que el proceso de este bloque ha concluido.	1
tamano	Tamaño de la imagen recepcionado (dato enviado por el usuario).	tsize
filas	Número de filas de la imagen original (dato enviado por el usuario).	tsize
columnas	Número de columnas de la imagen original (dato enviado por el usuario)	tsize

escribeoriginal	Señal de escritura hacia la memoria externa SDRAM,	1
datooriginal	Dato de la imagen original para escribir en la memoria externa SDRAM.	doutsize
direscribeoriginal	Dirección para escribir en la memoria externa SDRAM.	asize
coeficienteRe	Parte real de los coeficientes para cargar en el bloque escribe coeficientes (ver sección 2.7).	csize
coeficienteIm	Parte imaginaria de los coeficientes para cargar en el bloque escribe coeficientes (ver sección 2.7).	csize
Escribecoef	Señal que indica que se debe cargar el coeficiente en el bloque escribe coeficientes (ver sección 2.7).	1
Cargarhilbert	Señal que se deben cargar los coeficientes para la obtención de la imagen analítica.	1
iniciaSDRAM	Señal que indica que la memoria externa SDRAM debe inicializarse.	1
burstSDRAM	Bus de datos que envía el número de ciclos *burst* con que se configurará la memoria externa SDRAM.	4

Tabla 2.15. Parámetros, señales de entrada y salida del bloque cargar datos y coeficientes del usuario.

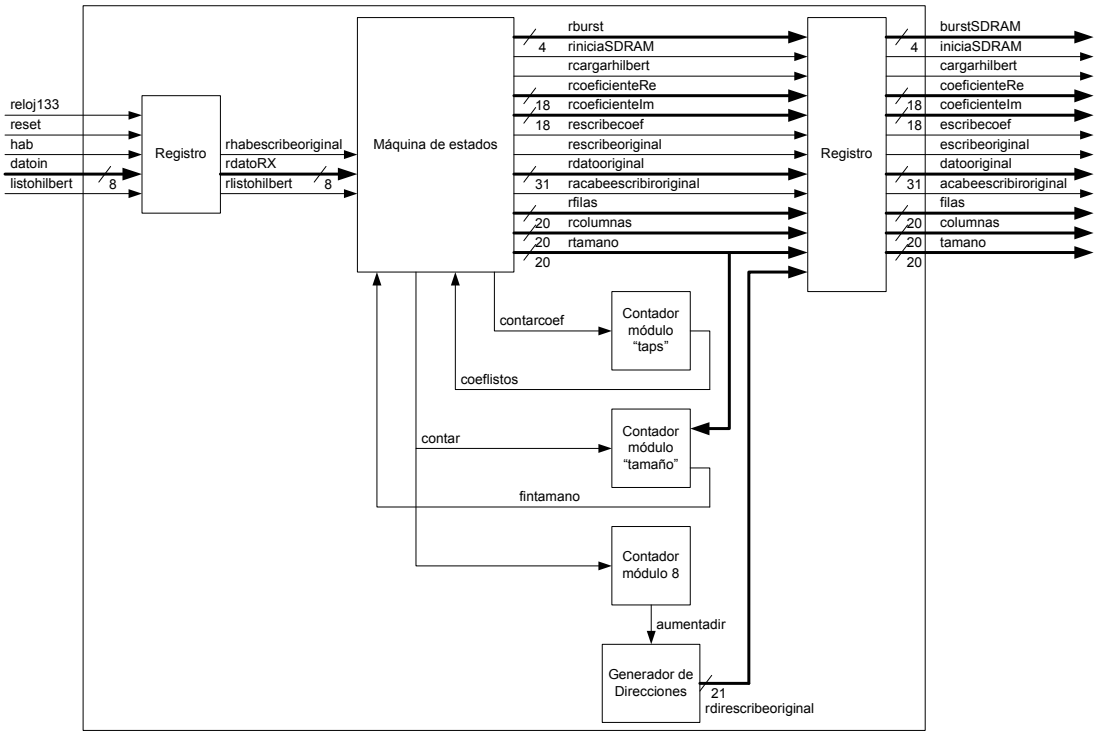


Figura 2.18. Bloque cargar datos y coeficientes del usuario.

2.8.3 Resultados independientes

El bloque fue compilado utilizando los siguientes valores de los parámetros: tsize = 20, dinsize = 8, doutsize = 32, csize = 18, taps = 7, burst = 8. El bloque diseñado ocupa solamente 432 LEs en el FPGA STRATIX EP1S25F672C6 utilizado, alcanzando una frecuencia máxima de trabajo de 151.54MHz con una compilación optimizada en velocidad, mientras que 418 LEs y 116.04MHz con

una optimización para el área a utilizar. La tabla 2.16 muestra los resultados obtenidos ante diferentes tipos de STRATIX, STRATIX II y de ACEX1K.

		ÁREA			VELOCIDAD		
	FPGA	LEs	%LEs	fmax (MHz)	LEs	%LEs	fmax (MHz)
Stratix	EP1S10F672-6	418	3.95	116.04	432	4.09	151.54
	EP1S10F672-7	418	3.95	110.32	432	4.09	145.20
	EP1S25F672-6	418	1.63	116.04	432	1.68	151.54
	EP1S25F672-7	418	1.63	110.32	432	1.68	145.20
	EP1S60F1508-6	418	0.73	116.04	432	0.76	151.54
	EP1S60F1508-7	418	0.73	110.32	432	0.76	145.20
Stratix II	EP2S15F484-3	333	2.13	204.67	333	2.13	204.67
	EP2S15F484-4	333	2.13	194.86	333	2.13	194.86
	EP2S15F484-5	333	2.13	181.19	333	2.13	181.19
	EP2S30F672-3	333	0.98	204.67	333	0.98	204.67
	EP2S30F672-4	333	0.98	194.86	333	0.98	194.86
	EP2S30F672-5	333	0.98	181.19	333	0.98	181.19
Acex	EP1K10QC208-1	446	77.43	108.58	461	80.03	113.64
	EP1K10QC208-2	446	77.43	88.42	461	80.03	99.01
	EP1K10QC208-3	446	15.49	108.58	461	16.01	113.64

Tabla 2.16. Resultados obtenidos usando distintos FPGAs para el bloque cargar datos y coeficientes del usuario.

2.9 Bloque leer imagen original

2.9.1 Comportamiento funcional

La función de este bloque es leer los datos de la imagen original almacenados en la memoria externa SDRAM para enviarlos al bloque convolución (ver sección 2.4) para obtener la imagen analítica. Debido a que el proceso de leer la imagen original trabaja conjuntamente con la escritura de la imagen analítica en la memoria externa SDRAM, el bloque lee los datos de ocho en ocho (este valor es definido porque se utilizó un número de ciclos burst igual a ocho), envía los datos y no vuelve a leer datos hasta que el resultado haya sido procesado por el bloque escribir imagen filtrada en filas (ver sección 2.10). Debido a que el número de coeficientes utilizado es menor a dieciséis, el bloque al terminar de leer una fila, envía dieciséis ceros al bloque convolución (ver sección 2.4) para obtener los resultados correctos (ver figura 1.9 y sección 1.4.2).

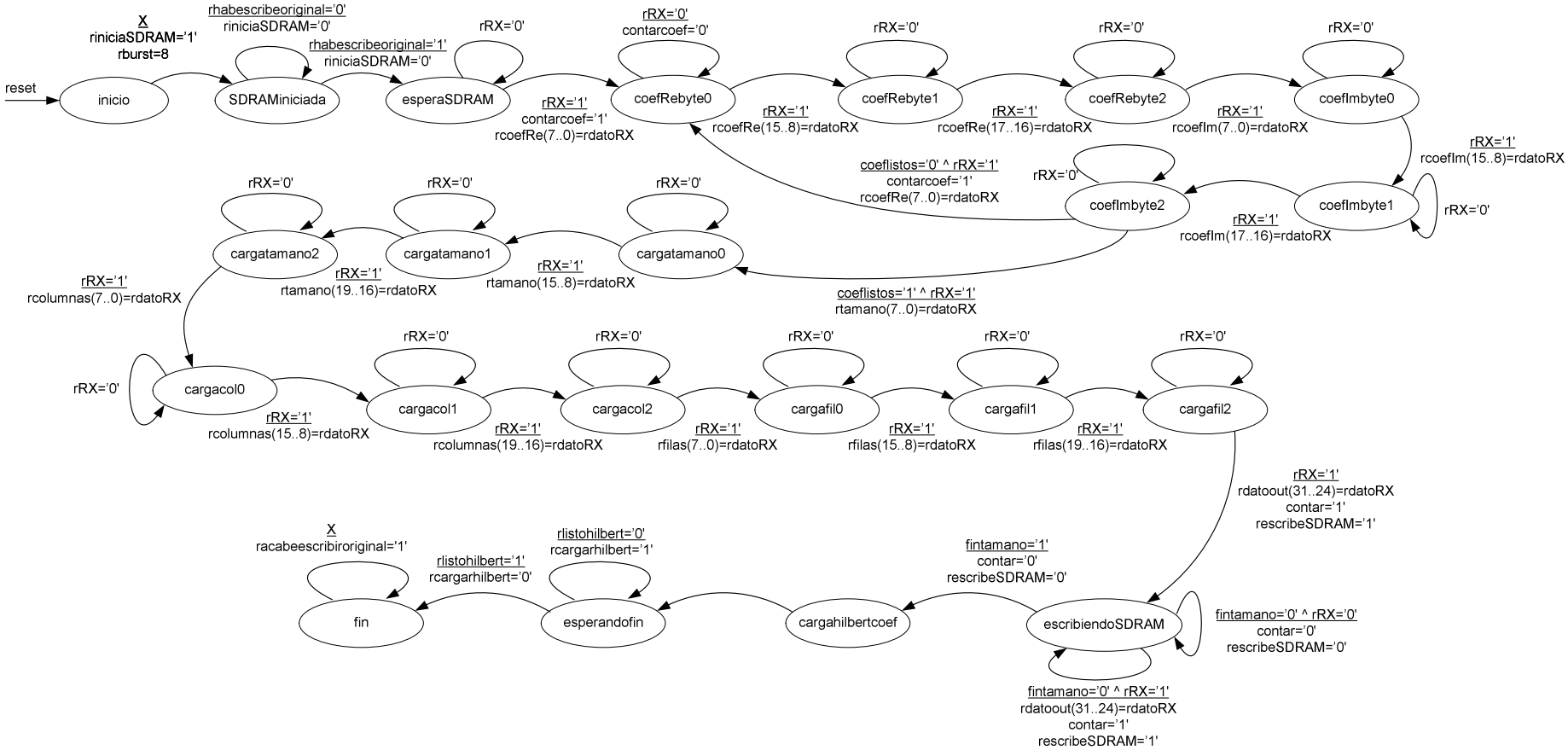


Figura 2.19. Máquina de estados del bloque cargar datos y coeficientes del usuario.

2.9.2 Desarrollo

El bloque desarrollado (ver LEEORIGINAL.VHD en sección Anexos) se basa en una máquina de estados que es controlada por el habilitador generado por la Unidad de Control (ver sección 2.6) y por las señales enviadas de los contadores internos implementados. Estos contadores indican el número de datos que se leyó, si se terminó de enviar los datos de la fila, además del número de datos enviados (ver figura 2.20). El bloque lee de la memoria externa SDRAM los ocho datos de la imagen original y los almacena internamente. Para evitar el uso de registros, utiliza memoria interna del FPGA, y una vez almacenados los envía al boque convolución. Este trabajo de almacenar temporalmente los datos permite realizar el envío en forma continua, un dato por ciclo, o en forma más lenta según la frecuencia de procesamiento que se tenga.

La escritura en la memoria interna es controlada por la correcta recepción de datos de la memoria externa SDRAM, y debido a que la parte imaginaria de la imagen original es cero, sólo se almacena la parte real. De esta forma, los 16 bits más significativos del dato leído, bits del 31 al 16, son enviados como la parte imaginaria de la imagen, para hallar la imagen analítica. Los 2 bits menos significativos del dato enviado al bloque convolución (ver sección 2.3) son puestos a '0'. Debido a que la parte real de la imagen analítica tiene el mismo valor de la imagen original, la información enviada como parte imaginaria es también enviada a los bloques retardadores (ver sección 2.18), pero dicho control de envío lo realiza la Unidad de Control (ver sección 2.6).

La máquina de estados que controla este bloque (ver figura 2.21) consta de ocho estados: *inicio, enviando1, enviando2, enviando3, resetdatos, waitceros, resetdatos2,* y *finoriginal.* De esta forma controla el funcionamiento de los contadores, incluyendo la generación de las direcciones de lectura, que se encuentran en el rango de 000000H a 07FFFFH. Una vez terminado el proceso, el bloque le indica a la Unidad de Control (ver sección 2.6) que ha terminado (ver tabla 2.17).

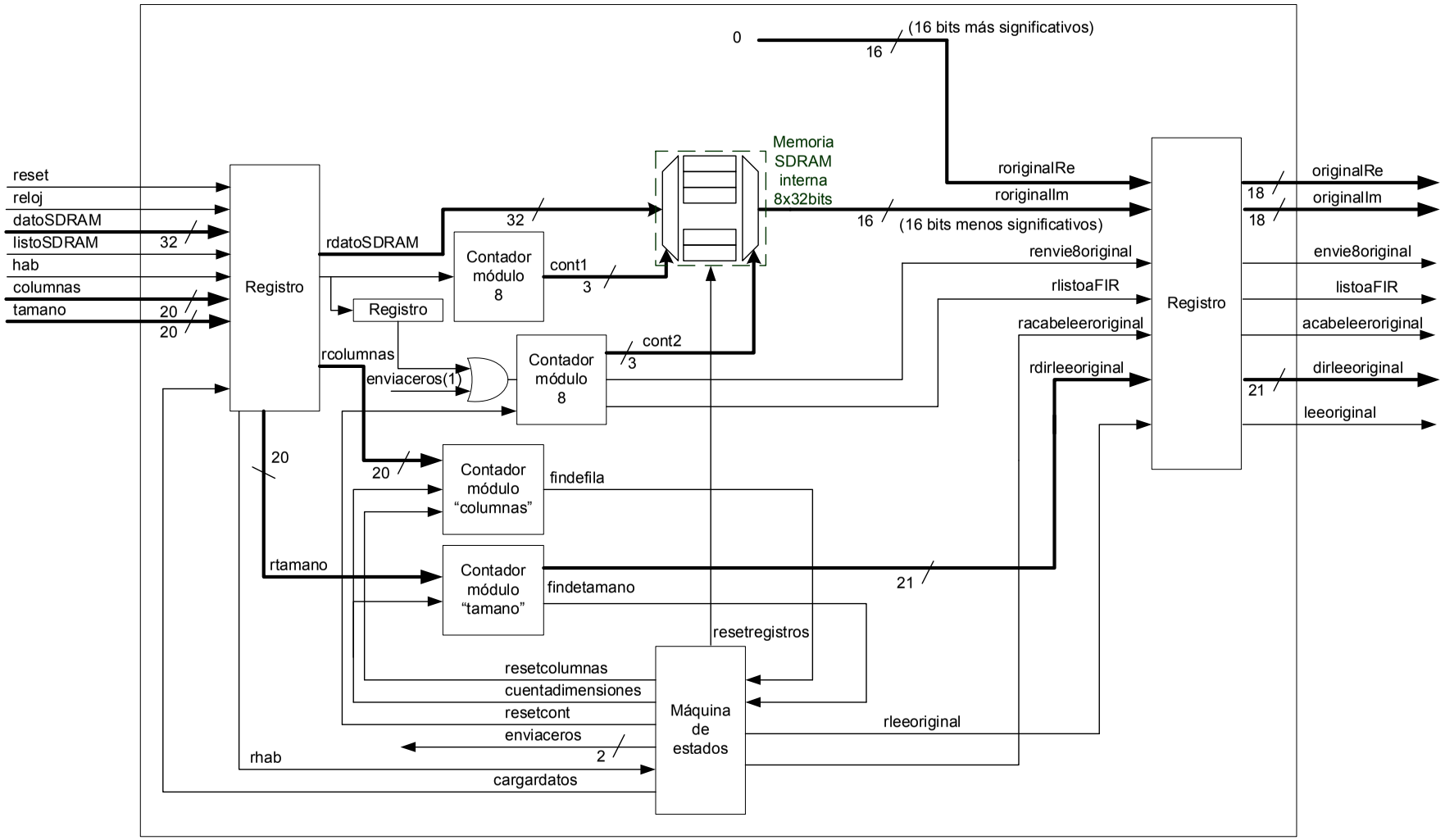


Figura 2.20. Bloque leer imagen original.

Parámetro	**Función**	
tsize	Indica el tamaño en bits del tamaño, número de columnas y número de filas.	
dinsize	Indica el tamaño en bits del dato de entrada, que es el tamaño de palabra de la memoria externa SDRAM.	
doutsize	Indica el tamaño en bits del dato de salida, que es el tamaño del dato de entrada del bloque convolución (ver sección 2.3).	
asize	Indica el tamaño en bits de la dirección.	
burst	Indica el número de ciclos *burst* utilizados.	
Señal de entrada	**Función**	**Número de bits**
reloj	Señal de sincronismo del bloque.	1
reset	Señal de reinicio del bloque.	1
hab	Señal que habilita el bloque.	1
datoSDRAM	Datos leídos de la memoria externa SDRAM.	dinsize
listoSDRAM	Señal que indica que el dato de la memoria externa SDRAM es correcto.	1
columnas	Número de columnas de la imagen original.	tsize
tamano	Tamaño en bytes de la imagen original.	tsize
Señal de salida	**Función**	**Número de bits**
leeoriginal	Señal de lectura hacia la memoria externa SDRAM,	1
dirleeoriginal	Dirección para leer en la memoria externa SDRAM.	asize
originalRe	Parte real del dato a enviar al bloque convolución (ver sección 2.3).	doutsize
originalIm	Parte imaginaria del dato a enviar al bloque convolución (ver sección 2.3).	doutsize
listoaFIR	Señal que indica le indica al bloque convolución el dato es correcto.	1
envie8original	Señal que indica que ha enviado 8 datos al bloque convolución.	1
acabeleeroriginal	Señal que indica que el proceso de este bloque ha concluido.	1

Tabla 2.17. Parámetros, señales de entrada y salida del bloque leer imagen original.

2.9.3 Resultados independientes

El bloque fue compilado utilizando los siguientes valores de los parámetros: tsize = 20, dinsize = 32 y doutsize = 18. El bloque diseñado ocupa solamente 432 LEs en el FPGA STRATIX EP1S25F672C6 utilizado, alcanzando una frecuencia máxima de trabajo de 151.54MHz con una compilación optimizada en velocidad, mientras que 418 LEs y 116.04MHz con una optimización para el área a utilizar. La tabla 2.16 muestra los resultados obtenidos ante diferentes tipos de STRATIX, STRATIX II y de ACEX1K.

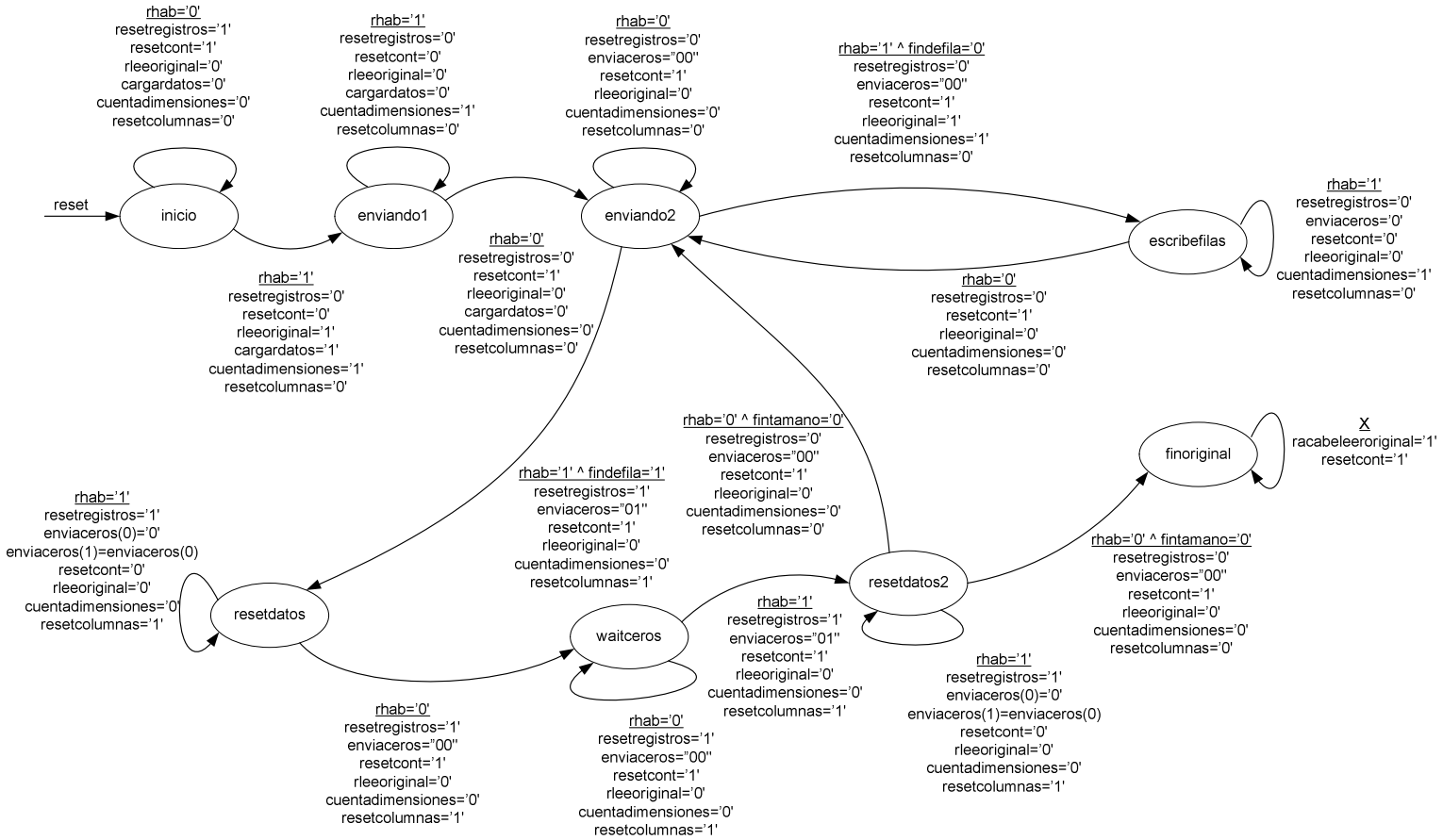


Figura 2.21. Máquina de estados del bloque leer imagen original.

		ÁREA			VELOCIDAD		
	FPGA	LEs	%LEs	Fmax (MHz)	LEs	%LEs	fmax (MHz)
Stratix	EP1S10F672-6	418	3.95	116.04	432	4.09	151.54
	EP1S10F672-7	418	3.95	110.23	432	4.09	145.20
	EP1S25F672-6	418	1.63	116.04	432	1.68	151.54
	EP1S25F672-7	418	1.63	110.23	432	1.68	145.20
	EP1S60F1508-6	418	0.73	116.04	432	0.76	151.54
	EP1S60F1508-7	418	0.73	110.23	432	0.76	145.20
Stratix II	EP2S15F484-3	333	2.13	204.67	333	2.13	204.67
	EP2S15F484-4	333	2.13	194.86	333	2.13	194.86
	EP2S15F484-5	333	2.13	181.19	333	2.13	181.19
	EP2S30F672-3	333	0.98	204.67	333	0.98	204.67
	EP2S30F672-4	333	0.98	194.86	333	0.98	194.86
	EP2S30F672-5	333	0.98	181.19	333	0.98	181.19
Acex	EP1K10QC208-1	446	77.43	108.58	461	80.03	113.64
	EP1K10QC208-2	446	77.43	88.42	461	80.03	99.01

Tabla 2.18. Resultados obtenidos usando distintos FPGAs para el bloque leer imagen original.

2.10 Bloque escribir imagen analítica

2.10.1 Comportamiento funcional

La función de este bloque es recepcionar los datos provenientes del bloque convolución, y determinar si el dato es utilizable o no según el número de coeficientes utilizados (ver sección 1.4.2). Una vez obtenido estos datos, trunca el resultado obtenido de manera que sean almacenables en la memoria externa SDRAM para que se almacene tanto la parte real como la imaginaria en la misma palabra. La escritura se realiza enviando datos de ocho en ocho para utilizar el tipo de escritura *burst* definido. El bloque luego de recibir ocho datos, le indica a la Unidad de Control que los ha recepcionado correctamente de manera que el bloque leer imagen original (ver sección 2.9) envíe nuevos datos, y una vez que termina de escribir la cantidad de datos equivalente al tamaño de la imagen, le indica a la Unidad de Control (ver sección 2.6) que ha terminado el proceso.

2.10.2 Desarrollo

El bloque desarrollado (ver ESCRIBEHILBERT.VHD en sección Anexos) se basa en una máquina de estados que es controlada por el habilitador generado

por la Unidad de Control (ver sección 2.6) y por las señales enviadas de los contadores internos implementados. Estos contadores indican el número de datos que se están recibiendo del bloque convolución, si recibió una fila completa, además si se terminó de recibir el número de datos correspondientes al tamaño de la imagen original (ver figura 2.23), y son controlados al mismo tiempo por la máquina de estados y por la correcta recepción de datos del bloque convolución (ver sección 2.3).

Debido a que el bloque leer imagen original envía dieciséis ceros luego de terminar una fila, este bloque utiliza el número de coeficientes utilizados en la convolución para determinar si el dato recibido es utilizable o no (ver sección 1.4.2, figura 1.9 y figura 2.22), utilizando para esto el parámetro *ele*, el cual tiene el valor del redondeo de la división entre el número de coeficientes y el número dos. Este procedimiento lo realiza utilizando los contadores de módulo "ele", módulo "16-ele" y de módulo "columnas" (ver figura 2.23). Los datos recepcionados son truncados de manera que se consideran sólo los dieciséis bits más significativos, de manera que se genera el dato de 32 bits a escribir en la memoria externa SDRAM.

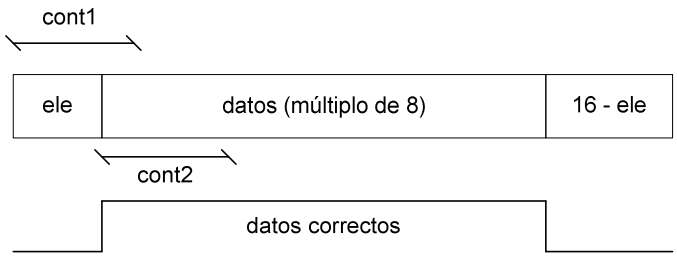


Figura 2.22. Selección de datos utilizables según el parámetro "ele".

La máquina de estados que controla este bloque (ver figura 2.24) consta de cinco estados: *inicio, recibe1, espera, recibe2* y *finhilbert.* De esta forma controla el funcionamiento de los contadores, incluyendo la generación de las direcciones de escritura, que se encuentran en el rango de 000000H a 07FFFFH. Una vez terminado el proceso, el bloque le indica a la Unidad de Control (ver sección 2.6) que ha terminado (ver tabla 2.19).

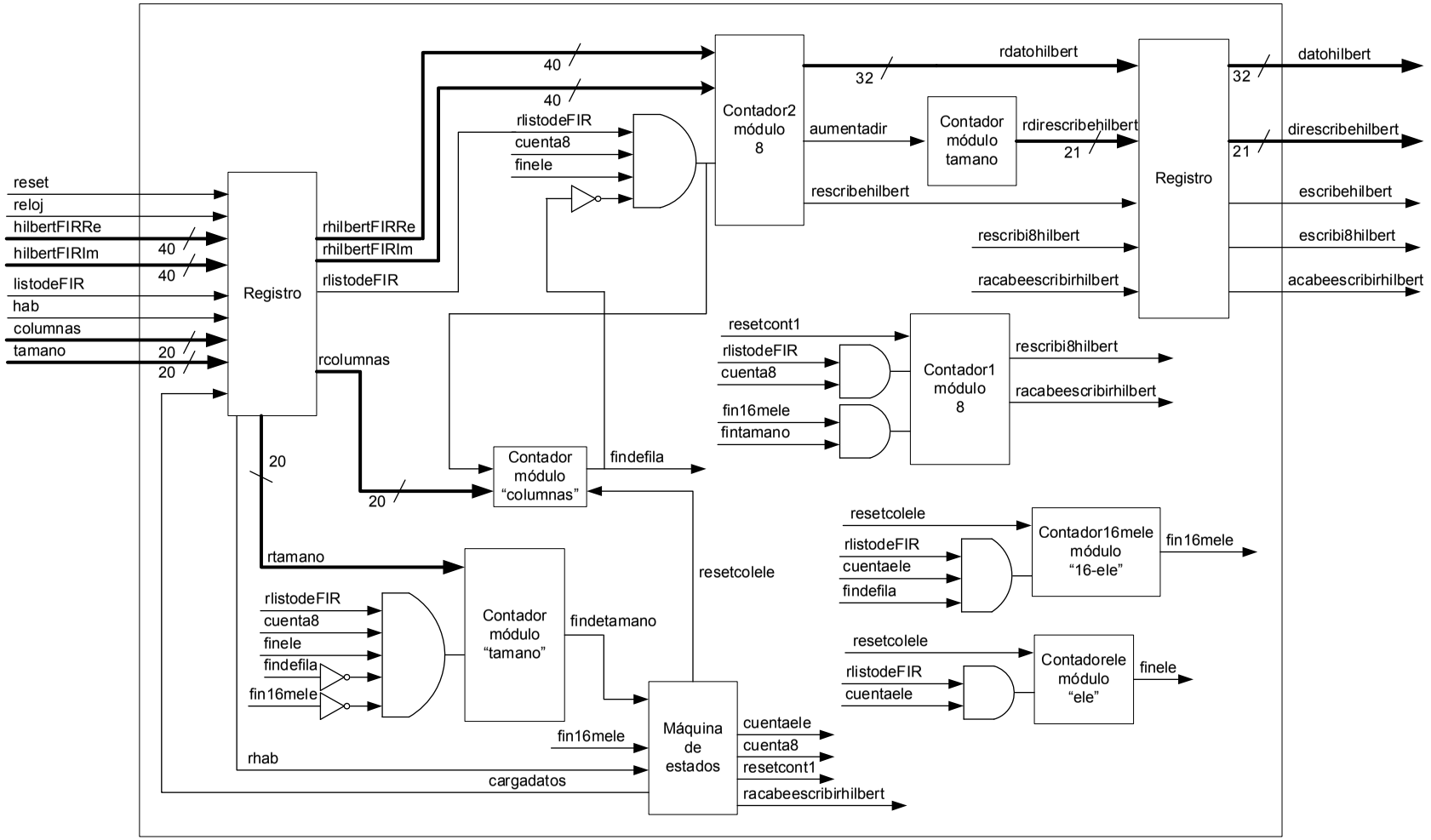


Figura 2.23. Bloque escribir imagen analítica.

Parámetro	**Función**	
tsize	Indica el tamaño en bits del tamaño, número de columnas y número de filas.	
dinsize	Indica el tamaño en bits del dato de entrada, que es el tamaño del dato de salida del bloque convolución (ver sección 2.3).	
doutsize	Indica el tamaño en bits del dato de salida, que es el tamaño de palabra de la memoria externa SDRAM.	
asize	Indica el tamaño en bits de la dirección.	
burst	Indica el número de ciclos *burst* utilizados.	
ele	Indica el número redondeado de la división entre el número de coeficientes utilizados y el número dos.	
Señal de entrada	**Función**	**Número de bits**
reloj	Señal de sincronismo del bloque.	1
reset	Señal de reinicio del bloque.	1
hab	Señal que habilita el bloque.	1
columnas	Número de columnas de la imagen original.	tsize
tamano	Tamaño en bytes de la imagen original.	tsize
hilbertFIRRe	Parte real del resultado obtenido en el bloque convolución.	dinsize
hilbertFIRIm	Parte imaginaria del resultado obtenido en el bloque convolución.	dinsize
listodeFIR	Señal que indica que el dato obtenido en el bloque convolución es correcto.	1
Señal de salida	**Función**	**Número de bits**
escribehilbert	Señal de escritura hacia la memoria externa SDRAM,	1
direscribehilbert	Dirección para escribir en la memoria externa SDRAM.	asize
datohilbert	Dato a escribir en la memoria externa SDRAM.	doutsize
escribi8hilbert	Señal que indica que ha recibido 8 datos del bloque convolución.	1
acabeescribirhilbert	Señal que indica que el proceso de este bloque ha concluido.	1

Tabla 2.19. Parámetros, señales de entrada y salida del bloque escribir imagen analítica.

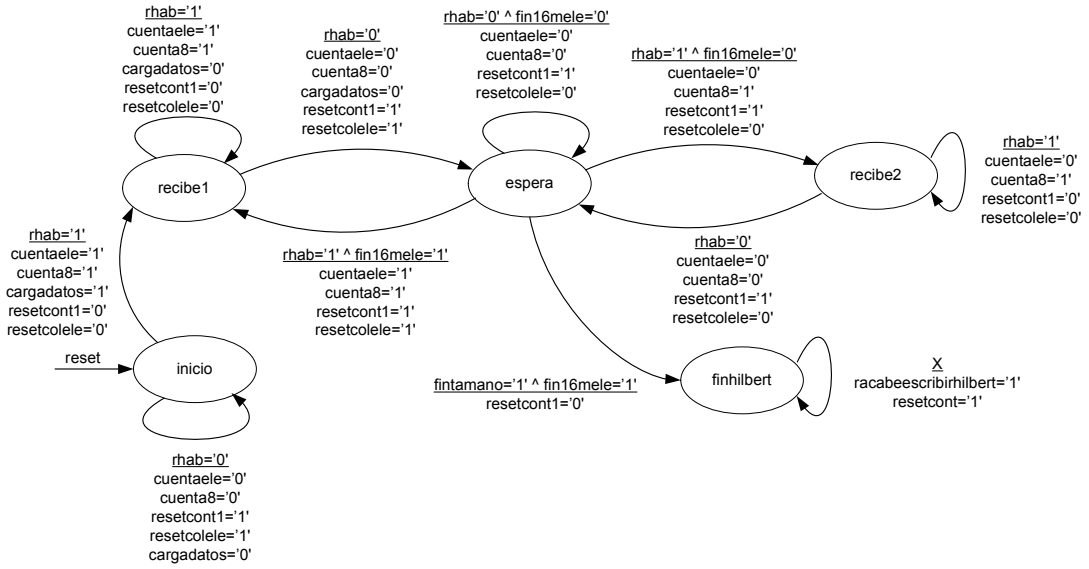


Figura 2.24. Máquina de estados del bloque escribir imagen analítica.

2.10.3 Resultados independientes

El bloque fue compilado utilizando los siguientes valores de los parámetros: tsize = 20, dinsize = 32 y doutsize = 18. El bloque diseñado ocupa 344 LEs en el FPGA STRATIX EP1S25F672C6 utilizado, alcanzando una frecuencia máxima de trabajo de 151.54MHz con una compilación optimizada en velocidad, mientras que 340 LEs y 133.46MHz con una optimización para el área a utilizar. La tabla 2.20 muestra los resultados obtenidos ante diferentes tipos de STRATIX, STRATIX II y de ACEX1K.

		ÁREA			VELOCIDAD		
	FPGA	LEs	%LEs	fmax (MHz)	LEs	%LEs	fmax (MHz)
Stratix	EP1S10F672-6	340	3.22	133.46	344	3.25	151.54
	EP1S10F672-7	340	3.22	128.65	344	3.25	144.86
	EP1S25F672-6	340	1.33	133.46	344	1.34	151.54
	EP1S25F672-7	340	1.33	128.65	344	1.34	144.86
	EP1S60F1508-6	340	0.60	133.46	344	0.60	151.54
	EP1S60F1508-7	340	0.60	128.65	344	0.60	144.86
Stratix II	EP2S15F484-3	289	1.85	204.67	289	1.85	204.67
	EP2S15F484-4	289	1.85	194.86	289	1.85	194.86
	EP2S15F484-5	289	1.85	181.19	289	1.85	181.19
	EP2S30F672-3	289	0.85	204.67	289	0.85	204.67
	EP2S30F672-4	289	0.85	194.86	289	0.85	194.86
	EP2S30F672-5	289	0.85	181.19	289	0.85	181.19
Acex	EP1K10QC208-1	403	69.97	100.50	407	70.66	102.56
	EP1K10QC208-2	403	69.97	91.32	407	70.66	92.17
	EP1K10QC208-3	403	69.97	67.80	407	70.66	69.69
	EP1K50FC484-1	403	13.99	104.71	407	14.13	106.95
	EP1K50FC484-2	403	13.99	88.11	407	14.13	89.69
	EP1K50FC484-3	403	13.99	72.20	407	14.13	74.35

Tabla 2.20. Resultados obtenidos usando distintos FPGAs para el bloque escribir imagen analítica.

2.11 Bloque leer imagen analítica

2.11.1 Comportamiento funcional

La función de este bloque es leer los datos de la imagen analítica almacenados en la memoria externa SDRAM para enviarlos al bloque convolución (ver sección 2.3) para obtener la imagen filtrada en filas. El comportamiento

funcional de este bloque es similar al del bloque leer imagen original (ver sección 2.9).

2.11.2 Desarrollo

El bloque desarrollado (ver LEEHILBERT.VHD en sección Anexos) se basa en una máquina de estados que es controlada por el habilitador generado por la Unidad de Control (ver sección 2.6) y por las señales enviadas de los contadores internos implementados. El desarrollo de este bloque es similar al del bloque leer imagen original (ver sección 2.9) con la diferencia que se utiliza una memoria interna adicional para de esta manera almacenar los datos de la parte real y de la parte imaginaria almacenados en la memoria externa SDRAM (ver figura 2.25), y que este bloque envía como parte real del dato los 2 bytes más significativos del dato leído, y los restantes como parte imaginaria. La máquina de estados que controla este bloque es similar al del bloque leer imagen original (ver sección 2.9) y el rango de direcciones que utiliza es el mismo. Los parámetros, señales de entrada y de salida correspondientes a este bloque son mostrados en la tabla 2.21.

Parámetro	**Función**	
tsize	Indica el tamaño en bits del tamaño, número de columnas y número de filas.	
dinsize	Indica el tamaño en bits del dato de entrada, que es el tamaño de palabra de la memoria externa SDRAM.	
doutsize	Indica el tamaño en bits del dato de salida, que es el tamaño del dato de entrada del bloque convolución (ver sección 2.3).	
asize	Indica el tamaño en bits de la dirección.	
burst	Indica el número de ciclos *burst* utilizados.	
Señal de entrada	**Función**	**Número de bits**
reloj	Señal de sincronismo del bloque.	1
reset	Señal de reinicio del bloque.	1
hab	Señal que habilita el bloque.	1
datoSDRAM	Datos leídos de la memoria externa SDRAM.	dinsize
listoSDRAM	Señal que indica que el dato de la memoria externa SDRAM es correcto.	1
filas	Número de filas de la imagen original.	tsize
columnas	Número de columnas de la imagen original.	tsize
tamano	Tamaño en bytes de la imagen original.	tsize
Señal de salida	**Función**	**Número de bits**
leehilbert	Señal de lectura hacia la memoria externa SDRAM,	1
dirleehilbert	Dirección para leer en la memoria externa SDRAM.	asize
hilbertRe	Parte real del dato a enviar al bloque convolución.	doutsize
hilbertIm	Parte imaginaria del dato a enviar al bloque convolución.	doutsize

listoaFIR	Señal que indica le indica al bloque convolución el dato es correcto.	1
envie8hilbert	Señal que indica que ha enviado 8 datos al bloque convolución.	1
acabeleerhilbert	Señal que indica que el proceso de este bloque ha concluido.	1

Tabla 2.21. Parámetros, señales de entrada y salida del bloque leer imagen analítica.

2.12 Bloque escribir imagen filtrada en filas

El comportamiento, desarrollo y resultados de este bloque es similar a los del bloque escribir imagen analítica (ver sección 2.10), con la única diferencia que el rango de direcciones utilizado es desde 080000H hasta 0FFFFFH ver ESCRIBEFILAS.VHD en sección Anexos).

2.13 Bloque leer imagen filtrada en filas

2.13.1 Comportamiento funcional

La función de este bloque es leer los datos de la imagen filtrada en filas almacenados en la memoria externa SDRAM para enviarlos al bloque convolución (ver sección 2.3). Debido a que el proceso de leer la imagen filtrada en filas trabaja conjuntamente con la escritura de la fase obtenida en la memoria externa SDRAM, el bloque lee los datos de ocho en ocho (este valor es definido porque se utilizó un número de ciclos *burst* igual a ocho), pero sólo envía el primer dato, ya que ahora la lectura se realiza columna por columna, lo que involucra que los datos no están ubicados en una secuencia continua en la memoria externa SDRAM. El bloque no vuelve a leer datos hasta que el resultado haya sido procesado por el bloque escribir fase obtenida (ver sección 2.14). Debido a que el número de coeficientes utilizado es menor a dieciséis, el bloque al terminar de leer una columna, envía dieciséis ceros al bloque convolución (ver sección 2.3) para obtener los resultados correctos (ver figura 1.9 y sección 1.4.2).

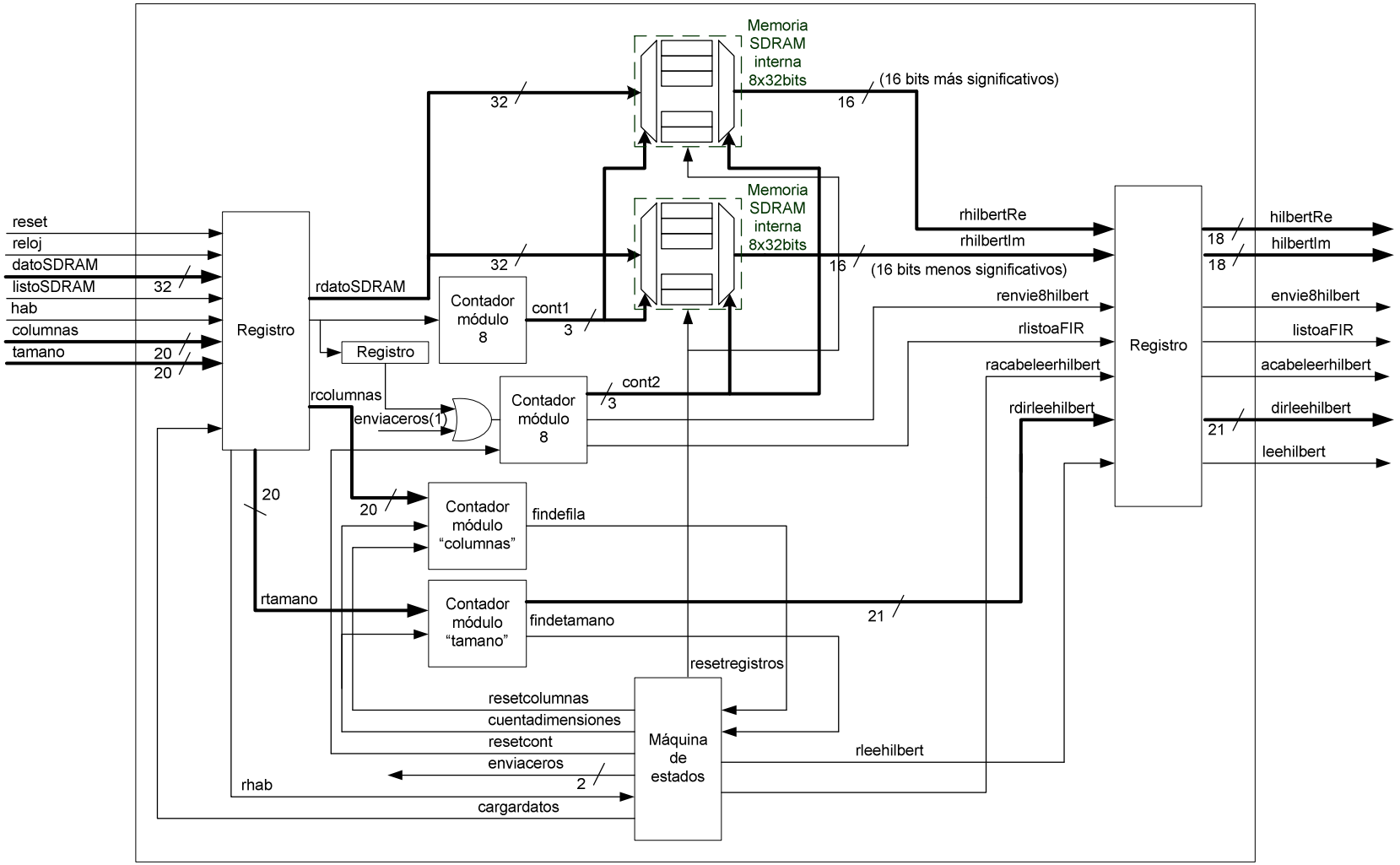


Figura 2.25. Bloque leer imagen analítica.

2.13.2 Desarrollo

El bloque desarrollado (ver LEEFILAS.VHD en sección Anexos) se basa en una máquina de estados que es controlada por el habilitador generado por la Unidad de Control (ver sección 2.6) y por la señales enviadas de los contadores internos implementados. Estos contadores indican el número de datos que se leyó, si se terminó de enviar los datos de la columna, además del número de datos enviados (ver figura 2.26).

Al leer los datos de la memoria externa SDRAM, los 16 bits más significativos, bits del 31 al 16, son enviados como la parte real, y los 16 bits menos significativos como la parte imaginaria. Los 2 bits menos significativos del dato enviado al bloque convolución (ver sección 2.3) son puestos a '0'.

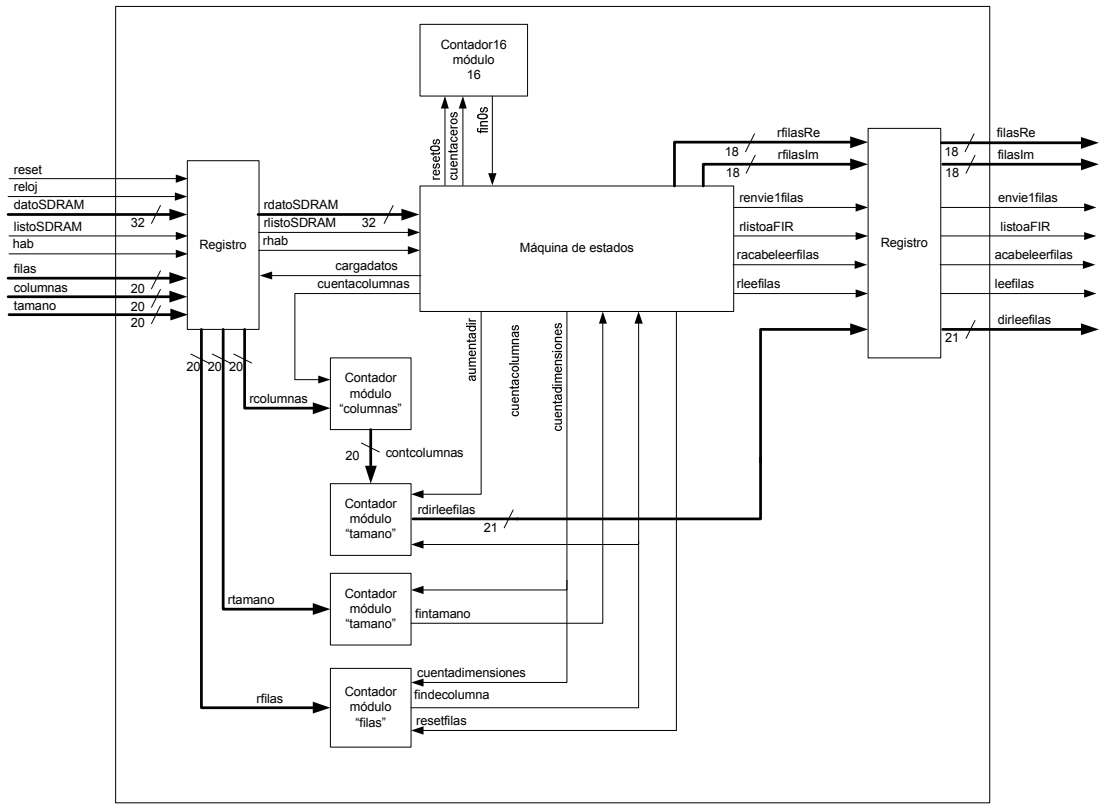


Figura 2.26. Bloque leer imagen filtrada en filas.

La máquina de estados que controla este bloque (ver figura 2.27) consta de siete estados: *inicio, espera1, enviando, espera2, ceros, duda* y *finfilas.* De esta forma controla el funcionamiento de los contadores, incluyendo la generación de las direcciones de lectura, que se encuentran en el rango de 080000H a

0FFFFFH. Una vez terminado el proceso, el bloque le indica a la Unidad de Control (ver sección 2.6) que ha terminado (ver tabla 2.22).

Parámetro	**Función**	
tsize	Indica el tamaño en bits del tamaño, número de columnas y número de filas.	
dinsize	Indica el tamaño en bits del dato de entrada, que es el tamaño de palabra de la memoria externa SDRAM.	
doutsize	Indica el tamaño en bits del dato de salida, que es el tamaño del dato de entrada del bloque convolución (ver sección 2.3).	
asize	Indica el tamaño en bits de la dirección.	
burst	Indica el número de ciclos *burst* utilizados.	
Señal de entrada	**Función**	**Número de bits**
reloj	Señal de sincronismo del bloque.	1
reset	Señal de reinicio del bloque.	1
hab	Señal que habilita el bloque.	1
datoSDRAM	Datos leídos de la memoria externa SDRAM.	dinsize
listoSDRAM	Señal que indica que el dato de la memoria externa SDRAM es correcto.	1
columnas	Número de columnas de la imagen original.	tsize
tamano	Tamaño en bytes de la imagen original.	tsize
Señal de salida	**Función**	**Número de bits**
leefilas	Señal de lectura hacia la memoria externa SDRAM,	1
dirleefilas	Dirección para leer en la memoria externa SDRAM.	asize
filasRe	Parte real del dato a enviar al bloque convolución.	doutsize
filasIm	Parte imaginaria del dato a enviar al bloque convolución.	doutsize
listoaFIR	Señal que indica le indica al bloque convolución el dato es correcto.	1
envie1filas	Señal que indica que ha enviado 8 datos al bloque convolución.	1
acabeleerfilas	Señal que indica que el proceso de este bloque ha concluido.	1

Tabla 2.22. Parámetros, señales de entrada y salida del bloque leer imagen filtrada en filas.

2.13.3 Resultados independientes

El bloque fue compilado utilizando los siguientes valores de los parámetros: tsize = 20, dinsize = 32 y doutsize = 18. El bloque diseñado ocupa solamente 451 LEs en el FPGA STRATIX EP1S25F672C6 utilizado, alcanzando una frecuencia máxima de trabajo de 148.32MHz con una compilación optimizada en velocidad, mientras que 448 LEs y 133.46 con una optimización para el área a utilizar. La tabla 2.23 muestra los resultados obtenidos ante diferentes tipos de STRATIX, STRATIX II y de ACEX1K.

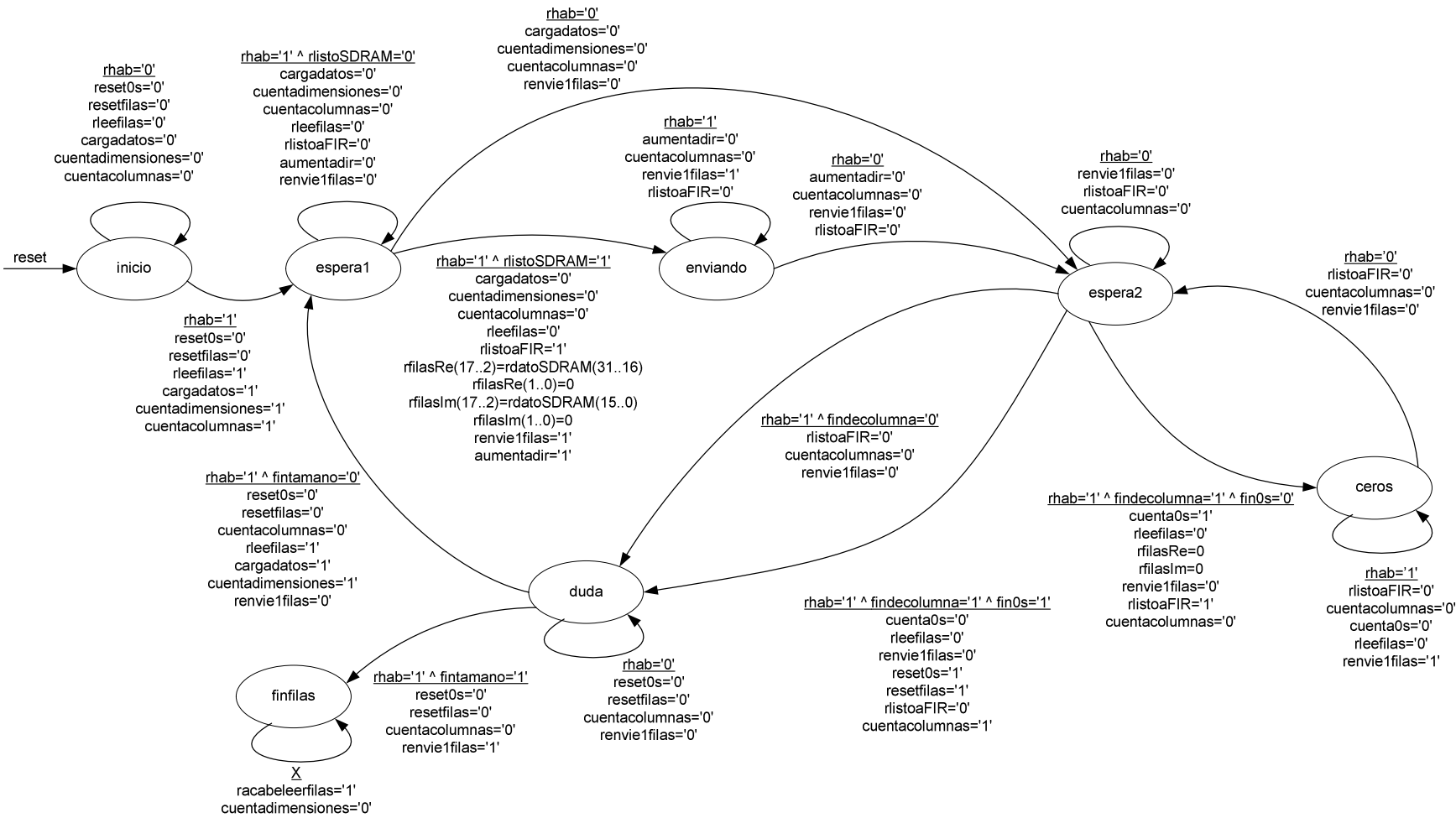


Figura 2.27. Máquina de estados del bloque leer imagen original.

		ÁREA			VELOCIDAD		
	FPGA	LEs	%LEs	fmax (MHz)	LEs	%LEs	fmax (MHz)
Stratix	EP1S10F672-6	448	4.24	133.46	451	4.27	148.32
	EP1S10F672-7	448	4.24	128.65	451	4.27	141.60
	EP1S25F672-6	448	1.75	133.46	451	1.76	148.32
	EP1S25F672-7	448	1.75	128.65	451	1.76	141.60
	EP1S60F1508-6	448	0.78	133.46	451	0.79	148.32
	EP1S60F1508-7	448	0.78	128.65	451	0.79	141.60
Stratix II	EP2S15F484-3	347	2.22	194.48	347	2.22	198.69
	EP2S15F484-4	347	2.22	184.88	347	2.22	192.98
	EP2S15F484-5	347	2.22	171.61	347	2.22	180.21
	EP2S30F672-3	347	1.02	194.48	347	1.02	198.69
	EP2S30F672-4	347	1.02	184.88	347	1.02	192.98
	EP2S30F672-5	347	1.02	171.61	347	1.02	180.21
Acex	EP1K10QC208-1	558	96.88	100.50	567	98.44	102.56
	EP1K10QC208-2	558	96.88	91.32	567	98.44	92.17
	EP1K10QC208-3	558	96.88	67.80	567	98.44	69.69
	EP1K50FC484-1	558	19.38	104.71	567	19.69	106.95
	EP1K50FC484-2	558	19.38	88.11	567	19.69	89.69
	EP1K50FC484-3	558	19.38	72.20	567	19.69	74.35

Tabla 2.23. Resultados obtenidos usando distintos FPGAs para el bloque leer imagen filtrada en filas.

2.14 Bloque escribir fase obtenida

2.14.1 Comportamiento funcional

La función de este bloque es recepcionar la fase obtenida del bloque arcotangente (ver sección 2.5), truncarla y escribirla en la memoria externa SDRAM. La escritura la hace en posiciones de memoria no continuas debido a que se recepcionan a manera de columna por columna (diferente a la escritura original de fila por fila). Este bloque le informa a la Unidad de Control (ver sección 2.6) una vez que ha escrito el dato y cuando ha terminado de escribir todos los datos.

2.14.2 Desarrollo

El bloque desarrollado (ver ESCRIBEFASE.VHD en sección Anexos) se basa en una máquina de estados que es controlada por el habilitador generado por la Unidad de Control (ver sección 2.6) y por las señales enviadas de los contadores internos implementados. Estos contadores indican el número de

datos que se recepcionó, si se terminó de escribir los datos de la columna, además del número de datos escritos (ver figura 2.28).

Debido a que el bloque leer imagen filtrada en filas envía dieciséis ceros luego de terminar una columna, este bloque utiliza el número de coeficientes utilizados en la convolución para determinar si el dato recibido es utilizable o no (ver sección 1.4.2, figura 1.9 y figura 2.22), utilizando para esto el parámetro *ele*. Este procedimiento lo realiza utilizando los contadores de módulo "ele", módulo "16-ele" y de módulo "columnas" (ver figura 2.28). Los datos recepcionados son truncados de manera que se consideran sólo los treinta y dos bits más significativos, de manera que se genera el dato de 32 bits a escribir en la memoria externa SDRAM. La máquina de estados que controla este bloque (ver figura 2.29 y tabla 2.24) consta de ocho estados: *inicio, recibe, espera1, espera2, nosirve1, nosirve2, duda,* y *finfase.* De esta forma controla el funcionamiento de los contadores, incluyendo la generación de las direcciones de escritura, que se encuentran en el rango de 100000H a 1FFFFFH. Una vez terminado el proceso, el bloque le indica a la Unidad de Control (ver sección 2.6) que ha terminado (ver tabla 2.25).

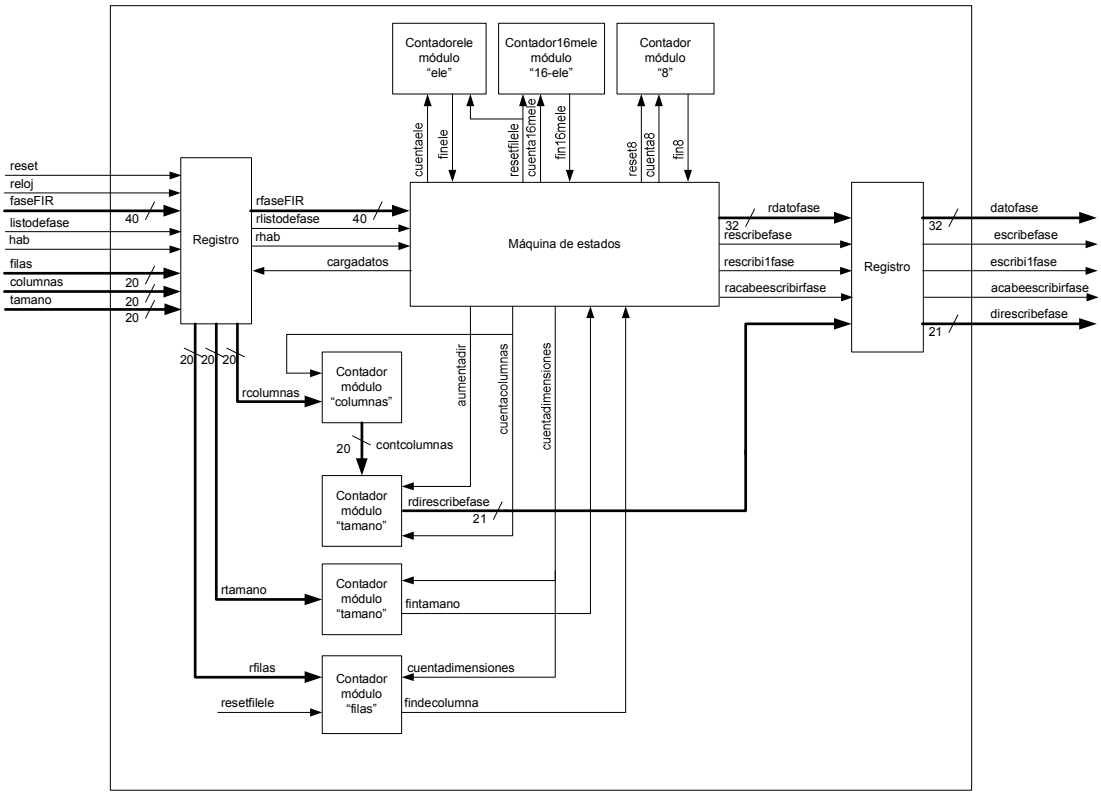


Figura 2.28. Bloque escribir fase obtenida.

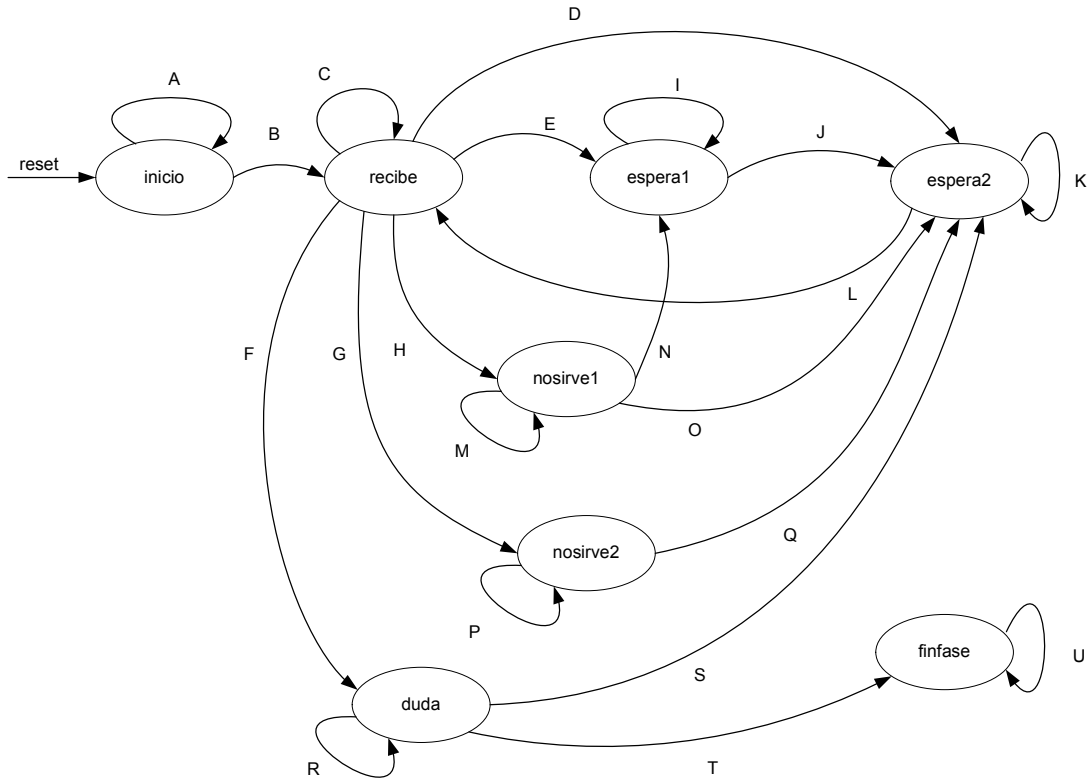


Figura 2.29. Máquina de estados del bloque escribir fase obtenida.

SALTO	CONDICIÓN	SALIDA
A	rhab='0'	reset8='1'; resetfilele='1'; cargadatos='0'; rescribi1fase='0'
B	rhab='1'	reset8='0'; resetfilele='0'; cargadatos='1'; rescribi1fase='0'
C	rhab='1' ^ rlistodeFIR='0'	cargadatos='0'; rescribefase='0'; rescribi1fase='0'; cuentacolumnas='0'
D	rhab='0'	cargadatos='0'; rescribi1fase='0'
E	rhab='1' ^ rlistodeFIR='1' ^ finele='0'	cargadatos='0'; cuentaele='1'; cuentadimensiones='0'; rescribefase='0'; rescribi1fase='1'; cuentacolumnas='0'
F	rhab='1' ^ rlistodeFIR='1' ^ finele='1' ^ findecolumna='1' ^ fin16mele='1'	cargadatos='0'; rescribefase='0'; resetfilele='1'; aumentadir='1'; rescribi1fase='1'; cuentacolumnas='1'
G	rhab='1' ^ rlistodeFIR='1' ^ finele='1' ^ findecolumna='1' ^ fin16mele='0'	cargadatos='0'; rescribefase='0'; cuenta8='0'; aumentadir='0'; cuentadimensiones='0'; cuenta16mele='1'; cuentacolumnas='0'
H	rhab='1' ^ rlistodeFIR='1' ^ finele='1' ^ findecolumna='0'	cargadatos='0'; rescribefase='0'; rdatofase=rfaseFIR(39..8); cuenta8='1'; aumentadir='0'; cuentadimensiones='1'; cuenta16mele='0'; rescribi1fase='0'; cuentacolumnas='0'
I	rhab='1'	rescribefase='0'; aumentadir='0'; cuentaele='0'; rescribi1fase='1'; reset8='0'; cuentadimensiones='0'
J	rhab='0'	rescribefase='0'; aumentadir='0'; rescribi1fase='0'
K	rhab='0'	rescribi1fase='0'

L	rhab='1'	rescribi1fase='0'
M	rhab='1' ^ fin8='0'	rescribefase='1'; cuenta8='1'; aumentadir='0'; cuentadimensiones='0'; rescribi1fase='0'
N	rhab='1' ^ fin8='1'	rescribefase='0'; reset8='1'; cuenta8='0'; aumentadir='1'; cuentadimensiones='0'; rescribi1fase='1'
O	rhab='0'	cuenta8='0'; rescribefase='0'; rescribefase='0'; rescribi1fase='0'
P	rhab='1'	cuenta16mele='0'; rescribi1fase='1'
Q	rhab='0'	rescribi1fase='0'
R	rhab='1' ^ fintamano='0'	cuentacolumnas='0'; resetfilele='0'; aumentadir='0'; racabeescribirfase='0'
S	rhab='0'	cuentacolumnas='0'; resetfilele='0'; aumentadir='0'; rescribi1fase='0'
T	rhab='1' ^ fintamano='1'	cuentacolumnas='0'; rescribi1fase='1'; racabeescribirfase='1'
U	X	racabeescribirfase='1'; cuenta8='0'; cuentaele='0'; cuentadimensiones='0'; aumentadir='0'; cuentacolumnas='0'

Tabla 2.24. Condiciones de la máquina de estados del bloque escribir fase obtenida.

Parámetro	**Función**	
tsize	Indica el tamaño en bits del tamaño, número de columnas y número de filas.	
dinsize	Indica el tamaño en bits del dato de entrada, que es el tamaño del dato de salida del bloque arcotangente (ver sección 2.5).	
doutsize	Indica el tamaño en bits del dato de salida, que es el tamaño de palabra de la memoria externa SDRAM.	
asize	Indica el tamaño en bits de la dirección.	
burst	Indica el número de ciclos *burst* utilizados.	
ele	Indica el número redondeado de la división entre el número de coeficientes utilizados y el número dos.	
Señal de entrada	**Función**	**Número de bits**
reloj	Señal de sincronismo del bloque.	1
reset	Señal de reinicio del bloque.	1
hab	Señal que habilita el bloque.	1
filas	Número de filas de la imagen original.	tsize
columnas	Número de columnas de la imagen original.	tsize
tamano	Tamaño en bytes de la imagen original.	tsize
faseFIR	Fase obtenida del bloque arcotangente.	dinsize
listodefase	Señal que indica que el dato obtenido en el bloque arcotangente es correcto.	1
Señal de salida	**Función**	**Número de bits**
escribefase	Señal de escritura hacia la memoria externa SDRAM,	1
direscribefase	Dirección para escribir en la memoria externa SDRAM.	asize
datofase	Dato a escribir en la memoria externa SDRAM.	doutsize
escribi1fase	Señal que indica que ha recibido 8 datos del bloque convolución.	1
acabeescribirfase	Señal que indica que el proceso de este bloque ha concluido.	1

Tabla 2.25. Parámetros, señales de entrada y salida del bloque escribir fase obtenida.

2.14.3 Resultados independientes

El bloque fue compilado utilizando los siguientes valores de los parámetros: tsize = 20, dinsize = 32 y doutsize = 18. El bloque diseñado ocupa solamente 497 LEs en el FPGA STRATIX EP1S25F672C6 utilizado, alcanzando una frecuencia máxima de trabajo de 148.32MHz con una compilación optimizada en velocidad, mientras que 490 LEs y 133.46MHZ con una optimización para el área a utilizar. La tabla 2.26 muestra los resultados obtenidos ante diferentes tipos de STRATIX, STRATIX II y de ACEX1K.

		ÁREA			VELOCIDAD		
	FPGA	LEs	%LEs	fmax (MHz)	LEs	%LEs	fmax (MHz)
Stratix	EP1S10F672-6	490	4.64	133.46	497	4.70	148.32
	EP1S10F672-7	490	4.64	128.65	497	4.70	141.60
	EP1S25F672-6	490	1.91	133.46	497	1.94	148.32
	EP1S25F672-7	490	1.91	128.65	497	1.94	141.60
	EP1S60F1508-6	490	0.86	133.46	497	0.87	148.32
	EP1S60F1508-7	490	0.86	128.65	497	0.87	141.60
Stratix II	EP2S15F484-3	399	2.56	203.42	399	2.56	203.42
	EP2S15F484-4	399	2.56	193.72	399	2.56	193.72
	EP2S15F484-5	399	2.56	180.21	399	2.56	180.21
	EP2S30F672-3	399	1.18	203.42	399	1.18	203.42
	EP2S30F672-4	399	1.18	193.72	399	1.18	193.72
	EP2S30F672-5	399	1.18	180.21	399	1.18	180.21
Acex	EP1K10QC208-1	668	115.97	100.50	676	117.36	102.56
	EP1K10QC208-2	668	115.97	91.32	676	117.36	92.17
	EP1K10QC208-3	668	115.97	67.80	676	117.36	69.69
	EP1K50FC484-1	668	23.19	104.71	676	23.47	106.95
	EP1K50FC484-2	668	23.19	88.11	676	23.47	89.69
	EP1K50FC484-3	668	23.19	72.20	676	23.47	74.35

Tabla 2.26. Resultados obtenidos usando distintos FPGAs.

2.15 Bloque leer fase obtenida y contador

2.15.1 Comportamiento funcional

La función de este bloque es enviar al bloque transmisor serial (ver sección 2.18) la cuenta del tiempo tomado en el proceso total (ver sección 2.19) además de la fase obtenida almacenada en la memoria externa SDRAM. El bloque envía un byte del dato correspondiente y espera a que haya sido transmitido para enviar el siguiente.

2.8.2 Desarrollo

El bloque desarrollado (ver LEEFASE.VHD en sección Anexos) se basa en una máquina de estados que es controlada por el tamaño de la imagen que se está enviando y el correcto envío del dato. El bloque recepciona los datos de los contadores para transmitirlos y luego lee la memoria externa. El bloque cuenta internamente con multiplexores (ver figura 2.30) para seleccionar el dato correcto a transmitir, sea el contador deseado (cuenta total o cuenta de obtención de la imagen analítica, ver sección 2.19) o el dato almacenado en la memoria externa SDRAM. El tamaño de la fase obtenida que se utiliza es de 16 bits, por lo que se considera sólo los 2 bytes más significativos del dato almacenado en la memoria SDRAM. De estos 2 bytes seleccionados, se envía primero el byte más significativo y luego el menos significativo.

La máquina de estados que controla este bloque (ver figura 2.31) consta de trece estados: *inicio, Tu, Td, Tc, Tm, Hu, Hd, Hc, Hm, espera, enviaMSB, enviaLSB* y *finleefase.* En los estados desde *Tu* hasta *Hm* se envía la cuenta total del proceso (unidades, decenas, centenas y miles, respectivamente) y la cuenta de la obtención de la imagen analítica (unidades, decenas, centenas y miles, respectivamente), para luego enviar la información referente a la fase obtenida. De esta forma controla el funcionamiento de los contadores, incluyendo la generación de las direcciones de lectura, que se encuentran en el rango de 100000H a 1FFFFFH. Una vez terminado el proceso, el bloque le indica a la Unidad de Control (ver sección 2.6) que ha terminado (ver tabla 2.27).

2.15.3 Resultados independientes

El bloque fue compilado utilizando los siguientes valores de los parámetros: tsize = 20, dinsize = 32 y doutsize = 18. El bloque diseñado ocupa solamente 612 LEs en el FPGA STRATIX EP1S25F672C6 utilizado, alcanzando una frecuencia máxima de trabajo de 149.84MHz con una compilación optimizada en velocidad, mientras que 608 LEs y 128.57MHz con una optimización para el área a utilizar. La tabla 2.28 muestra los resultados obtenidos ante diferentes tipos de STRATIX, STRATIX II y de ACEX1K.

Parámetro	**Función**	
tsize	Indica el tamaño en bits del tamaño, número de columnas y número de filas.	
dinsize	Indica el tamaño en bits del dato de entrada, que es el tamaño de palabra de la memoria externa SDRAM.	
doutsize	Indica el tamaño en bits del dato de salida.	
asize	Indica el tamaño en bits de la dirección.	
csize	Indica el tamaño en bits de cada cuenta a enviar (ver sección 2.19).	
Señal de entrada	**Función**	**Número de bits**
reloj133	Señal de sincronismo del bloque.	1
reset	Señal de reinicio del bloque.	1
hab	Señal que habilita el bloque.	1
columnas	Número de columnas de la imagen original.	tsize
tamano	Tamaño en bytes de la imagen original.	tsize
enviadoTX	Señal que indica que el dato ya ha sido enviado.	1
listoSDRAM	Señal que indica que el dato de la memoria externa SDRAM es correcto.	1
datoSDRAM0	Dato “0” leído de la memoria externa SDRAM.	dinsize
datoSDRAM1	Dato “1” leído de la memoria externa SDRAM.	dinsize
datoSDRAM2	Dato “2” leído de la memoria externa SDRAM.	dinsize
datoSDRAM3	Dato “3” leído de la memoria externa SDRAM.	dinsize
datoSDRAM4	Dato “4” leído de la memoria externa SDRAM.	dinsize
datoSDRAM5	Dato “5” leído de la memoria externa SDRAM.	dinsize
datoSDRAM6	Dato “6” leído de la memoria externa SDRAM.	dinsize
datoSDRAM7	Dato “7” leído de la memoria externa SDRAM.	dinsize
tiempoTu	Unidades del tiempo total empleado.	csize
tiempoTd	Decenas del tiempo total empleado.	csize
tiempoTc	Centenas del tiempo total empleado.	csize
tiempoTm	Miles del tiempo total empleado.	csize
tiempoHu	Unidades del tiempo empleado en la obtención de la imagen analítica.	csize
tiempoHd	Decenas del tiempo empleado en la obtención de la imagen analítica.	csize
tiempoHc	Centenas del tiempo empleado en la obtención de la imagen analítica.	csize
tiempoHm	Miles del tiempo empleado en la obtención de la imagen analítica.	csize
Señal de salida	**Función**	**Número de bits**
leefase	Señal de lectura hacia la memoria externa SDRAM,	1
dirleefase	Dirección para leer en la memoria externa SDRAM.	asize
fase	Dato de la fase para transmitir.	doutsize
enviafase	Señal que indica que el dato se debe enviar.	1
acabeleefase	Señal que indica que el bloque terminó su proceso.	1

Tabla 2.27. Parámetros, señales de entrada y salida del bloque leer fase obtenida y cuentas.

		ÁREA			VELOCIDAD		
	FPGA	LEs	%LEs	fmax (MHz)	LEs	%LEs	fmax (MHz)
Stratix	EP1S10F672-6	608	5.75	128.57	612	5.79	149.84
	EP1S10F672-7	608	5.75	123.55	612	5.79	144.86
	EP1S25F672-6	608	2.37	128.57	612	2.39	149.84
	EP1S25F672-7	608	2.37	123.55	612	2.39	144.86

	EP1S60F1508-6	608	1.06	128.57	612	1.07	149.84
	EP1S60F1508-7	608	1.06	123.55	612	1.07	144.86
Stratix II	EP2S15F484-3	435	2.79	201.01	435	2.79	201.01
	EP2S15F484-4	435	2.79	194.86	435	2.79	194.86
	EP2S15F484-5	435	2.79	181.19	435	2.79	181.19
	EP2S30F672-3	435	1.28	201.01	435	1.28	201.01
	EP2S30F672-4	435	1.28	194.86	435	1.28	194.86
	EP2S30F672-5	435	1.28	181.19	435	1.28	181.19
Acex	EP1K10QC208-1	706	122.57	100.50	726	126.04	102.56
	EP1K10QC208-2	706	122.57	91.32	726	126.04	92.17
	EP1K10QC208-3	706	122.57	67.80	726	126.04	69.69
	EP1K50FC484-1	706	24.51	104.71	726	25.21	106.95
	EP1K50FC484-2	706	24.51	88.11	726	25.21	89.69
	EP1K50FC484-3	706	24.51	72.20	726	25.21	74.35

Tabla 2.28. Resultados obtenidos usando distintos FPGAs para el bloque lee fase obtenida y contador.

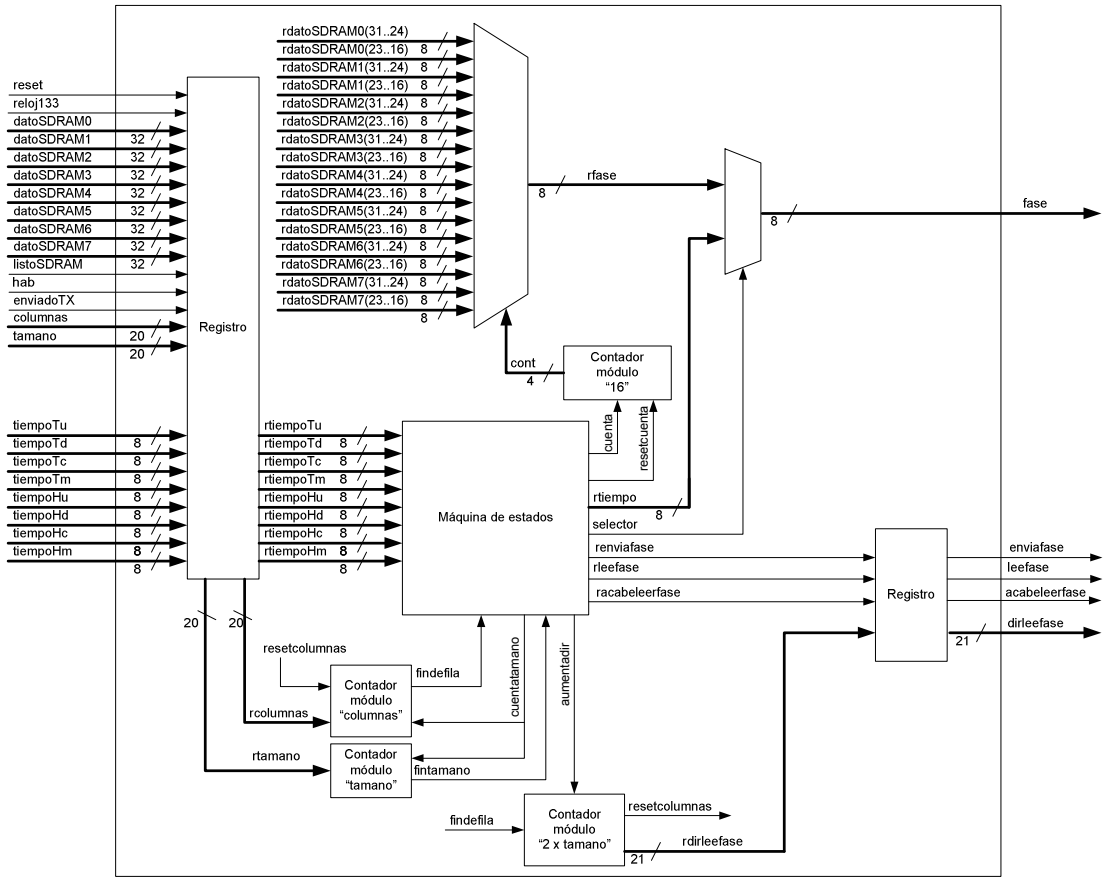


Figura 2.30. Bloque leer fase obtenida y cuentas.

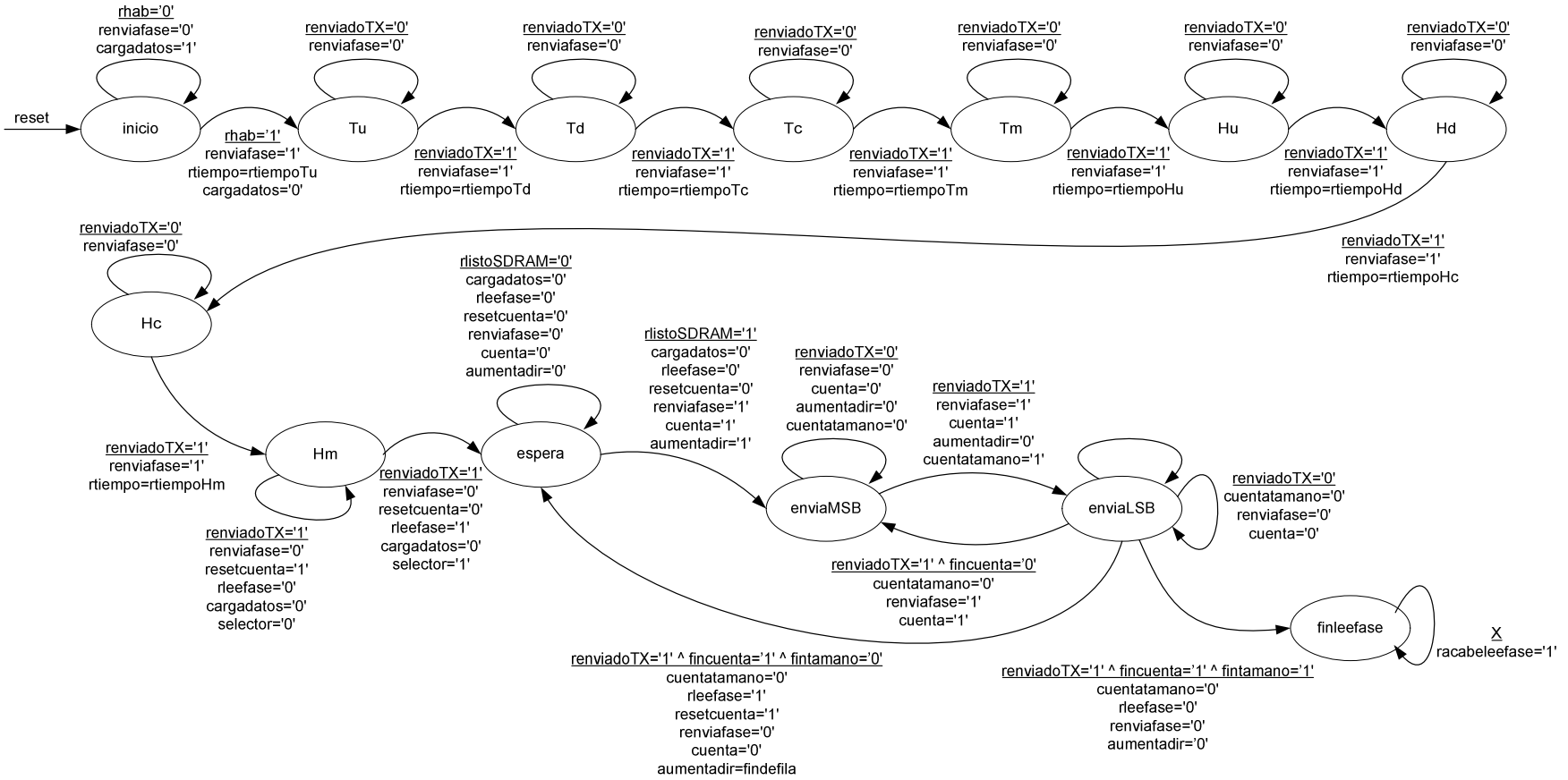


Figura 2.31. Máquina de estados del bloque leer fase obtenida y cuentas.

2.16 Demultiplexor y multiplexores

2.16.1 Introducción

En este trabajo se utilizaron dos tipos de multiplexores: los simples y los diseñados, además de utilizarse un demultiplexor diseñado. Los multiplexores simples (ver sección 2.16.2) fueron implementados utilizando el *Megawizard* del programa Quartus II, por lo que su utilización fue directa. Los multiplexores diseñados fueron descritos usando VHDL y tienen características distintas a las de los multiplexores simples (ver sección 2.16.3). El demultiplexor utilizado fue descrito usando VHDL y tiene características similares a la de los multiplexores diseñados (ver sección 2.16.4).

Las siguientes líneas muestran el funcionamiento de ellos además de indicar en qué partes de la arquitectura desarrollada se utilizaron.

2.16.2 Multiplexores simples

Estos multiplexores (ver figura 2.32.a) son utilizados los bloques indicados en la tabla 2.29.

Multiplexor	Selector	Señales de entrada	Señal de Salida	Tamaño de palabra
Escritura de memoria externa SDRAM	selectorMUXescribir	escribeoriginal escribehilbert escribefilas escribefase	escribirSDRAM	1
Lectura de memoria externa SDRAM	selectorMUXleer	leeoriginal leehilbert leefilas leefase	leerSDRAM	1
Datos de entrada a escribir en memoria externa SDRAM	selectorMUXdatos	datooriginal datohilbert datofilas datofase	datoinSDRAM	32
Dirección de la memoria externa SDRAM	selectorMUXADDR	direscribeoriginal dirleeoriginal direscribehilbert dirleehilbert direscribefilas dirleefilas direscribefase dirleefase	dirSDRAM	21

Tabla 2.29. Multiplexores simples utilizados en la arquitectura desarrollada.

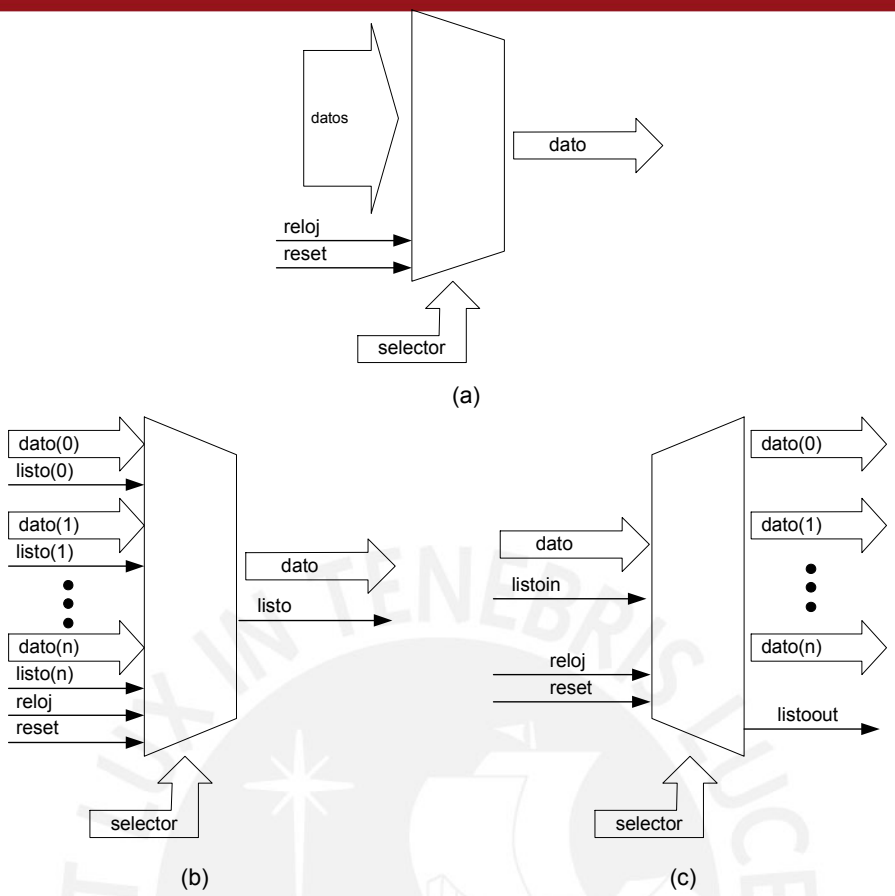


Figura 2.32. (a) Multiplexor simple. (b) Multiplexor diseñado. (c) Demultiplexor diseñado.

2.16.3 Multiplexores diseñados

Estos multiplexores tienen la característica de que cada dato a la entrada tiene su propia señal de cargar datos, con lo que no es suficiente con seleccionar el dato deseado, sino que además debe cargarse. El multiplexor diseñado (ver figura 2.32.b) genera a su salida una señal que indica que el dato a la salida es correcto. Estos multiplexores son utilizados los bloques indicados en la tabla 2.30.

2.16.4 Demultiplexor diseñado

Este demultiplexor tiene la característica de que el dato de entrada tiene su propia señal de cargar datos, con lo que no es suficiente con seleccionar la salida deseada, sino que además debe cargarse. El demultiplexor diseñado (ver figura 2.32.c) genera a su salida una señal que indica que el dato a la

salida es correcto. El demultiplexor es utilizado en el bloque indicado en la tabla 2.31.

Multiplexor	Selector	Señales de entrada	Señal de Salida	Tamaño de palabra
Datos de entrada del bloque convolución.	selectorMUXFIR	originalRe originalIm	Re Im	18 18
		hilbertRe hilbertIm		
		filasRe filasIm		
Parte real de la convolución obtenida.	selectorMUXRe	Retardo	deFIRReDEMUX	40
		filtradoRe		

Tabla 2.30. Multiplexores diseñados utilizados en la arquitectura desarrollada.

Multiplexor	Selector	Señal de entrada	Señales de salida	Tamaño de palabra
Datos de salida del bloque convolución.	selectorDEMUXFIR	Re Im	hilbertFIRRe hilbertFIRIm	40
			filasFIRRe filasFIRIm	
			faseFIRRe faseFIRIm	

Tabla 2.31. Demultiplexor diseñado utilizado en la arquitectura desarrollada.

2.17 Bloques retardadores

2.17.1 Comportamiento funcional y desarrollo

La función de estos bloques producir un retardo, mediante el desplazamiento en registros, de la propagación de la señal deseada. El bloque presenta parámetros que indican el número de ciclos de retardo que se desea obtener, además del tamaño del dato. El bloque desarrollado (ver RETARDADOR.VHD en sección Anexos) se basa en registros colocados en cascada (ver figura 2.33). El parámetro *retardo* determina el número de ciclos de retardo que se requiere y con ello la creación de los registros de tamaño de palabra *dsize* (ver tabla 2.32).

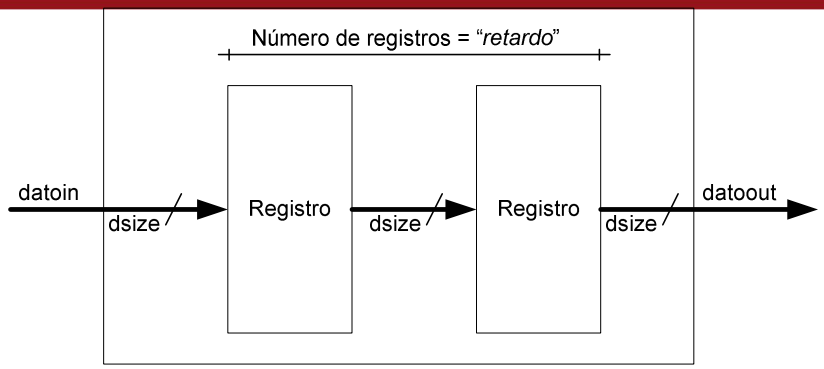


Figura 2.33. Bloques retardadores.

Parámetro	**Función**	
Dsize	Indica el tamaño en bits del dato.	
Retardo	Indica el número de ciclos de retardo que se desean.	
Señal de entrada	**Función**	**Número de bits**
Reloj	Señal de sincronismo del bloque.	1
Reset	Señal de reinicio del bloque.	1
Datoin	Dato de que se desea retardar.	dsize
Señal de salida	**Función**	**Número de bits**
Datoout	Dato retardado.	dsize

Tabla 2.32. Parámetros, señales de entrada y salida de los bloques retardadores.

		ÁREA ó VELOCIDAD						
		Retardo = 5			Retardo = 10			fmax (MHz)
	FPGA	Tamaño de palabra	LEs	%LEs	Tamaño de palabra	LEs	%LEs	
Stratix	EP1S25F672-6	18	108	1080	18	198	0	422.12
	EP1S25F672-7	18	108	1080	18	198	0	390.02
	EP1S25F672-6	40	240	1080	40	440	0	422.12
	EP1S25F672-7	40	240	1080	40	440	0	390.02
Stratix II	EP2S30F672-3	18	108	2400	18	198	0	422.12
	EP2S30F672-4	18	108	2400	18	198	0	422.12
	EP2S30F672-5	18	108	1080	18	198	0	390.02
	EP2S30F672-3	40	240	2400	40	440	0	422.12
	EP2S30F672-4	40	240	2400	40	440	0	422.12
	EP2S30F672-5	40	240	2400	40	440	0	390.02

Tabla 2.33. Resultados obtenidos usando distintos FPGAs para los bloques retardadores.

2.17.2 Resultados independientes

El bloque fue compilado variando el tamaño del dato de entrada y el número de ciclos de retardo deseado ante diferentes tipos de STRATIX y STRATIX II (ver tabla 2.33).

2.18 Transmisor serial

2.18.1 Comportamiento funcional

Este bloque es el complemento del bloque receptor serial (ver sección 2.2) para un correcto intercambio de datos entre el FPGA y el STRATIX. Al igual que el receptor serial, el bloque es parametrizable en cuanto a la velocidad de transferencia de datos, además cuenta con una entrada para los datos y una entrada de reinicio asíncrono. Como salida debe generar serialmente la secuencia necesaria para transmitir el dato e indicar una vez que lo ha transmitidos.

Se decidió utilizar el formato de envío con 8 bits de datos, sin paridad, y con dos bits de parada.

2.18.2 Desarrollo

El desarrollo del bloque utiliza los sub-bloques similares al bloque receptor serial (ver sección 2.2) pero dispuestos de forma diferente (ver figura 2.34). La máquina de estados que controla el bloque utiliza la misma idea del receptor serial con la diferencia que mientras en la recepción se recibe 1 bit por estado, en esta máquina se envía 1 bit por estado (ver figura 2.35). La tabla 2.34 muestra los parámetros, señales de entrada y de salida del bloque.

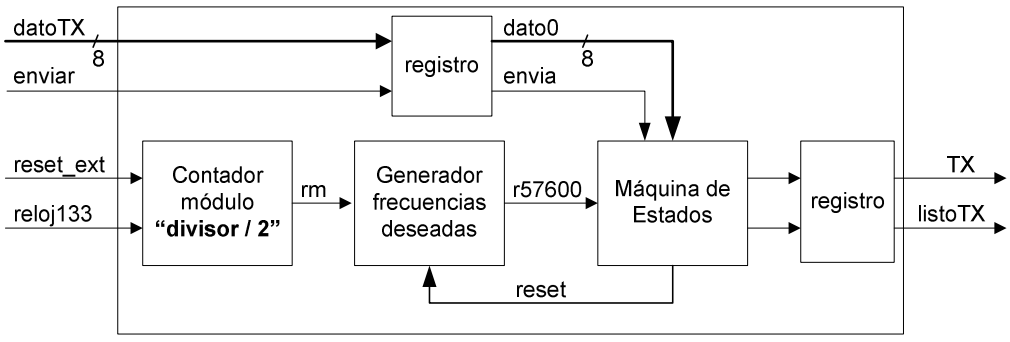


Figura 2.34. Bloque transmisor serial.

Parámetro	Función	
divisor	Controla la velocidad de transferencia de datos (ver ecuación 14).	
Señal de entrada	**Función**	**Número de bits**
reloj133	Señal de sincronismo del bloque.	1
reset_ext	Señal de reinicio del bloque.	1
datoTX	Bus de datos de la palabra a enviar.	8
enviar	Señal que indica que el dato se debe enviar.	1
Señal de salida	**Función**	**Número de bits**
TX	Señal de envío de datos serialmente.	1
listo	Señal que indica que el dato ha sido enviado.	1

Tabla 2.34. Parámetro y señales de entrada, y salida, del bloque transmisor serial.

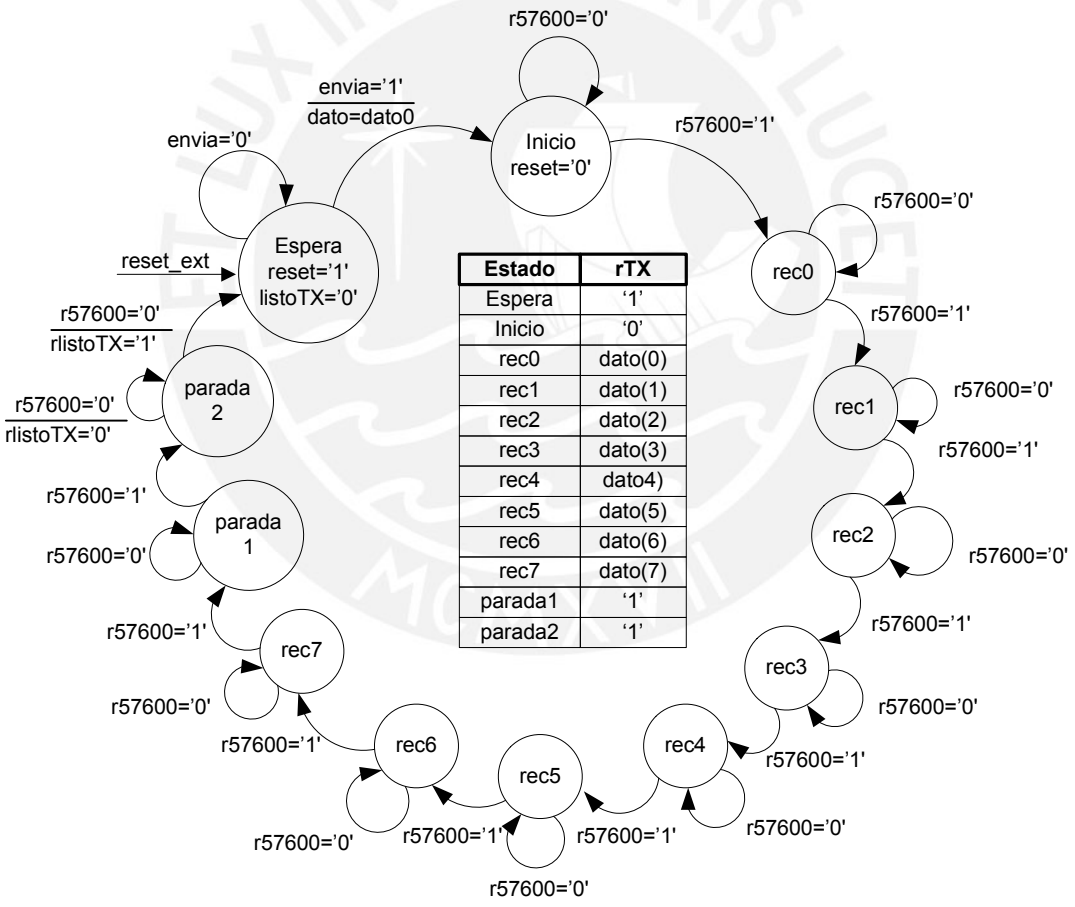


Figura 2.35. Máquina de estados del bloque transmisor serial.

2.18.3 Resultados independientes

El bloque fue compilado utilizando un *divisor* igual a 1157. El bloque diseñado ocupa solamente 63 LEs en el FPGA STRATIX EP1S25F672C6 utilizado,

alcanzando una frecuencia máxima de trabajo de 190.62MHz con una compilación optimizada en velocidad, mientras que 62 LEs y 190.62MHz con una optimización para el área a utilizar. La tabla 2.35 muestra los resultados obtenidos ante diferentes tipos de STRATIX, STRATIX II y de ACEX1K.

		ÁREA			VELOCIDAD		
	FPGA	LEs	%LEs	fmax (MHz)	LEs	%LEs	fmax (MHz)
Stratix	EP1S10F672-6	62	0.59	190.62	63	0.60	190.62
	EP1S10F672-7	62	0.59	180.96	63	0.60	180.96
	EP1S25F672-6	62	0.24	190.62	63	0.25	190.62
	EP1S25F672-7	62	0.24	180.96	63	0.25	180.96
	EP1S60F1508-6	62	0.11	190.62	63	0.11	190.62
	EP1S60F1508-7	62	0.11	180.96	63	0.11	180.96
Stratix II	EP2S15F484-3	46	0.29	311.33	46	0.29	311.33
	EP2S15F484-4	46	0.29	293.00	46	0.29	293.00
	EP2S15F484-5	46	0.29	268.24	46	0.29	268.24
	EP2S30F672-3	46	0.14	311.33	46	0.14	311.33
	EP2S30F672-4	46	0.14	293.00	46	0.14	293.00
	EP2S30F672-5	46	0.14	268.24	46	0.14	268.24
Acex	EP1K10QC208-1	62	10.76	161.81	62	10.76	182.48
	EP1K10QC208-2	62	10.76	141.24	62	10.76	154.32
	EP1K10QC208-3	62	10.76	108.93	62	10.76	120.77
	EP1K50FC484-1	62	2.15	173.01	62	2.15	189.39
	EP1K50FC484-2	62	2.15	137.36	62	2.15	147.49
	EP1K50FC484-3	62	2.15	108.93	62	2.15	117.92

Tabla 2.35. Resultados obtenidos usando distintos FPGAs para el bloque receptor serial.

2.19 Contador del proceso

2.19.1 Comportamiento funcional

Este bloque funciona de forma independiente a la unidad de control (ver sección 2.6) usando las señales de fin de proceso de los bloques: cargar datos y coeficientes del usuario (ver sección 2.8), escribir imagen analítica (ver sección 2.10), y escribir fase obtenida (ver sección 2.14). El bloque realiza la cuenta de ciclos transcurridos del proceso completo, además de almacenar la cuenta del proceso hasta obtener la imagen analítica.

2.19.2 Desarrollo

Debido a que mientras mayor es el número de bits en un contador, menor es la frecuencia de operación, se realizó la arquitectura utilizando cuatro contadores, cada de 8 bits, conectados en serie (ver figura 2.36), que son controlados por una máquina de estados.

Debido a la analogía a un cronómetro, se les llamó a las cuentas producidas por este contador: unidades, decenas, centenas y miles que son los datos *rTu, rTd, rTc* y *rTm*. Estas cuentas son almacenadas en un registro interno una vez que el proceso de escribir la imagen analítica ha finalizado mediante la señal *cargaH*. La máquina de estados (ver figura 2.37) posee cuatro estados controlados por las señales de entrada indicadas en la sección anterior: *inicio, contando1, contando2* y *fin*. De esta forma, el tiempo transcurrido viene dado por las ecuaciones 20 y 21. La tabla 2.36 muestra las señales de entrada y salida del bloque.

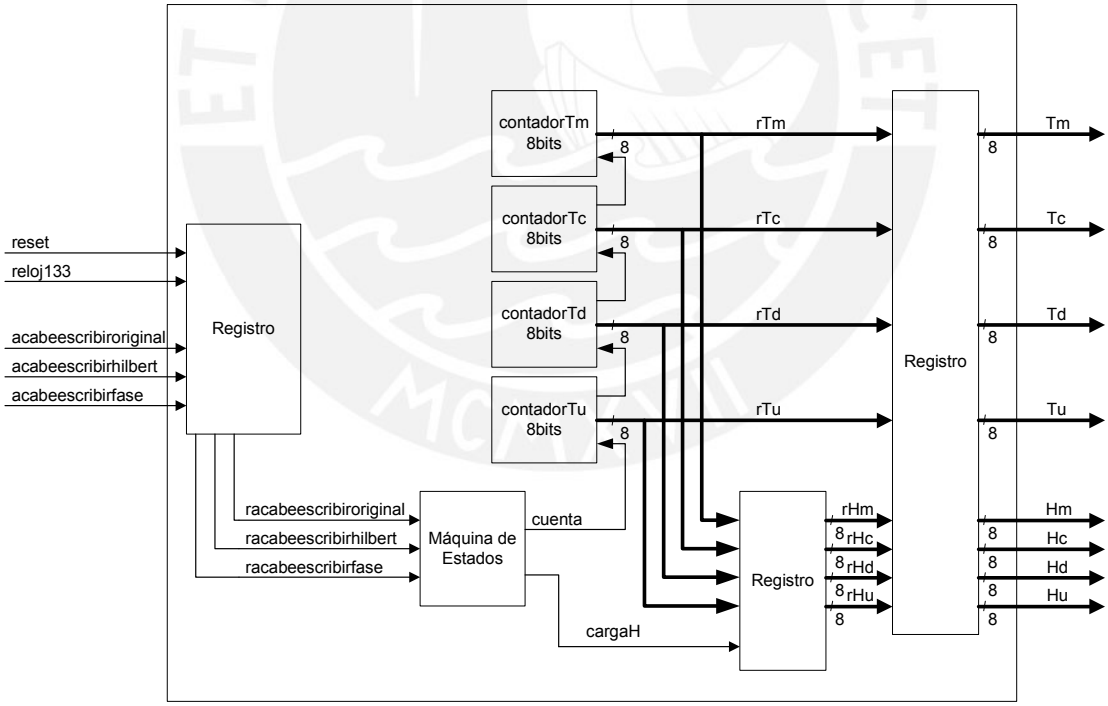


Figura 2.36. Bloque contador del proceso.

$$Tiempo\ total = \frac{Tu + Td \times 256 + Tc \times 256 \times 256 + Tm \times 256 \times 256 \times 256}{frecuencia\ de\ reloj\ del\ sistema} \qquad (20)$$

$$Tiempo\ Hilbert = \frac{Hu + Hd \times 256 + Hc \times 256 \times 256 + Hm \times 256 \times 256 \times 256}{frecuencia\ de\ reloj\ del\ sistema} \quad (21)$$

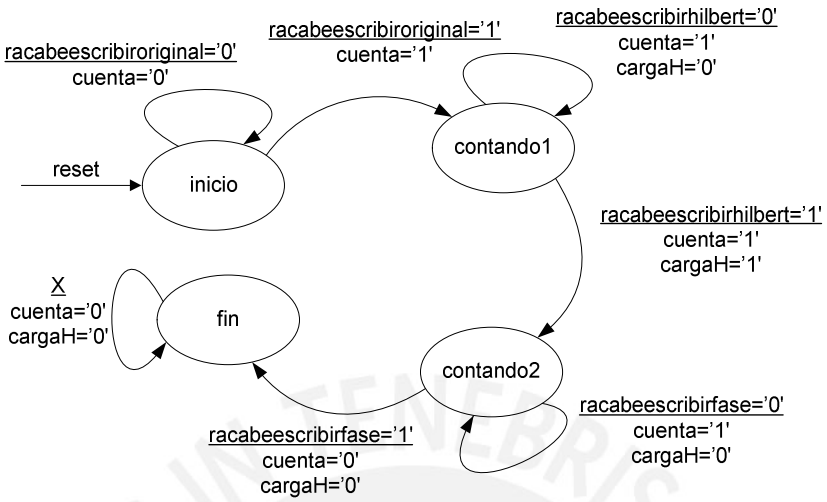


Figura 2.37. Máquina de estados del bloque contador del proceso.

Parámetro	**Función**	
csize	Indica el tamaño en bits de cada cuenta a enviar.	
Señal de entrada	**Función**	**Número de bits**
reloj133	Señal de sincronismo del bloque.	1
reset	Señal de reinicio del bloque.	1
acabeescribiroriginal	Señal que indica que el bloque cargar datos y coeficientes del usuario terminó de escribir los datos originales y cargar los coeficientes de Hilbert en el bloque convolución (ver sección 2.8).	
acabeescribirhilbert	Señal que indica que el bloque escribir imagen analítica terminó de escribir toda la imagen analítica en la memoria SDRAM (ver sección 2.10).	
acabeescribirfase	Señal que indica que el bloque escribir fase obtenida terminó de escribir toda la fase obtenida en la memoria SDRAM (ver sección 2.14).	
Señal de salida	**Función**	**Número de bits**
tiempoTu	Unidades del tiempo total empleado.	csize
tiempoTd	Decenas del tiempo total empleado.	csize
tiempoTc	Centenas del tiempo total empleado.	csize
tiempoTm	Miles del tiempo total empleado.	csize
tiempoHu	Unidades del tiempo empleado en la obtención de la imagen analítica.	csize
tiempoHd	Decenas del tiempo empleado en la obtención de la imagen analítica.	csize
tiempoHc	Centenas del tiempo empleado en la obtención de la imagen analítica.	csize
tiempoHm	Miles del tiempo empleado en la obtención de la imagen analítica.	csize

Tabla 2.36. Parámetros, señales de entrada y salida del bloque contador del proceso.

2.19.3 Resultados independientes

El bloque diseñado ocupa solamente 114 LEs en el FPGA STRATIX EP1S25F672C6 utilizado, alcanzando una frecuencia máxima de trabajo de 243.43MHz con una compilación optimizada en velocidad, mientras que 255.23MHZ con una optimización para el área a utilizar. La tabla 2.37 muestra los resultados obtenidos ante diferentes tipos de STRATIX, STRATIX II y de ACEX1K.

		ÁREA			VELOCIDAD		
	FPGA	LEs	%LEs	fmax (MHz)	LEs	%LEs	fmax (MHz)
Stratix	EP1S10F672-6	114	1.08	255.23	114	1.08	243.43
	EP1S10F672-7	114	1.08	236.35	114	1.08	233.10
	EP1S25F672-6	114	0.44	255.23	114	0.44	243.43
	EP1S25F672-7	114	0.44	236.35	114	0.44	233.10
	EP1S60F1508-6	114	0.20	255.23	114	0.20	243.43
	EP1S60F1508-7	114	0.20	236.35	114	0.20	233.10
Stratix II	EP2S15F484-3	107	0.69	368.46	107	0.69	368.46
	EP2S15F484-4	107	0.69	344.83	107	0.69	344.83
	EP2S15F484-5	107	0.69	313.38	107	0.69	313.38
	EP2S30F672-3	107	0.32	368.46	107	0.32	368.46
	EP2S30F672-4	107	0.32	344.83	107	0.32	344.83
	EP2S30F672-5	107	0.32	313.38	107	0.32	313.38
Acex	EP1K10QC208-1	114	19.79	250.00	114	19.79	250.00
	EP1K10QC208-2	114	19.79	200.00	114	19.79	200.00
	EP1K10QC208-3	114	19.79	188.68	114	19.79	185.19
	EP1K50FC484-1	114	3.96	250.00	114	3.96	250.00
	EP1K50FC484-2	114	3.96	200.00	114	3.96	200.00
	EP1K50FC484-3	114	3.96	166.67	114	3.96	166.67

Tabla 2.37. Resultados obtenidos usando distintos FPGAs para el bloque contador del proceso.

3. PRUEBAS Y ANÁLISIS DE RESULTADOS EXPERIMENTALES

3.1 Introducción

La arquitectura diseñada fue implementada y probada en la tarjeta de desarrollo MJL Stratix Development Kit [12] que cuenta con un FPGA STRATIX EP1S25F672C6 [3], esta tarjeta permite la comunicación con la PC, entre otros medios, a través del bus SERIAL. Las pruebas de obtención de la fase se realizaron mediante un programa simple desarrollado en C (ver sección ANEXOS); dicho programa configura el puerto serial a la velocidad deseada, luego envía los datos de la forma:

i. Envío de los coeficientes del usuario. El envío se realiza del byte menos significativo al byte más significativo. Primero se envía la parte real del coeficiente y luego la parte imaginaria.
ii. Envío de los 3 bytes que contiene la información respecto al tamaño de la imagen. Se envía desde el byte menos significativo al byte más significativo.
iii. Envío de los 3 bytes que contiene la información respecto al número de columnas de la imagen. Se envía desde el byte menos significativo al byte más significativo.
iv. Envío de los 3 bytes que contiene la información respecto al número de filas de la imagen. Se envía desde el byte menos significativo al byte más significativo.
v. Envío de los datos, píxel por píxel (que es lo mismo, para este caso, byte por byte). La forma de envío se realiza desde la primera fila, columna por columna, hasta la última fila.

Luego de enviarse la información, el programa recepciona los resultados obtenidos. Debido a que la fase obtenida tiene un tamaño de palabra de 16 bits (ver secciones 2.14 y 2.15), se recepciona primero el byte más significativo y luego el menos significativo.

En la comprobación de los resultados obtenidos por el sistema, se utilizó el programa MATLAB, de manera que no sólo se supiera si los resultados obtenidos son correctos, sino también para comparar los tiempos de procesamiento entre dicho programa y la arquitectura implementada.

3.2 Resultados de la arquitectura completa usando Quartus II

La arquitectura completa fue compilada y sintetizada utilizando el programa Quartus II v.4.0 configurándose para utilizar 7 coeficientes obteniéndose los siguientes resultados para el FPGA STRATIX EP1S25F672C6:

Elementos Lógicos utilizados: 23171 (90.30%)
Pines utilizados: 59 (12.45%)
Bits de memoria utilizados: 2000 (0.1%)
DSPs de 9 bits utilizados: 56 (70%)
PLLs utilizados: 1 (16%)
Frecuencia máxima de trabajo: 167.17MHz

Se realizó el análisis y síntesis de la arquitectura para diferentes tipos de STRATIX, además de una comparación con distintos tipos de STRATIX II (ver tabla 3.1 y figuras 3.1, 3.2 y 3.3). Cabe señalar que los resultados obtenidos utilizando el análisis y síntesis son diferentes a los obtenidos finalmente luego de la compilación completa debido a optimizaciones que se realizan al momento de crear el archivo final de configuración del FPGA (ver capítulo 4).

		Optimización: ÁREA						
	FPGA	LEs	%LEs	fmax (MHz)	bits	%bits	DSPs 9bits	%DSPs 9bits
Stratix	EP1S25F672-6	24095	93.90	93.26	2000	0.10	56	70.00
	EP1S25F672-7	24095	93.90	89.73	2000	0.10	56	70.00
	EP1S60F1508-6	24095	42.18	93.26	2000	0.04	56	38.89
	EP1S60F1508-7	24095	42.18	89.73	2000	0.04	56	38.89
Stratix II	EP2S30F672-3	19159	56.55	96.76	3008	0.22	56	43.75
	EP2S30F672-4	19159	56.55	93.76	3008	0.22	56	43.75
		Optimización: VELOCIDAD						
	FPGA	LEs	%LEs	fmax (MHz)	bits	%bits	DSPs 9bits	%DSPs 9bits
Stratix	EP1S25F672-6	24159	94.15	141.41	2000	0.10	56	70.00
	EP1S25F672-7	24159	94.15	135.87	2000	0.10	56	70.00
	EP1S60F1508-6	24159	42.30	141.41	2000	0.04	56	38.89
	EP1S60F1508-7	24159	42.30	135.87	2000	0.04	56	38.89
Stratix II	EP2S30F672-3	19159	56.55	97.32	3008	0.22	56	43.75
	EP2S30F672-4	19159	56.55	95.00	3008	0.22	56	43.75

Tabla 3.1. Resultados luego del análisis y síntesis.

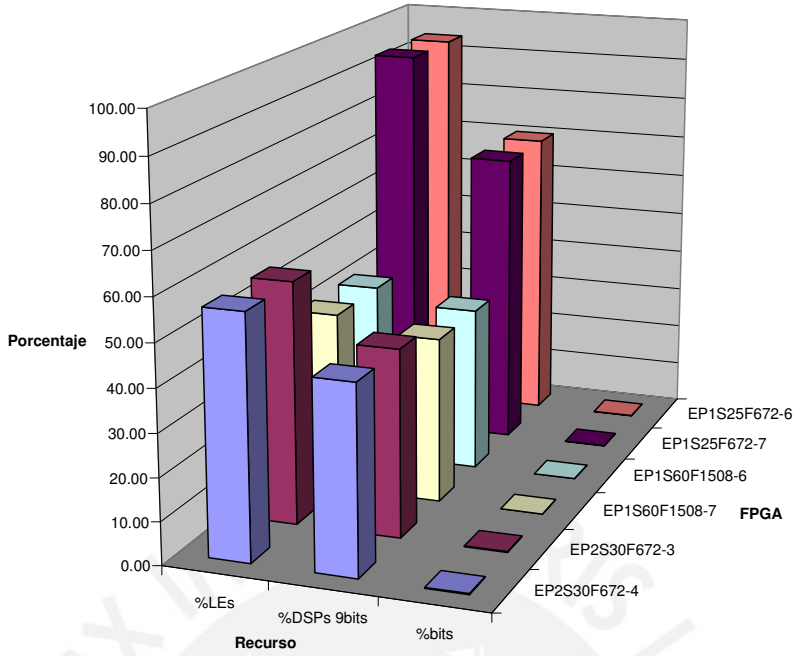


Figura 3.1. LEs, DSPs y bits utilizados usando un optimización en ÁREA según el FPGA.

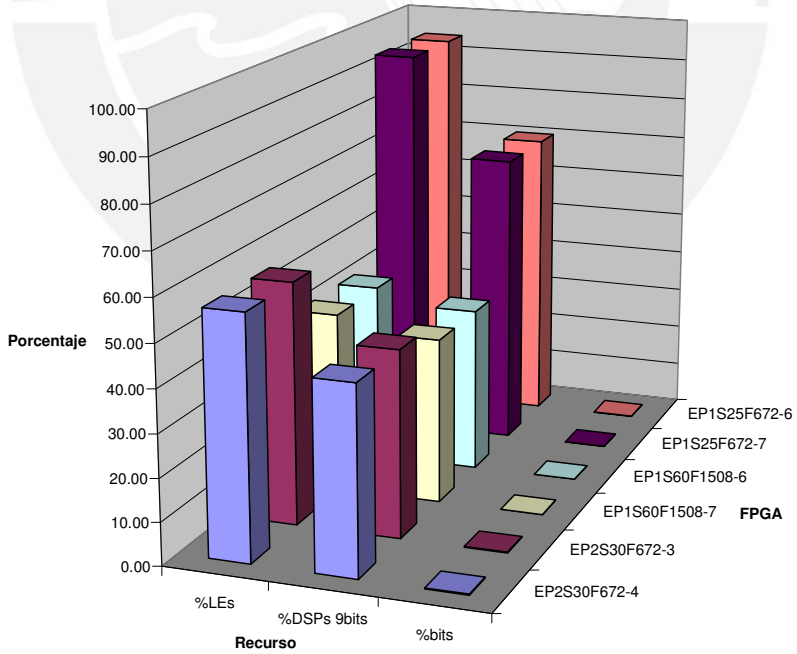


Figura 3.2. LEs, DSPs y bits utilizados usando un optimización en VELOCIDAD según el FPGA.

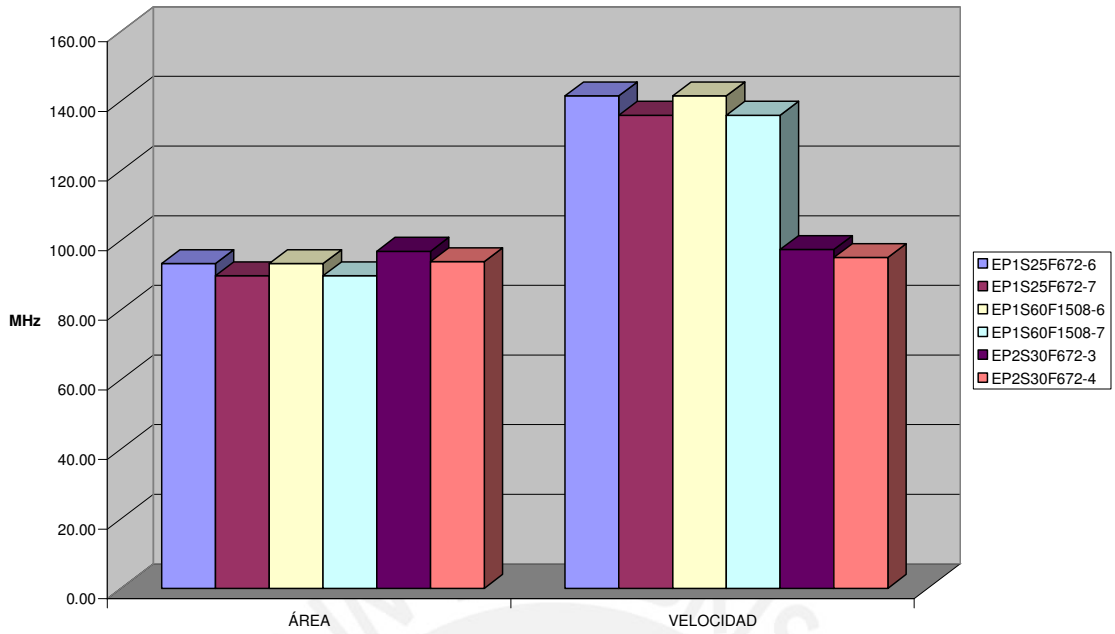


Figura 3.3. Frecuencia máxima según el tipo de optimización y el FPGA utilizado.

3.3 Resultados obtenidos

Las pruebas se realizaron transfiriendo matrices de datos aleatorios de distintas dimensiones como si se trataran de imágenes para comparar los datos obtenidos en el sistema desarrollado con los resultados obtenidos utilizando Matlab con el objetivo de conocer el error relativo (ver ecuación 22) en la obtención de la fase obtenida y con eso conocer el error relativo que se obtendrá al utilizar la fase para obtener la imagen con uno, dos y tres armónicos (ver sección 1.3). La tabla 3.2 muestra los errores relativos obtenidos.

	ERROR RELATIVO PROMEDIO (%)	ERROR RELATIVO MÍNIMO (%)	ERROR RELATIVO MÁXIMO (%)
FASE	1.1642	0	2.6763
CON 1 ARMÓNICO	1.125	0	2.634
CON 2 ARMÓNICOS	1.12	0	2.6032

Tabla 3.2. Errores relativos obtenidos luego de las pruebas.

$$error\ relativo = \left| \frac{dato\ obtenido - dato\ correcto}{dato\ correcto} \right| \times 100\% \qquad (22)$$

Al obtener los tiempos de procesamiento experimentales, ocurrieron problemas con la tarjeta utilizada. Al realizarse pruebas independientes con sistemas simples con dicha tarjeta, se encontraron errores originados por el FPGA, por lo que se deduce que ese es el motivo por el cual no se hayan procesado correctamente todas las imágenes procesadas. Al no contar con otra tarjeta similar, no se pudo realizar nuevas pruebas experimentales. La figura 3.4 muestra una comparación entre la recuperación de la imagen utilizando cinco armónicos con la fase obtenida mediante Matlab y con la fase obtenida en la arquitectura desarrolla, mientras que la tabla 3.3 muestra los tiempos de procesamiento experimentales obtenidos tanto con la arquitectura desarrollada (utilizando la frecuencia de trabajo de 133.33MHz y las ecuaciones 20 y 21) como con el algoritmo implementado en Matlab (ver PRUEBAFINAL.M en sección Anexos). El algoritmo en Matlab fue implementado en una PC Pentium IV de 3.06GHz con 256MB de memoria RAM.

Figura 3.4. Recuperación de la imagen utilizando 5 armónicos. Derecha: Fase obtenida en Matlab. Izquierda: Fase obtenida en el FPGA.

Dimensiones				Tiempo (ms)		
Tamaño	Filas	Columnas	Medio	proceso total	imagen analítica	sin imagen analítica
32	2	16	MATLAB	125.00	0.00	125.00
			Arquitectura	0.19	0.01	0.19
32	4	8	MATLAB	125.00	0.00	125.00
			Arquitectura	0.12	0.01	0.11
64	8	8	MATLAB	219.00	0.00	219.00
			Arquitectura	0.16	0.01	0.14
4096	64	64	MATLAB	11016.00	47.00	10969.00
			Arquitectura	4.55	0.45	4.10
8192	64	128	MATLAB	22828.00	47.00	22781.00
			Arquitectura	8.95	0.82	8.13
16384	128	128	MATLAB	42922.00	109.00	42813.00
			Arquitectura	*	*	*
65536	256	256	MATLAB	181437.00	1406.00	180031.00
			Arquitectura	63.47	6.30	57.17
131072	256	512	MATLAB	376343.00	4312.00	372031.00
			Arquitectura	126.37	12.32	114.06
262144	512	512	MATLAB	778297.00	13344.00	764953.00
			Arquitectura	*	*	*
(*) error en prueba						

Tabla 3.3. Tiempos experimentales obtenidos.

Se observa que los tiempos obtenidos utilizando la arquitectura desarrollada son mucho menores a los que se demora el programa Matlab. Lamentablemente, no se pudo realizar pruebas en otra tarjeta similar, pero con los tiempos obtenidos y el modelo de la arquitectura desarrollada, se puede hacer una estimación del tiempo de procesamiento dependiente del número de filas (n), de columnas (m), y del tamaño total ($n \times m$) (ver ecuaciones 23 y 24). De esta forma, el tiempo estimado de procesamiento, utilizando los 7 coeficientes indicados en la sección 3.2, queda definido por la ecuaciones 25 y 26.

$$tiempo\ total\ (\mu s) \approx a_T n + b_T m + c_T (n \times m) \quad (23)$$

$$tiempo\ imagen\ analítica\ (\mu s) \approx a_H n + b_H m + c_H (n \times m) \quad (24)$$

donde:

a_T proporción de las filas de la imagen para el tiempo total.

b_T proporción de las columnas de la imagen para el tiempo total.

c_T proporción del tamaño de la imagen para el tiempo total.

a_H proporción de las filas de la imagen para el tiempo de la imagen analítica.

b_H proporción de las columnas de la imagen para el tiempo de la imagen analítica.

c_H proporción del tamaño de la imagen para el tiempo de la imagen analítica.

$$tiempo\ total\ (\mu s) \approx 2.226925n + 9.841431m + 0.921367(n \times m) \quad (25)$$

$$tiempo\ imagen\ analítica\ (\mu s) \approx 1.113778n + 0.007961m + 0.091762(n \times m) \quad (26)$$

Con estas estimaciones, los tiempos estimados con un imagen de 512x512 píxeles son:
Tiempo total ≈ 247.7099 ms
Tiempo sin obtención de imagen analítica ≈ 223.0807 ms

Debido a la poca literatura que existe que den información acerca del tiempo de procesamiento utilizando otros métodos, se utilizó la comparación con el algoritmo utilizando el programa Matlab indica antes. Con la estimación realizada anteriormente, se puede comparar con los resultados obtenidos por el Ing. Paul Rodríguez en las pruebas de la tesis doctoral mencionada en la sección Introducción [1]. Los siguientes datos fueron obtenidos utilizando programación en SIMD en una PC Pentium IV con HT de 3.06GHz con 1GB de memoria RAM:

- Imagen de 512x512 píxeles utilizando 11 coeficientes: 54.69131 ms.
- Imagen de 512x512 píxeles utilizando 21 coeficientes: 79.46997 ms.
- Imagen de 512x512 píxeles utilizando 31 coeficientes: 129.81801 ms.

Lo que indica que la arquitectura desarrolla obtiene los datos en un tiempo mayor en 5 veces que los algoritmos desarrollados utilizando SIMD.

4. CONCLUSIONES

- La arquitectura diseñada mostró una alta utilización tanto de espacio (LEs) como de bloques DSPs que posee el FPGA STRATIX EP1S25F672C6, que fue del 90.30% y 70% respectivamente. El uso de una memoria externa es necesario debido al almacenamiento temporal de resultados parciales.

- El valor mínimo de la frecuencia máxima de trabajo que se puso como objetivo (135MHz) fue superada, alcanzando los 167.17MHz en el FPGA utilizado.

- Es la primera vez a nivel mundial, según se ha buscado información, que se desarrolla en un FPGA una arquitectura relacionada a la demodulación AM-FM, lo que abre un camino de posibilidades para continuar el trabajo y/o para presentarlo en diversos congresos relacionados al tema.

- El uso de los bloques DSPs de los FPGAs STRATIX ha demostrado un alto desempeño para procesos de multiplicación, lográndose resultados a altas velocidades y sin utilizar LEs. Se concluye que todo sistema que requiera el uso de multiplicadores en FPGAs de la familia STRATIX debe utilizar estos bloques.

- El uso de los PLLs con que cuentan los FPGAs STRATIX permite alcanzar frecuencias estables de oscilación de valores múltiplos de la frecuencia de oscilación de entrada, con esto es posible obtener altas frecuencias de señales de reloj con cristales externos de bajas frecuencias. Además, como ya se mencionó, el utilizar el PLL para generar la misma frecuencia inclusive presenta un mejor desempeño ya que la señal resultante es más estable.

- Si bien el tiempo en la obtención del resultado es mucho menor que al transcurrido en la obtención utilizando Matlab, hay un procedimiento utilizando SIMD que lo obtiene en un tiempo 5 veces más rápido. Esto implica que utilizando un STRATIX de mayor frecuencia de trabajo y una memoria más rápida, no sólo se podría utilizar un mayor número de coeficientes en los filtros, sino que el tiempo de procesamiento puede superar el tiempo máximo actual.

- El análisis y síntesis de la arquitectura utilizando el FPGA STRATIX II muestra que las frecuencias de trabajo son altamente superadas en la mayoría de bloques, pero no es así en el bloque arcotangente lo que implica que este último bloque debe ser adaptado de una manera distinta si se desea utilizar dicho FPGA.

- El análisis y síntesis de la arquitectura desarrollada utilizando el programa Quartus II v. 4.0 presenta un estimado tanto en la frecuencia de operación como en el número de celdas lógicas utilizadas. Las cifras reales obtenidas luego de la compilación total muestran una mayor eficiencia en el consumo de LEs y un gran aumento de la frecuencia máxima de trabajo. La mejora se debe a optimizaciones en la configuración que se le da al FPGA cuando se requiere configurarlo físicamente.

- La arquitectura diseñada funcionó en simulación, pero no funcionó como era esperado una vez implementado. Luego de la prueba independiente mencionada en el capítulo 3, se concluye que el FPGA de la tarjeta utilizada presenta problemas.

- Los bloques encargados de la transmisión y recepción serial funcionaron correctamente y pueden ser utilizados como bloques independendientes en cualquier sistema implementado en un FPGA. En ese sentido, el bloque fue entregado y utilizado por los alumnos del curso “Procesamiento Digital de Señales” (de la especialidad Ingeniería Electrónica) en su proyecto “Implementación de un Filtro Digital eliminador de ruido de línea de una señal de voz usando el FPGA Stratix” con grandes resultados.

- El bloque convolución ocupa un elevado número de LEs en el FPGA utilizado (28.88%) debido a los sumadores involucrados en él, pero trabaja a una alta velocidad logrando dar el resultado en 144.54ns luego de ingresar el primer dato, lo que lo hace útil para procesamiento de imágenes que requieren respuesta rápida. Además, el hecho que el bloque es amigable lo hace utilizable en una manera sencilla sin tener un alto conocimiento del tema. Actualmente (julio del 2004), el

trabajo referente a este bloque se encuentra en la segunda etapa de evaluación de proyectos del XI Intercon 2004.

- El controlador de la memoria SDRAM (memoria incluida en la tarjeta de desarrollo utilizada) es amigable, fácil de usar y no ocupa muchos LEs, además que permite su configuración del tipo de escritura o lectura en forma sencilla. Es decir, el bloque queda como aporte, al igual que los dos bloques mencionados en párrafos anteriores, para ser utilizada en cualquier aplicación que requiera usar dicha memoria.

- El bloque arcotangente ocupa el mayor número de LEs de la arquitectura (33.31%). Modificar este bloque para un menor consumo de LEs es posible pero implica una menor frecuencia de trabajo, con lo que el *pipeline* deseado para el sistema no sería posible a la frecuencia de trabajo utilizada (133MHz). Por otro lado, el bloque puede ser utilizado por cualquier sistema que requiera el procesamiento de datos a una frecuencia menor a 140MHz. Para trabajar con una mayor resolución de datos y a altas velocidades, se debería implementar el algoritmo CORDIC en un FPGA independiente.

- El trabajo desarrollado permitió experimentar por primera vez el uso de FPGAs de la familia STRATIX en la Pontificia Universidad Católica del Perú. Debido a los satisfactorios resultados que se alcanzaban parcialmente, el Grupo de Microelectrónica (GuE) y el Grupo de Procesamiento Digital de Señales e Imágenes (GPDSI) han adquirido, cada uno, una nueva tarjeta de desarrollo basada en dicha familia de FPGA para el desarrollo de dos proyectos auspiciados por la Dirección Académica de Investigación (DAI). Estas tarjetas presentan mejores prestaciones que la utilizada, siendo similares sólo en el FPGA.

5. RECOMENDACIONES

- El bloque convolución implementado utilizó 7 coeficientes, siendo el límite 10 coeficientes debido al número de bloques DSP con que cuenta el FPGA STRATIX EP1S25F672C6. En la obtención de la imagen analítica se utilizó ese número de coeficientes, pero el resultado que se obtiene pierde resolución debido a que un número óptimo de coeficientes en la obtención de la transformada de Hilbert implicada es como mínimo 35 coeficientes [1]. Para lograr dicha cantidad de coeficientes se puede utilizar dos STRATIX EP1S60 con lo que se puede obtener 36 coeficientes siempre y cuando no se desea utilizar LEs. Otra solución es implementar un bloque que utilizando el bloque ya desarrollado, realice la convolución de la forma que se realiza en la programación en SIMD [1], con lo que si bien la frecuencia de trabajo no disminuiría, el tiempo en obtener el dato aumentaría.

- Este trabajo ha sido desarrollado para la obtención de la fase, en una segunda etapa, se puede desarrollar las etapas que continúan en la obtención de la amplitud estimada [2] basándose en los resultados obtenidos. Una vez desarrollado ese bloque, como una tercera etapa se puede obtener las múltiples componentes AM-FM [2].

- Dada la alta velocidad de procesamiento del sistema, utilizando una interfase PC-FPGA de alta velocidad se puede procesar imágenes a altas velocidades en tiempo real. Posibilidades para este tipo de transferencia son: desarrollar un controlador para la salida USB de la tarjeta, desarrollar un controlador para la salida Ethernet de la tarjeta, o acoplar el sistema en las nuevas tarjetas adquiridas por GuE o GPDSI de modo que se haga la transferencia mediante el bus PCI de la PC.

- Como se mencionó en el capítulo anterior, el desarrollo del algoritmo CORDIC para hallar la raíz cuadrada y las funciones arcotangente, seno, coseno y exponencial en un FPGA aislado, sería de gran utilidad para este trabajo y para otros que se requieren el campo del Procesamiento Digital de Señales e Imágenes.

- La tarjeta de desarrollo adquirida presentó muchos problemas en cuanto al control de sus buses y dispositivos de memoria, por lo que en caso de adquirir una nueva tarjeta es recomendable que se asegure la inclusión de dichos controladores, tal como ha ocurrido con las nuevas tarjetas adquiridas.

- La arquitectura implementada físicamente no funcionó como se esperaba por problemas indicados en los capítulos 3 y 4. Se recomienda utilizar el FPGA de esa tarjeta de manera que el número de LEs utilizados no esté cerca al total de LEs disponibles. Además, se recomienda no sólo utilizar registros de entrada y salida de datos como se ha hecho, sino que las señales generadas tengan la opción de control de datos, ya sea mediante la visualización de éstas o mediante su almacenamiento.

REFERENCIAS

[1] Paul Rodríguez Valderrama, *"Fast AM-FM Demodulation with Applications to Real-time Image and Motion Mode Video Analysis"*, Ph.D. thesis proposal, 2003, The University of New Mexico, USA.

[2] Joseph P. Havlicek, David S. Harding, and Alan C. Bovick, *"The multicomponent AM-FM Image Representation",* IEEE Transactions on Image Processing, Vol 5, No. 6, June 1996.

[3] Altera Corporation, *"Stratix Device Handbook"*, 2004.

[4] Marios S. Pattichis, George Panayi, Alan C. Bovick and Shun-Pin Hsu, *"Fingerprint Classification Using an AM-FM Model",* IEEE Transactions on Image Processing, Vol. 10, No. 6, June 2001 Pages: 951 – 954.

[5] Joseph P. Havlicek, David S. Harding, and Alan C. Bovick, *"The Analytic Image",* 1997 International Conference on Image Processing (ICIP '97) 3-Volume Set-Volume 2. October 26 - 29, 1997.

[6] Oppenheim, Alan V., *"Discrete-time signal processing"*, Eglewood Cliffs, NJ. : Prentice-Hall, 1989. Capítulo 11.

[7] Marios S. Pattichis, *"Least Squares FIR Filter Design Using Frequency Domain Piecewise Polynomial Approximations",* Proceedings of X European Signal Processing Conference, Tampere, Finland, September 5-8, 2000.

[8] Hussain Zahid, "*Digital Image Processing: Practical applications of parallel processing techniques*", Ellis Horwood Limited, 1991.

[9] A. L. Abbott, R. M. Haralick, and X. Zhuang, "*Pipeline Architectures for Morphologic Image Analysis*", Machine Vision and Applications 1 (1); 23-40.

[10] Altera Corporation, *"Stratix II Device Handbook"*, 2004.

[11] Altera Corporation, *"ACEX 1K: Programmable Logic Device Family"*, May. 2003, ver. 3.4.

[12] MJL Technology, Ltd., "*MJL Stratix Development Board",* December 2002, ver. 1.0.

[13] Joseph P. Havlicek, John W. Havlicek, Ngao D. Mamuya and Alan C. Bovik, *"Skewed 2D Hilbert Transforms and Computed AM-FM Models"*,

1998 International Conference on Image Processing, 1998. ICIP 98. Proceedings, Volume: 1, 4-7 Oct. 1998 Pages:602 - 606 vol.1.

[14] Pattichis, M.S.; Pattichis, C.S.; Avraam, M.; and Bovik, A.; Kyriacou, K., "*AM-FM Texture Segmentation in Electron Microscopic Muscle Imaging*", IEEE Transactions on Medical Imaging, Volume: 19, Issue: 12 , Dec. 2000.Pages:1253 – 1257.

[15] Tangsukson, T. and Havlicek, J.P., "*AM-FM Image Segmentation*", 2000 International Conference on Image Processing Proceedings., Volume: 2 , 10-13 Sept. 2000 Pages:104 - 107 vol.2.

[16] Ray, N.; Havlicek, J.; Acton, S.T.; and Pattichis, M., "*Active Contour Segmentation Guided by AM-FM Dominant Component Analysis*", 2001 International Conference on Image Processing, 2001. Proceedings, Volume: 1 , 7-10 Oct. 2001 Pages:78 - 81 vol.1.

[17] Joseph P. Havlicek, John W. Havlicek, Ngao D. Mamuya and Alan C. Bovik, "*Skewed 2D Hilbert Transforms and Computed AM-FM Models*", 1998 International Conference on Image Processing, 1998. ICIP 98. Proceedings, Volume: 1 , 4-7 Oct. 1998 Pages:602 - 606 vol.1.

[18] V. Katkovnik and L. Stankovic, *"Instantaneous Frequency Estimation Using the Wigner Distribution with Varying and Data-Driven Window Length"*, IEEE Transactions on Signal Processing, Vol. 46, No. 9, September 1998.

[19] Paulo M. Oliveira and Victor Barroso, "*Instantaneous Frequency of Multicomponent Signals*", IEEE Signal Processing Letters, Vol. 6, No. 4, April 1999.

[20] M. S. Pattichis "*AM-FM Transform with Applications*" Ph.D. diss. The University of Texas at Austin 1998.

[21] J. P. Havlicek, "*AM-FM Image Models*" Ph.D. diss., The University of Texas at Austin, 1996. J. P. Havlicek, A. C. Bovik Ch. 4.4 "Handbook of Image and Video Processing" Academic Press 2000.

[22] A. C. Bovik, N. Gopal, T. Emmoth and A. Restrepo, "*Localized measurement of emergent image frequencies by Gabor wavelets",* IEEE Transactions Inform. Theory, Vol. 38, No. 2, Mar. 1992. Pages: 691 - 712.

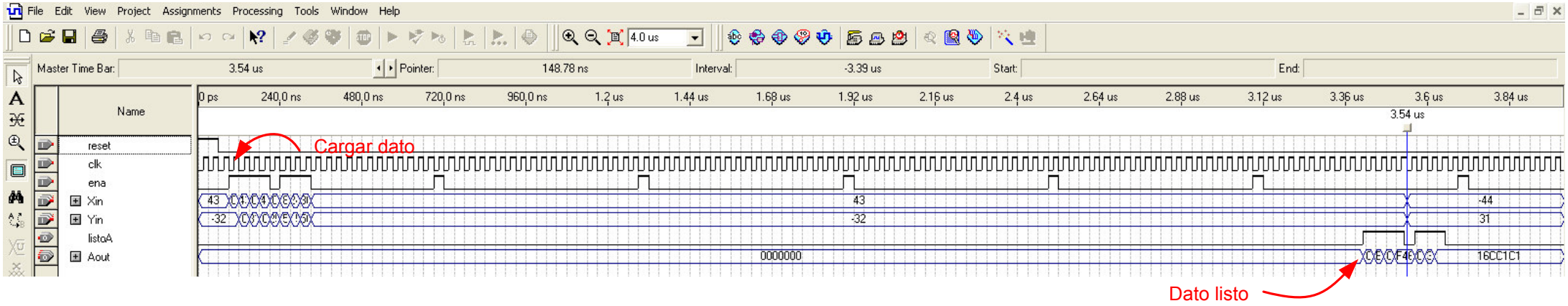
[23] Jordán Vitella, "*Implementación de una Arquitectura para un filtro morfológico de Imagenes Digitales en Escala de Grises en un FPGA de Altera",* Tesis de Licenciatura, Pontificia Universidad Católica del Perú, 2003.

[24] Integrated Device Technology, Inc., "*3.3V CMOS Static RAM 1 Meg (64K x 16-Bit) IDT71V016SA Datasheet*", June 2002.

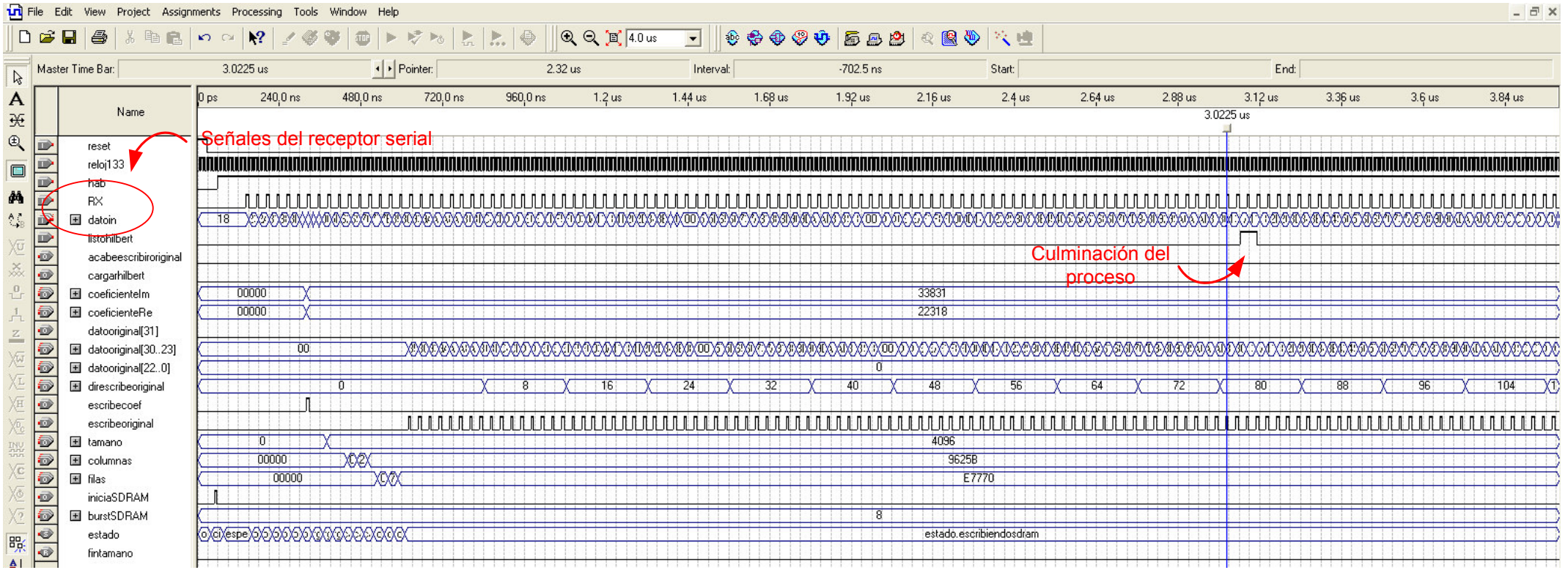
[25] Micron Technology, Inc., "*64Mb: x32 SDRAM MT48LC2M32B2 Datasheet*", Mar. 2004.

[26] Altera Corporation, *"SDR SDRAM Controller"*, August 2002.

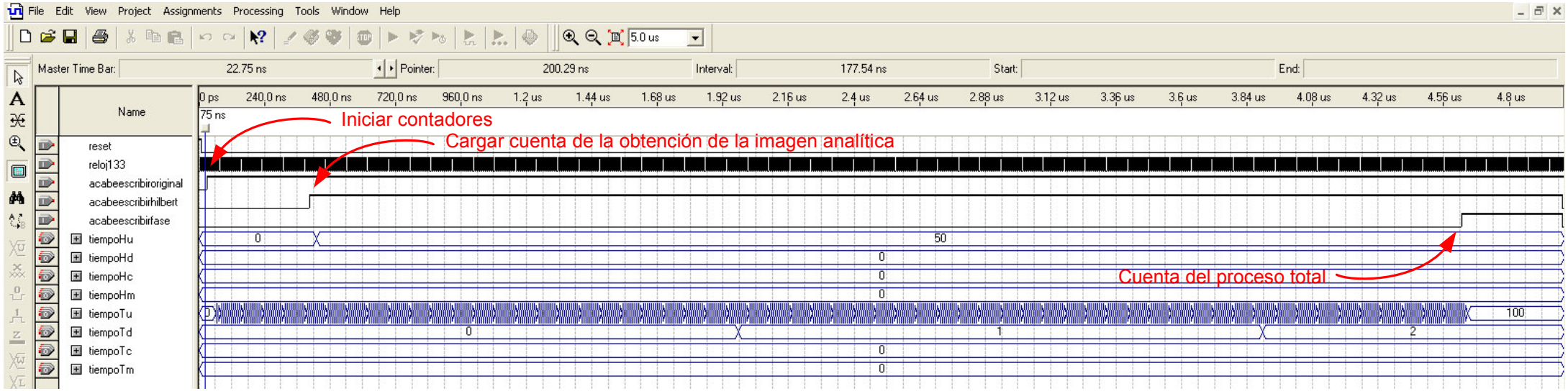
Arcotangente



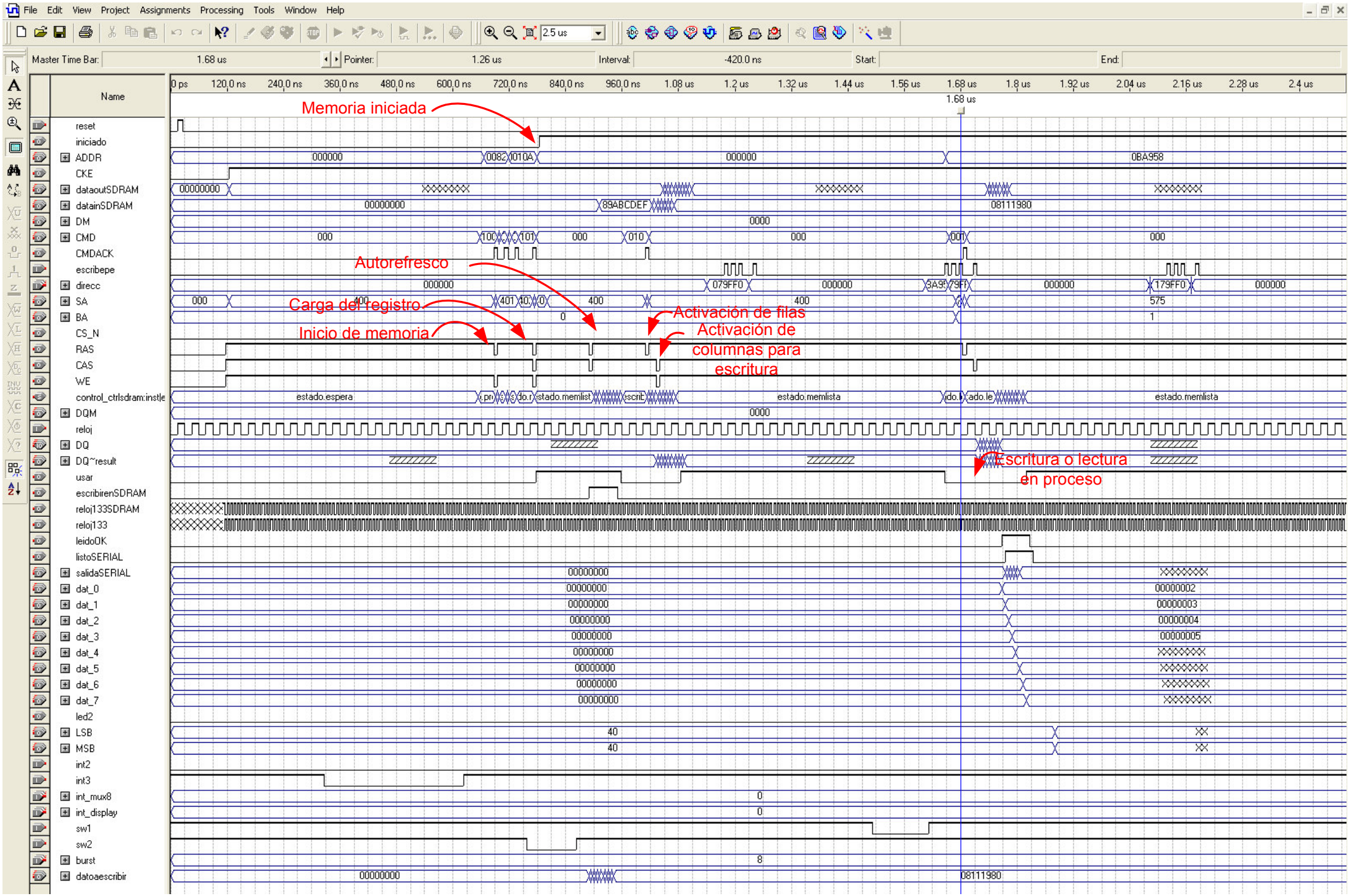
Cargar datos y coeficientes del usuario



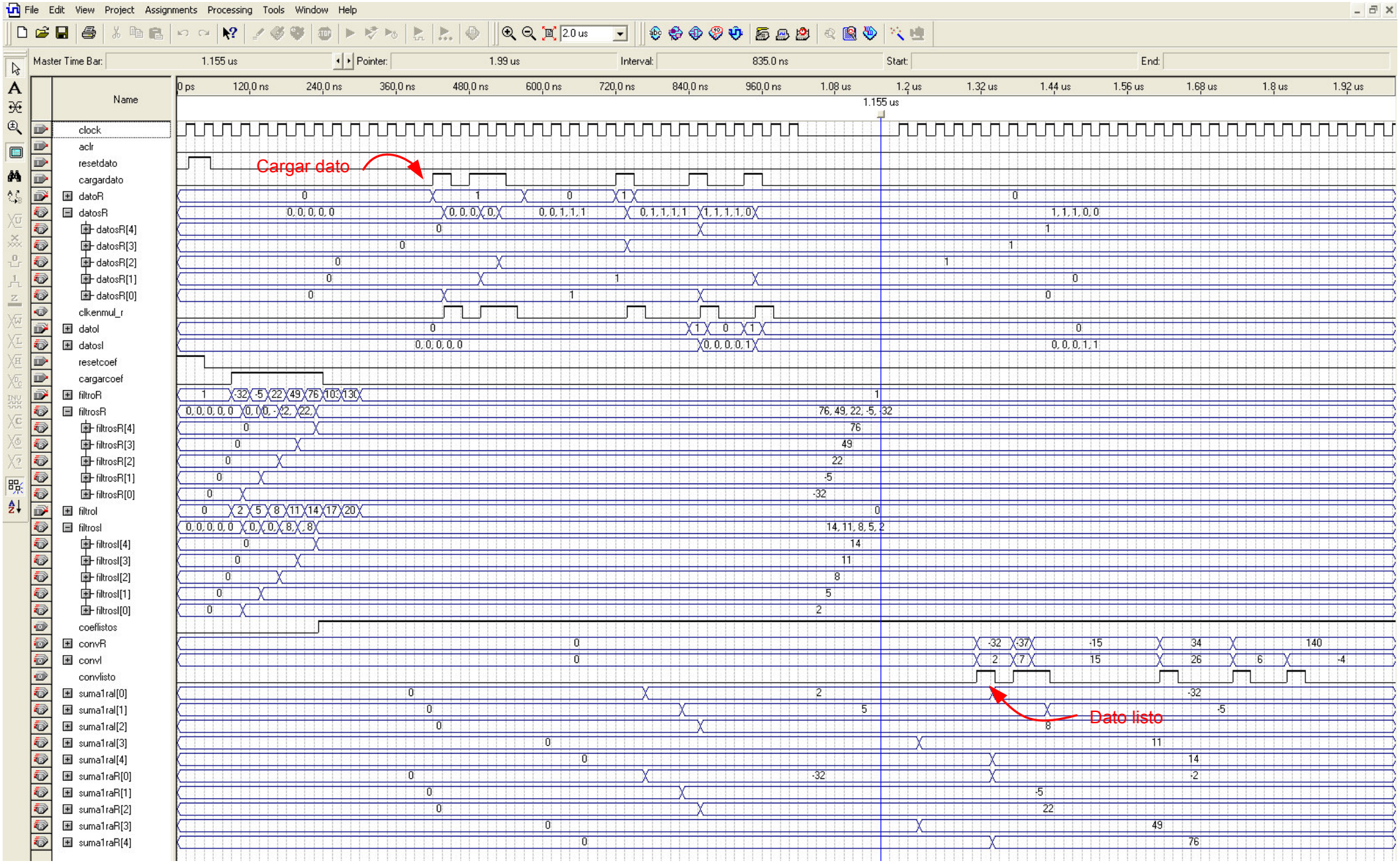
Contador del proceso



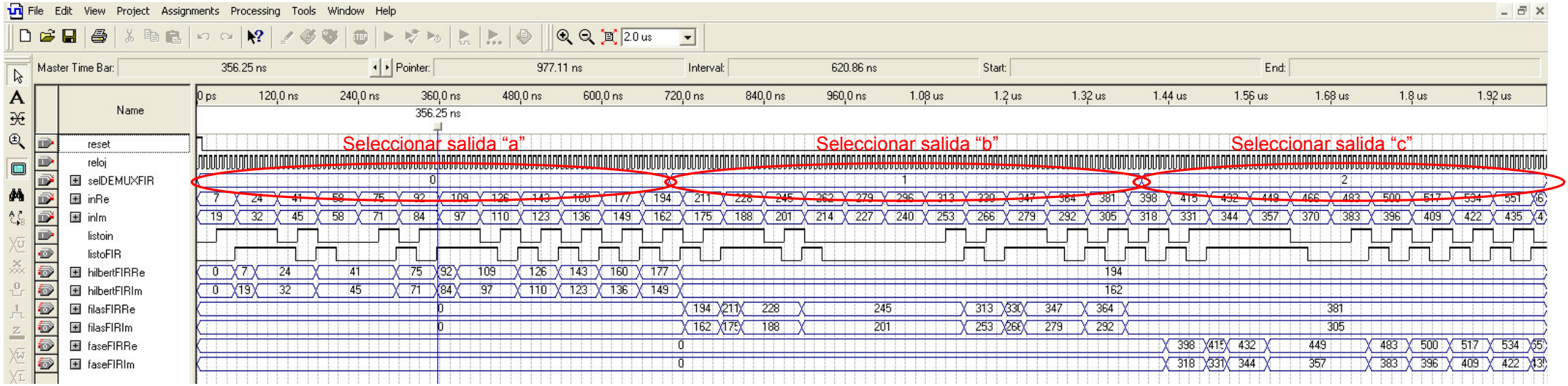
Controlador de la memoria SDRAM



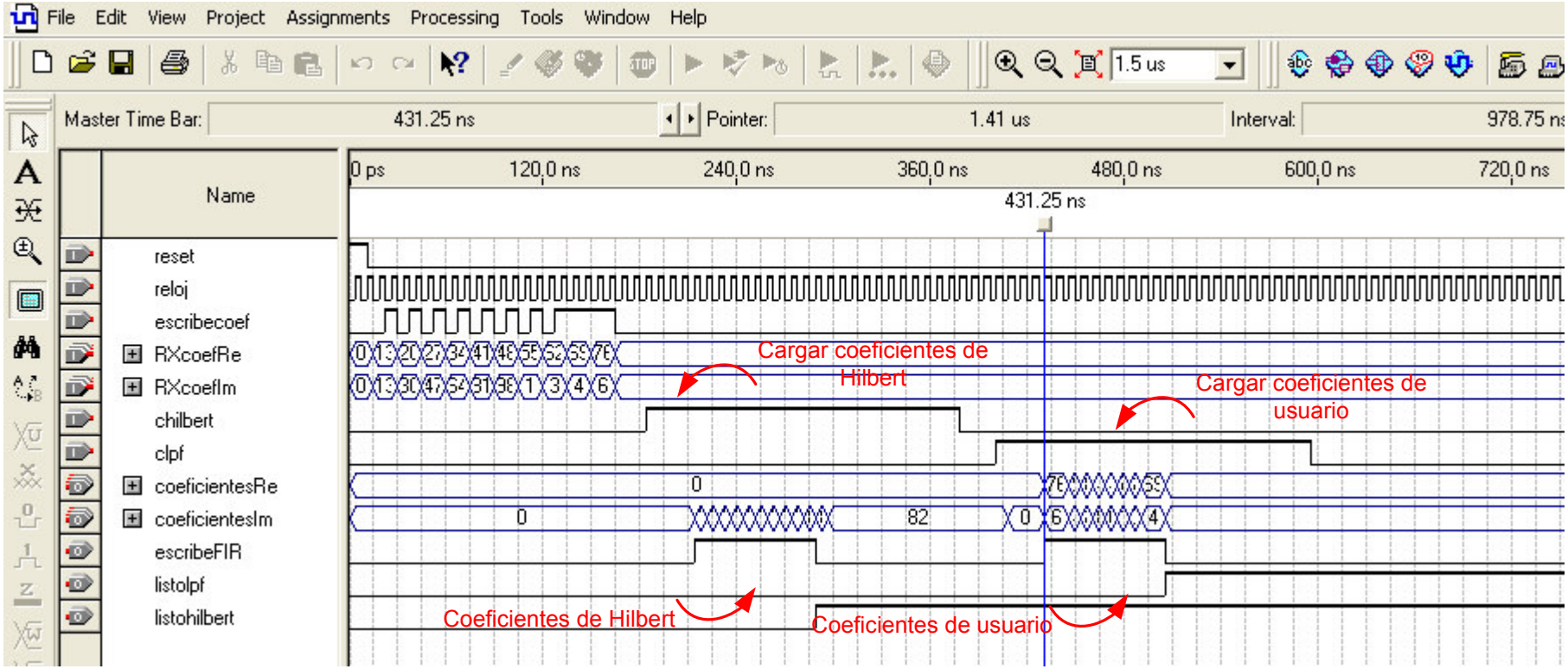
Convolución



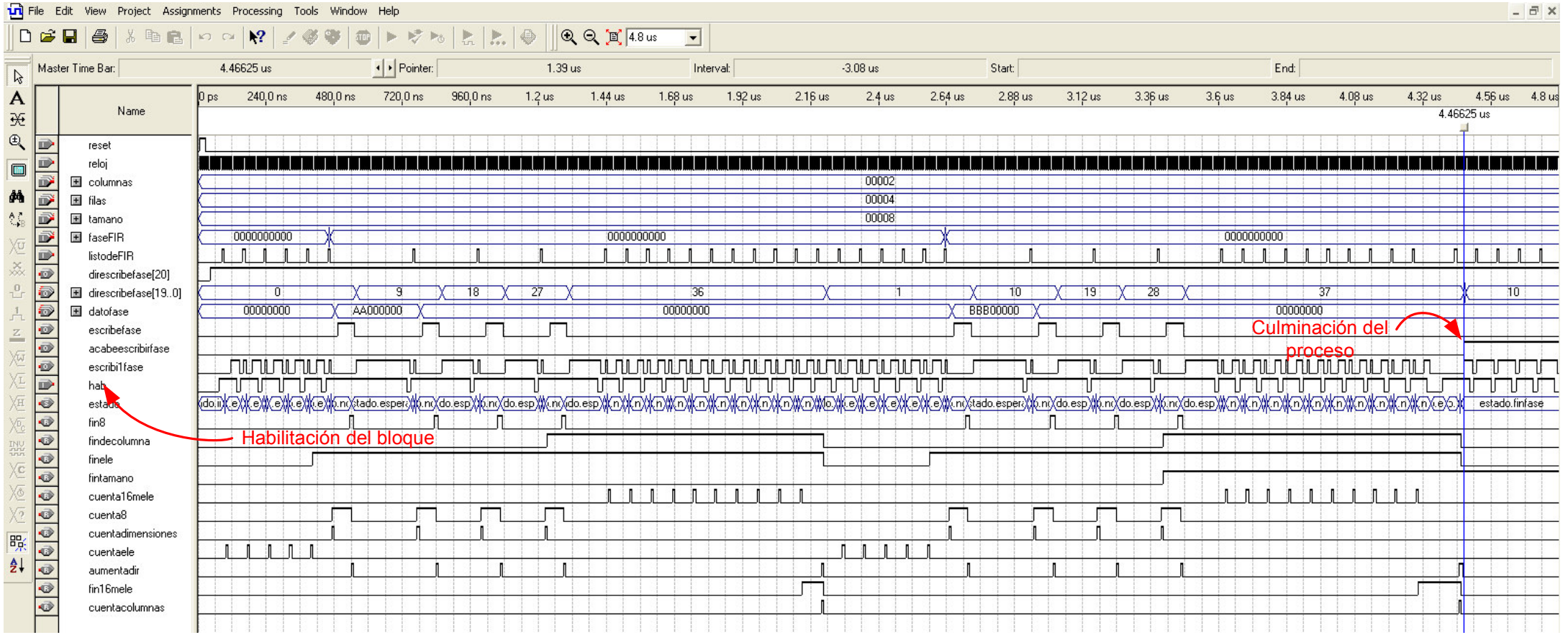
Demultiplexor diseñado



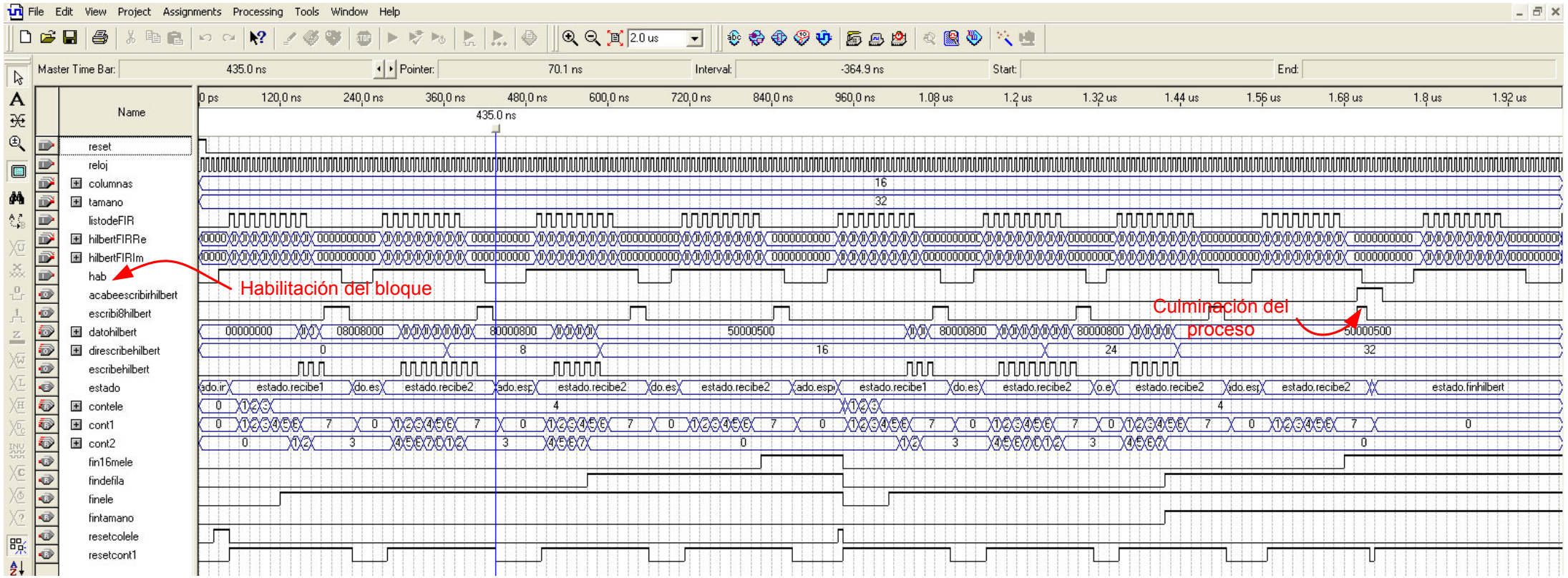
Escribe coeficientes



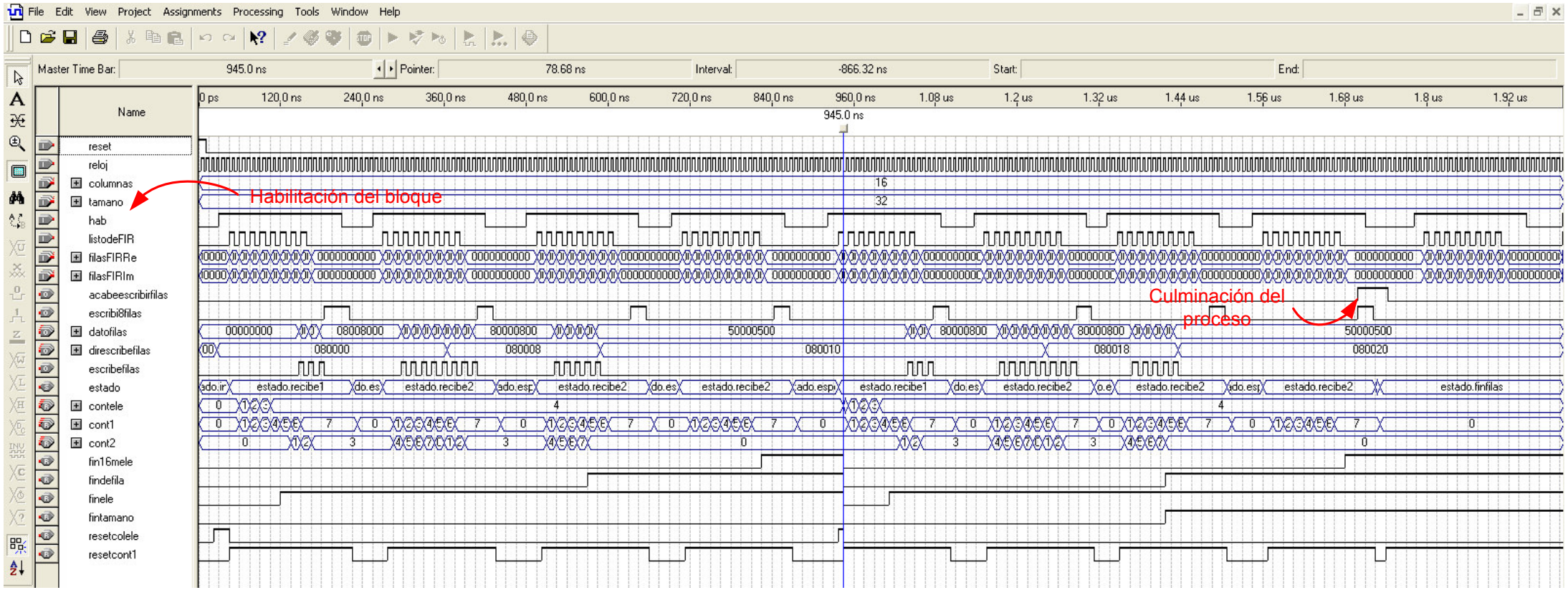
Escribir fase obtenida



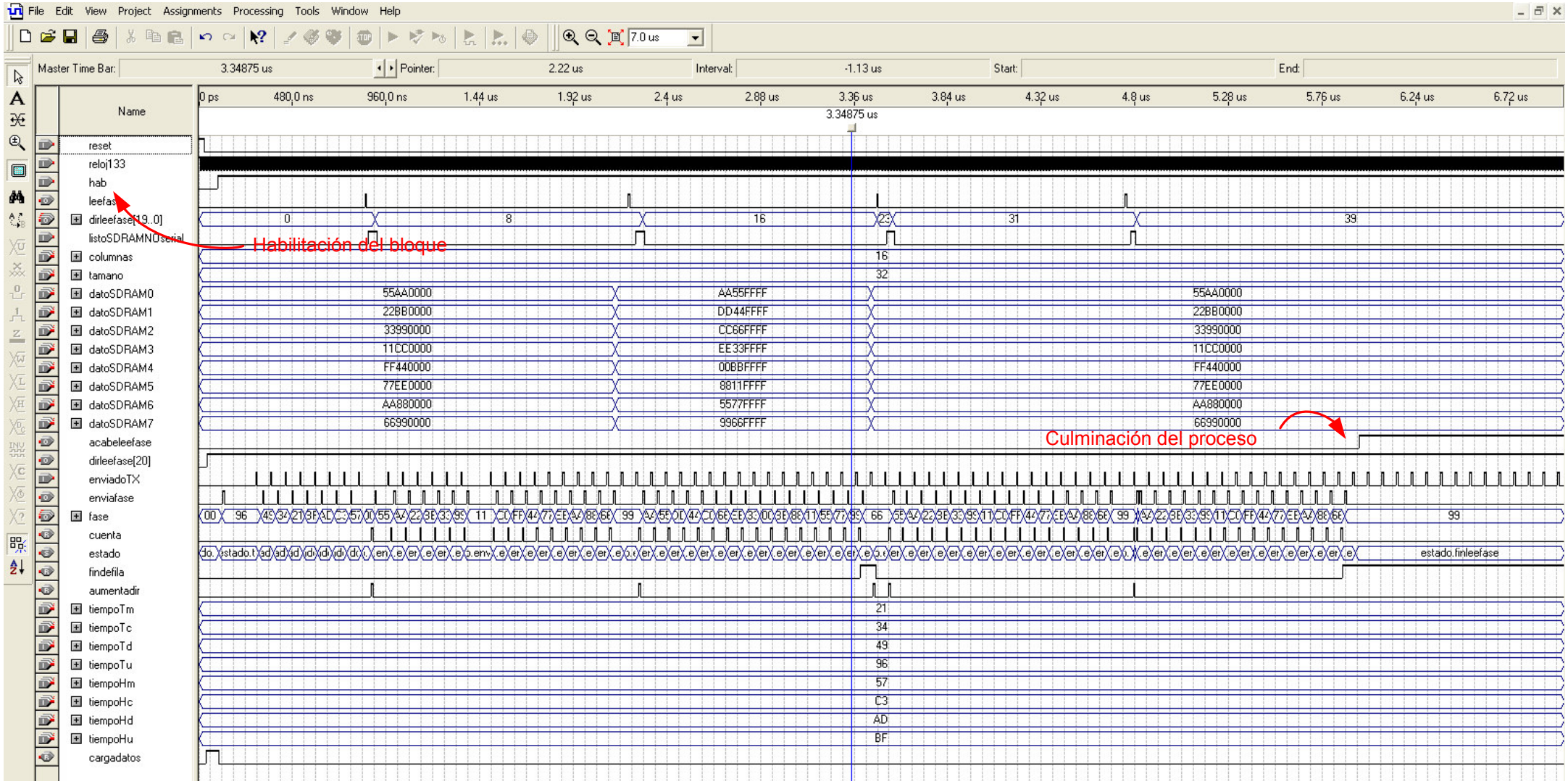
Escribir imagen analítica



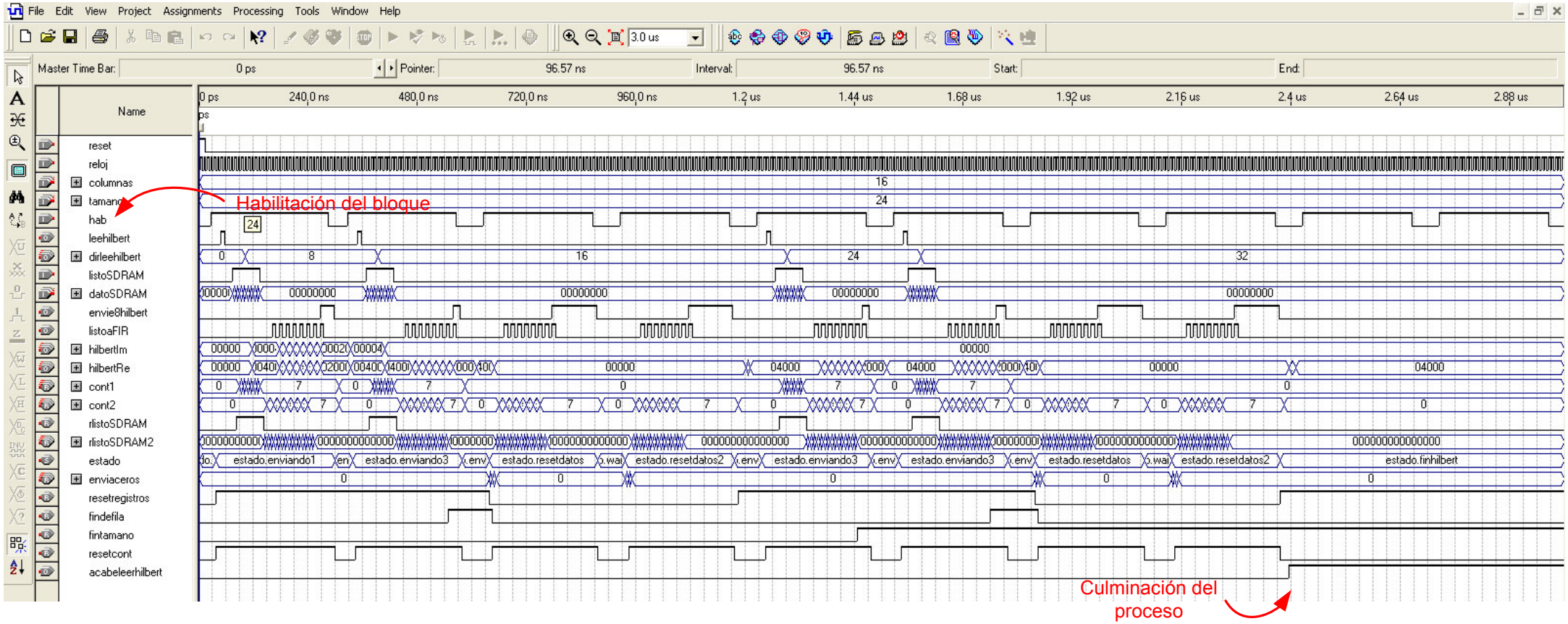
Escribir imagen filtrada en filas



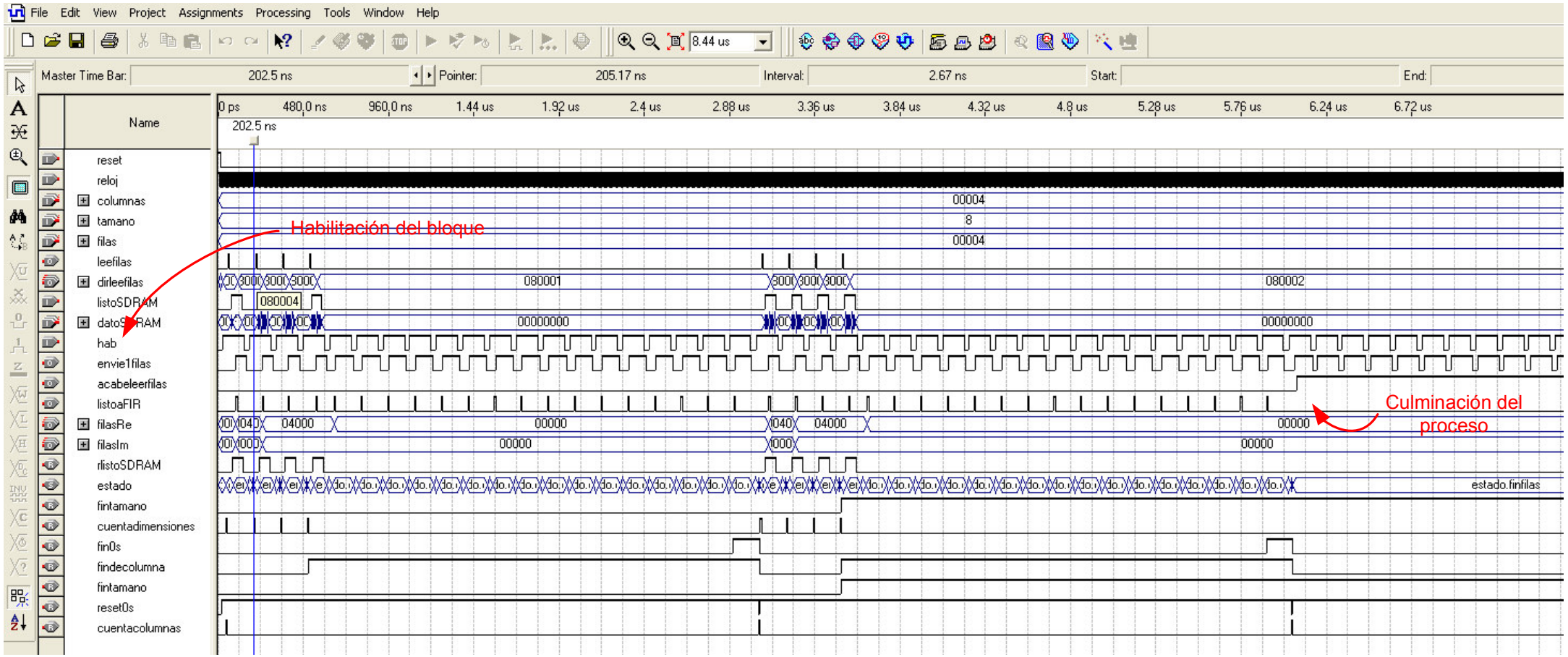
Leer fase obtenida y contador



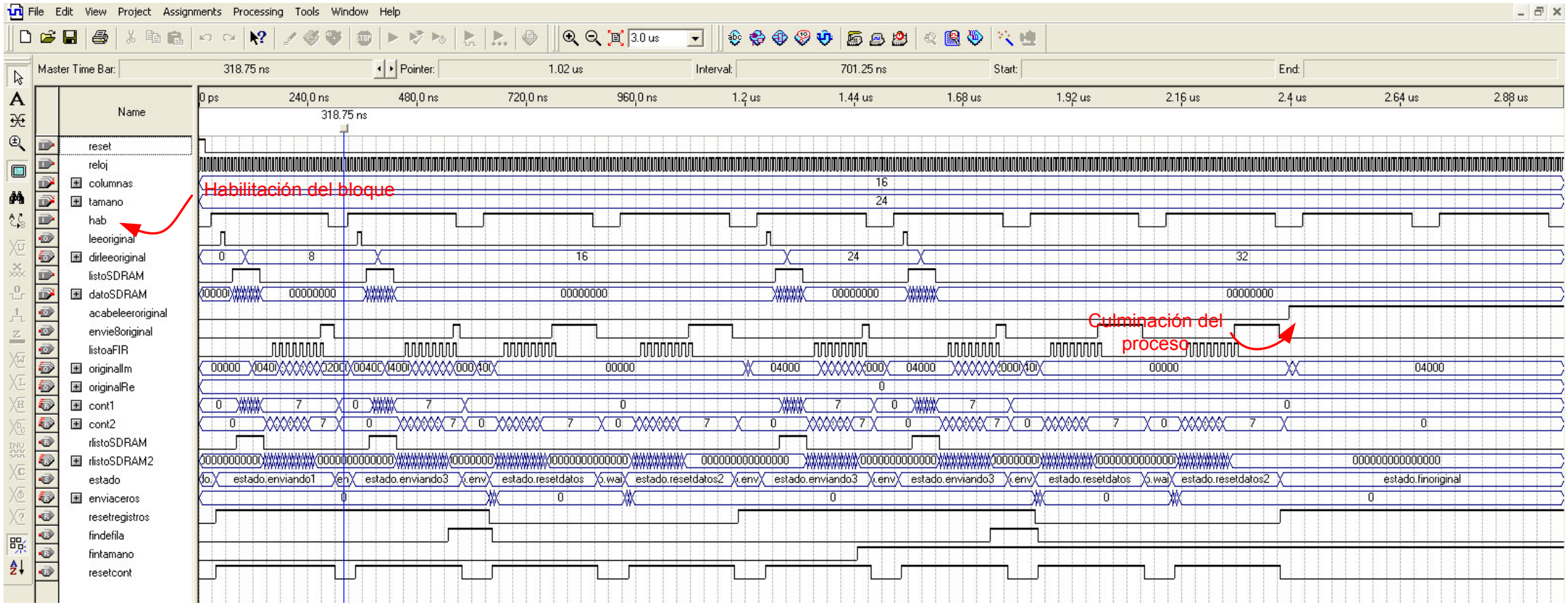
Leer imagen analítica



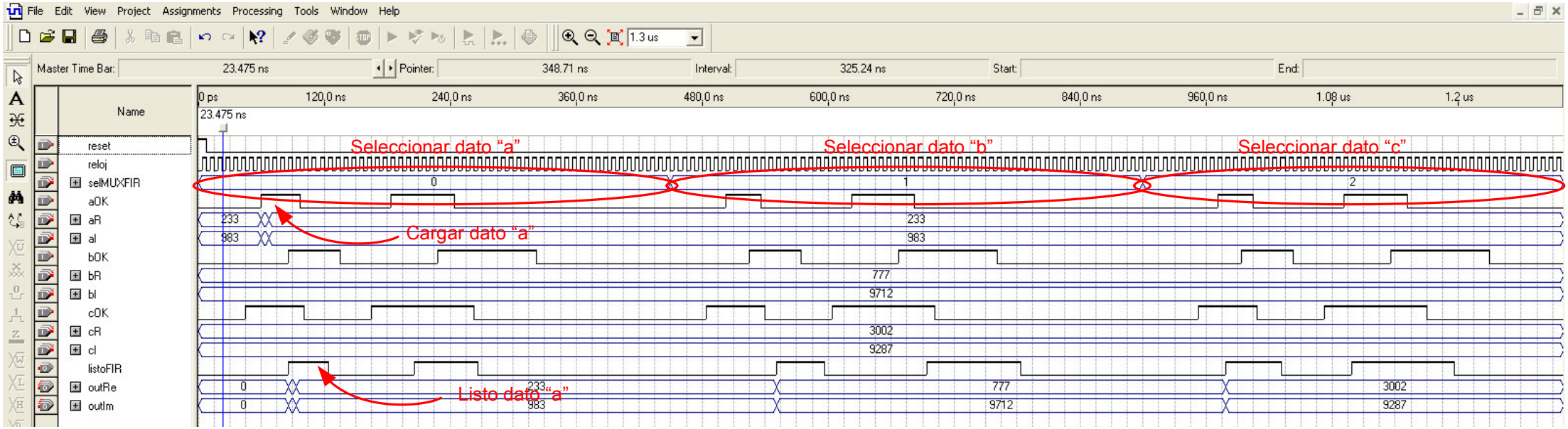
Leer imagen filtrada en filas

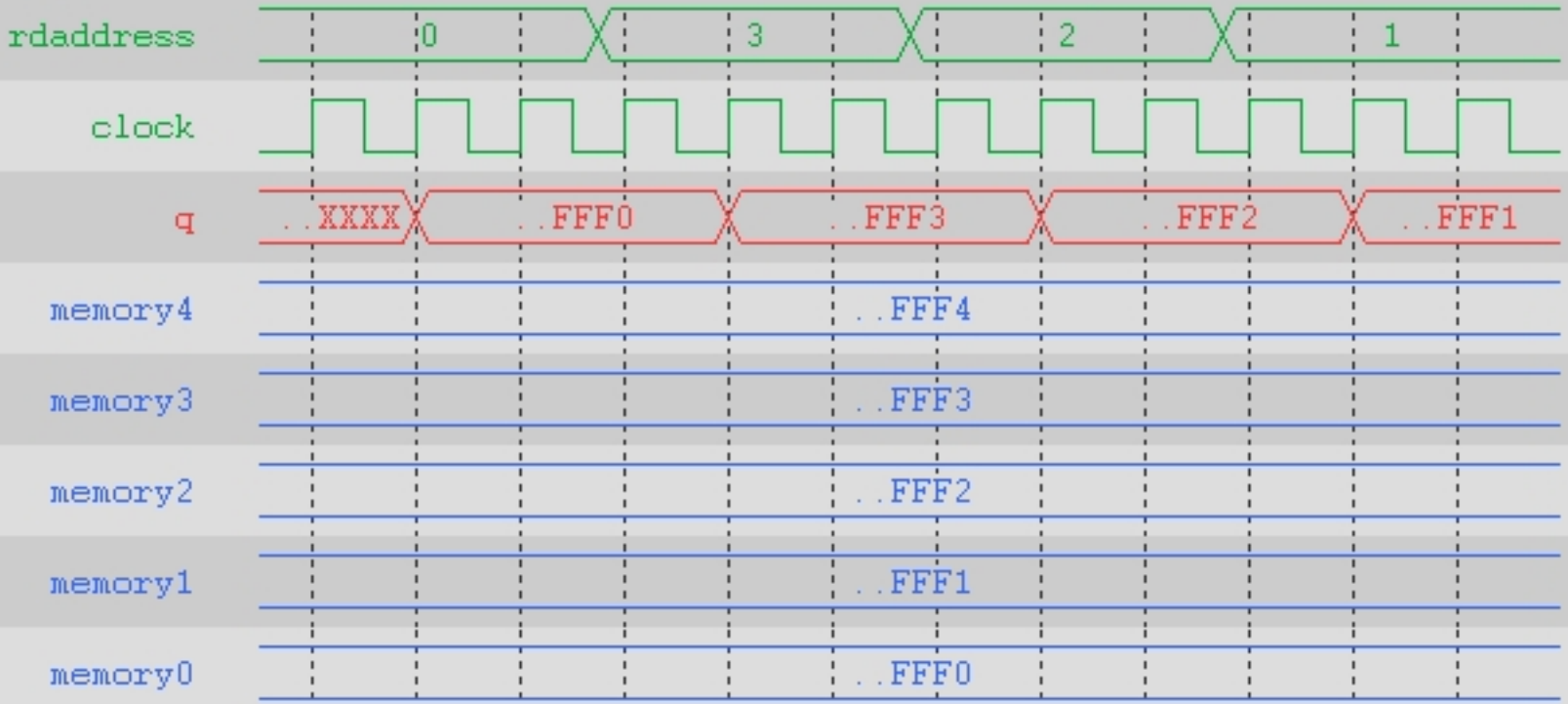


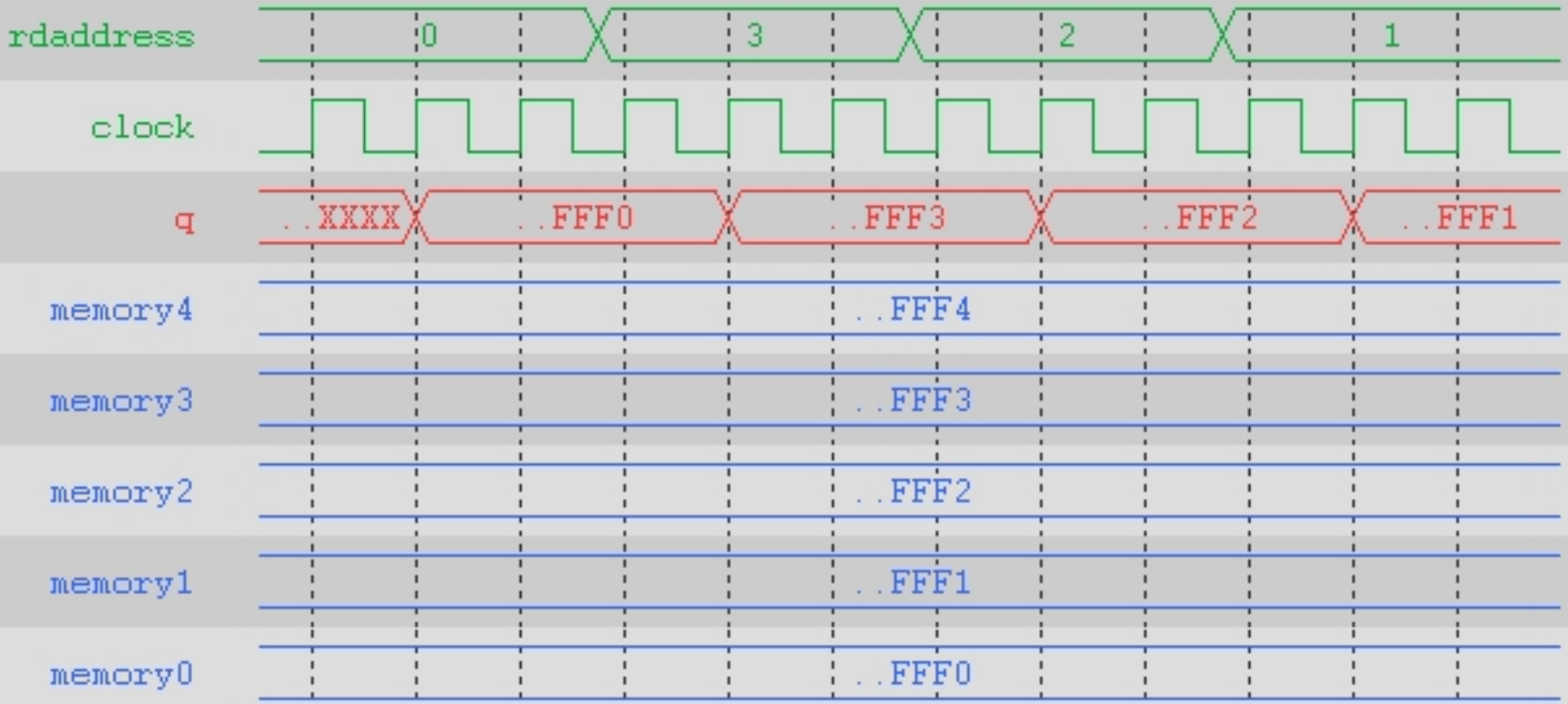
Leer imagen original

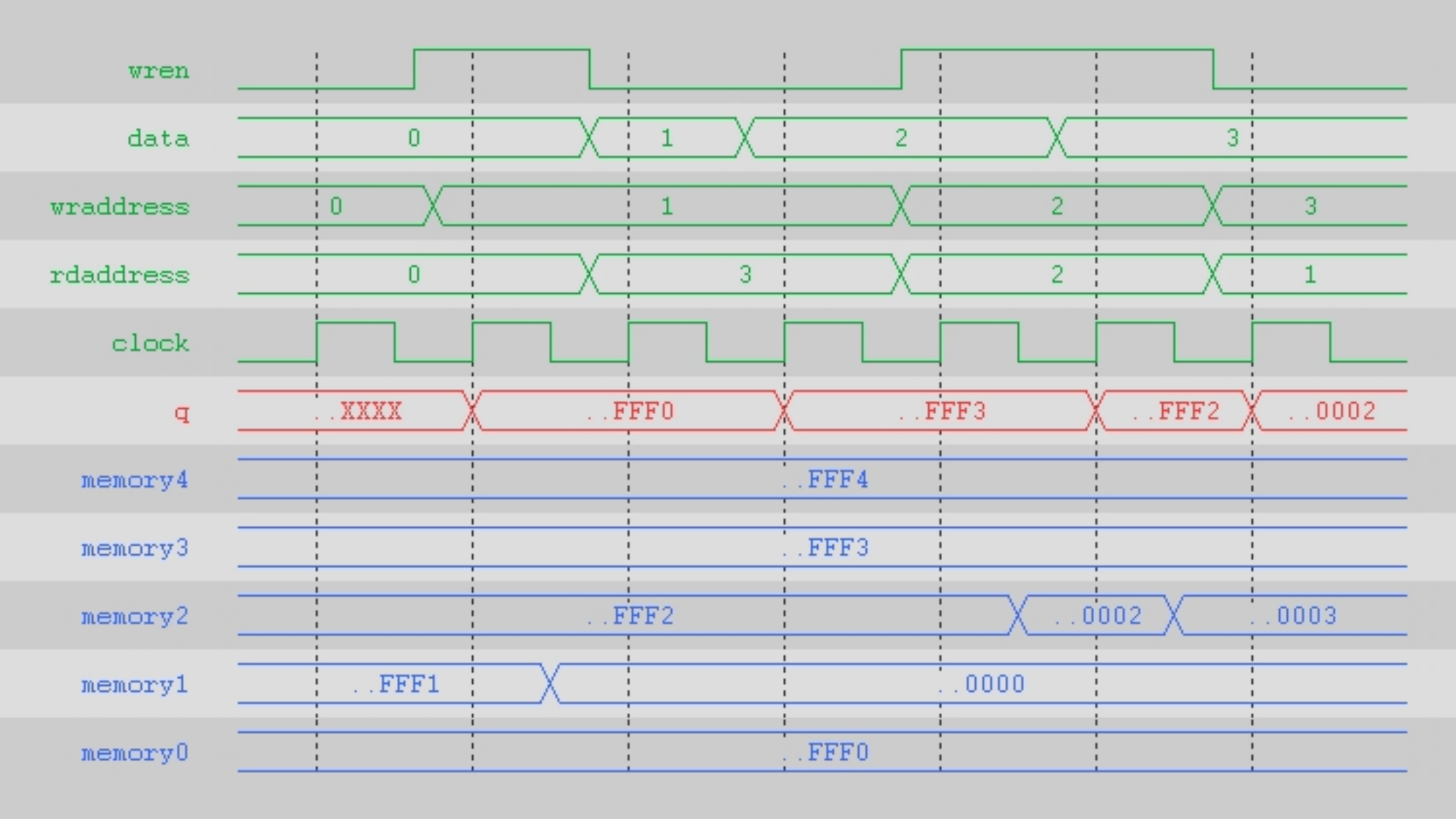


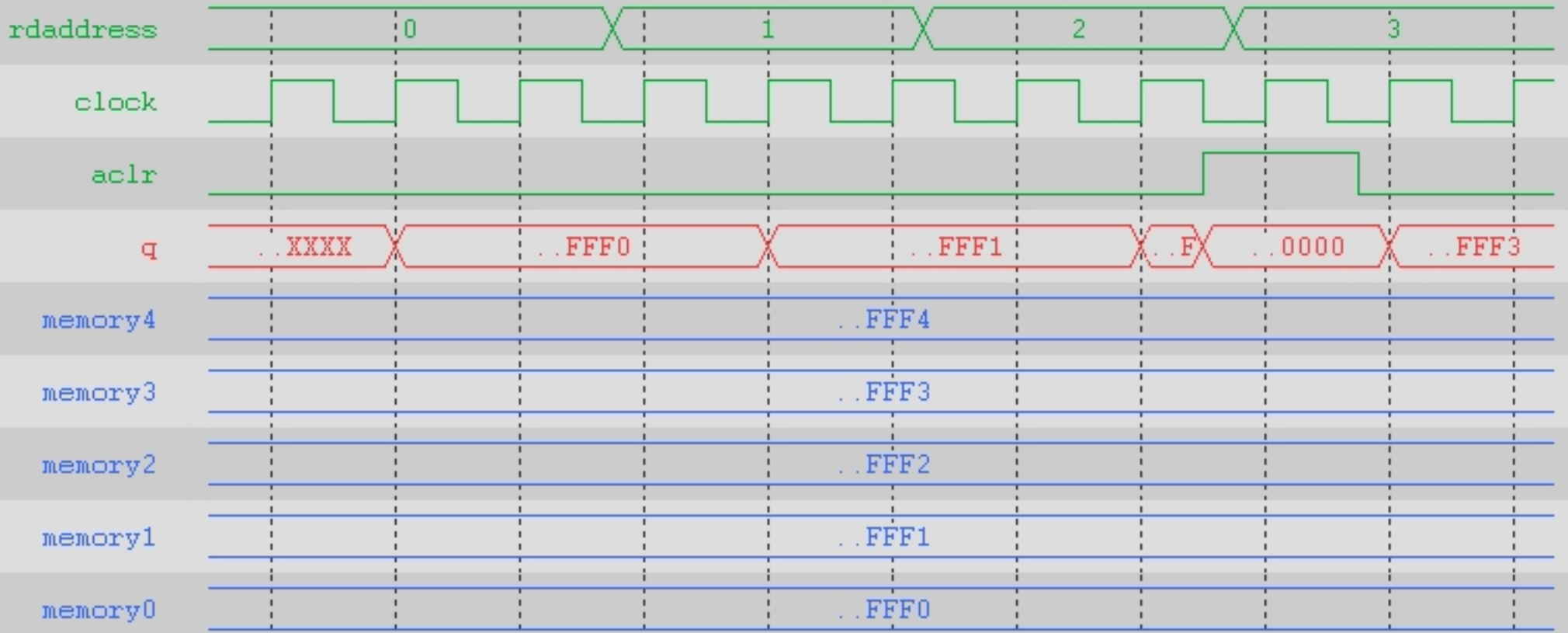
Multiplexor diseñado

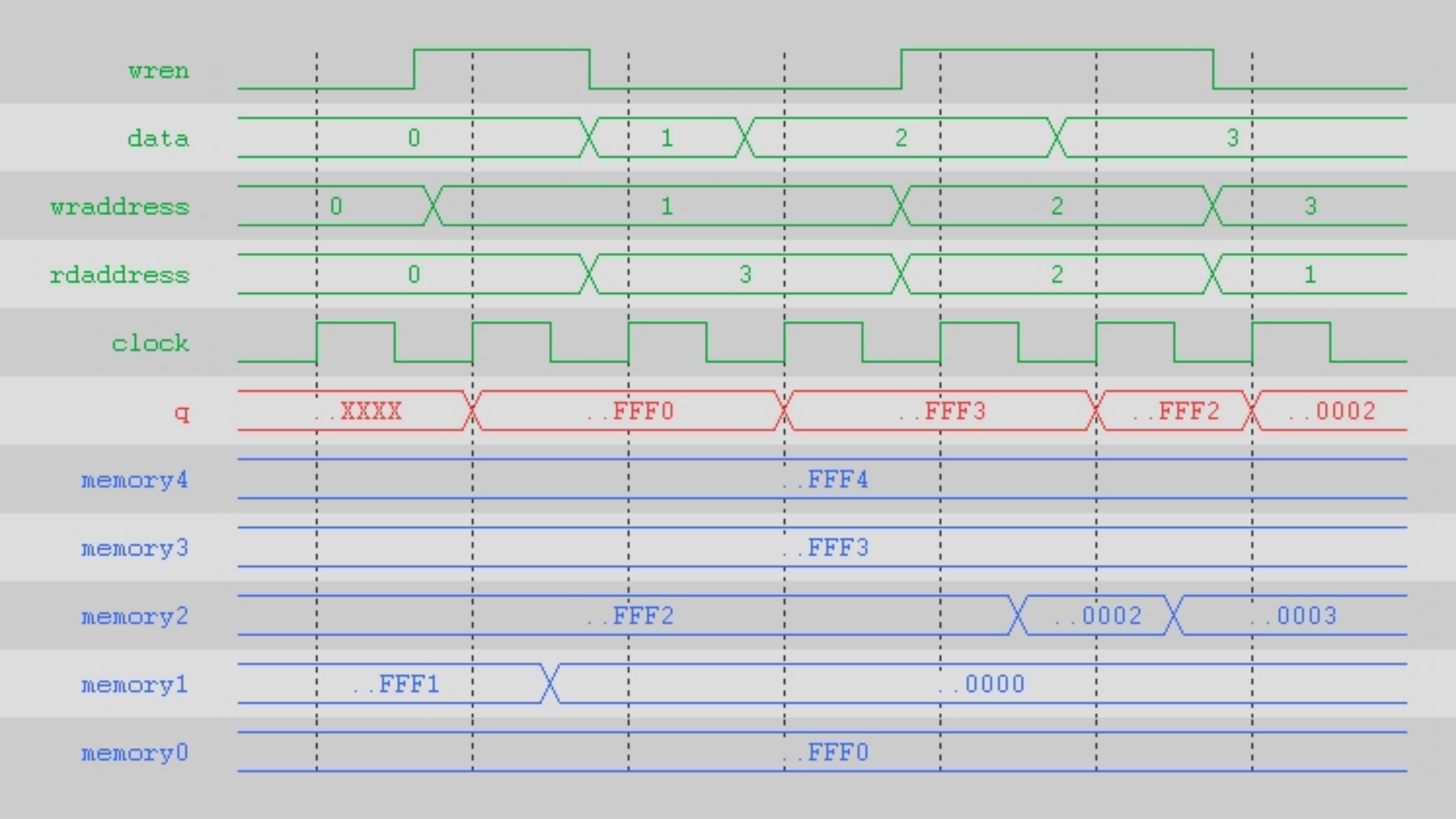


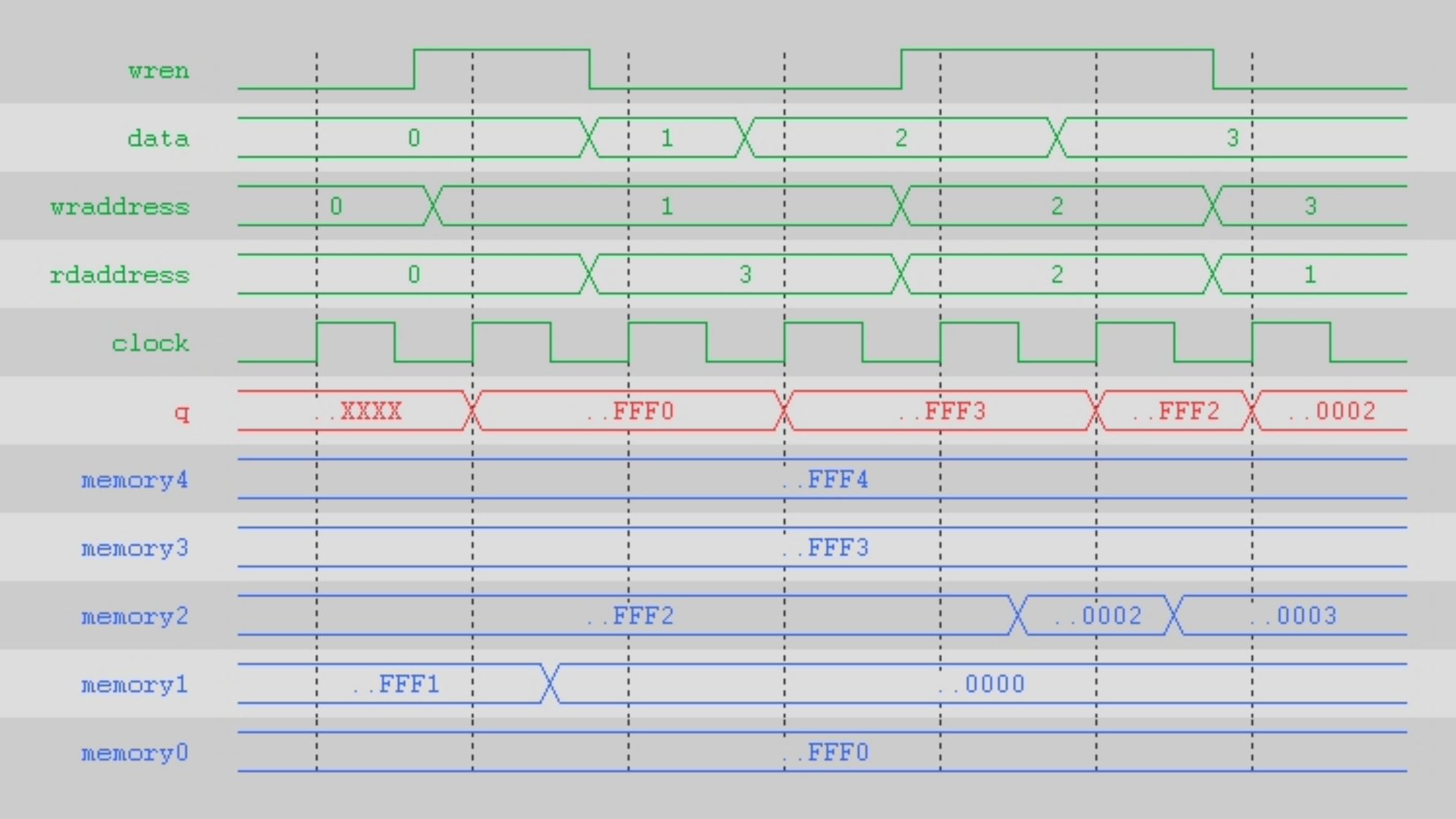

rdaddress
clock
q
memory4
memory3
memory2
memory1
memory0
0
3
2
1
..XXXX
..FFF0
..FFF3
..FFF2
..FFF1
..FFF4
..FFF3
..FFF2
..FFF1
..FFF0

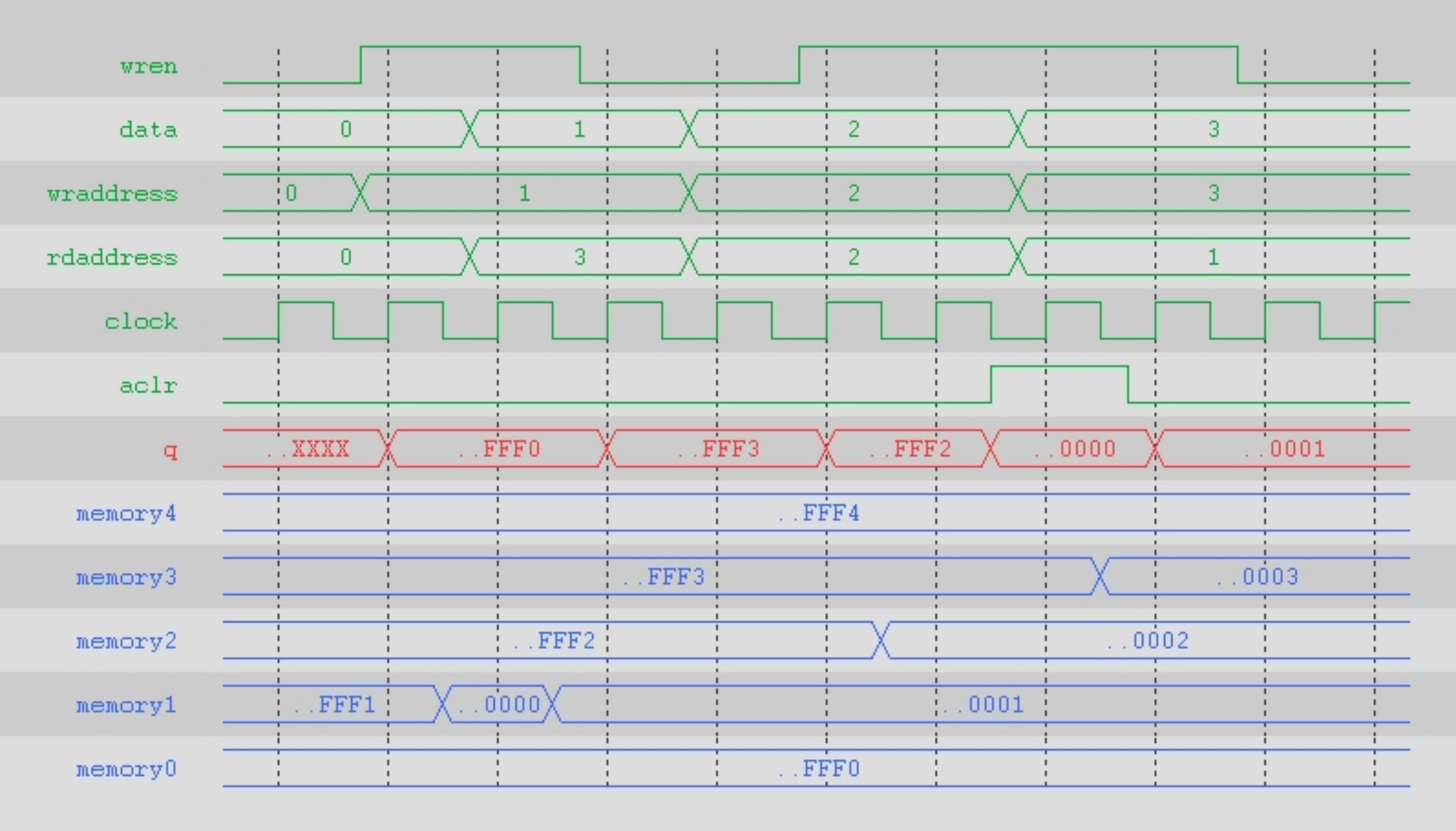

rdaddress
clock
q
memory4
memory3
memory2
memory1
memory0
0
3
2
1
..XXXX
..FFF0
..FFF3
..FFF2
..FFF1
..FFF4
..FFF3
..FFF2
..FFF1
..FFF0

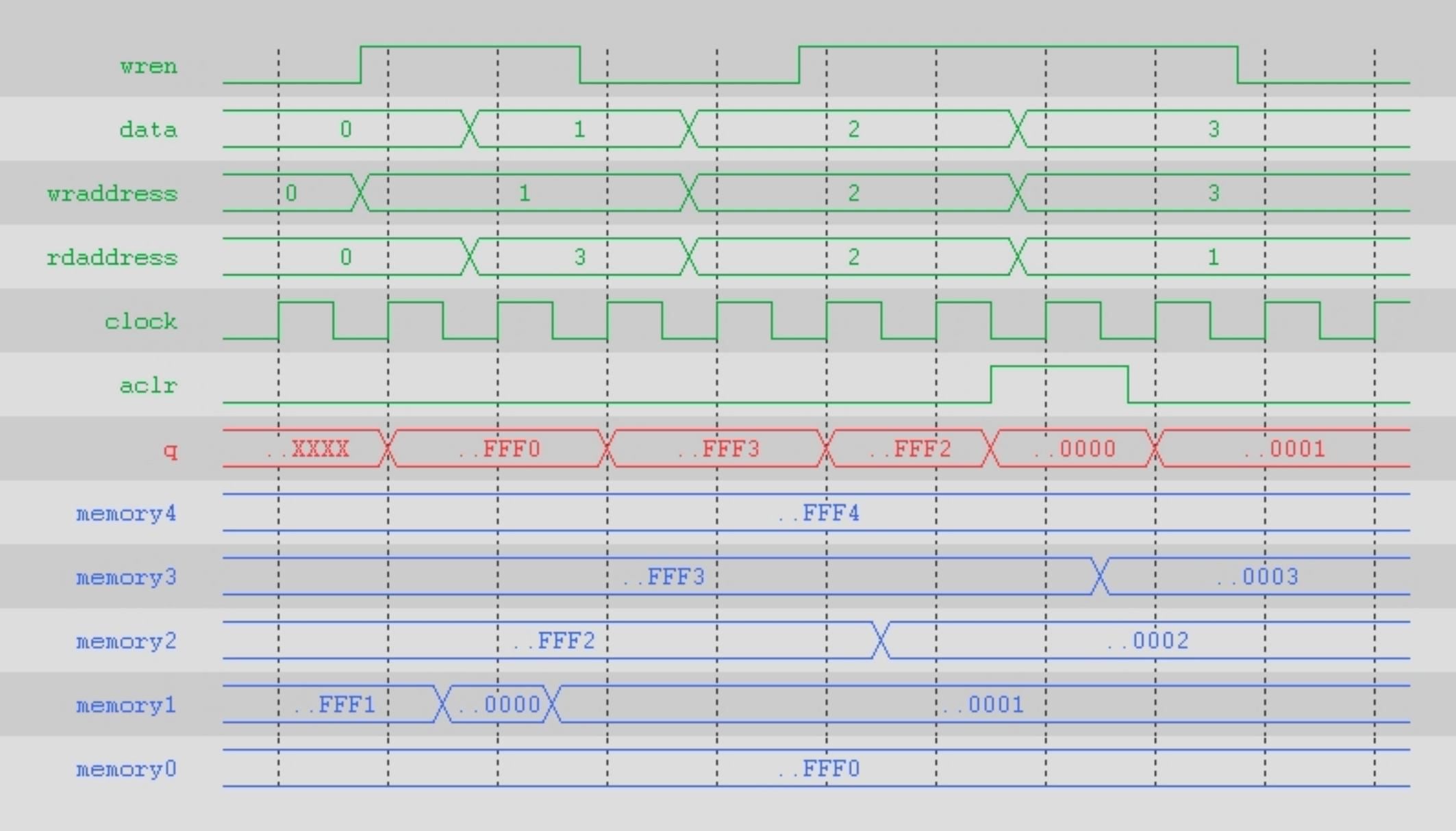

wren
data
0
1
2
3
wraddress
0
1
2
3
rdaddress
0
3
2
1
clock
q
..XXXX
..FFF0
..FFF3
..FFF2
..0002
memory4
..FFF4
memory3
..FFF3
memory2
..FFF2
..0002
..0003
memory1
..FFF1
..0000
memory0
..FFF0


rdaddress
clock
aclr
q
memory4
memory3
memory2
memory1
memory0
0
1
2
3
..XXXX
..FFF0
..FFF1
..F
..0000
..FFF3
..FFF4
..FFF3
..FFF2
..FFF1
..FFF0


wren
data
0
1
2
3
wraddress
0
1
2
3
rdaddress
0
3
2
1
clock
q
..XXXX
..FFF0
..FFF3
..FFF2
..0002
memory4
..FFF4
memory3
..FFF3
memory2
..FFF2
..0002
..0003
memory1
..FFF1
..0000
memory0
..FFF0


wren
data
0
1
2
3
wraddress
0
1
2
3
rdaddress
0
3
2
1
clock
q
..XXXX
..FFF0
..FFF3
..FFF2
..0002
memory4
..FFF4
memory3
..FFF3
memory2
..FFF2
..0002
..0003
memory1
..FFF1
..0000
memory0
..FFF0


wren
data
0
1
2
3
wraddress
0
1
2
3
rdaddress
0
3
2
1
clock
aclr
q
..XXXX
..FFF0
..FFF3
..FFF2
..0000
..0001
memory4
..FFF4
memory3
..FFF3
..0003
memory2
..FFF2
..0002
memory1
..FFF1
..0000
..0001
memory0
..FFF0


wren
data
0
1
2
3
wraddress
0
1
2
3
rdaddress
0
3
2
1
clock
aclr
q
..XXXX
..FFF0
..FFF3
..FFF2
..0000
..0001
memory4
..FFF4
memory3
..FFF3
..0003
memory2
..FFF2
..0002
memory1
..FFF1
..0000
..0001
memory0
..FFF0

Sample behavioral waveforms for design file ram2.vhd

The following waveforms show the behavior of altsyncram megafunction for the chosen set of parameters in design ram2.vhd . For the purpose of this simulation, the contents of the memory at the start of the sample waveforms is assumed to be (..FFF0, ..FFF1, ..FFF2, ..FFF3, ..FFF4, ...). The design ram2.vhd has one read port and one write port. The read port has 32 words of 18 bits each and the write port has 32 words of 18 bits each. The ram block type of the design is AUTO . The output of the read port is registered by clock.

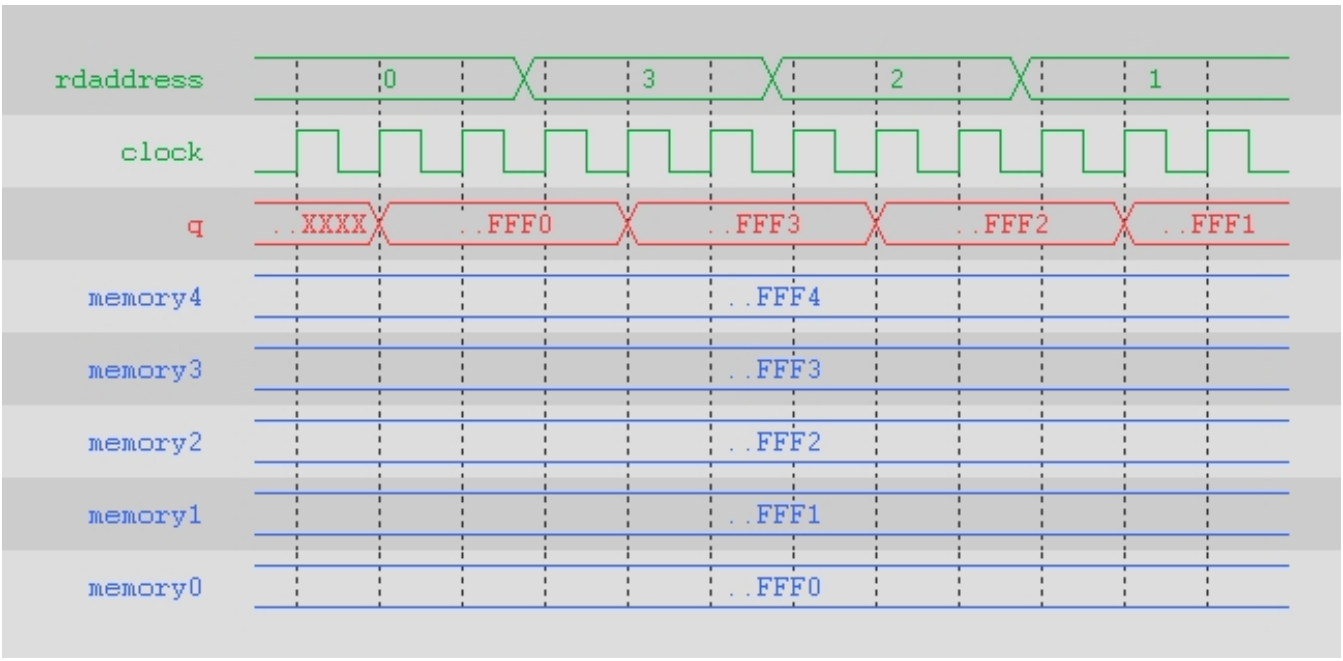


Fig. 1 : Wave showing read operation.

The above waveform shows the behavior of the design under normal read conditions. The read happens at the rising edge of the enabled clock cycle. The output from the RAM is undefined until after the first rising edge of the read clock.

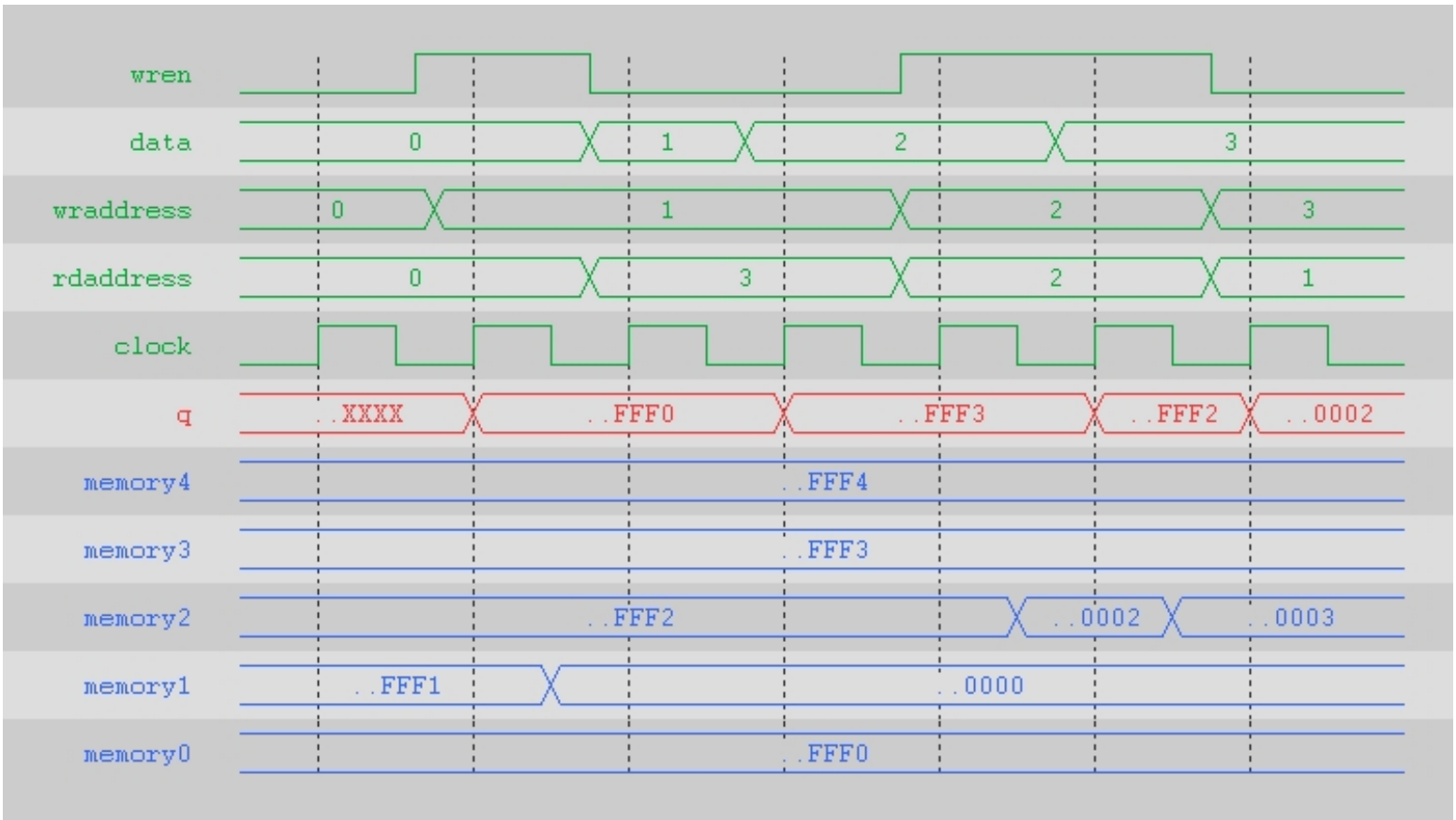


Fig. 2 : Waveform showing write operation

The above waveform shows the behavior of the design under normal write conditions. The write cycle is assumed to be from the rising edge of the enabled clock in which wren is high till the rising edge of the next clock cycle. In DUAL_PORT mode, when the write happens at the same address as the one being read in the other port, the read output is the old data at the address. Actual write into the RAM happens at the rising edge or falling edge of the write clock, depending on whether the RAM blocks are assigned to M-RAM or not. In the sample waveforms, they are shown to be on the falling edge of the write clock.

Sample behavioral waveforms for design file ram2.vhd

The following waveforms show the behavior of altsyncram megafunction for the chosen set of parameters in design ram2. vhd . For the purpose of this simulation, the contents of the memory at the start of the sample waveforms is assumed to be (..FFF0, ..FFF1, ..FFF2, ..FFF3, ..FFF4, ...). The design ram2.vhd has one read port and one write port. The read port has 8 words of 32 bits each and the write port has 8 words of 32 bits each. The ram block type of the design is AUTO . The output of the read port is registered by clock.

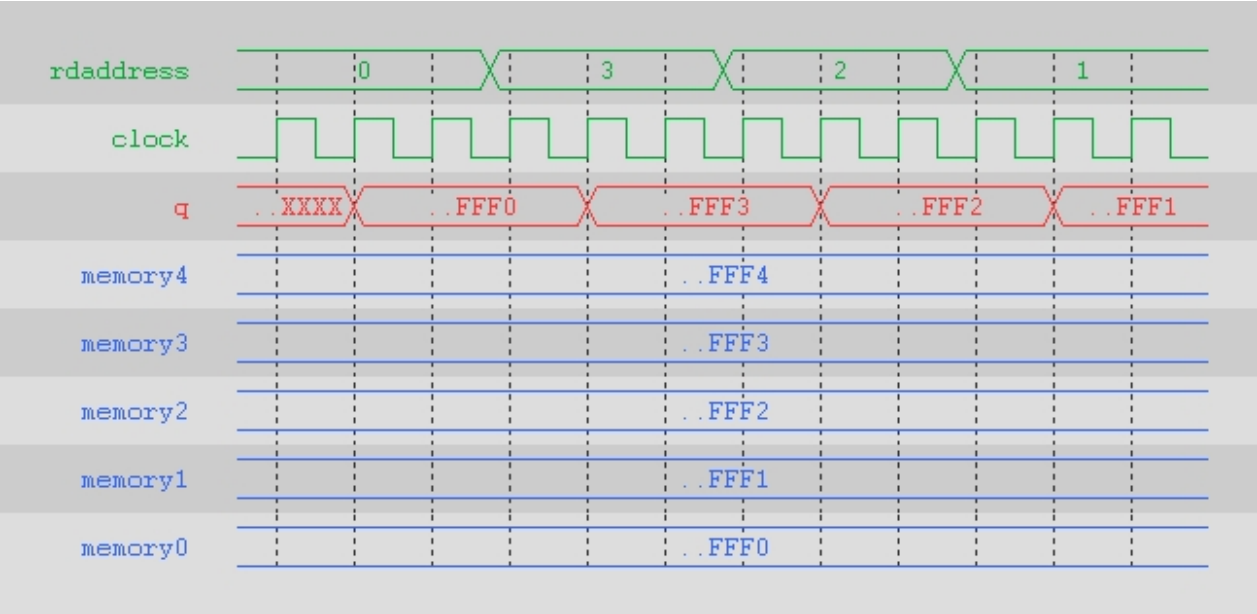


Fig. 1 : Wave showing read operation.

The above waveform shows the behavior of the design under normal read conditions. The read happens at the rising edge of the enabled clock cycle. The output from the RAM is undefined until after the first rising edge of the read clock.

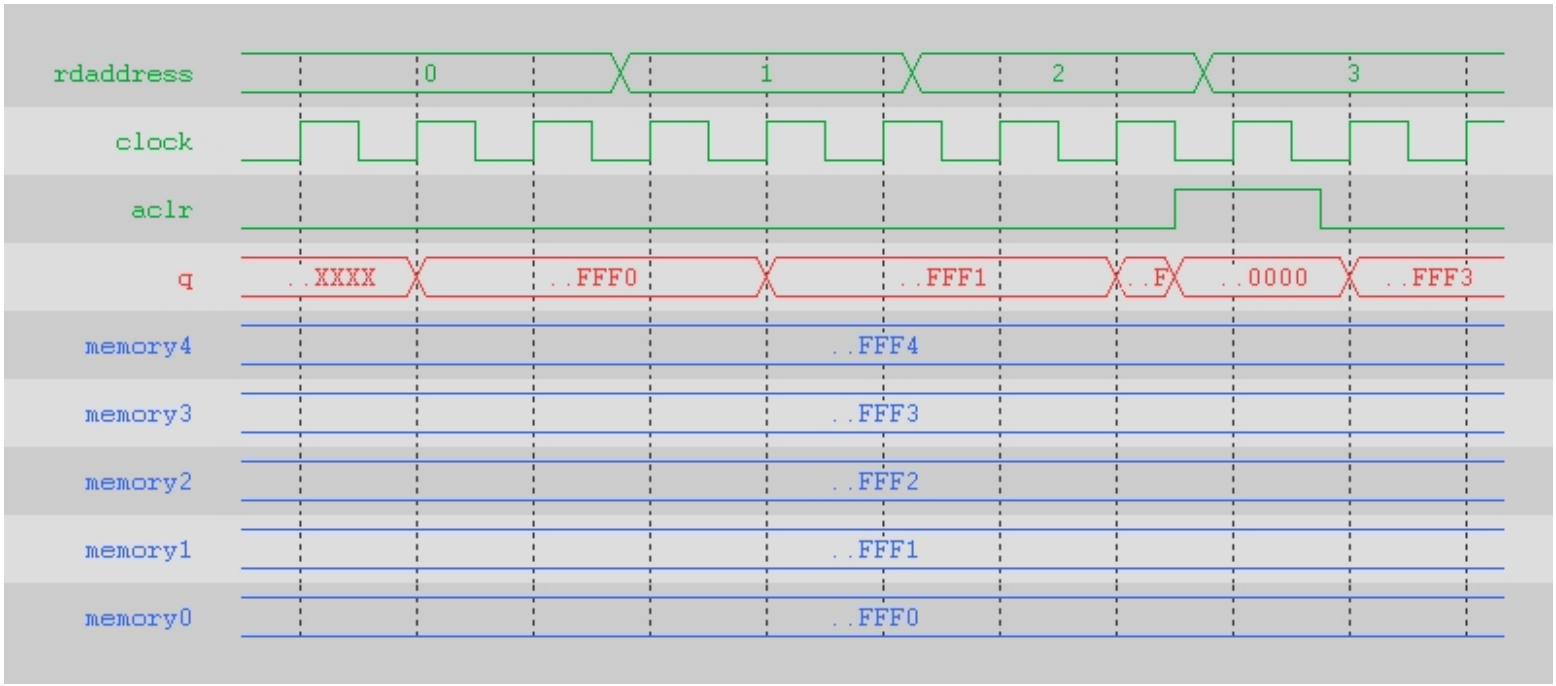


Fig. 2 : Waveform showing read operation with clear(s)

The above waveform shows the behavior of the design under read conditions with clears on input and/or output registers. The read happens at the rising edge of the enabled clock cycle. The read cycle is assumed to be from the rising edge of the clock cycle till the next rising clock edge. The clear on the output register is asynchronous.

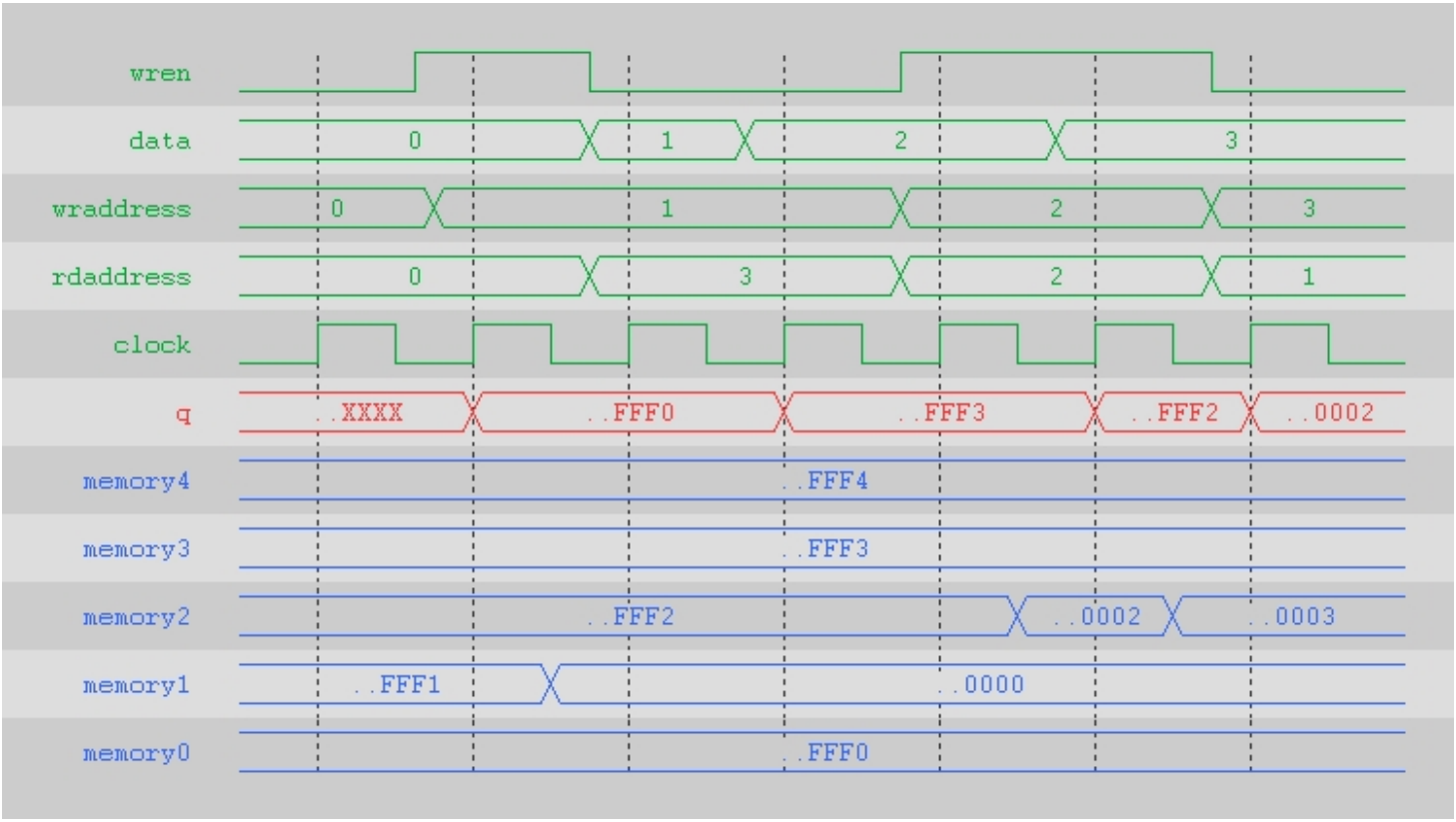


Fig. 3 : Waveform showing write operation

The above waveform shows the behavior of the design under normal write conditions. The write cycle is assumed to be from the rising edge of the enabled clock in which wren is high till the rising edge of the next clock cycle. In DUAL_PORT mode, when the write happens at the same address as the one being read in the other port, the read output is the old data at the address. Actual write into the RAM happens at the rising edge or falling edge of the write clock, depending on whether the RAM blocks are assigned to M-RAM or not. In the sample waveforms, they are shown to be on the falling edge of the write clock.

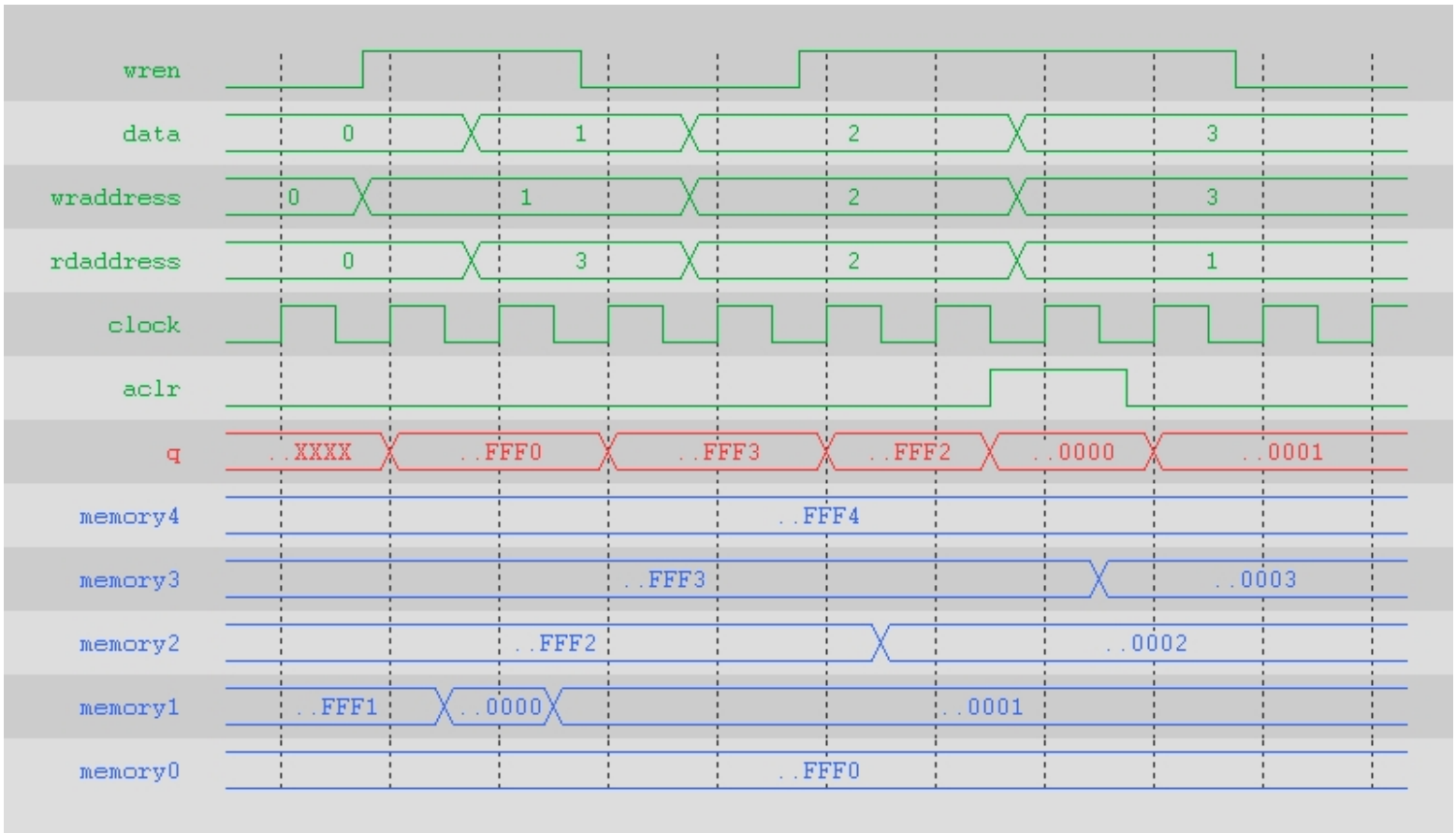
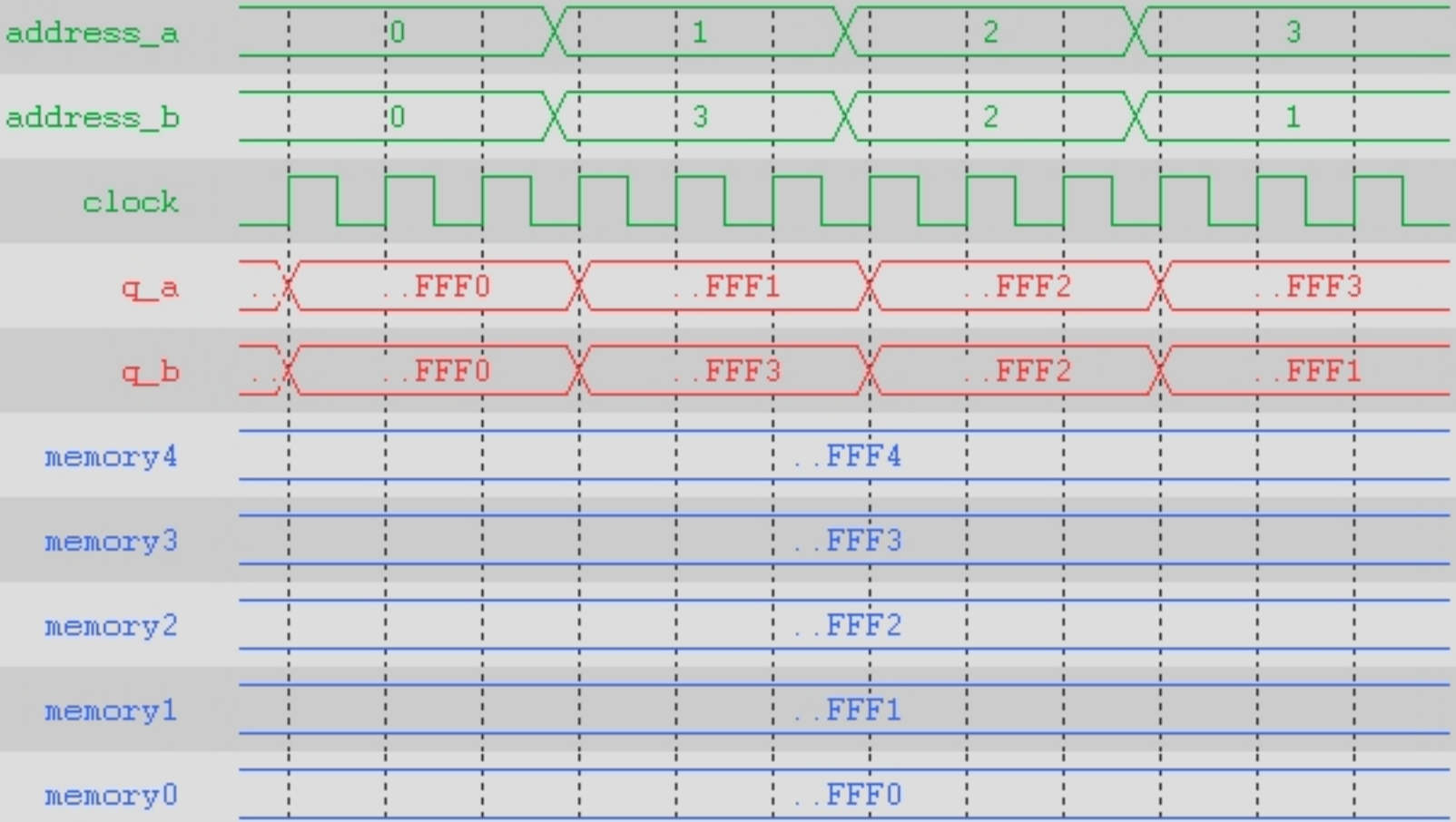
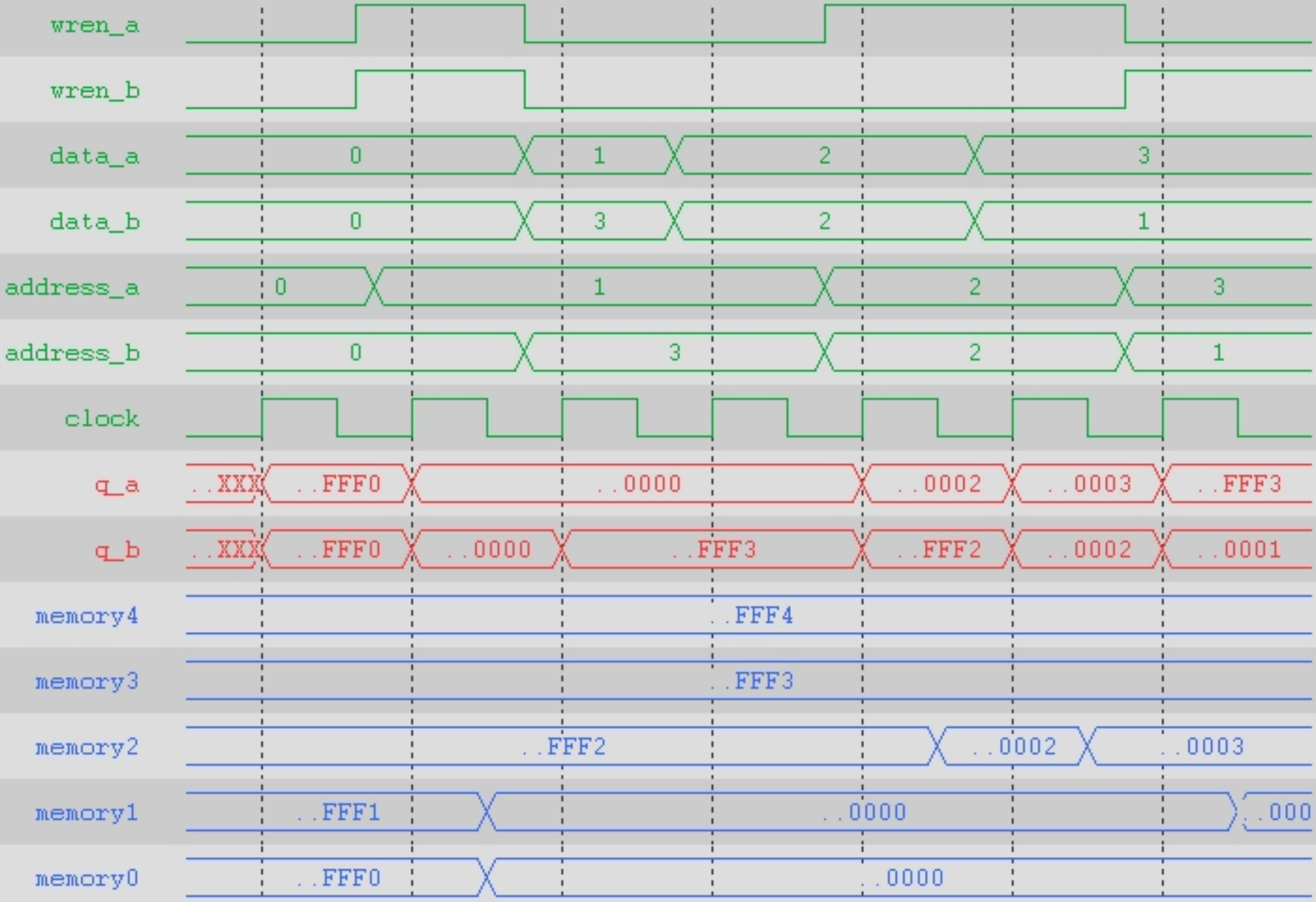


Fig. 4 : Waveform showing write operation with clear(s)

The above waveform shows the behavior of the design under write conditions with clear(s).


address_a
0
1
2
3
address_b
0
3
2
1
clock
q_a
..FFF0
..FFF1
..FFF2
..FFF3
q_b
..FFF0
..FFF3
..FFF2
..FFF1
memory4
..FFF4
memory3
..FFF3
memory2
..FFF2
memory1
..FFF1
memory0
..FFF0


wren_a
wren_b
data_a
0
1
2
3
data_b
0
3
2
1
address_a
0
1
2
3
address_b
0
3
2
1
clock
q_a
..XXX
..FFF0
..0000
..0002
..0003
..FFF3
q_b
..XXX
..FFF0
..0000
..FFF3
..FFF2
..0002
..0001
memory4
..FFF4
memory3
..FFF3
memory2
..FFF2
..0002
..0003
memory1
..FFF1
..0000
..000
memory0
..FFF0
..0000

Sample behavioral waveforms for design file ram2ports.vhd

The following waveforms show the behavior of altsyncram megafunction for the chosen set of parameters in design ram2ports.vhd . For the purpose of this simulation, the contents of the memory at the start of the sample waveforms is assumed to be (..FFF0, ..FFF1, ..FFF2, ..FFF3, ..FFF4, ...). The design ram2ports.vhd has two read/write ports. Read/write port A has 64 words of 18 bits each and Read/write port B has 64 words of 18 bits each. The ram block type of the design is AUTO . The output of the read/write port A is unregistered. The output of the read/write port B is unregistered.

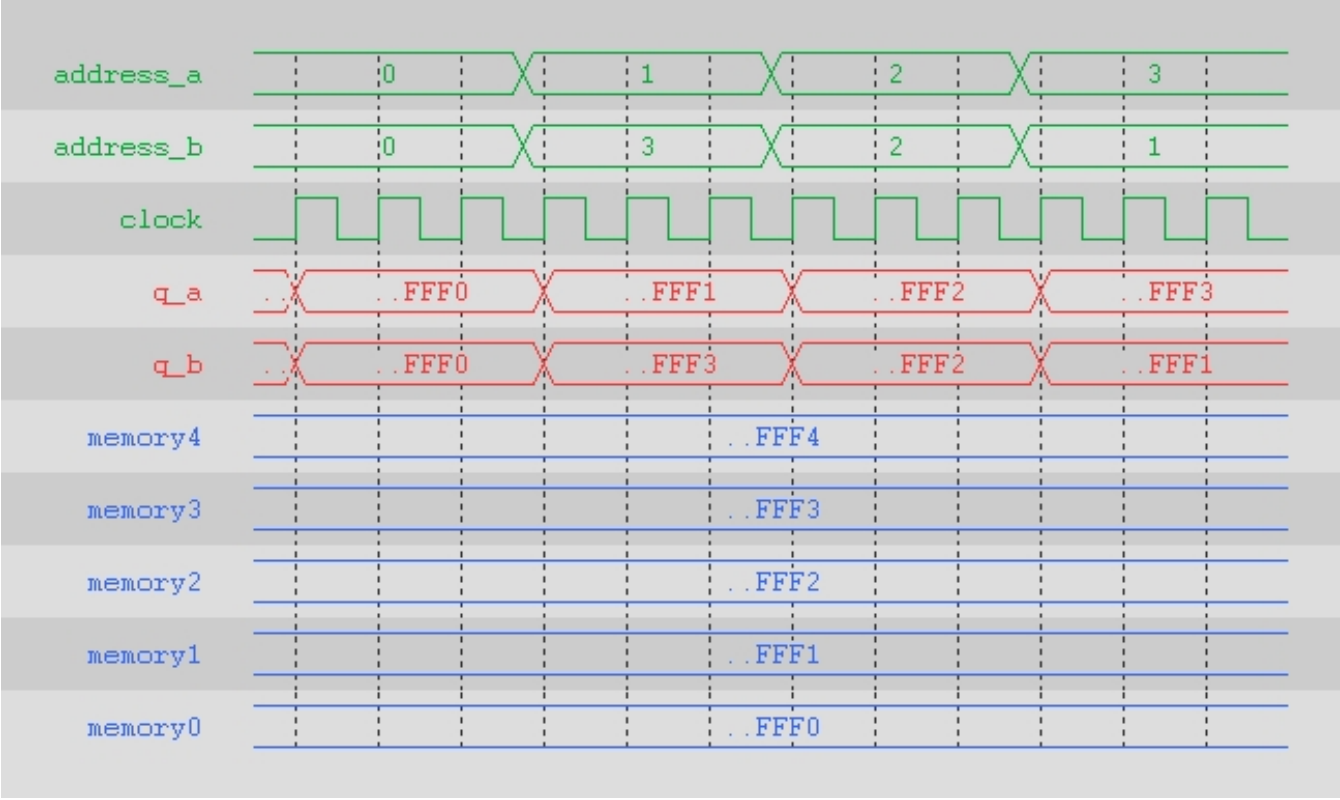


Fig. 1 : Wave showing read operation.

The above waveform shows the behavior of the design under normal read conditions. The read happens at the rising edge of the enabled clock cycle. The output from the RAM is undefined until after the first rising edge of the read clock.

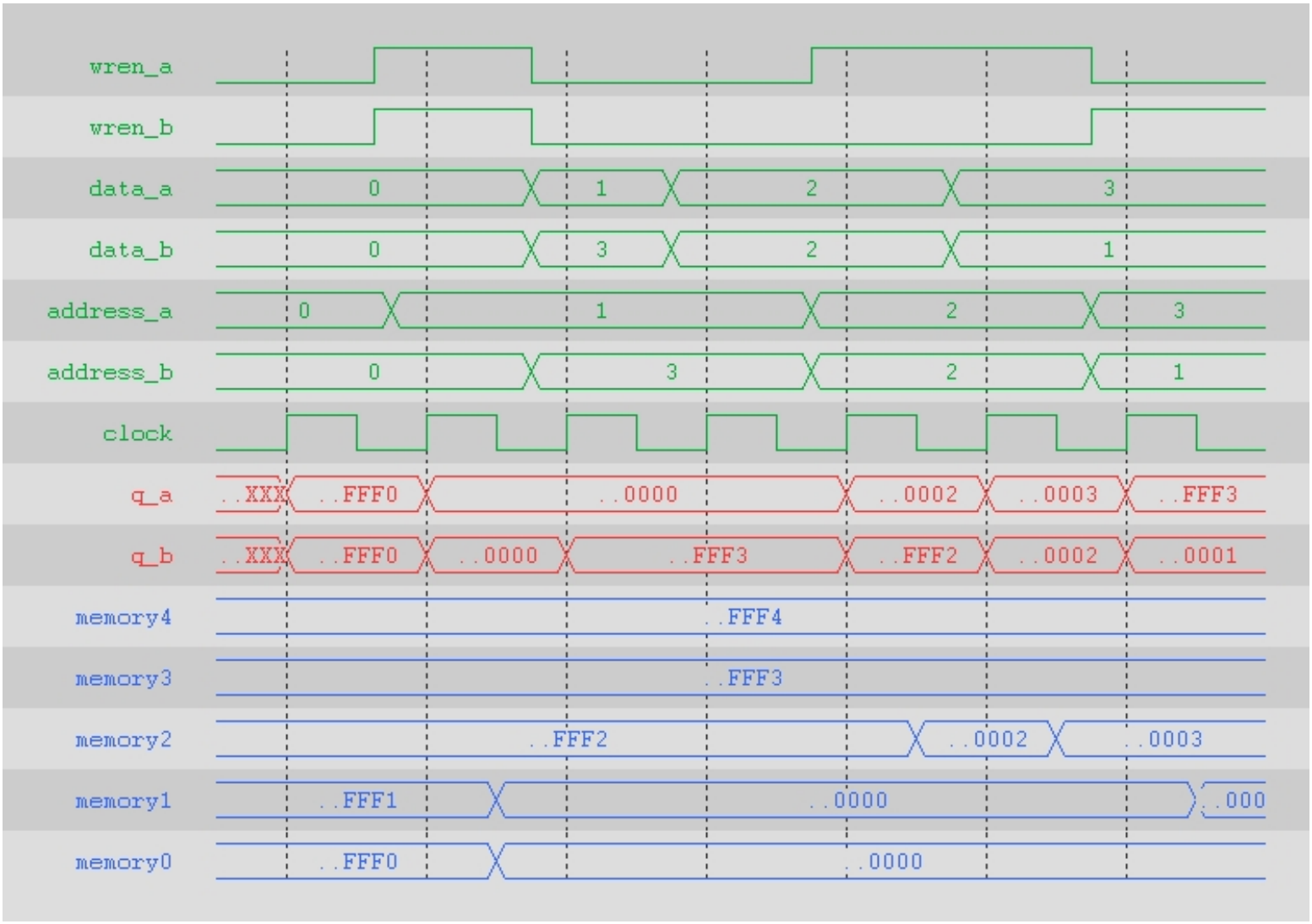
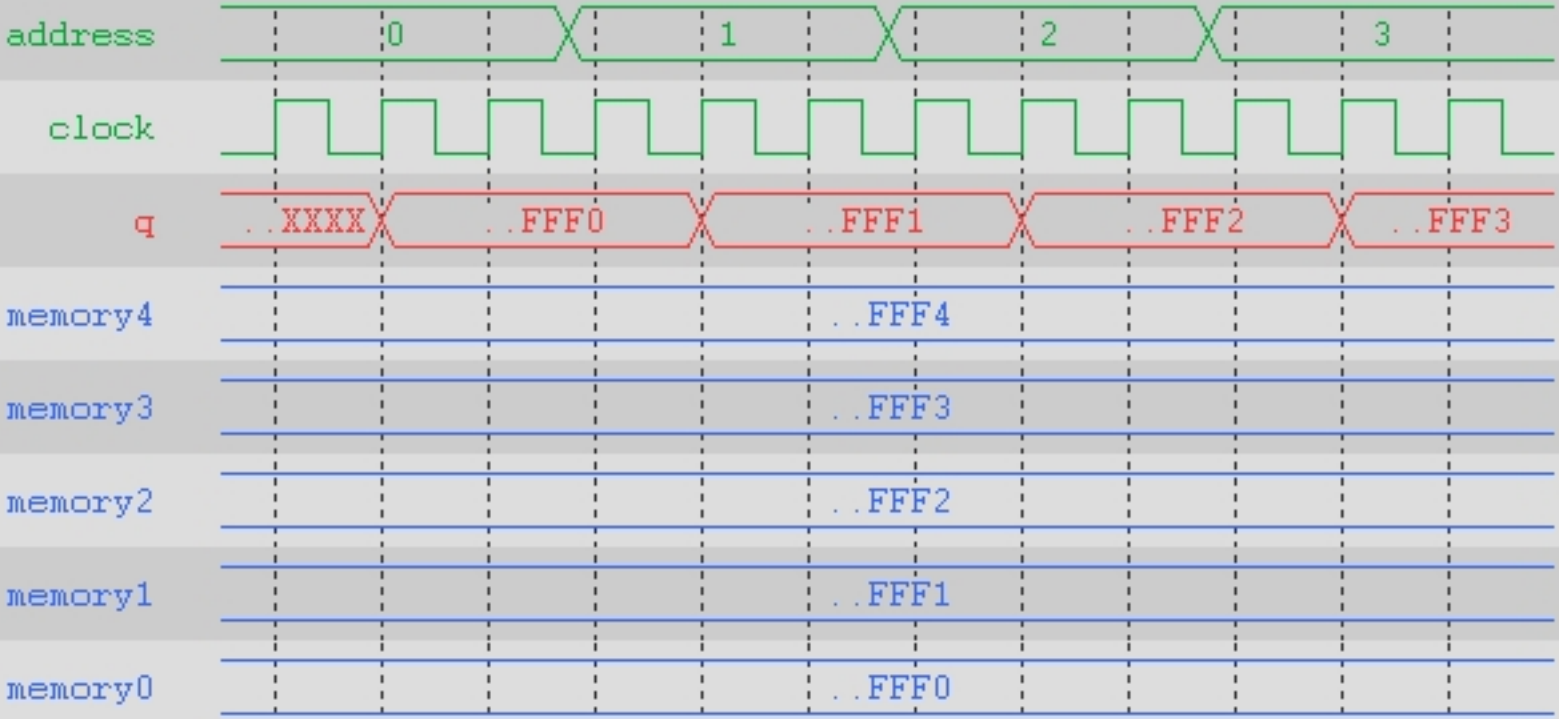


Fig. 2 : Waveform showing write operation

The above waveform shows the behavior of the design under normal write conditions. The write cycle is assumed to be from the rising edge of the enabled clock in which wren is high till the rising edge of the next clock cycle.
In BIDIR_DUAL_PORT mode, when the write happens at the same address as the one being read in the other port, the read output is the old data at the address. During a write cycle on a port (A or B), the new data flows through to the output of the same port. Actual write into the RAM happens at the rising edge or falling edge of the write clock, depending on whether the RAM blocks are assigned to M-RAM or not. In the sample waveforms, they are shown to be on the falling edge of the write clock.


address
clock
q
memory4
memory3
memory2
memory1
memory0
0
1
2
3
..XXXX
..FFF0
..FFF1
..FFF2
..FFF3
..FFF4
..FFF3
..FFF2
..FFF1
..FFF0

Sample behavioral waveforms for design file ramhilbert.vhd

The following waveforms show the behavior of altsyncram megafunction for the chosen set of parameters in design ramhilbert.vhd . For the purpose of this simulation, the contents of the memory at the start of the sample waveforms is assumed to be (..FFF0, ..FFF1, ..FFF2, ..FFF3, ..FFF4, ...). The design ramhilbert.vhd has one read port. The read port has 32 words of 18 bits each. The ram block type of the design is AUTO . The output of the read port is registered by clock.

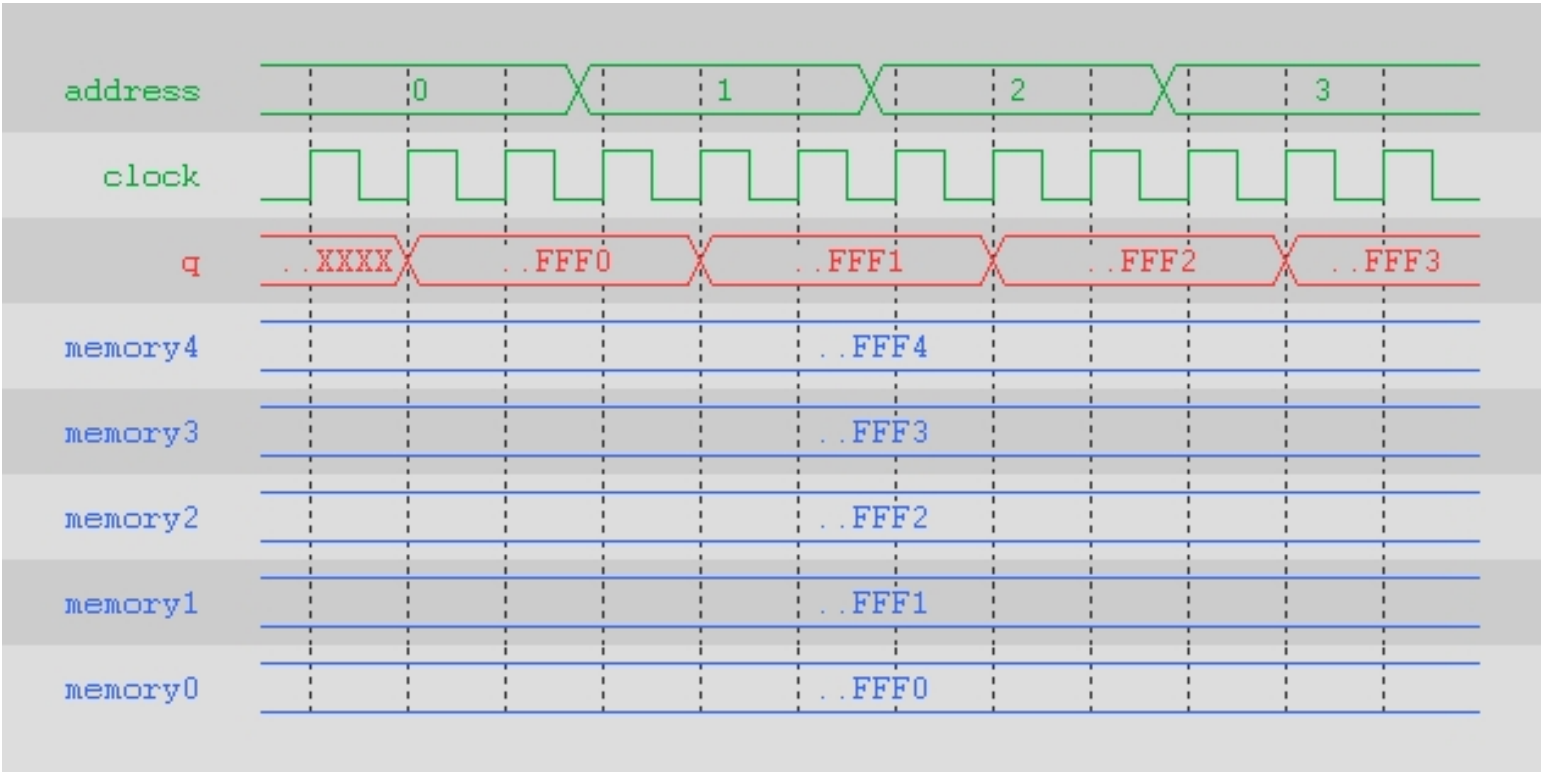
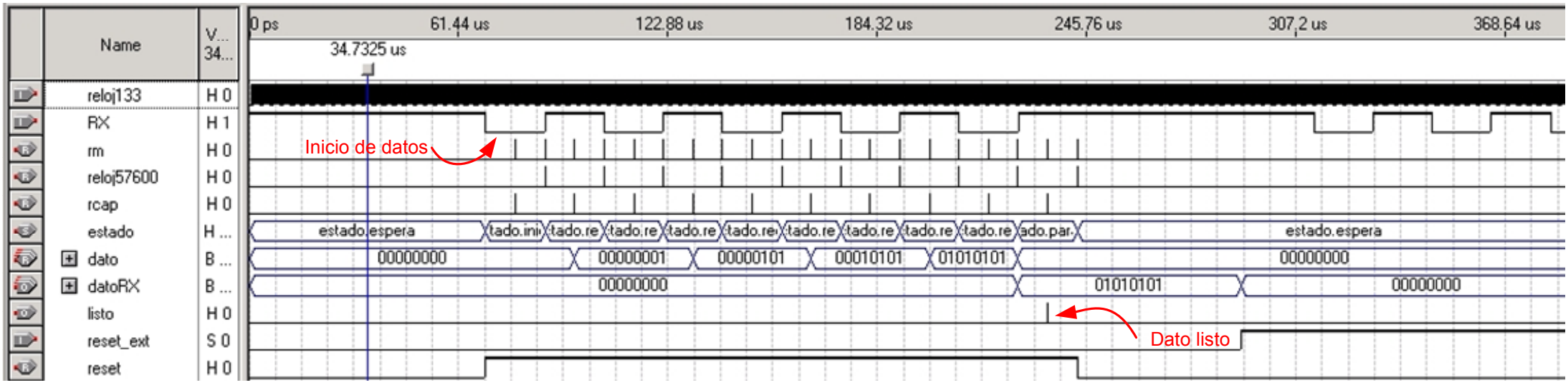


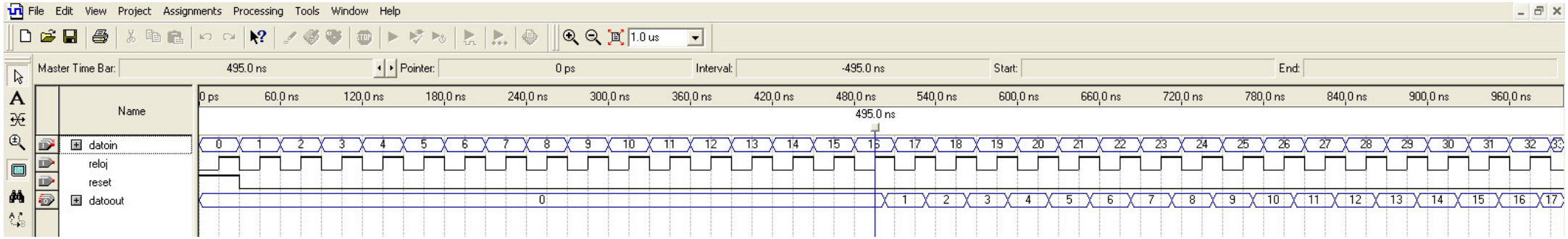
Fig. 1 : Wave showing read operation.

The above waveform shows the behavior of the design under normal read conditions. The read happens at the rising edge of the enabled clock cycle. The output from the RAM is undefined until after the first rising edge of the read clock.

Receptor serial



Retardador

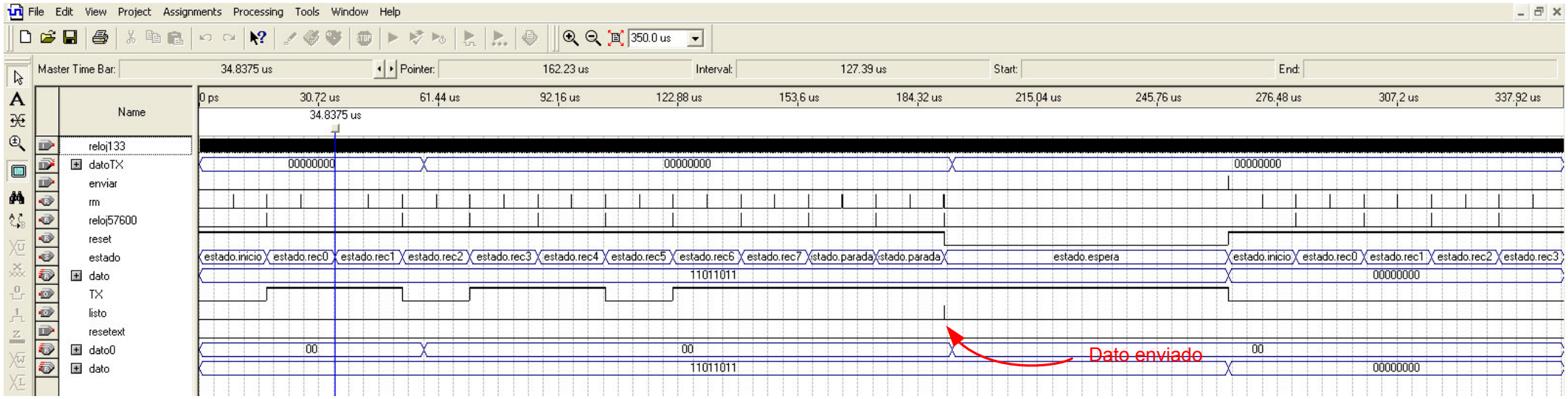

File Edit View Project Assignments Processing Tools Window Help
1.0 us
Master Time Bar: 495.0 ns
Pointer: 0 ps
Interval: -495.0 ns
Start:
End:
Name
0 ps
60.0 ns
120.0 ns
180.0 ns
240.0 ns
300.0 ns
360.0 ns
420.0 ns
480.0 ns
540.0 ns
600.0 ns
660.0 ns
720.0 ns
780.0 ns
840.0 ns
900.0 ns
960.0 ns
495.0 ns
datoin
reloj
reset
datoout

<internal_error>
<executable>quartus.exe</executable>
<sub_system>AFC</sub_system>
<file>./afc_app.cpp</file>
<line>2535</line>
<error>Unhandled exception detected</error>
<date>Sat Jun 26 11:15:11 2004</date>
<version>Quartus II Version 4.0 Build 190 1/28/2004 SJ Full Version</version>
<full_error>Internal Error: Sub-system: AFC, File: ./afc_app.cpp, Line: 2535
Unhandled exception detected
Quartus II Version 4.0 Build 190 1/28/2004 SJ Full Version </full_error>
</internal_error>
<internal_error>
<executable>quartus.exe</executable>
<sub_system>CSET</sub_system>
<file>./cset_signaltap_page.cpp</file>
<line>1403</line>
<error>parent != NULL</error>
<date>Sat Jun 26 13:36:39 2004</date>
<version>Quartus II Version 4.0 Build 190 1/28/2004 SJ Full Version</version>
<full_error>Internal Error: Sub-system: CSET, File: ./cset_signaltap_page.cpp, Line: 1403
parent != NULL
Quartus II Version 4.0 Build 190 1/28/2004 SJ Full Version </full_error>
</internal_error>
<internal_error>
<executable>quartus.exe</executable>
<sub_system>AFC</sub_system>
<file>./afc_app.cpp</file>
<line>2535</line>
<error>Unhandled exception detected</error>
<date>Sat Jun 26 13:36:42 2004</date>
<version>Quartus II Version 4.0 Build 190 1/28/2004 SJ Full Version</version>
<full_error>Internal Error: Sub-system: AFC, File: ./afc_app.cpp, Line: 2535
Unhandled exception detected
Quartus II Version 4.0 Build 190 1/28/2004 SJ Full Version </full_error>
</internal_error>

<internal_error>
<executable>quartus.exe</executable>
<sub_system>AFC</sub_system>
<file>./afc_app.cpp</file>
<line>2535</line>
<error>Unhandled exception detected</error>
<date>Fri Jun 11 19:46:27 2004</date>
<version>Quartus II Version 4.0 Build 190 1/28/2004 SJ Full Version</version>
<full_error>Internal Error: Sub-system: AFC, File: ./afc_app.cpp, Line: 2535
Unhandled exception detected
Quartus II Version 4.0 Build 190 1/28/2004 SJ Full Version </full_error>
</internal_error>

<internal_error>
<executable>quartus.exe</executable>
<sub_system>AFC</sub_system>
<file>./afc_app.cpp</file>
<line>2535</line>
<error>Unhandled exception detected</error>
<date>Tue Jun 29 10:40:23 2004</date>
<version>Quartus II Version 4.0 Build 190 1/28/2004 SJ Full Version</version>
<full_error>Internal Error: Sub-system: AFC, File: ./afc_app.cpp, Line: 2535
Unhandled exception detected
Quartus II Version 4.0 Build 190 1/28/2004 SJ Full Version </full_error>
</internal_error>

Transmisor serial



Unidad de control