UNIVERSIDAD MICHOACANA DE SAN NICOLÁS DE HIDALGO

FACULTAD DE INGENIERÍA ELÉCTRICA

Diseño e Implementación De Un Controlador De Memoria SDRAM Para Un Procesador De Imagen Por Medio De Un FPGA

TESIS

QUE PARA OBTENER EL TÍTULO DE
INGENIERO EN ELECTRÓNICA

PRESENTA
LUIS ANTONIO GUZMÁN MONGE

ASESOR
M.C. ALBERTO CARLOS SALAS MIER

Febrero 2017
MORELIA, MICHOACÁN

Índice General

Índice de Figuras

Índice de Tablas

Agradecimientos

Primeramente quiero agradecer a Dios por las excelentes personas que ha puesto en mi camino y que gracias a todos ellos he llegado hasta donde estoy. De igual forma quiero agradecer infinitamente a mis padres que siempre me ha apoyado a cumplir todas mis metas, su trabajo y dedicación han sido un ejemplo para mí. Siempre me motivaron a ser mejor y a trabajar por lo que quiero. A ellos, muchas gracias.

También quiero agradecer a mi hermana que ha sido ejemplo de constancia, dedicación y trabajo duro. A mi tía Mari que ha sido como una segunda madre para mí y me ha apoyado en cada momento y siempre ha estado ahí para mí.

A todos mis amigos con los que he vivido momentos increíbles, su apoyo y su compañía a través de este camino han sido de gran ayuda para poder concluir este logro personal, les agradezco por todas las buenas charlas y sus consejos.

Quiero también agradecer a todos mis profesores, de cada uno de ellos aprendí algo en particular, todos dejan una huella en mí y agradezco su dedicación y su vocación, gracias a eso logro hoy cumplir esta meta. En especial quiero agradecer a mi asesor de tesis, el Maestro Alberto Carlos Salas Mier por todo el apoyo que me brindó desde el servicio social hasta elegir este tema de tesis y su apoyo para desarrollarla, sus consejos y conocimientos fueron fundamentales para poder realizar este trabajo.

Agradezco a la Facultad de Ingeniería Eléctrica por la oportunidad que me dio de estudiar una carrera y encontrar una de mis pasiones en la vida que es la electrónica.

Finalmente y no menos importante agradezco a la Universidad Michoacana de San Nicolás de Hidalgo por el apoyo académico y económico que me dio para poder finalizar mi formación profesional, siempre me sentiré orgulloso de ser “Nicolaita”.

A todos ellos muchas gracias.

Resumen

El proyecto en general aspira a crear un procesador de imagen que pueda mostrar un histograma en tiempo real y pueda detectar ciertas características de la imagen las cuales son de interés para el usuario.

Una parte fundamental dentro de este procesador de imagen es el manejo de los datos, estos deben ser almacenados dentro de una memoria, en este caso una SDRAM (Synchronized Dynamic Random Access Memory). Por lo que es necesario un controlador. Esta tesis se centra en el diseño y la implementación de un controlador de memoria que formará parte del procesador de imagen. Éste controlador de memoria se creará con las necesidades específicas para el proyecto y en particular para la SDRAM IS42S16400J-6TL.

En esta tesis se hace uso de un FPGA (Field Programmable Gate Array), El cual tiene la ventaja de ser un hardware "programable", (un circuito genérico donde su funcionabilidad puede ser programada para una aplicación en particular), lo cual ayudará a ahorrar recursos valiosos que con una computadora de propósito general se desperdiciarían, además de que es relativamente barato.

Palabras clave: SDRAM, FPGA, Verilog, Quartus II, controlador.

Abstract

This project intends to create an image processor that can show a real time histogram and can detect some particular characteristics in the images that are useful for the user.

A fundamental part within this image processor the handling of the data. The data must be stored inside a memory, in this case an SDRAM (Synchronized Dynamic Random Access Memory). To handle this memories a controller is required. This particular thesis focuses in the design and the implementation of this memory controller that will be part of the image processor. This memory controller will be created with the specifications required for this project, in particular for the SDRAM IS42S16400J-6TL.

In this thesis an FPGA (Field Programmable Gate Array) is used. The advantages of this device is that this is a "programmable" hardware, a generic circuit where its functionality can be programmed for a particular application. This will help save valuable resources that with a general purpose computer would be wasted, plus it's relatively cheap.

Capítulo 1

1. Introducción

1.1. Introducción

En este capítulo se presenta una reseña histórica de los sistemas digitales y su evolución con el paso del tiempo, se describe la metodología que se usa; se presenta la justificación; se definen los objetivos; y finalmente se describe brevemente como se estructura esta tesis.

En el procesamiento de imagen, el manejo de la información es fundamental. Para poder procesarla es necesario su almacenamiento y es por esto que se necesita de una memoria. Existen varios tipos de memorias pero en este caso se utilizará una memoria SDRAM. Las memorias SDRAM necesitan de un controlador que inicialice la memoria y administre la manera en que se lee y se escribe la información. Esta tesis se diseña e implementa este controlador. Para esto se utilizan ciertas herramientas, una tarjeta de desarrollo con un FPGA y el software necesario para poder programar el FPGA.

1.2. Revisión Histórica

La evolución de los sistemas digitales en los últimos años es cada vez mayor y basándose en la ley de Moore se puede decir que cada 2 años se duplica el número de transistores en un circuito integrado, lo que permite aumentar cada vez más la capacidad de procesamiento (Brown & Vranesic, 2008).

En 1961 James L. Buie de TRW Inc inventa la tecnología TTL (Transistor-transistor logic) la cual se comenzó a utilizar en circuitos digitales. Esta tecnología conforme fue evolucionando permitió el desarrollo de las computadoras y en general de la electrónica digital (Laws, 2016) .

Fue a inicios de los años 70's cuando se trabaja por reemplazar esta tecnología y se crea la *Programmable Read Only Memory* (PROM) considerado el primer dispositivo programable. En esta misma década, *National Semiconductors* se vuelve pionero en el desarrollo del *Programmable Logic Array* (PLA) el cual es un conjunto de compuertas AND programables que se conectan a un juego de compuertas OR programables (Herrera Lozada, 2016).

En 1978 se introducen los dispositivos de *Programmable Array Logic* (PAL) los cuales poseen una arquitectura más sencilla que los dispositivos PLD al omitir las compuertas OR programables (Semiconductor Special Interest Group, 2016). El siguiente paso en la evolución de estos dispositivos fueron los dispositivos *Generic Array Logic* (GAL) inventados por *Lattice Semiconductor* en el año de 1985, la ventaja de estos dispositivos era que a diferencia de las PAL estos podían borrarse y reprogramarse (Lattice Semiconductor Corporation, 1998). Los dispositivos PAL y GAL solo cuentan con una pequeña cantidad de compuertas lógicas, por lo tanto para aplicaciones donde se requiria una cantidad mucho mayor de compuertas fueron creadas las *Complex Programmable Logic Device* (CPLDS) las cuales pueden tener miles de compuertas lógicas (Floyd, 2006).

En 1985 los cofundadores de Xilix, Ross Freeman y Bernard Vonderschmitt crearon la primera FPGA comercial (Muthuswamy & Banerjee, 2015). Desde la década de los 90's la industria de los FPGA tuvo un incremento exponencial. Actualmente son usados principalmente en las telecomunicaciones, la industria automotriz y varias aplicaciones industriales.

Las grandes ventajas de los FPGA son:

- Su rendimiento puede ser mucho mayor al de los DSPs, debido a su paralelismo de hardware.
- La flexibilidad de desarrollar prototipos y probarlos en hardware antes de pasar al proceso de fabricación.
- Su precio es mucho menor que el de un ASIC (*Application-Specific Integrated Circuit*).
- Al no necesitar sistemas operativos, mejoran la fiabilidad con ejecución paralela y hardware preciso dedicado a cada tarea.
- Por ser reconfigurables es fácil su actualización y es mucho más simple reconfigurar un FPGA que rediseñar un ASIC. (National Instruments, 2016).

1.3. Metodología

Se comienza por realizar una investigación en general de los tipos de memorias y sus clasificaciones para posteriormente centrarse en las memorias SDRAM.

Se estudian más a detalle las características de las memorias SDRAM con el objetivo de comprender correctamente su funcionamiento y poder diseñar un controlador de memoria.

Se analiza la plataforma con la que se trabajará, en este caso una tarjeta de desarrollo con un FPGA para de igual manera comprender su funcionamiento y poder utilizarla de la mejor manera posible.

Posteriormente se estudian los lenguajes de descripción de hardware y al software que se utilizará para manipular la FPGA.

Para realizar el diseño del controlador se estudia la hoja de datos de la memoria con la cual se trabaja y en base a las características de esta y de su funcionamiento se realiza un diseño y se implementa en la tarjeta de desarrollo.

Se realizan simulaciones de la implementación y en caso de encontrar errores se corrigen estos hasta finalmente obtener un controlador funcional.

1.4. Justificación

El proyecto principal pretende crear un procesador de imagen en un sistema embebido reconfigurable que servirá como guía para los estudiantes de la Facultad de Ingeniería Eléctrica en la introducción al uso de FPGA’s y el procesamiento de imágenes. Para poder crear este procesador son necesarias varias etapas donde una de ellas es el diseño del controlador de memoria. Este trabajo se centra en la creación de este controlador de memoria con el cual se podrá acceder a una memoria SDRAM y almacenar y leer información según las necesidades del usuario.

1.5. Objetivos

El objetivo de la tesis es diseñar, simular e implementar un controlador de memoria para la SDRAM IS42S16400J-6TL con el FPGA de Altera EP3C16F484C6N, posteriormente se almacenará una imagen en la memoria y se desplegará en un monitor VGA.

1.6. Descripción de los Capítulos

En el capítulo 1 se da una breve introducción histórica a los FPGA y se aclaran los objetivos y la justificación de esta tesis. En capítulo 2 trata de los fundamentos teóricos de las memorias, las máquinas de estados, los tipos de memoria y su funcionamiento, en particular de la SDRAM. En el capítulo 3 se da una introducción a la descripción del software y el hardware elegido. El capítulo 4 habla sobre el diseño y la implementación de los módulos para el controlador de memoria. Finalmente en el capítulo 5 se dan las conclusiones y el trabajo futuro a realizarse.

Capítulo 2

2. Marco Teórico

2.1. Sistemas Digitales

Los sistemas digitales son una combinación de dispositivos diseñados para manipular cantidades físicas o información que se represente de manera digital; es decir, que solo puede tomar valores discretos (Tocci & Widmer, 2003).

2.1.1. Circuitos Digitales

Los circuitos digitales manejan señales digitales las cuales son discretas a diferencia de los circuitos analógicos que tienen señales continuas, es decir, las señales digitales solo pueden tener dos valores, *verdadero* o *falso*, estos valores generalmente se representan con dos valores distintos de voltaje, el *falso* generalmente es un valor de voltaje cercano a la referencia o a 0V y el *verdadero* es normalmente un valor cercano al voltaje de alimentación.

La manera en que un circuito digital responde a una entrada es conocida como lógica del circuito, es por esto que los circuitos digitales también son conocidos como *circuitos lógicos*. (Tocci & Widmer, 2003)

2.1.2. Memoria

En general cuando a un circuito lógico se le aplica una señal de entrada, la salida varía en función de la entrada pero al retirar la señal de entrada, la señal de salida vuelve a su forma original. Este tipo de circuitos no presenta la propiedad de *memoria* ya que las señales de salida vuelven a su estado original. Cuando una señal se aplica a un circuito y la señal de salida se modifica y se mantiene aun cuando se retira la señal de entrada, se dice entonces que se tiene un circuito con *memoria.*

Los dispositivos con memoria son de gran importancia en los sistemas digitales ya que con éstos se pueden almacenar números binarios de forma temporal o permanente, y se puede modificar la información almacenada en cualquier instante (Tocci & Widmer, 2003).

2.1.3. Circuitos Lógicos Combinacionales

Son aquellos sistemas digitales en los que las salidas solo dependen del valor de la combinación de entradas en un momento determinado. En estos sistemas no afectan los estados anteriores o las salidas que se obtengan, es decir que no poseen la característica de memoria (Tocci & Widmer, 2003).

2.1.4. Circuitos Lógicos Secuenciales

En este tipo de sistemas, los valores de las salidas en un instante determinado dependen de la combinación lógica en las entradas y además del estado anterior del sistema o del estado actual. También son conocidos como *máquinas de estados.*

2.2. Máquinas de Estados

Las máquinas de estados o circuitos secuenciales se forman por una etapa lógica combinacional y una etapa de memoria, algunos pueden tener una señal de reloj los cuales son conocidos como síncronos, en caso de no tener esta señal de reloj se conocen como asíncronos. Las máquinas de estados pueden tener varias entradas y salidas las cuales determinan el camino que la máquina seguirá entra los distintos estados que pueda tener.

En la memoria de la máquina de estados se tiene siempre un estado, (considerado como estado actual) el cual en el caso de las máquinas de estados síncronas, cambia a un estado distinto cuando se da un impulso de reloj y dependiendo de las entradas y del estado actual la máquina pasa a un nuevo estado modificándose así las salidas.

Para diseñar una máquina de estados primeramente se necesita de un diagrama de estados el cual indica los distintos estados que se tendrán. De igual manera es necesaria tener una tabla de estados futuros en la cual se enumera cada estado y el estado al que cambia.

Existen dos tipos básicos de máquinas de estados. La máquina de estados de Moore donde la salida solo depende del estado interno y de algún reloj y la máquina de estado

Mealy donde las salidas están determinadas por el estado interno y por entradas que no están sincronizadas con el circuito (Floyd, 1997).

2.3. Tipos de Memorias

Las memorias son dispositivos de almacenamiento de datos binarios. Existen memorias semiconductoras, magnéticas y ópticas. Las memorias semiconductoras se forman por matrices de elementos de almacenamiento los cuales pueden ser latches, capacitores o cualquier elemento de carga (Floyd, 1997).

2.3.1. Memorias semiconductoras

Los principales tipos de memorias semiconductoras son las memorias volátiles y las no volátiles. Las memorias volátiles son aquellas que al interrumpirse su suministro de energía eléctrica pierden toda la información. De forma contraria las memorias no volátiles almacenan la información incluso si no están energizadas.

Estas memorias están construidas con semiconductores usándose principalmente dos tipos de tecnologías; la bipolar y la MOS. La tecnología MOS (Metal-Oxide Semiconductor, en español metal-óxido semiconductor), utiliza MOSFETs (Metal-oxide-semiconductor Field-effect transistor, en español transistor de efecto de campo metal-óxido semiconductor) la cual es la tecnología que más se utiliza en las memorias debido a su bajo costo y a que disipan menos potencia. Las memorias bipolares ya casi no se utilizan pero existe la tecnología BiMOS la cual es una combinación de bipolar y MOS (Floyd, 1997).

2.3.1.1. RAM

Las memorias RAM (Random Access Memory) son memorias en las que se puede escribir y leer rápidamente en la dirección que se quiera en el orden que se desee. La RAM se utiliza para almacenamiento de datos a corto plazo, ya que estos se pierden cuando la memoria se desconecta de la alimentación, por esto se conocen como memorias volátiles.

Las memorias RAM a su vez se subdividen en SRAM (Static RAM, RAM estática) y DRAM (Dynimic RAM, RAM dinámica). La SRAM usa elementos de almacenamiento como latches lo que permite almacenar información por un periodo indefinido de tiempo siempre y cuando esté conectada la alimentación. Las memorias DRAM almacenan los datos en capacitores, los cuales necesitan recargarse de manera periódica para mantener la información.(Floyd, 1997).

En la Figura 2.1 se observa un chip de una memoria RAM.

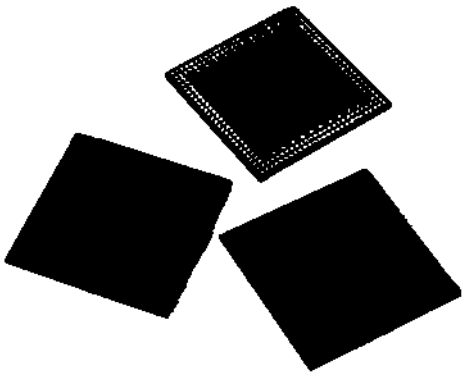

Figura 2.1 Chip de memoria RAM. (tecnomagazine).

2.3.1.2. ROM

Las memorias ROM (Read Only Memory) solo permiten la lectura debido a que los datos se almacenan de manera permanente o semipermanente. Los datos almacenados no se pueden modificar o se requiere de algún equipo especial para hacerlo. Los datos que se almacenan en la ROM son datos que se utilizan constantemente en ciertas aplicaciones, pueden ser tablas, conversiones, instrucciones programadas para la inicialización de algún sistema, etc.

Las memorias ROM son conocidas como memorias no volátiles ya que la información se almacena en estas incluso si se desconecta la alimentación.

Existen varios tipos de memorias ROM; están las PROM (Programmable Read Only Memory) la cual se puede modificar con cierto equipo. La EPROM (Erasable Programmable Read Only Memory) permite borrar los datos y existen dos tipos de EPROM, la UV-EPROM (Ultra-Violet Erasable Programmable Read Only Memory) y la EEPROM (Electrically Eresable Programmable Read Only Memory).

La UV-EPROM permite grabarle datos eléctricamente pero para borrar estos datos es necesaria la exposición a luz ultravioleta por un periodo de varios minutos. La EEPROM permite borrar los datos de manera eléctrica y se pueden borrar en pocos milisegundos (Floyd, 1997).

En la Figura 2.2 se puede ver una memoria UV-EPROM.

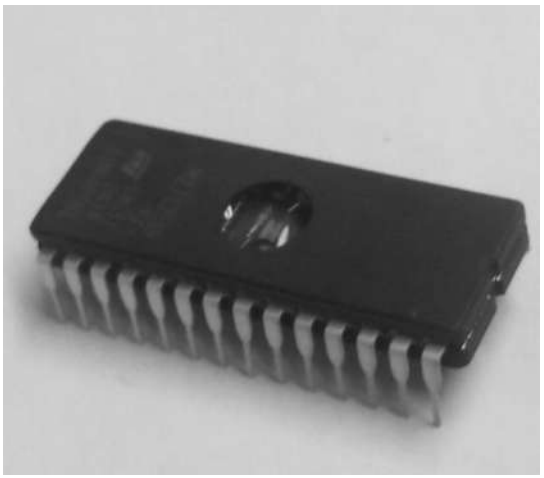

Figura 2.2 Memoria ROM M27C256B.

2.3.1.3. Memoria Flash

Las memorias flash son memorias no volátiles, tienen capacidad de lectura y escritura y tienen una gran rapidez de operación y una excelente relación calidad-precio.

Son memorias de alta densidad, lo que permite almacenar una gran cantidad de información en chips de tamaño muy pequeño. Estas memorias están formadas por un único transistor MOS de puerta flotante (Floyd, 1997).

En la Figura 2.3 se puede observar un chip de memoria flash.

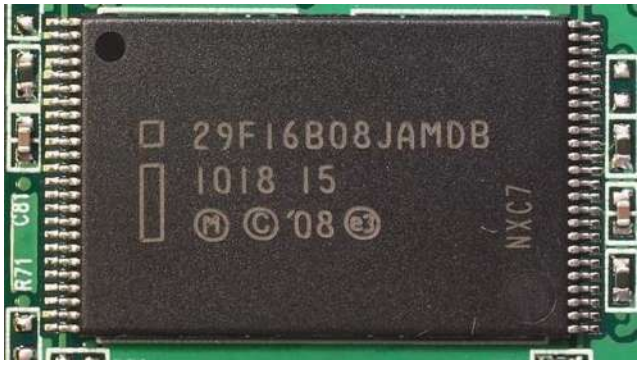

Figura 2.3 Chip de memoria flash. (www.abc.es).

2.3.2. Memorias Ópticas y Magnéticas

Este tipo de dispositivos se utiliza principalmente en sistemas de ordenadores para almacenamiento masivo de datos. Las memorias magnéticas tienen un tiempo de lectura mayor que las memorias semiconductoras por lo que su aplicación se limita a almacenamiento permanente o semipermanente de grandes cantidades de datos. Las memorias magnéticas se subdividen en tres categorías básicas: disco magnético, disco magneto-óptico y cinta. (Floyd, 1997)

En la Figura 2.4 se puede observar un disco duro magnético.

Figura 2.4 Hard Disk Drive. (www.pcactual.com).

En los últimos años han surgido nuevas tecnologías como el SSD (Solid State Drive) que actualmente ya están remplazando a los HDD (Hard Disk Drive) en varias aplicaciones.

2.4. Memoria SDRAM

La memoria SDRAM (Synchronous Dynamic Random Access Memory) es de la familia de las memorias DRAM. La diferencia con las memorias DRAM es que las SDRAM trabajan con una señal de reloj por lo que está sincronizada con el bus del sistema.

La SDRAM a grandes rasgos como se muestra en la Figura 2.5 tiene un bus de datos, un bus de direcciones y un bus de control. El bus de datos es bidireccional por lo que es a través de este bus que se leen y escriben los datos. Con el bus de dirección se decide a

que bloque de la memoria se quiere acceder eligiendo las filas y las columnas deseadas. El bus de control da los comandos necesarios para que la memoria opere de manera correcta.

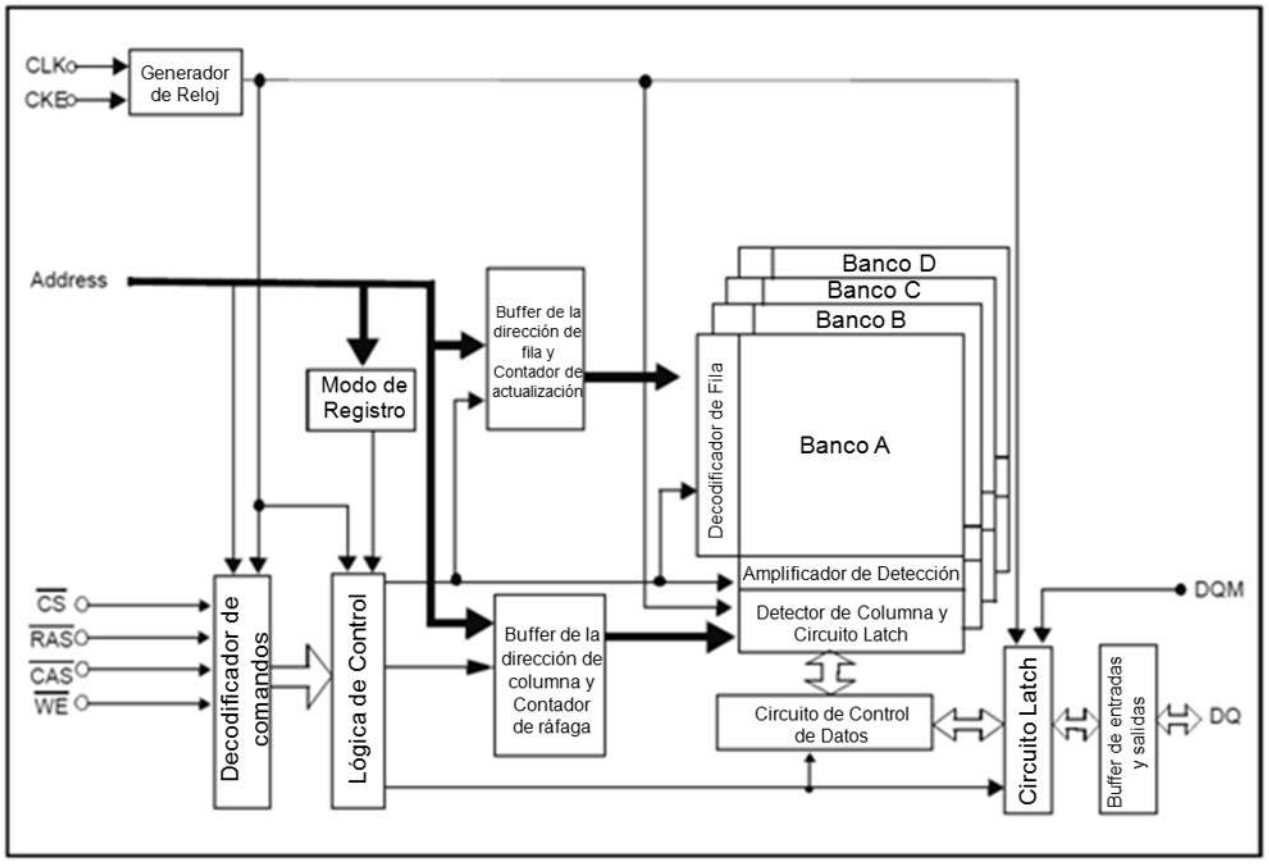

Figura 2.5 Diagrama Funcional de Bloques.

Las SDRAM se organizan en matrices de celdas, es decir, se tiene un número de bits para las direcciones de filas y otro número de bits para las direcciones de las columnas. Las memorias se distribuyen en bancos los cuales cuentan con una cantidad determinada de filas y columnas y cada celda tiene un tamaño determinado dependiendo de la memoria. Cada bit de memoria en la SDRAM es un transistor y un capacitor, el transistor suministra la carga y el capacitor almacena el estado de cada bit. La carga en los capacitores desaparece después de cierto tiempo, es por esto que se debe estar refrescando esta carga constantemente (Anon., 2016).

2.4.1. Descripción de Señales

Las memorias SDRAM tienen 8 señales de entrada y dos de entrada/salida, las cuales se pueden observar en la figura 2.5 y se describen a continuación:

- **CLK** (Clock, reloj): Es la señal de reloj que maneja a todo el sistema, todas las demás señales se referencian con los flancos de subida de esta señal.
- **CKE** (Clock Enable, habilitador del reloj): CKE activa (HIGH, alto) o desactiva (LOW, bajo) la señal de CLK. Esto es útil para activar o desactivar ciertos comandos.
- $\overline{\mathbf{CS}}$ (Chip Select, Seleccionar chip): $\overline{\text{CS}}$ activa (LOW) y desactiva (HIGH) el decodificador de comandos. Es decir cuando esta señal tiene un nivel ALTO todos los comandos son ignorados.
- $\overline{\mathbf{CAS}}$(Column Address Strobe, selección de dirección de columna): Con esta señal como su nombre lo dice permite elegir la dirección de la columna que se usará.
- $\overline{\mathbf{RAS}}$ (Row Addres Strobe, selección de dirección de fila): De igual manera como su nombre lo dice esta señal permite elegir la dirección de la fila que se utilizará.
- $\overline{\mathbf{WE}}$ (Write Enable, habilitar escritura): Junto con $\overline{\text{CS}}$ define el comando a ejecutar.
- **DQM** (Input/Output Mask, máscara de entrada/salida): Esta señal desactiva la entrada de datos para la escritura y los de salida para la lectura. DQM tiene un nivel ALTO en el ciclo de escritura y las líneas de salida se colocan en alta impedancia cuando DQM tiene un nivel ALTO durante un ciclo de lectura.
- **Address** (Address Inputs, entradas de dirección): Con este bus de señal se proporciona la dirección de la fila cuando se tiene el comando ACTIVE, la dirección de la columna y el bit de AUTO PRECHARGE para los comandos READ/WRITE. (La dirección de las filas se especifica de A0-A11 y la dirección de las columnas se especifica de A0-A7, esto en particular para la memoria IS42S16400). De igual forma permite definir en qué banco de memoria se trabajará.
- **DQ** (Data Inputs, entrada de datos): Este es el bus bidireccional de datos.

2.4.2. Diagrama de Estados

El funcionamiento de las memorias SDRAM puede ser comprendido mejor de manera muy general si se observa su diagrama de estados en la figura 2.6. Inmediatamente después de que la memoria SDRAM se energiza, se debe realizar una inicialización, posteriormente la memoria entra a un estado IDLE donde se mantiene hasta que la memoria recibe el comando ACT el cual prepara la memoria para escribir o leer, posteriormente la memoria espera la instrucción de WRITE (escribir) o READ (leer) a continuación la memoria pasa al estado de PRECHARGE y de manera automática pasa de nuevo al estado de IDLE.

2.4.3. Tabla de Comandos

Para realizar la transición entre los estados, la memoria SDRAM requiere de ciertos comandos los cuales se introducen a través de las entradas con las que cuenta la memoria. En la Tabla 2-1 se ven los comandos con los que cuenta esta memoria SDRAM.

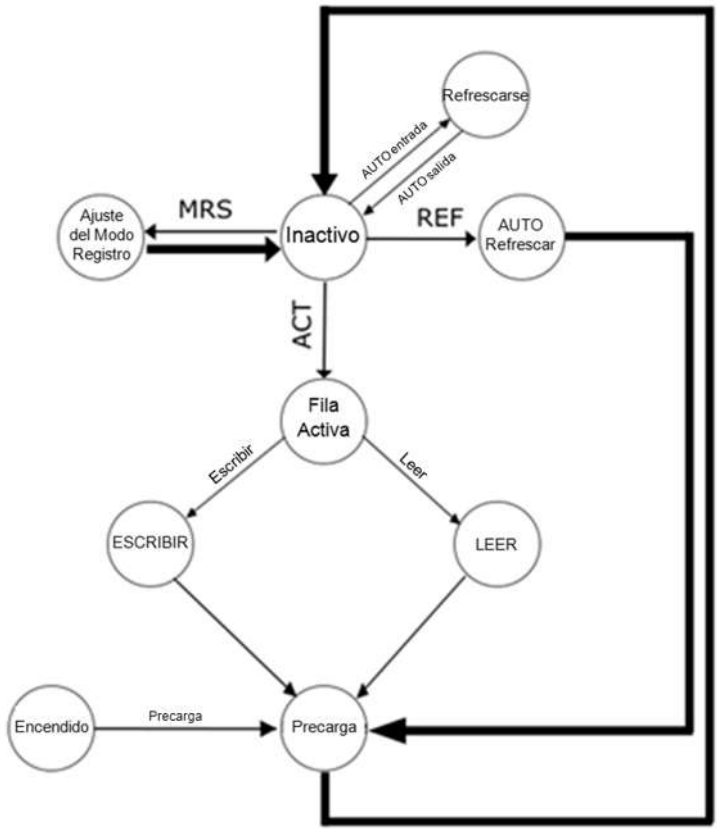

Figura 2.6 Diagrama de estados simplificado de una memoria SDRAM.

Tabla 2-1 Tabla de Comandos de la SDRAM IS42S16400J-6TL

Símbol	Comando	CKE n-1	CKE n	$\overline{CS}$	$\overline{RAS}$	$\overline{CAS}$	$\overline{WE}$	BA	A10	A11 A9-A0
DESL	Selector de Dispositivo	H	X	H	X	X	X	X	X	X
NOP	No operación	H	X	L	H	H	H	X	X	X
MRS	Ajuste del Modo Registro	H	X	L	L	L	L	L	L	V
ACT	Banco activo	H	X	L	L	H	H	V	V	V
READ	Leer	H	X	L	H	L	H	V	L	V
READA	Leer con Precarga	H	X	L	H	L	H	V	H	V
WRIT	Escribir	H	X	L	H	L	L	V	L	V
WRITA	Escribir con Precarga	H	X	L	H	L	L	V	H	V
PRE	Precargar el banco seleccionado	H	X	L	L	H	L	V	L	X
PALL	Precargar todos los bancos	H	X	L	L	H	L	X	H	X
BST	Alto de ráfaga	H	X	L	H	H	L	X	X	X
REF	CBR (Auto) Refrescar	H	H	L	L	L	H	X	X	X
SELF	Refrescarse	H	L	L	L	L	H	X	X	X

Notas:
H: Nivel alto
X: Nivel alto o bajo (No importa)
L: Nivel bajo
V: Dato de entrada válido

Capítulo 3

3. Descripción del Entorno de Trabajo

3.1. FPGA

3.1.1. Arquitectura del Cyclone III

Para este trabajo se utilizó una tarjeta con Cyclone III. Esta FPGA cuenta con una gran densidad de compuertas lógicas, memoria, multiplicadores embebidos, entradas y salidas (I/O) además soporta una gran cantidad módulos externos de memoria y varios protocolos de I/O.

La arquitectura de un FPGA, consta principalmente de 3 elementos: LEs (Logic Elements) que a su vez forman LABs (Logic Array Blocs), interconexiones programables y bloques de entrada/salida.

Los elementos de la FPGA que se utiliza en esta tesis se muestra en la Figura 3.1 y a continuación se explican sus elementos.

- LEs (Logic Elements) y LABs (Logic Array Blocks)

Los LABs consisten de 16 LEs y un LAB de control. Los LEs son la unidad lógica más pequeña dentro de la arquitectura del Cyclone III. Cada LE tiene 4 entradas, una LUT (Look-Up Table) de 4 entradas, un registro y una salida lógica. La LUT es una tabla que determina que salida se tendrá dependiendo de las entradas dadas.

- Interconexiones configurables (MultiTrack Interconnect)

En el Cyclone III las interconexiones entre las LEs, LABs, M9K (bloques de memoria), multiplicadores embebidos y los pines de I/O, está hecho por varias líneas de enrutamiento que pueden funcionar a diferentes velocidades, estas conexiones se generan con el software Quartus II de manera automática para conseguir el diseño más óptimo.

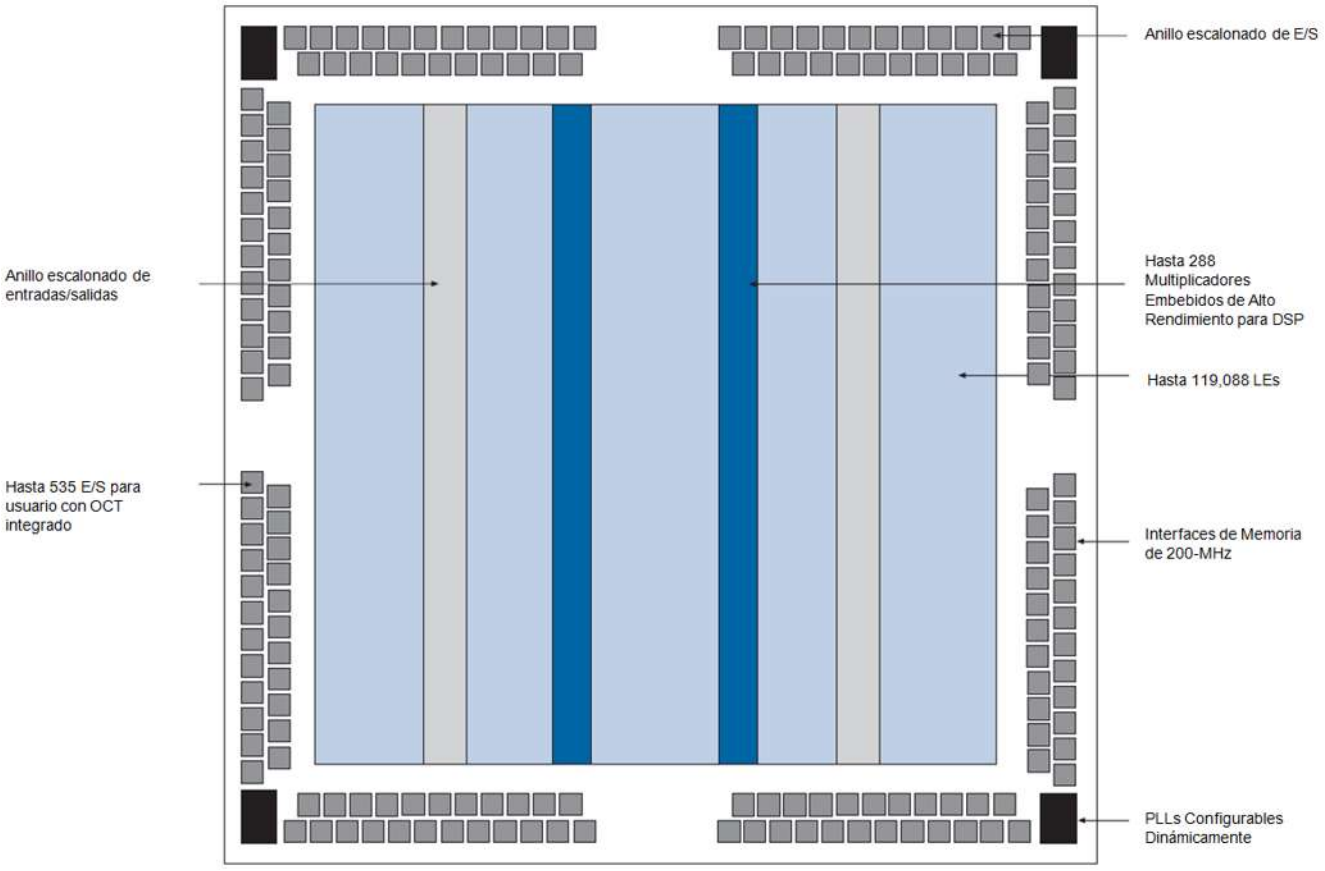

Figura 3.1 Vista general de la arquitectura del dispositivo Cyclone III. (Altera, 2008)

- Bloques de Memoria (Memory Blocks)

Cada Cyclone III tiene varios bloques M9K que es un bloque de memoria de 9 Kbits y está en un chip que puede operar a 315 MHz. La estructura de la memoria embebida consiste de columnas de bloques de memoria M9K que se pueden configurar como RAM, FIFO buffers o ROM. Estos bloques están optimizados para aplicaciones de procesamiento de video y almacenamiento de datos.

- Multiplicadores y Soporte para Procesamiento Digital de Señales

Cyclone III ofrece hasta 288 bloques de multiplicadores embebidos que soportan un multiplicador de 18x18 bit por bloque o dos multiplicadores de 9x9 bit por bloque.

Adicionalmente los FPGA Cyclone III incluyen una combinación de chips y recursos que permiten incrementar el desempeño, reducir costos y consumo de poder en el procesamiento digital de señales (DSP).

- Entradas y Salidas I/O

Los FPGA Cyclone III cuentan con entradas y salidas (I/O) divididas en 8 bancos. Todos estos bancos tienen entradas tanto referenciadas como diferenciales.

- Reloj y PLLs

El Cyclone III cuenta con 20 redes de relojes. Las señales de estos relojes pueden ser utilizada para pines dedicados al reloj, pines de doble propósito que requieren de reloj, lógica del usuario y PLLs.

También cuenta con 4 PLLs con 5 salidas por cada PLL para un mejor control del reloj. Los PLLs pueden ser utilizados en cascada para generar hasta 10 relojes internos y dos externos.

3.1.2. Tarjeta DE0

Para este trabajo de tesis se utilizó la tarjeta DE0, la cual como ya se mencionó cuenta con un FPGA Cyclone III

La tarjeta DE0 cuenta con los dispositivos que se muestran en el diagrama de la Figura 3.2

- FPGA Altera Cyclone III 3C16
- Dispositivo de Configuración Serial Altera
- USB Blaster
- 8-Mbyte SDRAM
- 4-Mbyte Flash memory
- SD Card Socket
- 3 Switches Pushbutton
- 10 Switches on/off
- 10 LEDs verdes
- 4 Displays de 7 segmentos
- Osciladores para fuente de reloj de 50-MHz
- VGA DAC con conector VGA de salida
- Transceptor RS-232
- Conector PS/2
- 2 puertos de expansión de 40 pines

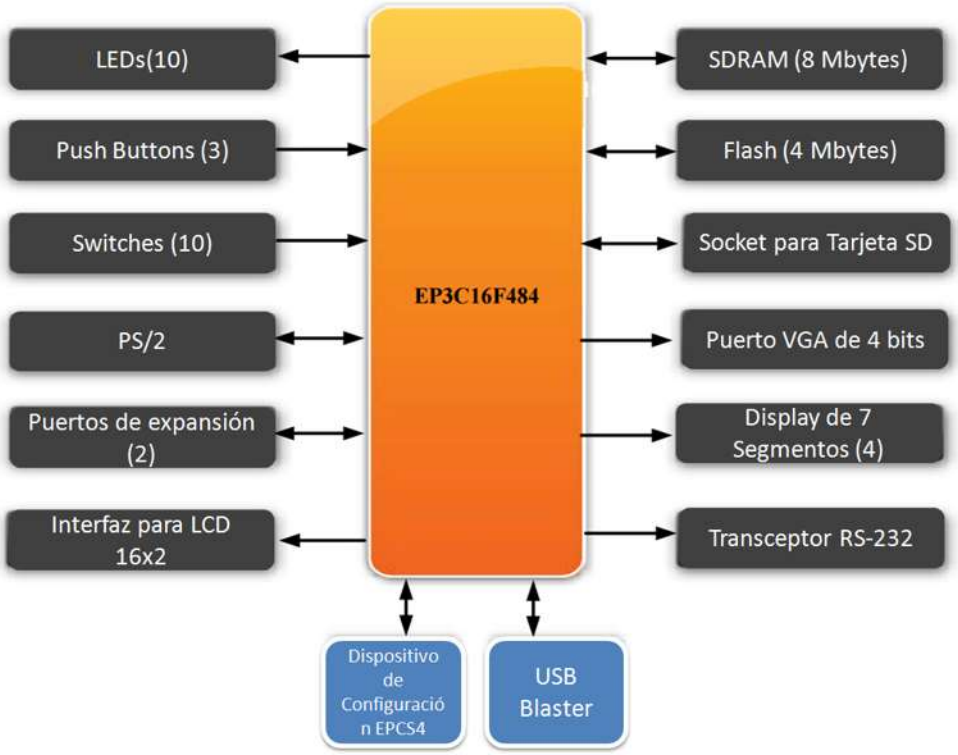

Figura 3.2 Diagrama de bloques de la tarjeta DE0 (Terasic Technologies, 2009).

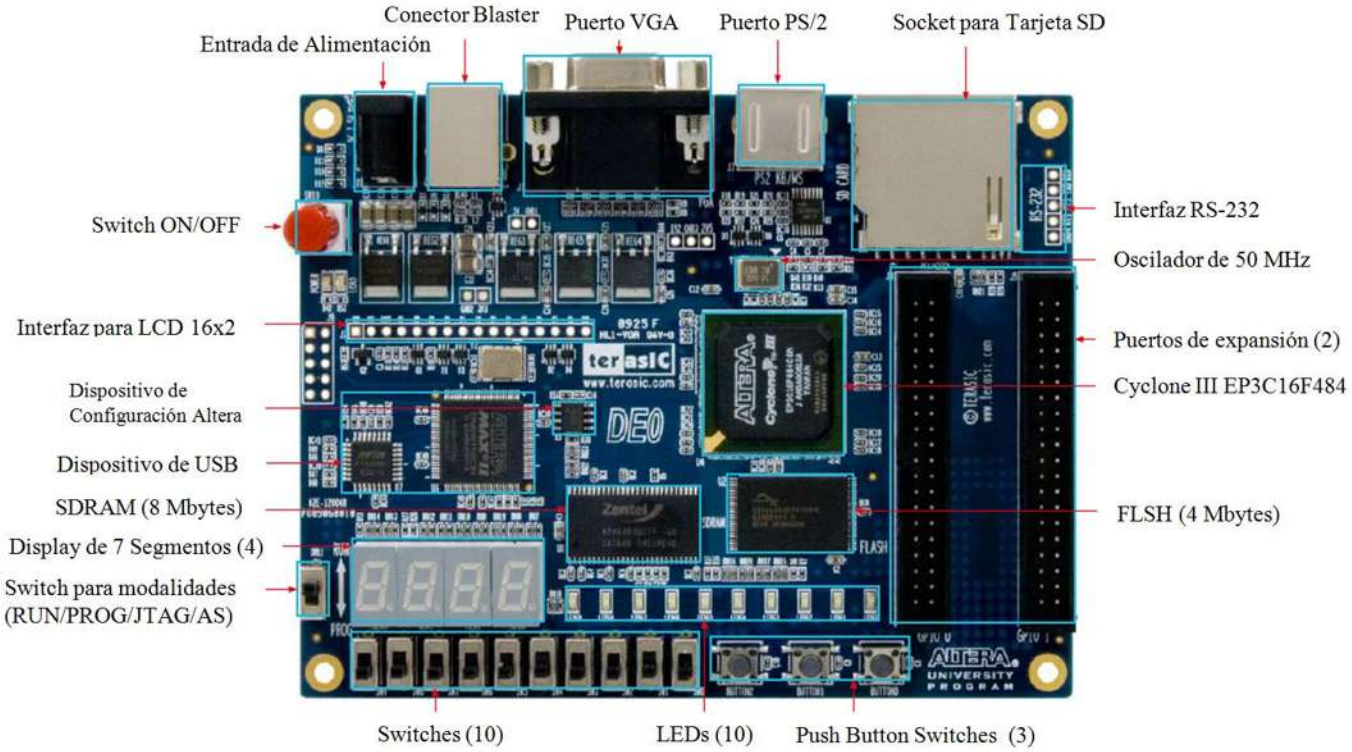

Figura 3.3 Tarjeta DE0 (Terasic Technologies, 2009).

En la Figura 3.3 se muestra una imagen de la tarjeta DE0.

3.2. Verilog

Verilog es un lenguaje de descripción de hardware (HDL, Hardware Description Language) basado en el lenguaje de programación C, pero hay que destacar que no es C. Está diseñado para modelar circuitos digitales, tiene una sintaxis muy similar a C pero su comportamiento es diferente. Usando este lenguaje se puede describir cualquier circuito digital de cualquier nivel, por ejemplo un microprocesador, una memoria o un flip-flop.

En verilog los módulos son como cajas negras que solo tienen entradas, salidas y una lógica interna, un equivalente en otros lenguajes de programación serían las funciones.

3.2.1. Operadores de Verilog

Los operadores que se utilizan en Verilog son los mostrados en la Tabla 3-1.

Tabla 3-1 Operadores de Verilog

Tipo de Operador	Símbolo del Operador	Operación Realizada
Aritmético	*	Multiplicación
	/	División
	+	Suma
	-	Resta
	%	Modulo
	+	Suma Unaria
	-	Resta Unaria
Lógico	!	NOT lógico
	&&	AND lógico
	||	OR lógico
Comparadores	>	Mayor que
	<	Menor que
	>=	Mayor o igual que
	<=	Menor o igual que
	==	Igual que

	!=	Distinto que
Reducción	~	Negación nivel bit
	~&	NAND
	|	OR
	~|	NOR
	^	XOR
	^~	XNOR
	~^	XNOR
Corrimiento	>>	Corrimiento a la derecha
	<<	Corrimiento a la izquierda
Concatenar	{}	Concatenación
Condicional	?	Condicional

3.2.2. Sentencias en Verilog

Las sentencias de control que utiliza Verilog son las siguientes:

- if
- else
- repeat
- while
- for
- case

Las sentencias **"if-else"** se utilizan para ejecutar cierta porción de código si se cumple cierta condición.

La sentencia **"case"** se utiliza cuando se tiene una variable que debe ser evaluada con varios valores. Las sentencias **"case"** inician justamente con la palabra reservada **"case**" y termina con la palabra reservada **"endcase"**.

La sentencia **"while"** ejecuta una porción de código de manera repetida hasta que se cumpla cierta condición.

La sentencia **“for”** es muy similar a como se utiliza en C. La única diferencia es que el ++ y el -- no se pueden utilizar en Verilog.

La sentencia **“repeat”** es muy similar a la sentencia **“for”** pero a diferencia del **“for”** donde primero se inicializa una variable, luego se ejecuta cierta acción mientras se cumpla una condición y finalmente se modifica la variable inicializada, en el caso del “**repeat”** solo se debe de indicar el número de veces que se quiere que se repita cierta porción de código.

En Verilog como en la electrónica digital se pueden utilizar elementos combinacionales o secuenciales, para realizar esto en Verilog se tienen dos sentencias reservadas, **“always”** y **“assign”.**

Para lógica secuencial se utiliza la sentencia **“always”,** lo que hace es que a diferencia de la mayoría de los lenguajes de programación donde cada sentencia se ejecuta una después de otra, cuando se utiliza **“always”** las sentencias se pueden ejecutar en paralelo. Todos los bloques marcados con **“always”** se ejecutarán al mismo tiempo.

Para la lógica combinacional se utiliza la sentencia **“assign”**, esta se utiliza para asignar el comportamiento a una variable siempre y cuando este comportamiento sea de lógica combinacional.

3.3. Quartus II

Para el uso de la tarjeta de desarrollo DE0 es necesario del software de Altera Quartus II Web Edition, el cual se puede conseguir directamente de la página web de Altera (Altera, 2016).

Este software permite la programación en Verilog y pasar este programa a la tarjeta DE0 mediante una simple conexión por puerto USB.

3.3.1. Crear proyecto en Quartus II

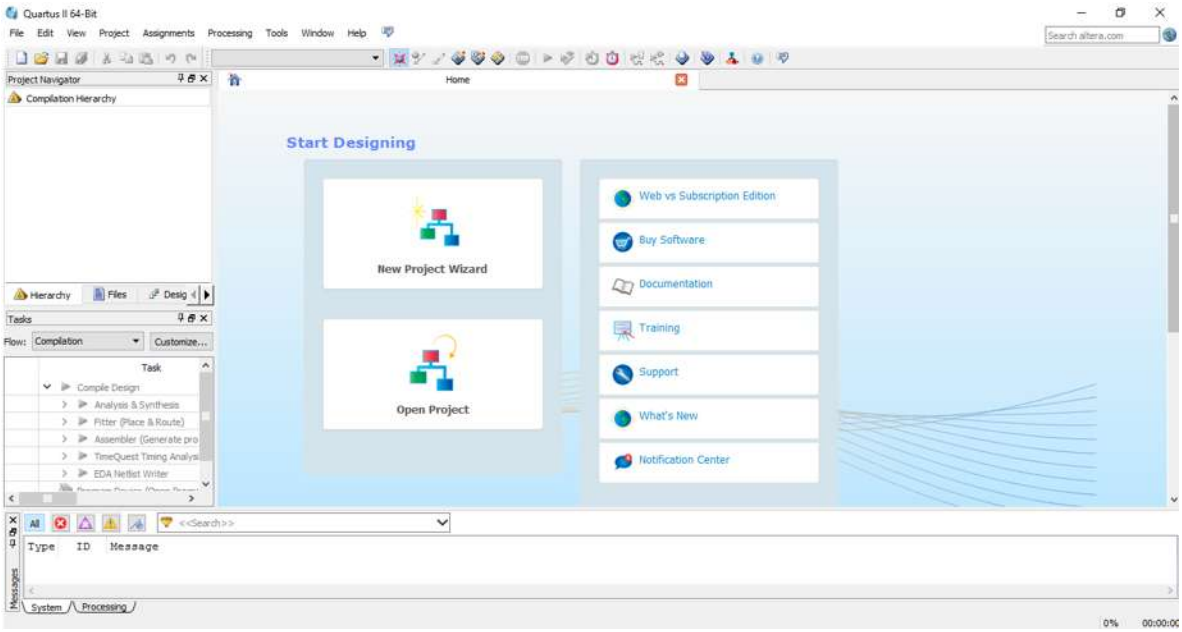

Figura 3.4 Interfaz principal Quartus II 13.1 (64-bit) Web Edition.

Para iniciar un proyecto nuevo se debe ir a la pestaña “File/New Project Wizard…” donde aparecerá una ventana como se ve en la Figura 3.4

Posteriormente se tiene la Figura 3.5 donde se da una breve introducción de la herramienta para crear un nuevo proyecto.

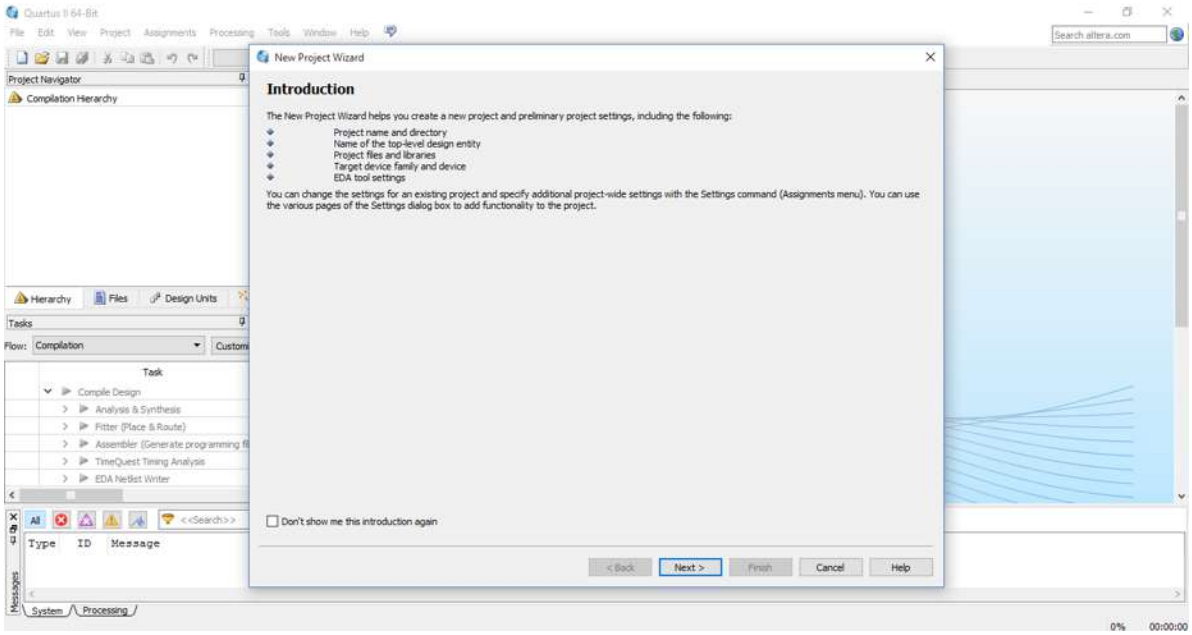

Figura 3.5 Creación de nuevo proyecto en Quartus II 13.1 (64-bit) Web Edition.

Posteriormente como se muestra en la Figura 3.6 se da click en “next >” y aparecerá una nueva ventana donde se elige la carpeta en la cual se guardará el proyecto creado y se elige el nombre del proyecto. A continuación se da click en “next >”.

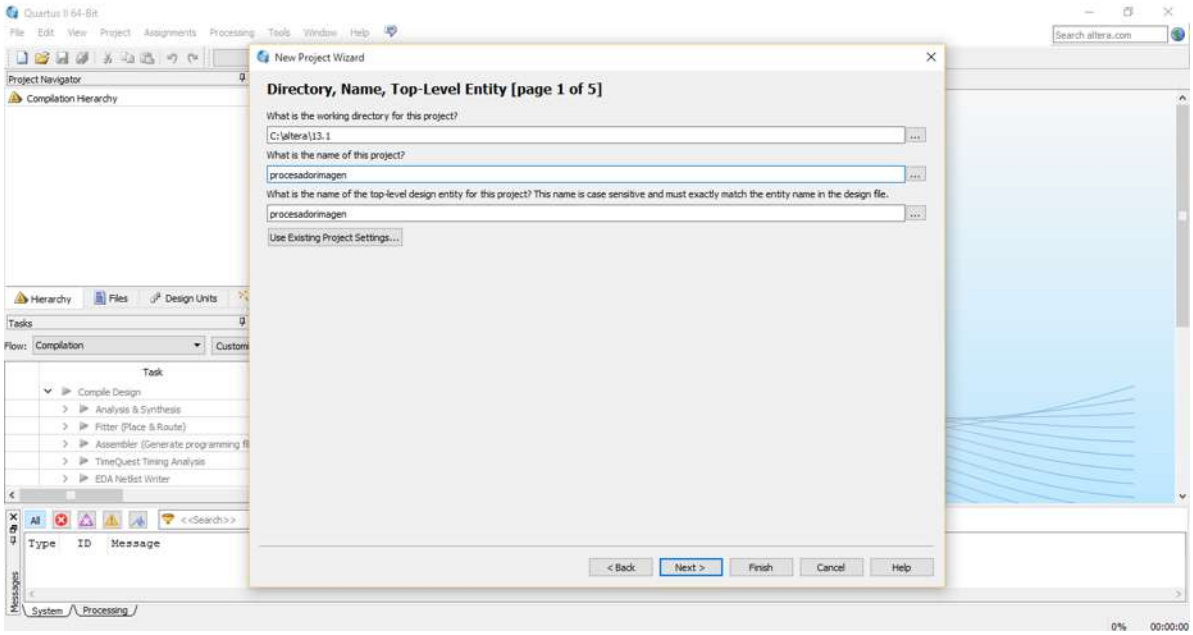

Figura 3.6 Se define en que carpeta y con qué nombre se guardará el proyecto.

A continuación se debe elegir que FPGA se usará en la pestaña de “Family” como se muestra en la Figura 3.7, se elige la Cyclone III y se busca la matrícula del FPGA que se usará en la ventana de “Available device”, la cual es EP3C16F484C6N, se da click en “next >”.

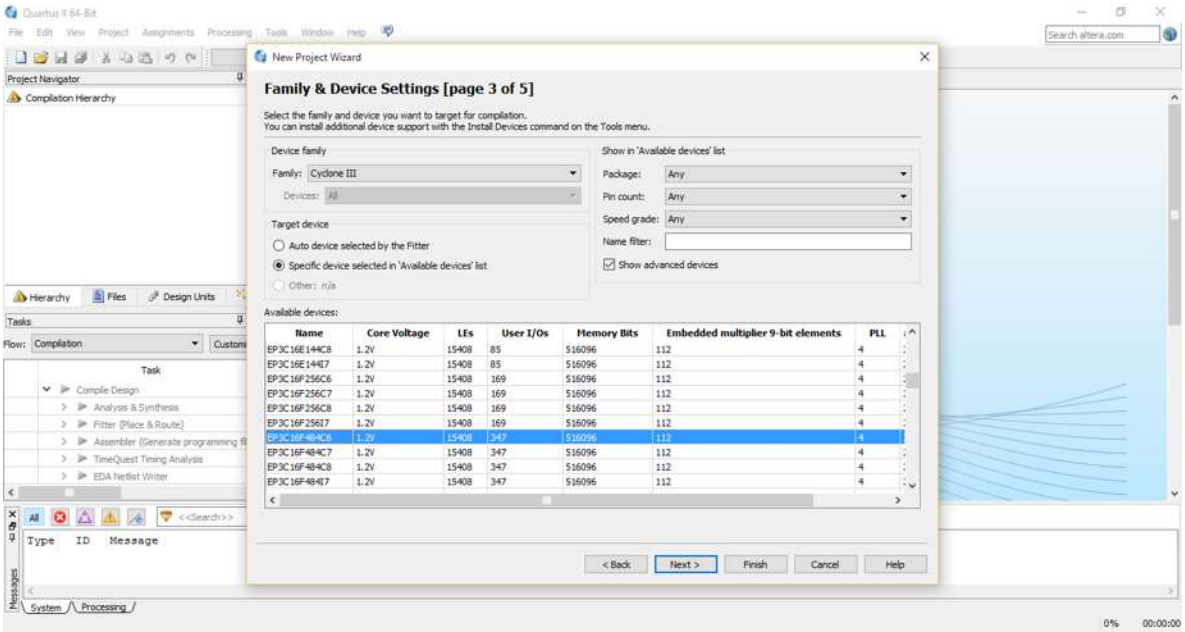

Figura 3.7 Elección del dispositivo a utilizar.

En la siguiente ventana, mostrada en la Figura 3.8, en la pestaña de “simulation”, “format(s)” se elige Verilog HDL y se da click en “next >”.

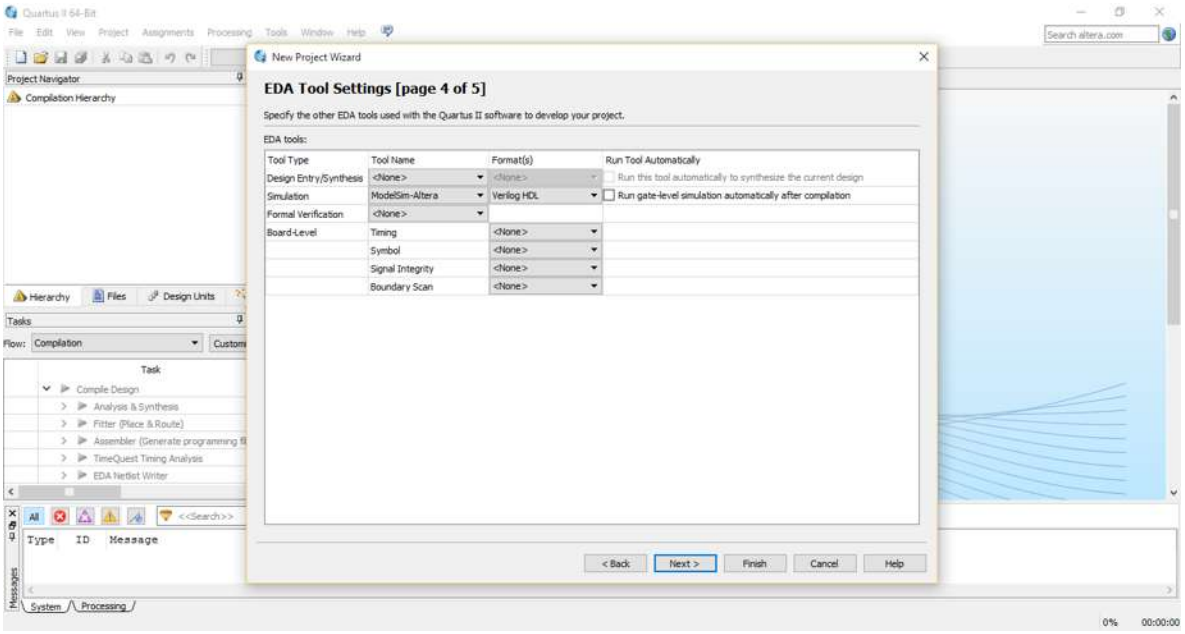

Figura 3.8 Elección del nombre del proyecto y ruta de guardado.

Finalmente como se muestra en la Figura 3.9, se tiene una ventana donde se muestra un resumen de las opciones elegidas, se verifica que se hayan elegido las opciones correctas y se da click en “Finish”.

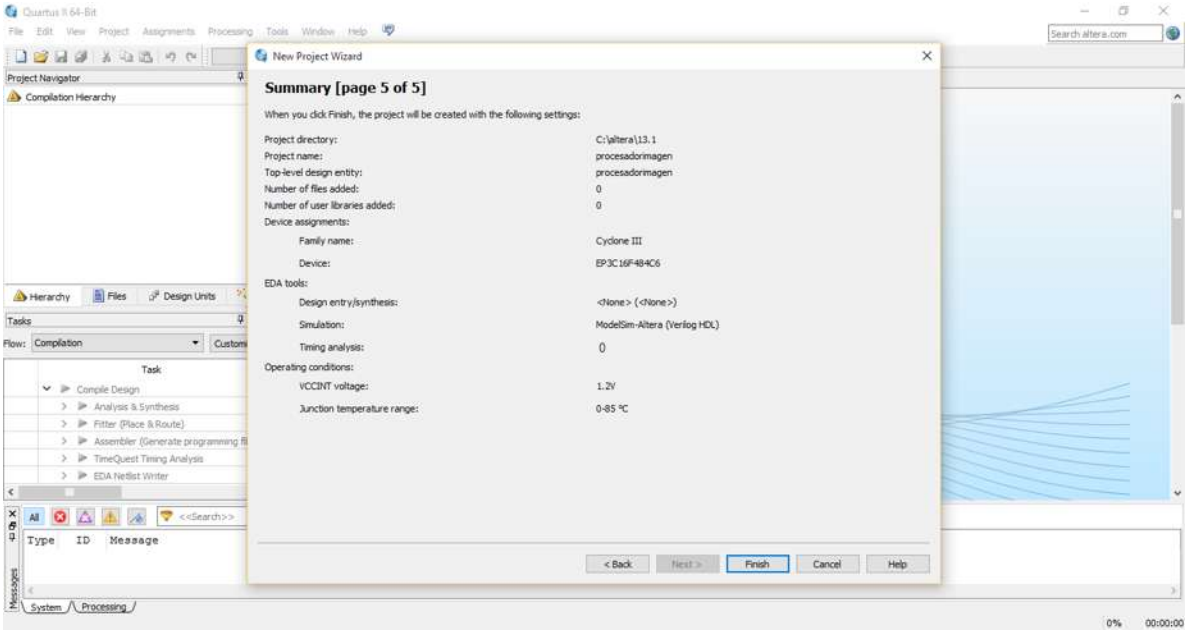

Figura 3.9 Resumen del proyecto creado.

3.4. Módulo de PLL

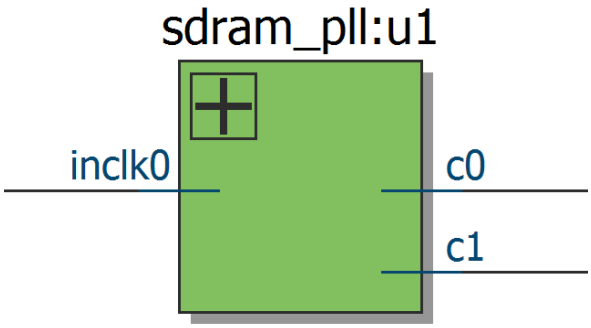

Figura 3.10 Módulo PLL

El módulo sdram_pll que se muestra en la Figura 3.10 tiene una entrada y dos salidas, en la entrada lleva el reloj que tiene la tarjeta DE0 que es de 50MHZ, este módulo lo que hace es multiplicar el reloj de tarjeta DE0 de 50MHZ a 125MHZ, esta frecuencia es la que se obtienen en las dos salidas del módulo.

Este módulo ya está incluido en el software de Quartus II como una mega función, el software de Quartus II tiene ciertos módulos prefabricados y entre ellos se encuentra el PLL, solo es necesario configurar las características con la que se desea que trabaje el módulo y se puede crear de una manera case automática.

3.5. Módulo Timer200µs

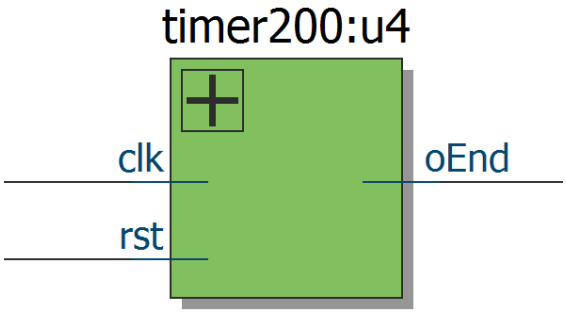

Figura 3.11 Módulo timer200

El módulo timer200 se muestra en la Figura 3.11 y consta de 2 entradas y una salida, la entrada ***"clk"*** lleva el reloj interno de la tarjeta DE0 de 50MHZ y la entrada ***"rst"*** la cual nos permite reiniciar el módulo. El modulo simplemente realiza un conteo de 200µs y al finalizar genera un pulso en la salida ***"oEnd",*** este módulo se utiliza para la inicialización de la SDRAM.

3.6. Módulo VGA_Controller

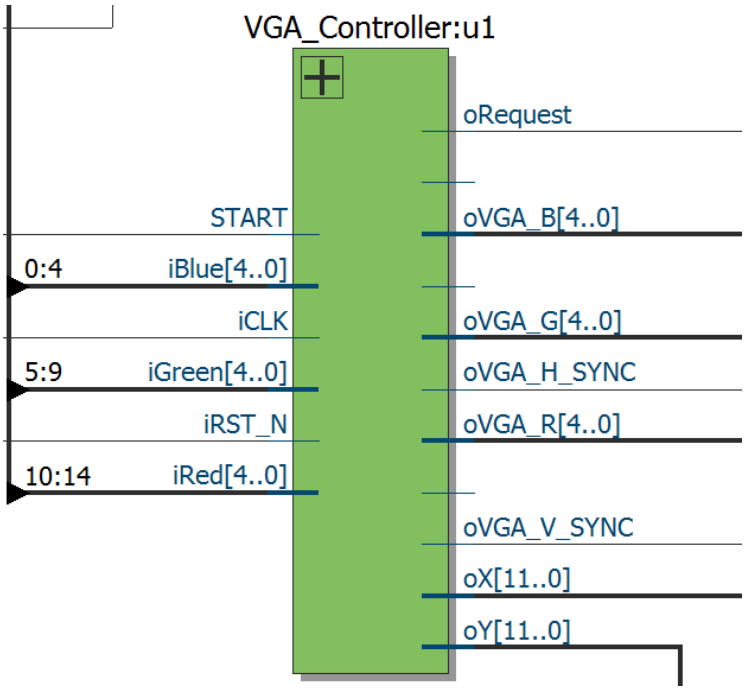

Figura 3.12 Módulo VGA_Controller

Este módulo es proporcionado por Terasic dentro de sus proyectos de demostración, en la Figura 3.12 se puede ver que cuenta con 6 entradas y 8 salidas, es un controlador de VGA el cual recibe los datos de la memoria SDRAM y los envía al VGA para ser observados en pantalla.

3.6.1. Entradas

- START.- Permite activar o desactivar el uso del módulo VGA_Controller.
- iBlue[4..0].- Es el bus de datos para el color azul.
- iCLK.- Es la entrada de la señal de reloj que utiliza el VGA_Controller.
- iGreen[4..0].- Es el bus de datos para el color verde.
- iRST_N.- Es la entrada para la señal de reinicio del módulo.
- iRed[4..0].- Es el bus de datos para el color rojo.

3.6.2. Salidas

- oRequest.- Esta salida manda una señal cada que el VGA requiere leer datos de la memoria.
- VGA_B[4..0].- Es el bus de datos de salida del color azul.
- VGA_G[4..0].- Es el bus de datos de salida del color verde.
- VGA_H_SYNC.- Esta es la señal de sincronización horizontal que requiere el VGA.
- VGA_R[4..0].- Es el bus de datos de salida del color rojo.
- VGA_V_SYNC.- Esta es la señal de sincronización vertical que se requiere para el funcionamiento del VGA.
- oX[11..0].- En esta salida se tiene la cuenta de la posición en el eje x que se está imprimiendo en pantalla.
- oY[11..0].- En esta salida se tiene la cuenta de la posición en el eje y que se está imprimiendo en pantalla.

3.7. Módulo VGA_CLK

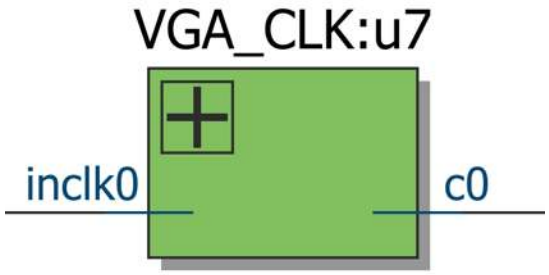

Figura 3.13 Módulo de VGA_CLK

Este módulo es proporcionado de igual manera por TERASIC, es simplemente un PLL que como se ve en la Figura 3.13 recibe a la entrada (***inclk0***) una señal de reloj de 50MHZ y a la salida (***c0***) se tiene una señal de reloj de 25MHZ que es la que necesita el módulo VGA_Controller para trabajar.

Capítulo 4

4. Diseño e implementación

4.1. Diseño de Controlador de Memoria

El controlador de memoria permitirá manipular a la memoria SDRAM con una interfaz sencilla y sin tener que preocuparse constantemente por ingresar todos los comandos necesarios para operarla.

En este proyecto se utilizará la memoria SDRAM que viene integrada dentro de la tarjeta DE0, la memoria IS42S16400J-6TL en la Figura 4.1 se muestra el diagrama de la memoria, la cual es una memoria de alta velocidad de 67,108,864 bit organizada en un arreglo de 1,048,576 x 16 x 4 (palabra x bit x banco).

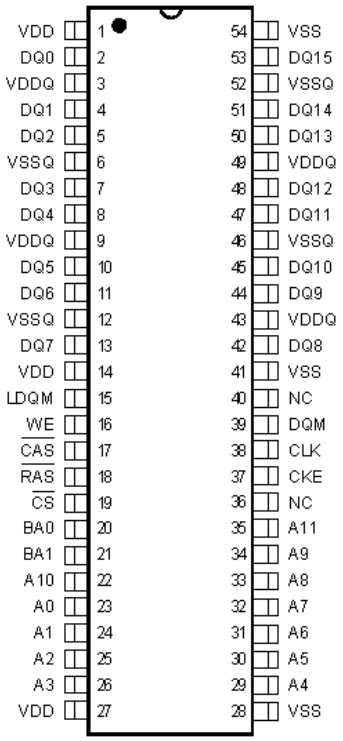

Figura 4.1 54-Pin TSOP-2 (IS42S16400). (Integrated Circuit Solution Inc., 2000)

Para el diseño del controlador en primer lugar es necesaria la hoja de datos que se encuentra dentro de la documentación que nos proporciona el disco de la tarjeta DE0. Dentro de la hoja de datos se puede observar un diagrama de bloques del funcionamiento de esta memoria, el cual se observa en la Figura 4.2.

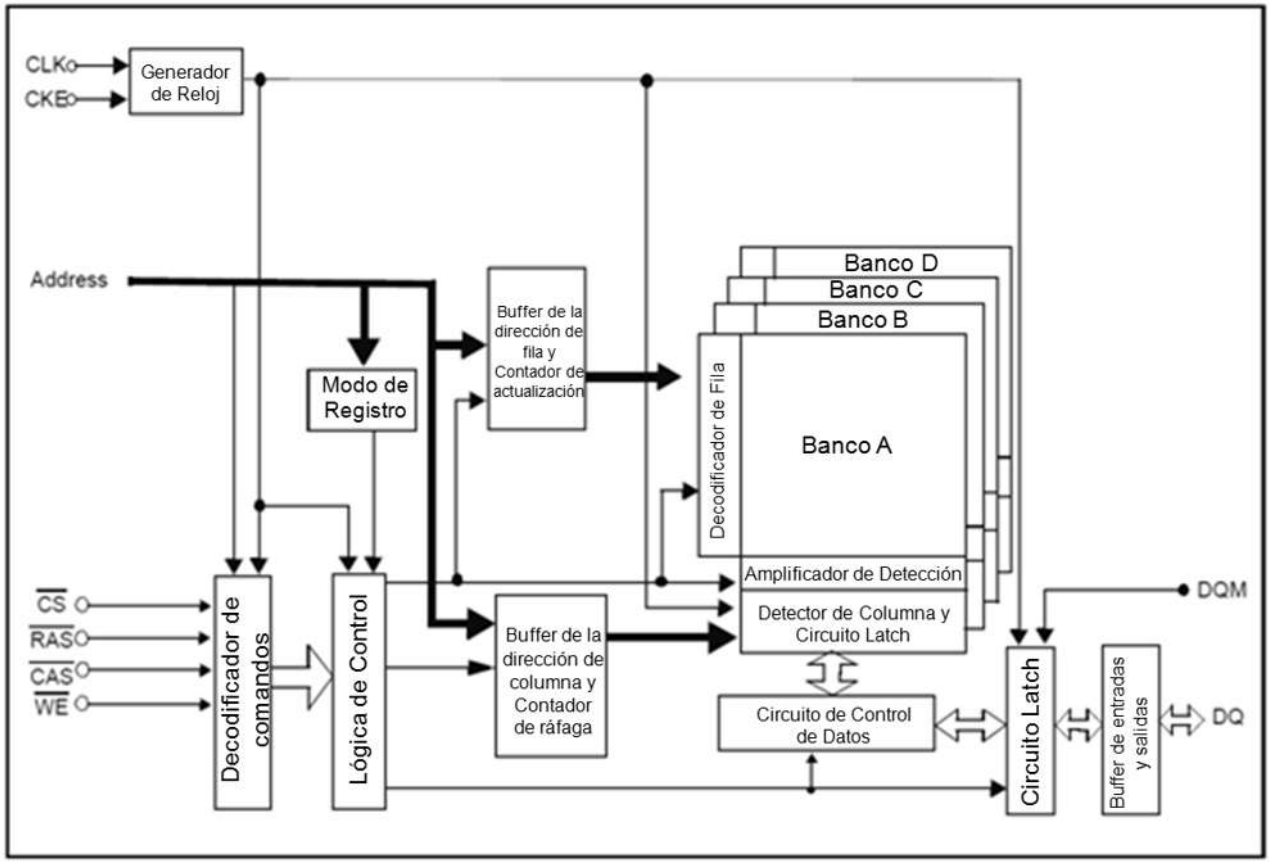

Figura 4.2 Diagrama de bloques del funcionamiento de la memoria IS42S16400.

La hoja de datos indica que se deben de seguir ciertos pasos para inicializar la memoria los cuales son:

1. Energizar e iniciar el reloj. Se debe intentar mantener en alto las entradas de CKE, DQN y NOP.
2. Mantener la energía estable y la entrada NOP durante un mínimo de 200µs.
3. Precargar todos los bancos con los comandos PRE o PREA.
4. Ya que todos los bancos se encuentren en estado idle, se deben aplicar 8 o más comandos de “auto-refresh”.
5. Se debe aplicar el comando de “mode register” para configurar la memoria.

Finalizando este proceso de inicialización, la SDRAM se debe encontrar en estado idle y lista para usarse.

El módulo que se diseñó “RAM_Controller” tiene 8 entradas, 10 salidas y un bus bidireccional como se muestra en la Figura 4.3.

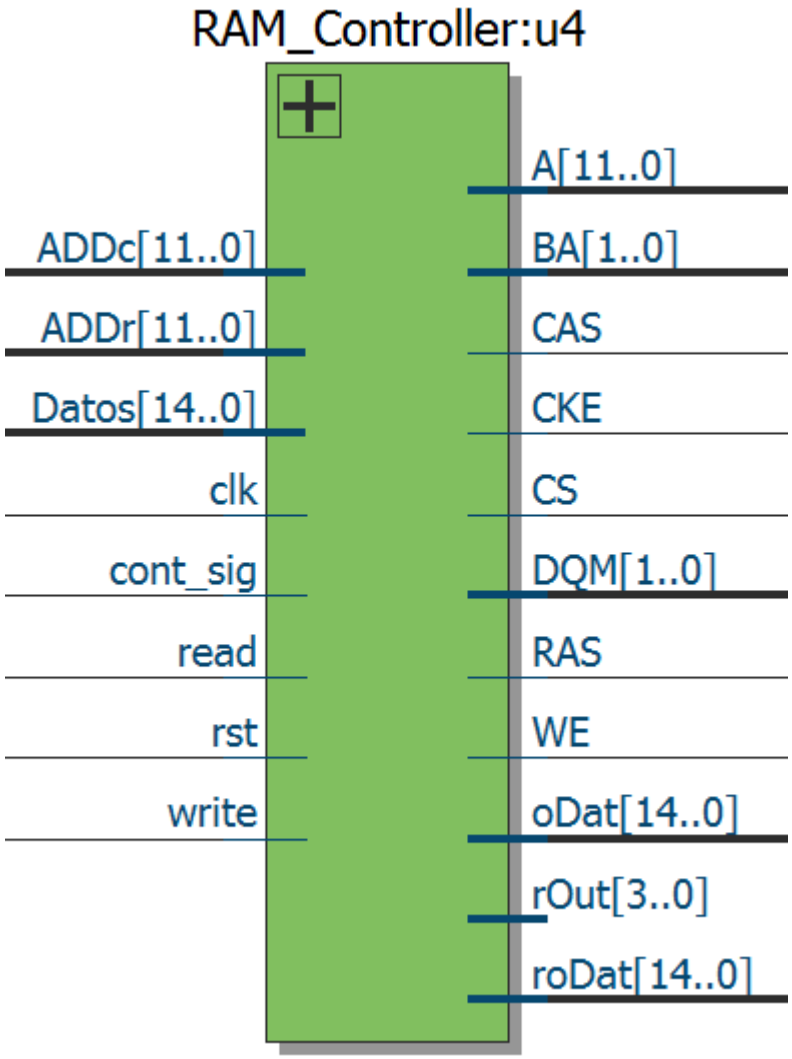

Figura 4.3 Modulo RAM_Controller

4.1.1. Entradas

- ADDc[11..0].- Este bus de entrada es el que da la dirección de la columna donde se almacenará el dato dentro de la memoria.

- ADDr[11..0].- Este bus de entrada indica la dirección del renglón donde se almacenará el dato en la memoria.
- Datos[14..0].- Este es el bus de 15 bits de los datos de entrada.
- clk.- Es la señal de reloj necesaria para el funcionamiento del módulo.
- cont_sig.- El módulo timer200 da una señal al terminar de contar 200µs los cuales son necesarios para la inicialización de la memoria.
- read.- Aquí se introduce una señal cuando se quiere realizar una lectura en la memoria.
- rst.- Es la entrada para la señal que reinicia el módulo.
- write.- Aquí se introduce una señal cuando se quiere realizar una escritura en la memoria.

4.1.2. Salidas

- A[11..0].- En este bus se ingresa la dirección donde se quiera almacenar la información, de igual manera sirve a la hora de configurar el modo de uso de la memoria.
- BA[1..0].- Este par de bits sirven para elegir el banco que se utilizará.
- RAS, CAS, WE.-Sirven para introducir los comandos.
- CKE.- Habilita o deshabilita el reloj.
- CS.- Es parte del comando que se da a la memoria.
- DQM[1..0].- Deshabilita la entrada de los datos cuando se escribe y deshabilita la salida de datos cuando se lee.
- rOut[3..0].- Esta es una salida que se creó para poder observar en la simulación en qué estado se encuentra el controlador.
- roDat[14..0].- Muestra los datos que se leen de la memoria.

4.1.3. Bus Bidireccional

- oDat[14..0].- Es el bus de datos que lleva tanto los datos de escritura como los datos de lectura a la memoria.

Internamente el controlador trabaja con estados, es una máquina de estados, dependiendo del estado anterior pasa a otro estado y en cada estado se envían ciertos comandos a la memoria.

Se tienen 14 estados los cuales son:

- Inicio.- En este estado se envían los comandos que la memoria debe recibir al iniciar, se mantienen el voltaje de entrada y algunas entradas deben mantenerse en ciertos estados. Este estado se debe mantener durante 200µS (según las especificaciones de la memoria) y al cumplirse esta condición se pasa al siguiente estado.
- Precarga.- En este estado es donde se deben de precargar todos los datos, esto se realiza debido a las instrucciones de inicialización de la memoria según la hoja de datos.
- RefreshX.- Conforme a la hoja de datos se deben de enviar 8 comandos de auto-refresh a la memoria. Estos 8 estados son los encargados de esto.
- Modreg.- Finalizando los estados de refresh se debe configurar la memoria, lo cual se realiza en este estado.
- Writa.- Este estado se da cuando se recibe la instrucción de escribir, el estado envía el comando para escribir.
- Reada.- A este estado se entra cuando se da la instrucción de leer. Envía el comando para leer.
- Nop.- Este estado es al cual la máquina va posteriormente del estado "modreg", este estado envía comandos de NOP para que la memoria se mantenga en Idle mientras no se reciba una instrucción de leer o de escribir, cuando se da alguna de estas instrucciones, envía un comando ACT y posteriormente pasa al estado "writa" o "reada" según sea el caso.

4.2. Módulo de Escritura y Lectura de Datos

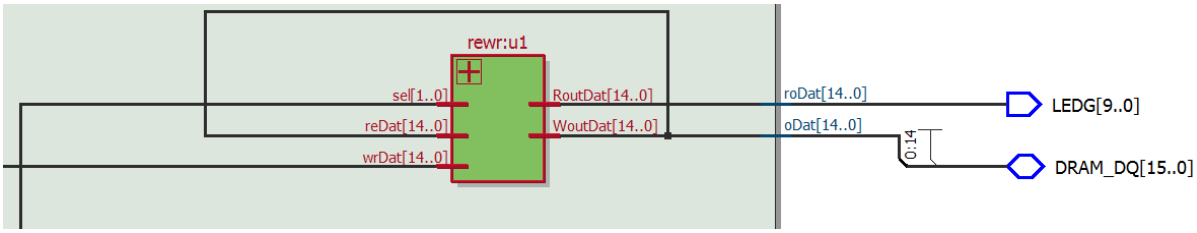

Figura 4.4 Módulo rewr

Dentro del módulo de "RAM_Controller" se tiene un módulo llamado "rewr" que se muestra en la Figura 4.4 el cual permite administrar los datos de lectura y de escritura. La

memoria tiene un bus bidireccional para estos datos por lo tanto al controlador le corresponde a la hora de lectura poder tomar estos datos y enviarlos a alguna interfaz que nos permita observar estos datos. Este módulo cuenta con 3 entradas y dos salidas. El funcionamiento del módulo es muy simple, el selector decide qué entrada y que salida se habilita dependiendo de si estamos leyendo o escribiendo. En la Figura 4.5 se muestra el módulo internamente.

4.2.1. Entradas

- reDat[14..0].- Cuando se va a leer los datos vienen de la memoria y entran por este bus.
- sel[1..0].- Este es un selector que indica si se lee o se escribe, el selector lo controla la máquina de estados.
- wrDat[14..0].- Cuando se escribe los datos van hacia la memoria, los datos entran por este bus.

4.2.2. Salidas

- RoutDat[14..0].- Cuando se lee, los datos vienen de la memoria y se envían a alguna interfaz que permita interpretar los datos leídos, en este caso unos leds.
- WoutDat[14..0].- Para escribir los datos vienen de la interfaz de que se utiliza para escribir y se envían a la memoria.

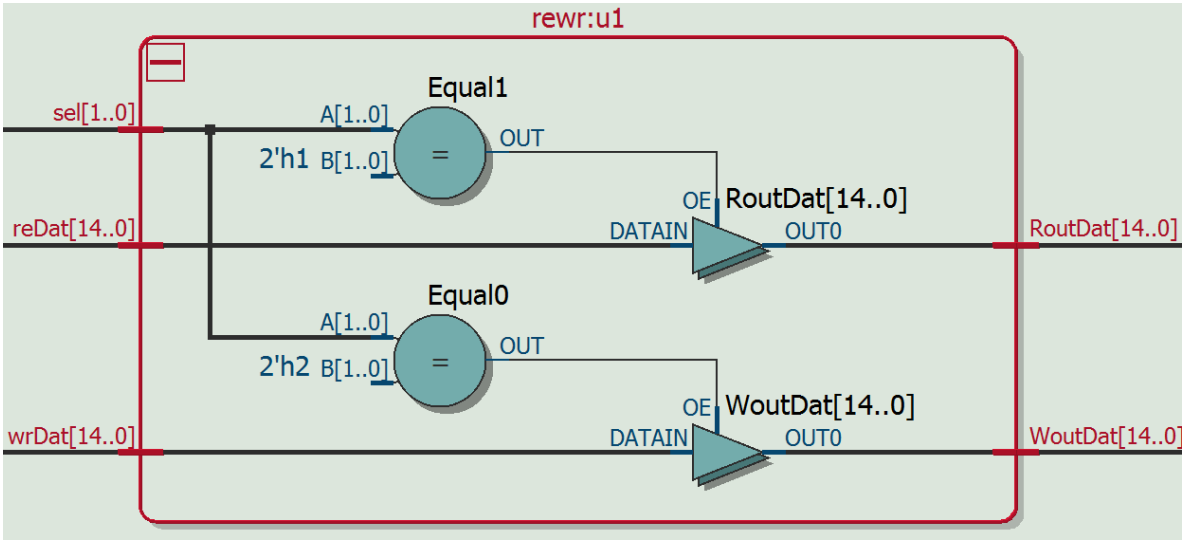

Figura 4.5 Interior del módulo rewr

4.3. Módulo de Generación de la Imagen

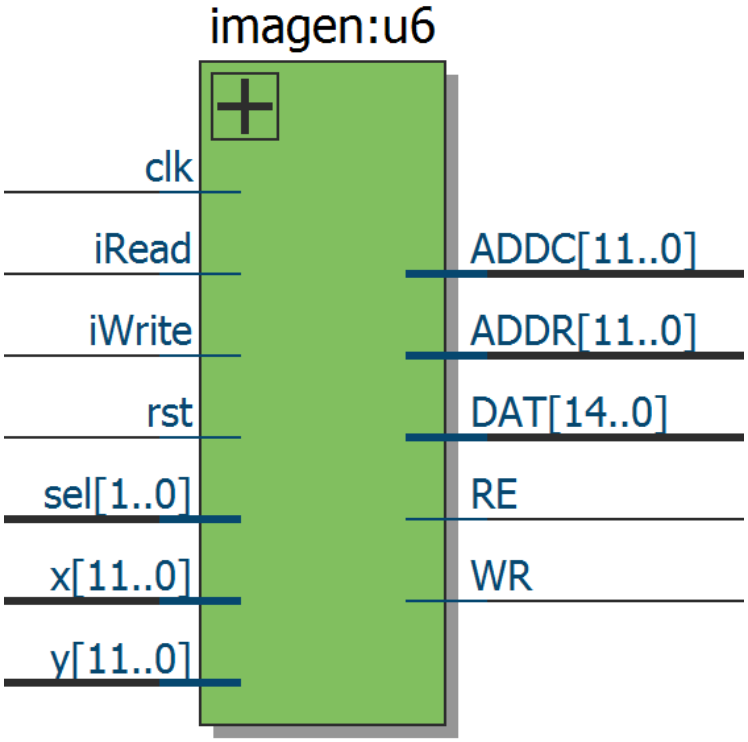

Figura 4.6 Módulo imagen

Este módulo que se muestra en la Figura 4.6 será el encargado de generar una imagen y guardarla dentro de la memoria SDRAM, cuenta con un selector el cual se conecta a un par de interruptores que permiten ir escribiendo en la memoria hasta 4 valores distintos en 4 direcciones diferentes. El módulo cuenta con 7 entradas y 5 salidas.

4.3.1. Entradas

- clk .- Esta es la entrada de reloj que permite el funcionamiento del módulo.
- iRead .- Esta entrada indica si el módulo VGA_Controller necesita leer.
- iWrite.- Esta entrada indica si el usuario necesita escribir en la memoria.
- rst .- Esta señal permite reiniciar el módulo.
- sel[1..0] .- Este selector permite cambiar las direcciones de memoria donde se almacenará la imagen.
- x[11..0] .- Este bus indica la posición en x.
- y[11..0] .- Este bus indica la posición en y.

4.3.2. Salidas

- ADDC[11..0].- Este bus de salida indica la dirección de la columna de la matriz de memoria donde se almacenará la imagen.
- ADDR[11..0].- Este bus de salida indica la dirección del renglón de la matriz de memoria donde se almacenará la imagen.

- DAT[14..0].-Este bus de salida contienen el dato de la imagen que se almacena en la memoria.
- RE.- Esta salida envía el comando de lectura al controlador de memoria.
- WR.- Esta salida envía el comando de escritura al controlador de memoria.

4.4. Simulaciones y Resultados

A continuación se presentan las simulaciones realizadas con el software Quartus II las cuales permiten observar el comportamiento en el tiempo de las señales en los diferentes módulos.

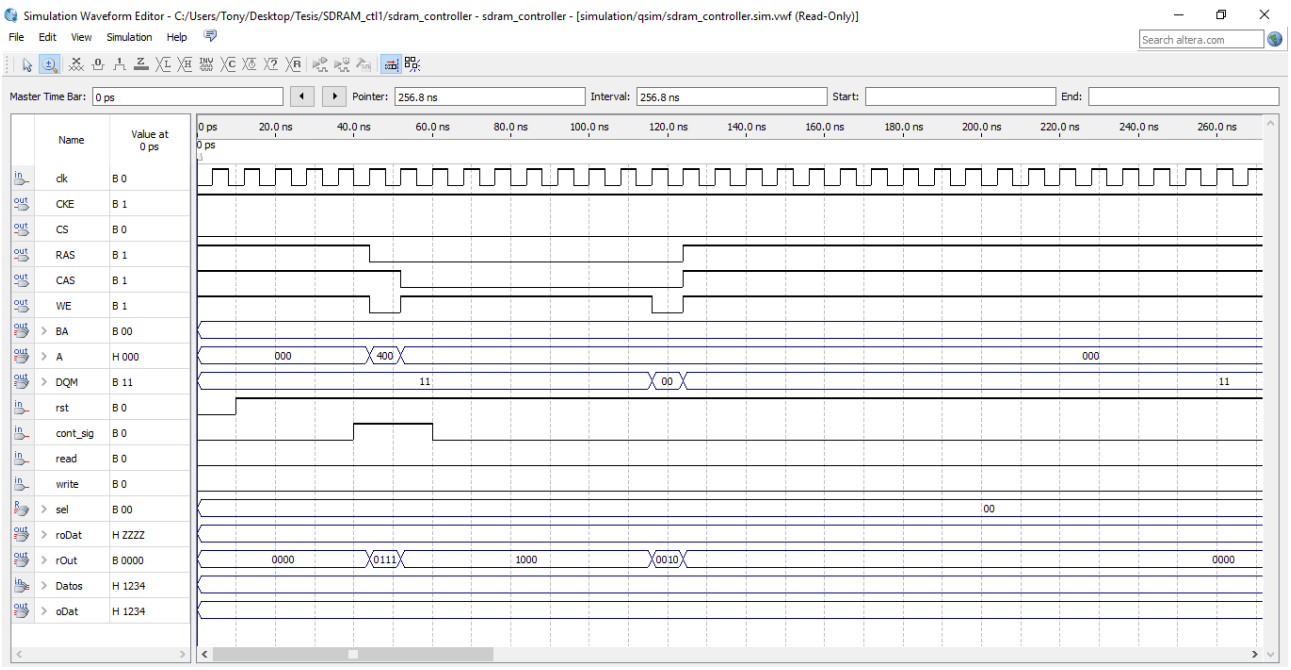

Figura 4.7 Simulación de la inicialización

En la Figura 4.7 se puede observar la simulación de la inicialización del controlador de memoria. Para la inicialización de la memoria se deben de realizar ciertos pasos antes de que la memoria funcione de manera normal.

1. Energizar la memoria e iniciar la señal de reloj. Se debe mantener CKE y DQN en alto y una condición de NOP en las entradas.
2. La energía se debe mantener estable y las entradas en condición de NOP por al menos 200µs.
3. Se debe enviar un comando de PRE a los bancos de memoria.
4. Ya que todos los bancos estén en estado “idle”, se debe tener 8 o más comandos de auto-refresh.
5. Se debe enviar un comando de “mode register” para inicializar el “mode register”.

Posterior a esto la memoria se debería encontrar en un estado “idle” lista para operar de manera normal.

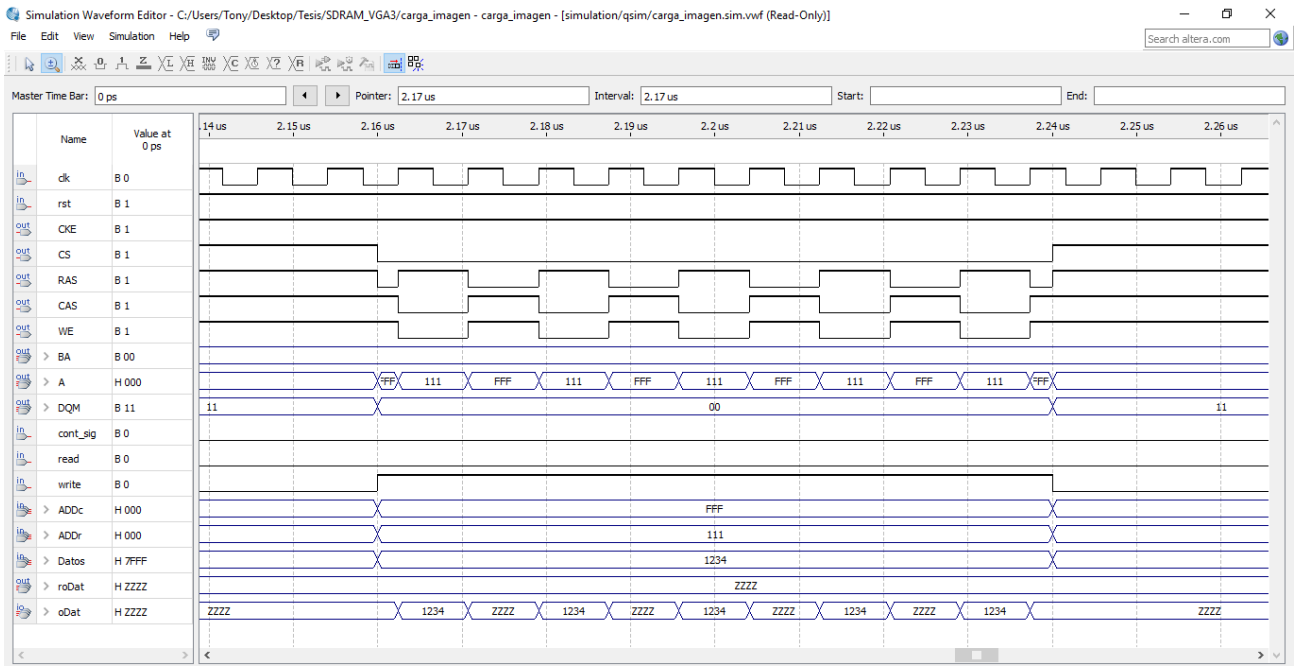

Figura 4.8 Simulación de la escritura

En la Figura 4.8 se puede observar como el controlador recibe un comando de escritura, las señales CS, RAS, CAS y WE son las señales que tiene que recibir la memoria SDRAM para ejecutar el comando de escritura, estos comandos están definidos en la Tabla 2-1. La señal de oDat la cual es una señal de entrada y salida, se habilita como salida cuando se tiene un comando de escritura y envía el dato que se encuentra en la señal de entrada “Datos”. ADDc y ADDr son las direcciones de la columna y de la fila en el banco de memoria respectivamente. Finalmente roDat son los datos que se leen, en este caso al ser un comando de escritura esta señal se encuentra en alta impedancia.

En la Figura 4.9 se tiene un comando de lectura, de igual forma que con el comando de escritura, las señales CS, RAS, CAS y WE envía el comando de escritura a la memoria SDRAM. En la señal oDat que en este caso se habilita como una entrada, se puede observar el dato que es enviando desde la memoria SDRM cuando recibe el comando de lectura, en este caso la señal de Datos no es de interés ya que no se escribirá nada. En roDat se observa el dato que se tiene en oDat ya que la señal de roDat es la que se envía al módulo que requiere leer el dato de la memoria SDRAM.

Finalmente en la Figura 4.10 se puede ver un patrón previamente cargado en la FPGA que ha sido almacenado en la memoria SDRAM y el módulo de VGA_Controller se está encargando de leer la memoria y desplegar el patrón en el monitor.

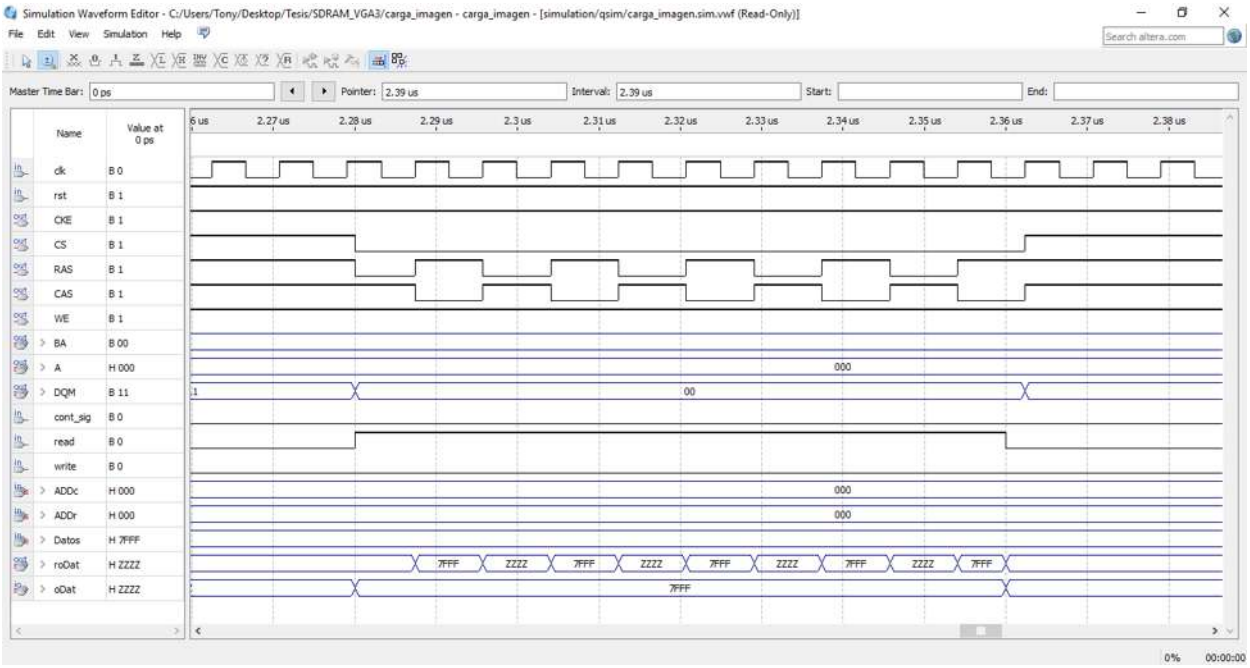

Figura 4.9 Simulación de lectura

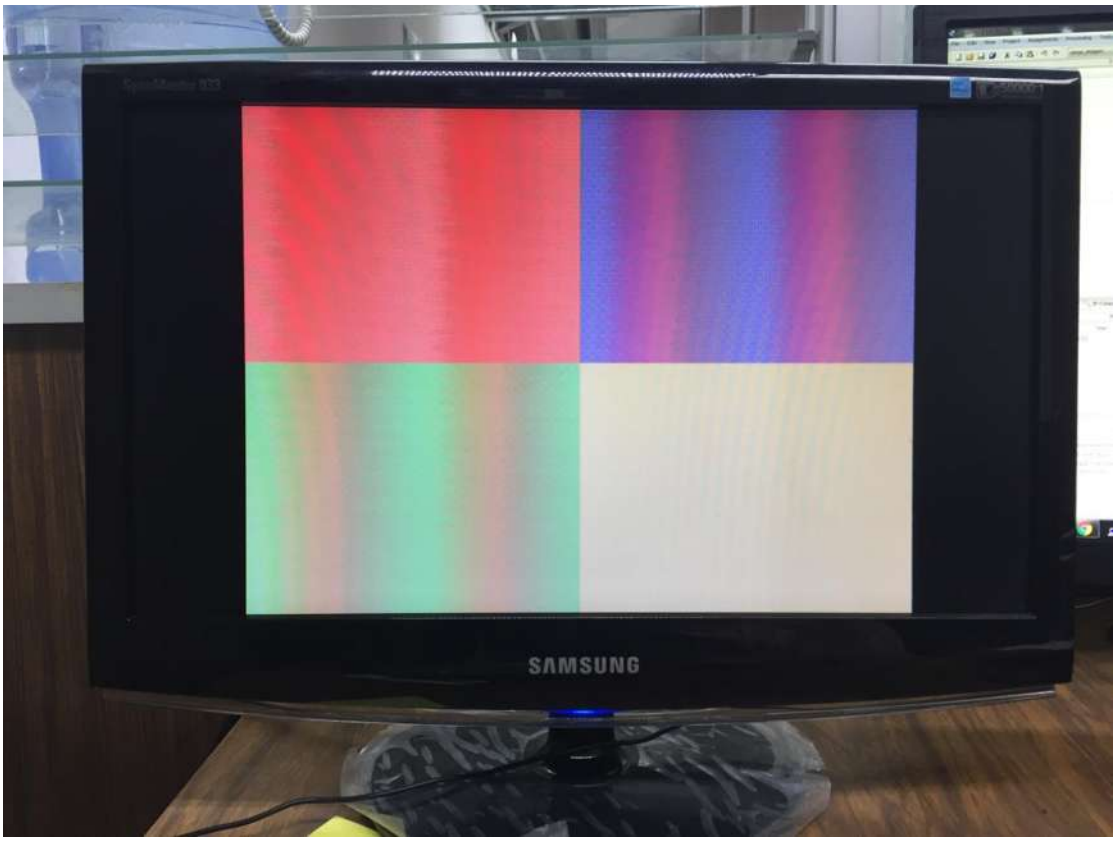

Figura 4.10 Patrón desplegado en pantalla

Capítulo 5

5. Conclusiones

5.1. Conclusiones

Aprovechando la gran flexibilidad que presentan los sistemas basados en lógica programable, se realizó el trabajo de acuerdo a la siguiente metodología de diseño de sistemas digitales de hardware:

- A partir desde las especificaciones de la memoria SDRAM IS42S16400J-6TL proporcionados por el fabricante en la hoja de datos, se diseñó un conjunto de módulos que realizan las funciones abstractas básicas: máquinas de estado, contadores, generadores de pulso, registros universales.

- Estos bloques funcionales fueron a continuación simulados y posteriormente integrados dentro del módulo controlador de alta jerarquía.

- Se simuló el funcionamiento del módulo completo, lo que proporcionó información sobre la factibilidad del diseño.

- Dada la simulación satisfactoria, se implementó el diseño en la tarjeta de pruebas. Se adaptó el diseño a un módulo de despliegue de imagen en un monitor VGA, comprobando con esto que la simulación arrojó resultados correctos.

- Finalmente el controlador cumple con lo planteado en un principio: permite escribir en diferentes direcciones de la memoria a través de una interfaz con el usuario. Para esto, se utiliza un push button que permite leer de la memoria; el módulo de VGA_Controller es el encargado de enviar el comando de lectura (solicitud de datos) y de procesar lo que la memoria le envía.

5.2 Trabajos Futuros

Éste controlador de memoria es muy básico y no se enfocó en reducir tiempos de escritura ni de lectura u optimizar la manera en que se almacena la información en las celdas del chip, sin embargo, se realiza con la intención de que en un futuro se pueda implementar un procesador de imagen, con la ventaja de que se tiene pleno conocimiento de cómo se realiza el control y acceso a la memoria. De igual forma solo tiene las funciones básicas necesarias para los objetivos que se pretenden cumplir en esta tesis. Por lo tanto este trabajo también queda abierto para que en algún futuro se puedan agregar más funciones u optimizar tiempos de lectura o escritura.

Referencias

Altera, 2008. *Cyclone III Device Handbook, Volume 1.* San José(California): s.n.

Altera, 2016. *Altera.* [En línea]
Available at: http://dl.altera.com/?edition=web.

Anon., 2016. *Qi Hardware.* [En línea]
Available at: http://en.qi-hardware.com/wiki/SDRAM/es

Brown, S. & Vranesic, Z., 2008. *Fundamentals of Digital Logic with Verilog Design.* New York: McGraw-Hill.

Floyd, T. L., 1997. *Fundamentos de Sistemas Digitales.* Madrid: Prentice Hall.

Floyd, T. L., 2006. *Fundamentos de Sistemas Digitales.* Madrid: Pearson Education.

Herrera Lozada, J. C., 2016. *jcrls en la Web.* [En línea]
Available at: http://embebidos-cidetec.com.mx/profesores/jcrls/doctos/tec_programables_cidetec.pdf

Integrated Circuit Solution Inc., 2000. *IS42S8800/IS42S8800L/IS42S16400/IS42S16400L.* HsinChu: s.n.

Lattice Semiconductor Corporation, 1998. *Introduction to GAL Device Architectures.* s.l.:s.n.

Laws, D., 2016. *The Rise of TTL: How Fairchild Won a Battle But Lost the War.* [En línea]
Available at: http://www.computerhistory.org/atchm/the-rise-of-ttl-how-fairchild-won-a-battle-but-lost-the-war/

Muthuswamy, B. & Banerjee, S., 2015. *A Route to Chaos Using FPGAs Volume I: Experimental Observations.* s.l.:Springer.

National Instruments, 2016. *Introducción a la Tecnología FPGA: Los Cinco Beneficios Principales.* [En línea]
Available at: http://www.ni.com/white-paper/6984/es/

Semiconductor Special Interest Group, 2016. *1978: PAL User-Programmable Logic Devices Introduced.* [En línea]
Available at: http://www.computerhistory.org/siliconengine/pal-user-programmable-logic-devices-introduced/

Terasic Technologies, 2009. *DE0 User Manual.* Jhubei City(HsinChu): s.n.

Tocci, R. J. & Widmer, N. S., 2003. *Sistemas Digitales Principios y aplicaciones.* México: Pearson Educación.