UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

DISEÑO DE UNA COMPUTADORA DE A BORDO EN UN FPGA PARA UN SISTEMA DE PERCEPCIÓN REMOTA DE IMÁGENES DE UN SATÉLITE BAJO EL ESTÁNDAR CUBESAT

TESIS

Que para obtener el título de

INGENIERO ELÉCTRICO-ELECTRÓNICO

P R E S E N T A:

Pérez Vicente Israel

DIRECTOR DE TESIS

Dr. Saúl de la Rosa Nieves

Ciudad Universitaria, Cd. Mx., 2023

Resumen

El presente trabajo tiene como finalidad diseñar la arquitectura de una computadora en un FPGA con capacidad para almacenar imágenes provenientes de un sensor de imagen CMOS y que incluya las interfaces necesarias para comunicarse con la computadora principal de un satélite del estándar CubeSat. Se emplea una memoria SDRAM DDR3 como medio de almacenamiento por sus características en cuanto a velocidad de transferencia y su relación en capacidad de almacenamiento/costo.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1. Presentación del problema

El desarrollo de nanosatélites bajo el estándar CubeSat se ha visto incrementado en los últimos años [1] debido a la posibilidad de no necesariamente usar componentes de calidad espacial o militar, dando lugar al uso de componentes conocidos como componentes comerciales salidos del estante (COTS, por sus siglas en inglés), haciendo posible que se reduzca el costo total de la misión, fomentando a su vez una mayor participación de universidades y organizaciones en su desarrollo.

Un aspecto clave a tener en cuenta para el uso de estos componentes es que no poseen un diseño acorde a las características requeridas para ser usados en ambiente espacial, por lo cual su funcionamiento puede verse afectado [2]. Los fenómenos de radiación presentes en el entorno espacial suponen un grave problema para los dispositivos electrónicos y por lo tanto para el desarrollo exitoso de la misión llevada a cabo, ya que estos elementos son vitales para el funcionamiento del satélite.

Anteriormente se llevó a cabo una cooperación tecnológica entre México y Rusia para llevar a cabo el desarrollo del proyecto CONDOR UNAM-MAI [3], el cual consistia en la propuesta de una plataforma de satélite bajo el estándar CubeSat, de la cual se destaca el módulo de carga útil compuesto por un sistema de percepción remota de imágenes. Este sistema debía incorporar una cámara multiespectral para la toma de imágenes con una resolución de 30 m, para posteriormente ser enviadas a la estación terrestre. El desarrollo de este proyecto no llegó a consolidarse, por lo cual varios de los subsistemas que conformarían al satélite no se construyeron.

Los FPGA basados en tecnología SRAM han sido usados en misiones espaciales y su uso se ha visto incrementado en años recientes [4]. Su capacidad para poder configurarse a nivel de hardware y capacidad de procesamiento en paralelo los hace versátiles para ser usados en aplicaciones embebidas. Esa capacidad de poder configurarse se ha aprovechado para implementar técnicas de tolerancia a fallas, haciendo que la confiabilidad de un FPGA comercial en ambiente espacial se vea incrementada [2] para mitigar los efectos de la radiación en el ambiente espacial [5], ocasionada principalmente por efectos de evento único (SEE, por sus siglas en inglés).

Tomando como referencia el concepto de la carga útil del proyecto CONDOR y el uso de componentes COTS en misiones espaciales como los FPGA, se plantea el diseño de una computadora en FPGA para un sistema de percepción remota de imágenes bajo el estándar CubeSat. El proyecto CONDOR contemplaba el uso de un sensor de imagen MT9P031, el cual requería de una memoria con capacidad para almacenar las imágenes tomadas en resolución completa y que fuera capaz de responder a la velocidad de transferencia de datos del sensor.

1.2. Hipótesis

El diseño de una computadora en un FPGA COTS con la capacidad para almacenar datos de un sensor de imagen se presenta como propuesta para ser integrada en un sistema de percepción remota de imágenes de un satélite del estándar CubeSat y ofrecer la posibilidad de implementar técnicas de tolerancia a fallas.

1.3. Objetivo

Desarrollar la arquitectura de una computadora de a bordo con la capacidad para almacenar datos de un sensor de imagen para implementarse en un sistema de percepción remota bajo el estándar CubeSat, basada en un FPGA COTS.

Objetivos particulares

- Diseñar la arquitectura funcional de la computadora para transferir las imágenes obtenidas por el sensor hacia la computadora principal de un satélite CubeSat.
- Implementar un controlador para una memoria que permita almacenar los datos del sensor de imagen a su máxima resolución y máxima tasa de transmisión.

- Desarrollar el módulo de la interfaz para el sensor seleccionado.
- Implementar el software para las pruebas del sistema.

1.4. Alcance

Los alcances de este trabajo contemplan el diseño de una computadora con la capacidad de almacenar los datos provenientes de un sensor de imagen, permitiendo que se pueda implementar como parte de un sistema de percepción remota de imágenes de un satélite CubeSat, además de que al ser implementada en un FPGA abre la posibilidad de implementar también técnicas de tolerancia a fallas que incrementen su fiabilidad en el ambiente espacial y presentarse como una propuesta a las opciones comerciales disponibles.

1.5. Justificación

Actualmente la tendencia de desarrollo en satélites de tipo CubeSat en universidades e institutos se ha visto favorecida debido a una mayor disponibilidad de los componentes y a las alternativas tecnológicas que han surgido y permitido abaratar los costos de desarrollo. Debido a que estos componentes no están diseñados para el entorno espacial es necesario llevar a cabo la implementación de técnicas de tolerancia a fallas que garanticen la fiabilidad de su funcionamiento en este ambiente, en donde los FPGA son dispositivos factibles para implementarlas.

Uno de los campos de desarrollo en este tipo de satélites se centra en el uso de los sensores de imagen, particularmente en los sistemas de percepción remota, los cuales han mostrado tener una gran diversidad de aplicaciones científicas, por lo que al desarrollar una computadora de a bordo para este sistema se abren nuevas oportunidades en el desarrollo de proyectos dentro del Laboratorio de Instrumentación en Sistemas Espaciales (LIESE) de la Facultad de Ingeniería.

Capítulo 2

Marco teórico

2.1. Satélites

Un satélite es cualquier objeto que órbita alrededor de otro, pudiendo ser de procedencia natural como una luna o planeta o bien artificial, es decir, construido por los seres humanos. Los satélites artificiales a lo largo de la historia han sido construidos para distintos propósitos, haciendo que el tamaño, altitud y diseño dependan de este mismo. Este tipo de artefactos son puestos en la órbita de un objeto por medio de cohetes lanzados desde la tierra. Actualmente se han lanzado una gran cantidad de satélites alrededor de la tierra. Los satélites artificiales tienen varias aplicaciones, desde las telecomunicaciones como la televisión, la radio y la telefonía, sistemas de posicionamiento global (GPS), sistemas de percepción remota.

En general la estructura de un satélite se puede agrupar en seis subsistemas diferentes.

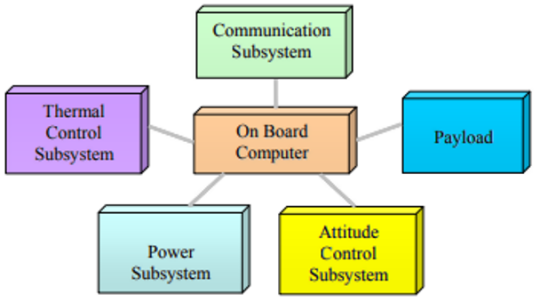


Figura 2.1. Subsistemas principales de un satélite [6].

2.1.1. Sistemas de percepción remota

Los sistemas de percepción remota forman parte de la carga útil del satélita (*payload* en inglés) y tienen como finalidad recolectar información del espectro electromagnético para fines educativos y de investigación. Para ello estos sistemas emplean sensores de imagen, puediendo ser del tipo multiespectrales, de espectro visible o infrarrojos. En varias de las misiones documentadas en el estado del arte se emplean con fines demostrativos de capacidad tecnológica, por lo cual no se detalla en el nivel de confiabilidad que ofrecen, sin embargo, para sistemas empleados con fines de investigación, se debe buscar garantizar el correcto funcionamiento del sistema, empleando una arquitectura más robusta para hacer frente a las condiciones del ambiente espacial.

2.1.2. Cámaras

Una cámara es un dispositivo óptico-electrónico con la capacidad para capturar imágenes. El sistema óptico, es el que se encarga de dirigir la luz de un objeto hacia el sensor de imagen de la cámara. Este sistema está compuesto de varios lentes que se encargan de ajustar la proyección de la imagen sobre el sensor. El sistema electrónico lo conforma el sensor de imagen, el cual se encarga de realizar la captura de la información visual a un formato que pueda ser procesado a nivel electrónico. Un módulo de cámara está compuesto por el sistema óptico y el sensor de imagen, los cuales se pueden montar en una estructura de ensamble.

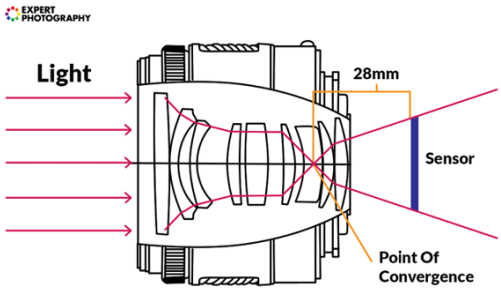


Figura 2.2. Estructura óptica de una cámara [7].

El sensor de imagen se encuentran principalmente en dos variantes: los de tecnología CCD (Charge-Coupled Devices) y los de tecnología CMOS (Complementary

Metal Oxide-Semiconductor, en inglés).

Una cámara embebida típicamente está compuesta de tres partes: ensamble óptico, sensor de imagen y la interfaz digital. La alianza MIPI ha especificado interfaces de comunicación para los sensores de imagen, desarrollando la Interfaz Paralela de Cámara (CPI, por sus siglas en inglés) y la Interfaz Serial de Cámara (CSI, por sus siglas en inglés), en sus versiones 1, 2 y 3.

La CPI especifica una interfaz I2C para el control de los parámetros de la cámara por medio de registros y una interfaz paralela para recibir los datos de la imagen capturada. El bus I2C es usado en muchas aplicaciones de sistemas embebidos, permitiendo la comunicación entre sensores a través de líneas de datos compartidas. En el caso de la especificación CPI, sirve para configurar los registros del sensor de imagen y poder así ajustar varios de los parámetros disponibles en el ajuste de la imagen final obtenida.

I2C es un bus de comunicación serial, actualmente propiedad de NXP, empleado en la comunicación con periféricos como memorias, relojes en tiempo real (RTC, por sus siglas en inglés), sensores, pantallas o comunicación entre controladores, todo esto a través de un solo bus compuesto de dos líneas de datos. Utiliza un modo de comunicación "half duplex", es decir, una sola línea de datos puede funcionar para transmitir y recibir, lo cual implica que la comunicación solo puede ser en una dirección a la vez. El bus I2C consiste de dos señales: SDA (datos) y SCL (Reloj).

La interfaz paralela está compuesta por un bus paralelo de n bits unidireccional, por el cual se transmite la información de video, así como las señales de sincronización Vertical (VSYNC), referencia horizontal (HREF) y reloj de pixel (PCLK). Los datos de la imagen están disponibles en la salida para cada flanco de subida de PCLK. HREF se mantiene en alto mientras se encuentra en salida una línea horizontal de la imagen, mientras que VSYNC se encuentra en alto al inicio de un fotograma. Los tiempos de sincronización son especificados en la hoja de datos del sensor empleado.

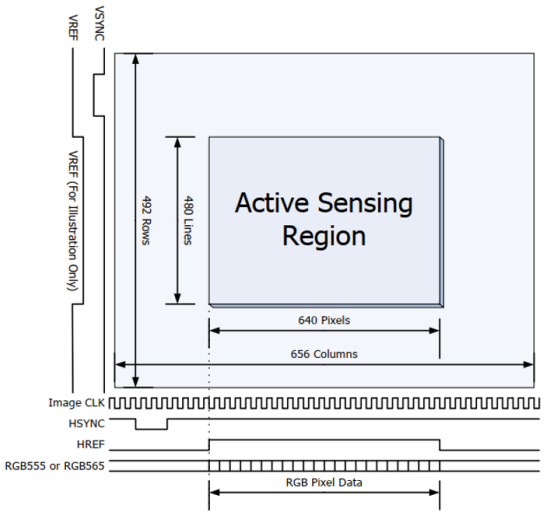


Figura 2.3. Diagrama de tiempos de un sensor de imagen [8].

2.2. Arquitectura de computadoras

Una computadora es una máquina electrónica programable que tiene la capacidad de operar datos binarios en base a una secuencia específica de instrucciones, permitiendo así ejecutar desde tareas simples hasta tareas complejas. Hoy en día existen una gran variedad de computadoras con diseños y características específicas de su funcionalidad, sin embargo tomaremos como referencia la arquitectura clásica de los componentes básicos y principales de una computadora, estos son: el procesador, las unidades de memoria (memoria de programa y de datos) y los puertos de entrada y salida. Un diagrama de bloques que muestra la estructura básica de una computadora se muestra en la figura 2.4.

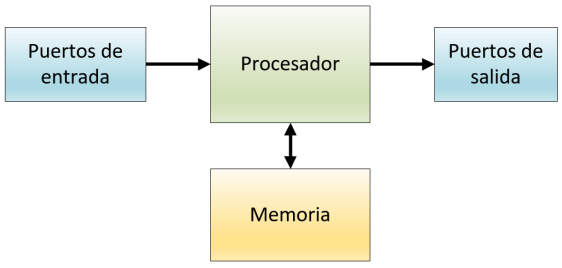


Figura 2.4. Diagrama de bloques de una computadora.

En la arquitectura de una computadora, existen dos modelos de clasificación conocidos como la arquitectura Von Neumann y Harvard. La arquitectura Von Neumann tiene la característica principal de que la memoria de instrucciones o de programa y la memoria de datos tienen acceso a través de un bus compartido de direcciones y de datos, mientras que en la arquitectura Harvard existen buses separados de acceso para la memoria de programa y para la memoria de datos. Ambas arquitecturas se muestran en la figura 2.5. La ventaja que presenta la arquitectura Harvard sobre la arquitectura Von Neumann es que permite el acceso simultáneo hacia la memoria de programa y la memoria de datos, lo cual incrementa su rendimiento. Esto ha hecho que se convierta en el modelo de arquitectura preferido para implementarse en los sistemas de cómputo actuales.

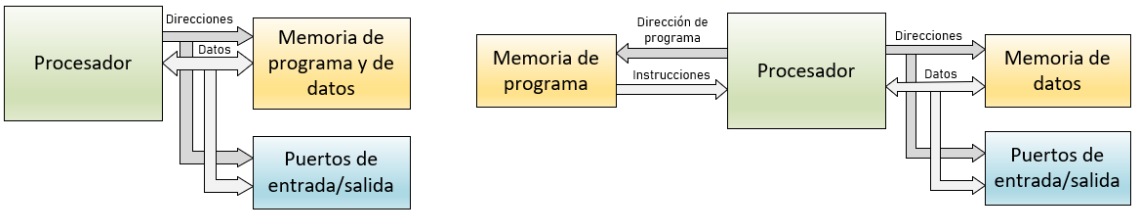


Figura 2.5. Arquitectura Von Neumann (Izquierda) y arquitectura Harvard (Derecha).

El **procesador** es el componente encargado de operar con los datos provenientes de la memoria de programa y de datos, así como de los dispositivos de entrada y salida, siguiendo una ejecución secuencial de instrucciones obtenidas de la memoria de programa. Los resultados de estas operaciones pueden permanecer en las unidades de almacenamiento temporal del procesador o bien ser enviados hacia la memoria de datos o hacia los dispositivos de entrada y salida.

Las **unidades de memoria** corresponden principalmente a la memoria de programa y a la memoria de datos. La memoria de programa es aquella que contiene el código del programa a ejecutar, definido por el conjunto de instrucciones específico de la arquitectura del procesador usado y generalmente es vista como una memoria de solo lectura para el procesador, mientras que la memoria de datos es usada para almacenar y mantener temporalmente resultados intermedios y variables del programa ejecutado, por lo cual es una memoria con capacidad de lectura y escritura.

Los **puertos de entrada y salida** son los componentes de hardware que funcionan como interfaces de comunicación de la computadora con el mundo exterior,

permitiendo así el flujo de datos tanto de entrada como de salida.

2.2.1. Arquitectura del conjunto de instrucciones (ISA, por sus siglas en inglés)

Es el conjunto de instrucciones que el procesador de una computadora puede interpretar y ejecutar, es decir, define los comandos y operaciones que el procesador puede ejecutar, así como los tipos de datos que puede manipular y el modelo de memoria que usa, de manera que se convierte en una interfaz entre el software y el hardware de una computadora.

Clasificación de la arquitectura del conjunto de instrucciones

Una ISA de una computadora puede clasificarse a través del almacenamiento interno empleado en el procesador para los operandos. De esta manera encontramos que existen cuatro principales arquitecturas que puede emplear una ISA:

1. *Arquitectura de pila.* En esta arquitectura los operandos se encuentran implícitamente en la pila (stack en inglés). Los operandos son cargados en el tope de la pila desde la memoria a través de operaciones PUSH, al realizar una operación los operandos en el tope de la pila son destruidos y el resultado ahora es depositado en el tope de esta. Para guardar el resultado que se encuentra en el tope de la pila se ejecuta una instrucción de almacenar. Una operación de POP quita el elemento del tope de pila.

2. *Arquitectura de acumulador.* En esta arquitectura uno de los operadores se encuentra implícito en un registro conocido como acumulador, mientras que el otro operador es explícito al ser directamente el dato de una dirección de memoria. El resultado de las operaciones es guardado en el acumulador y puede ser transferido desde el mismo hacia la memoria.

3. *Arquitectura de registro-memoria.* Esta arquitectura es semejante a la del acumulador, pero en este caso existen varios registros que pueden hacer la función del acumulador. Uno de los operandos se encuentra en un registro mientras que otro se toma explícitamente como dato de una dirección de memoria. El resultado puede guardarse en alguno de los registros.

4. *Arquitectura de carga-almacena.* En esta arquitectura los operadores y resultados residen únicamente en los registros y estos pueden transferirse desde y hacia la memoria únicamente a través de instrucciones de cargar-almacenar (load-store en inglés), en donde el contenido de los registros puede ser usado para direccionar datos en memoria. Actualmente es la arquitectura dominante en muchos procesadores actuales.

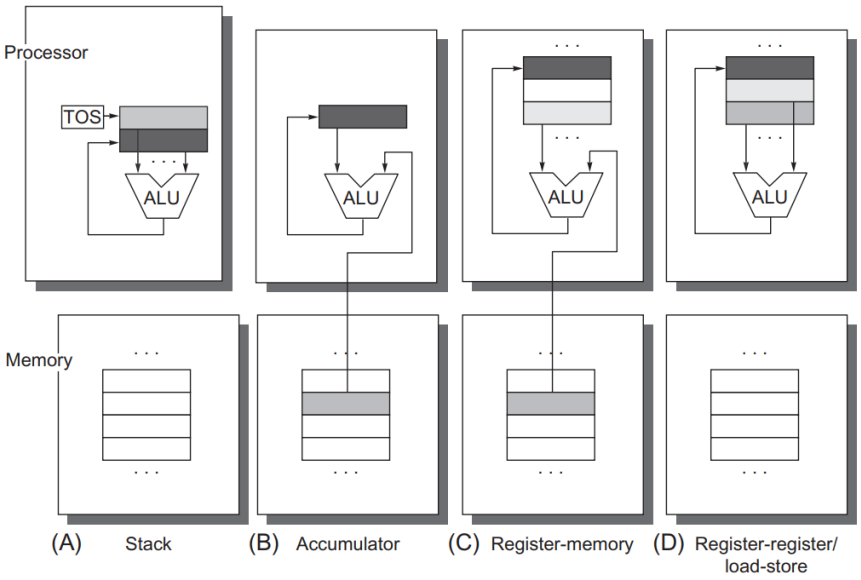


Figura 2.6. Ubicación de los operandos para los cuatro tipos de arquitecturas del set de instrucciones [9].

Modos de direccionamiento

Un modo de direccionamiento es la forma de especificar un operando a usarse en una instrucción, representando así el dato o la dirección en la cual la instrucción opera. Cada modo de direccionamiento provee características que ofrecen flexibilidad y eficiencia al momento de acceder a los datos o instrucciones almacenados en memoria. Entre los principales modos de direccionamiento usados se encuentran los siguientes:

1. Direccionamiento inmediato: en este modo el operando forma parte directa de la instrucción, por lo que generalmente es usado para operar con valores constantes y en operaciones de carga.

2. Direccionamiento por registro: el operando es un valor almacenado en uno de los registros del procesador.

3. Direccionamiento directo o absoluto: el operando es al dato que se encuentra en una dirección de memoria. La dirección de memoria a la que se hace referencia forma parte de la instrucción.

4. Direccionamiento indirecto: un registro funciona como puntero, es decir, contiene un valor que se interpreta como una dirección de memoria. El dato almacenado en esa dirección será el operando a usar en la instrucción. De este modo de direccionamiento se pueden encontrar variantes en las que se opera con la dirección base contenida en el registro.

    - Indexado o con desplazamiento: el operando se obtiene a partir del dato en memoria que se encuentra en la dirección que resulta de sumar el dato de un registro con una constante o con otro registro.
    - Con incremento o decremento: el valor del registro que funciona como puntero se incrementa o decrementa después de que ha sido referenciado. Este modo es útil cuando se requiere acceder a datos contiguos en memoria.

Operaciones en el conjunto de instrucciones

Son el conjunto de acciones que el procesador es capaz de realizar en base a las instrucciones. Para esto, requieren de la intervención de datos de entrada, conocidos como operandos y de un operador, que define la acción a realizar en base a estos datos para producir una salida, denominada como resultado de la operación. Particularmente estas operaciones toman lugar en la unidad aritmético-lógica del procesador.

Los tipos de operaciones principales usadas en la mayoría de los conjuntos de instrucciones son:

- Operaciones aritméticas: son procesos como la suma, resta, multiplicación y división en los que se emplean números enteros con signo o sin signo como operandos. El desplazamiento aritmético también forma parte de esta categoría.

- Operaciones lógicas: son aquellas basadas en el álgebra de Boole como AND, OR, NOT, XOR y NAND. En esta categoría también se incluye el desplazamiento lógico hacia la izquierda o hacia la derecha y las operaciones de comparación.

- Operaciones de transferencia de datos: son acciones que involucran el flujo de datos desde la memoria hacia el procesador, desde el procesador hacia la memoria o entre los elementos de almacenamiento interno del procesador.

- Operaciones de control: son aquellas que modifican el flujo del programa, por medio de saltos hacia secciones definidas en la memoria del programa, los cuales pueden ser condicionados o no condicionados. Los retornos de rutinas de interrupción entran también en esta categoría.

- Operaciones de punto flotante: son procesos como la suma, resta, multiplicación y división que actúan sobre datos interpretados como de tipo punto flotante.

- Operaciones del sistema y especiales: controlan funciones como llamadas del sistema operativo, privilegios de acceso o modifican el contenido de los registros especiales.

2.3. Memorias DRAM

La memoria dinámica de acceso aleatorio (DRAM, por sus siglas en inglés) es una tecnología de almacenamiento desarrollada por IBM que permite guardar información en forma de carga eléctrica almacenada en capacitores, lo cual hace también que pierda la información una vez que se interrumpe el suministro de energía (volátil). La memoria i1103 de Intel lanzada en 1970 fue una de las versiones exitosas de esta memoria que comenzaba a competir como memoria principal frente a las memorias basadas en núcleos magnéticos. Su simplicidad de hardware también le permitió posicionarse sobre la tecnología de la memoria SRAM, por lo que rápidamente se convirtió en la tecnología de memoria principal dominante en la industria de computadoras.

La celda básica de almacenamiento de la DRAM moderna se basa en un circuito base conformado por un capacitor y un transistor MOS (configuración 1T1C), como se muestra en la figura 2.7. El transistor opera en las regiones de corte y saturación a través de un nivel de voltaje aplicado a la compuerta, haciendo que funcione como un interruptor que se enciende y apaga, permitiendo así el acceso al capacitor que almacenará el dato.

Cuando el transistor se encuentra en saturación (encendido) permite que el capacitor adquiera el nivel lógico de la línea de datos, ya se cargándose con un nivel alto

(1) o descargándose con un nivel en bajo (0), permitiendo así que funcione como un elemento de almacenamiento. Cuando el transistor se encuentra en corte (apagado) no se puede acceder al capacitor para modificar o leer la información, sin embargo, debido las características físicas no ideales del circuito existen corrientes de fuga que provocan la descarga del capacitor y por lo tanto la pérdida de la información, de ahí su característica volátil de la DRAM.

Para mitigar el efecto de pérdida de información por la descarga del capacitor, se emplea un proceso conocido como refresco, el cual consiste en tomar lectura del voltaje producto de la carga almacenada en el capacitor a través de la línea de datos, establecer su valor lógico equivalente y posteriormente volver a cargar el capacitor con el nivel correspondiente al dato guardado. Este proceso de refresco de la información en las celdas de memoria, hace que sea una memoria dinámica.

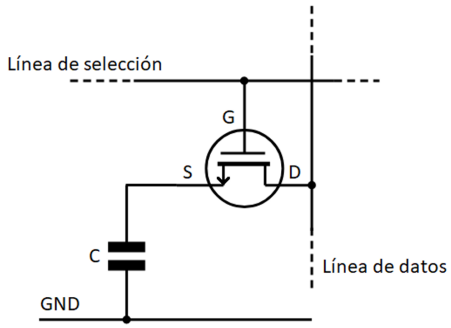


Figura 2.7. Celda de memoria DRAM.

A partir del transistor y capacitor que conforman la celda de memoria, es posible crear arreglos de estas, que en conjunto con la lógica de control, decodificación y acceso conforman la arquitectura base de la memoria DRAM. Un arreglo básico de una memoria DRAM se muestra en la Figura 2.8. A la línea de selección se le conoce también como línea de palabra (wordline, en inglés) o línea de fila y a la línea de datos como línea de bit (bitline, en inglés). En esta arquitectura se ha optado por realizar un multiplexado de la dirección de memoria, de manera que la dirección de fila y columna para el arreglo se carga en dos etapas a través de las mismas líneas de dirección, haciendo uso de las señales de sincronización strobe de dirección de fila (RAS, por sus siglas en inglés) y strobe de dirección de columna (CAS, por sus siglas en inglés).

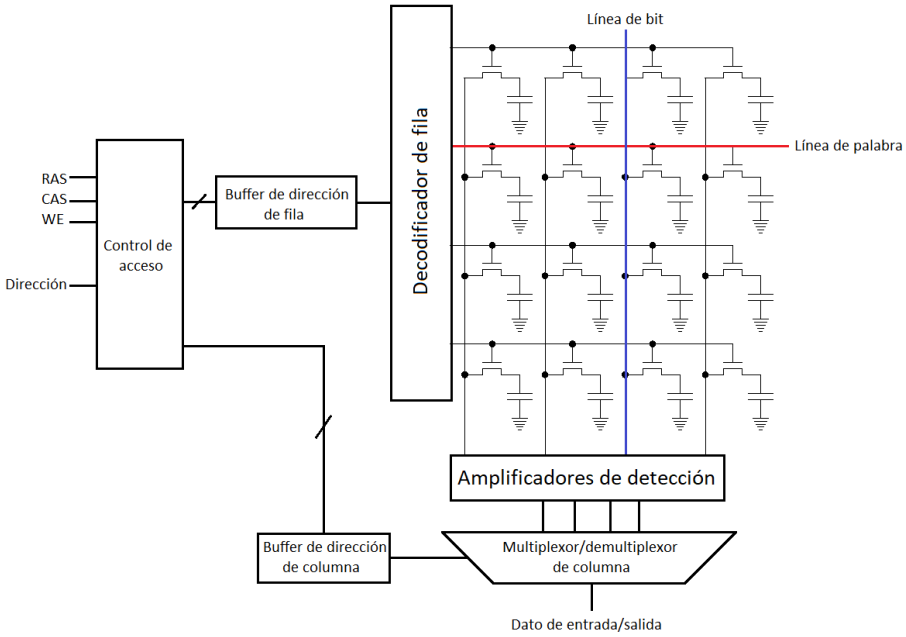


Figura 2.8. Arquitectura básica de una memoria DRAM.

En la Figura 2.9 se muestra el diagrama de tiempo de la operación de lectura en la memoria. Tomando como referencia el diagrama de la Figura 2.8, la dirección de una celda se carga en dos etapas. La dirección de fila se encuentra presente en el bus de dirección y a partir de la señal (RAS), se carga en el buffer de dirección de fila y es decodificada para seleccionar la correspondiente línea de palabra. A continuación, en el bus de dirección se encuentra la dirección de columna y se carga en el buffer por medio de la señal (CAS), esto permite seleccionar a la celda de memoria requerida y hace que el dato esté disponible a la salida.

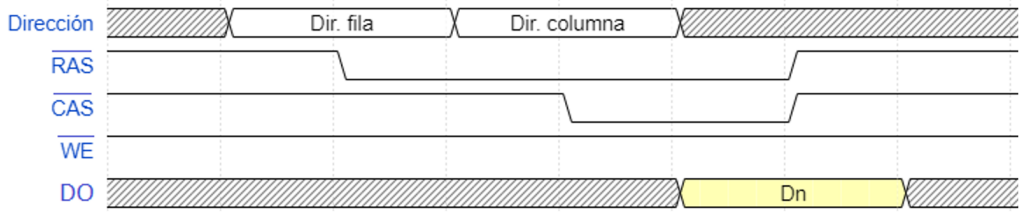


Figura 2.9. Diagrama de tiempo de operación de lectura en una DRAM.

En la Figura 2.10, se muestra el diagrama de tiempo para la operación de escritu-

ra, que al igual que en la lectura necesita de dos etapas para direccionar a una celda de memoria por medio de las señales (RAS) y (CAS), con la adición de la activación de la señal de escritura (WE) y la presencia del dato de entrada antes de activar la señal (CAS).

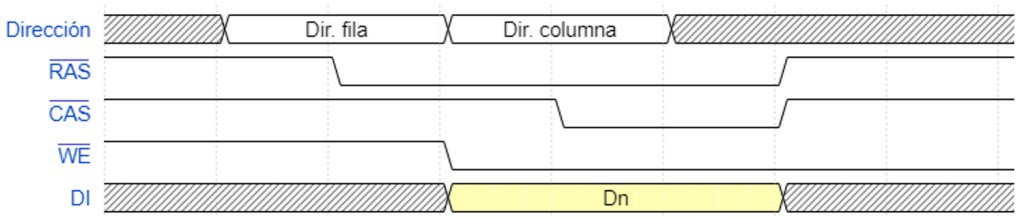


Figura 2.10. Diagrama de tiempo de operación de escritura de una DRAM.

A nivel de celda, el transistor de acceso es activado por medio del voltaje aplicado en la compuerta a través de la línea de fila. Los amplificadores de detección son necesarios para establecer un valor digital a la salida, debido a que el nivel de voltaje que se encuentra en la línea se ve afectado como consecuencia de la descarga de los capacitores.

La arquitectura presentada en la Figura 2.8 presenta la base de funcionamiento de una arquitectura de memoria DRAM, sin embargo, para fines prácticos se debe considerar que debido a las características físicas del arreglo, las líneas de fila y columnas son lo suficientemente grandes como para presentar una capacitancia que supera a la capacitancia de una sola celda de memoria, por lo que se presenta un escenario en el cual es difícil medir el pequeño cambio de carga producido por una celda de memoria sobre la línea de datos. Es por ello que en la arquitectura se ha optado por añadir una columna adicional de celdas de memoria (Figura 2.11), con el fin de hacer que para cada columna exista un par de estas líneas, de manera que se pueda medir mejor el cambio de voltaje a través del amplificador diferencial. La segunda línea añadida a cada columna presenta características idénticas a su adyacente, de manera que permite establecer una referencia estable para el amplificador diferencial.

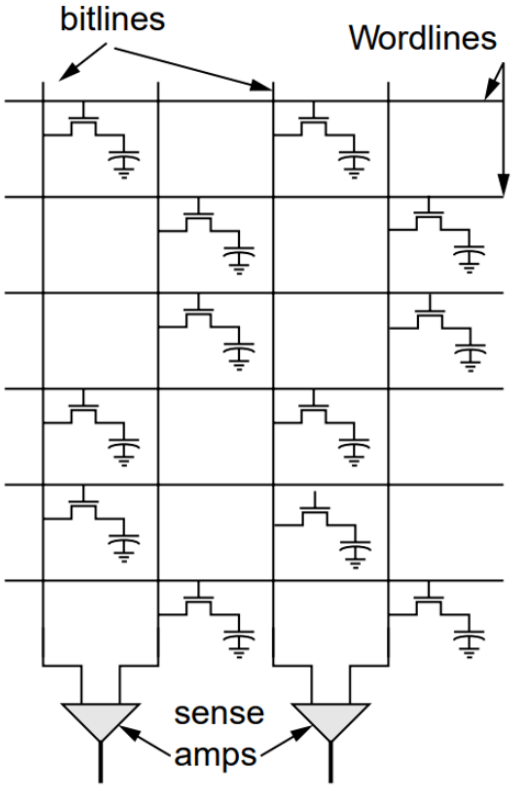


Figura 2.11. Arquitectura DRAM con doble línea de bit [10].

2.4. Memorias SDRAM DDRn

Las memorias SDRAM DDRn son un tipo de memorias síncronas, de acceso aleatorio, con tasa de datos doble, dinámicas y volátiles que emplean como base de operación a la memoria DRAM en conjunto con una arquitectura de control que le permite funcionar de manera síncrona y con doble tasa de datos. La interfaz DDRn SDRAM contempla la transferencia de datos tanto en el flanco de subida como el de bajada de la señal de reloj, es decir dos tiempos de bit en un solo periodo de tiempo, por lo cual es posible tener una transferencia al doble de la velocidad de reloj. A partir de esta interfaz se han desarrollado distintas generaciones que han ido añadiendo mejoras en la arquitectura denominadas como DDR o DDR1, DDR2, DDR3, DDR4 y DDR5.

Para la realización de este trabajo se contempla el uso de una memoria SDRAM

DDR3, particularmente el modelo MT41K256M16 de Micron, por lo cual se abordará el principio de funcionamiento y características aplicables a este modelo de memoria, de las cuales es posible encontrar también en otros modelos de memorias tipo DDR3 debido a las especificaciones del estándar.

2.5. Arquitectura de una memoria SDRAM DDR3

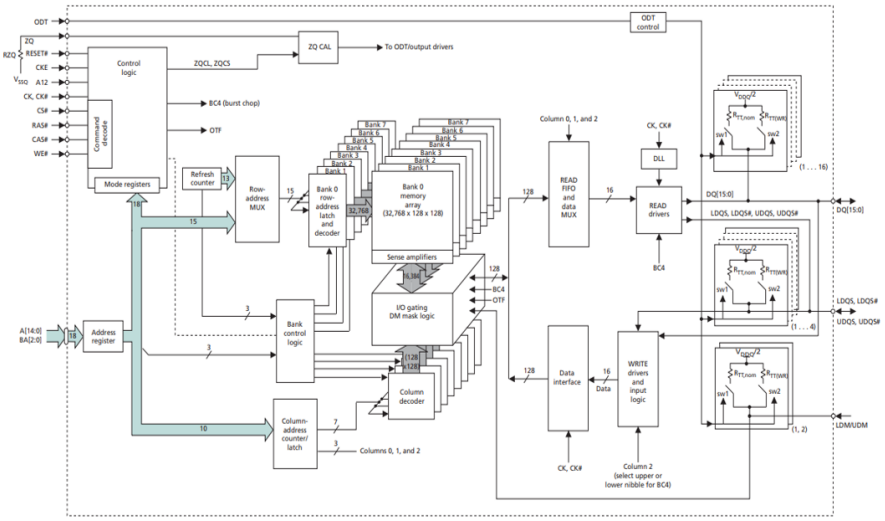


Figura 2.12. Diagrama de bloques de la arquitectura interna de una memoria SDRAM DDR3 [11].

Organización de la memoria

La memoria está organizada a través de bancos, filas, columnas y palabras. Para el caso de la arquitectura mostrada en la Figura 6, la memoria está organizada en 8 bancos con 32,768 filas cada uno y a su vez en cada una de estas 128 columnas con 128 bits (8 palabras de 16 bits) cada una. A su vez la configuración de la memoria indica el ancho del bus de datos y por lo tanto del tamaño de palabra, teniendo las configuraciones x4, x8 y x16.

Tipo de ráfaga (Burst type)

Indica la secuencia programada de accesos mediante los cuales se pueden realizar las operaciones en las columnas.

Longitud de ráfaga (Burst length)

Indica el número de columnas que serán leídas o escritas por cada comando de escritura o lectura. Para la generación DDR3 el tamaño puede ser de 4 (chop) u 8 columnas por ráfaga. El acceso a una columna durante una escritura se puede bloquear mediante el pin de máscara de datos (DM), evitando así que se modifique su contenido, permitiendo escribir un número de bytes menor al de la ráfaga.

Prefetch n

El tamaño de prefetch indica el número de palabras de una columna que se pueden almacenar temporalmente durante los procesos de lectura y escritura, donde el tamaño de palabra n está determinado por la configuración de la memoria. Para la generación de memorias DDR3 el prefetch es de 8n, lo cual significa que para una memoria en configuración x16 se puede operar hasta 8 palabras de 16 bits por cada operación de lectura o escritura.

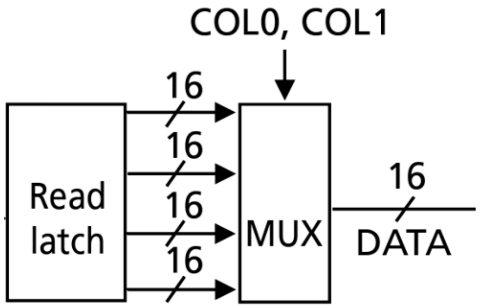


Figura 2.13. Representación del prefetch y salida de datos [11].

Cada circuito integrado de memoria contiene una estructura similar a la Figura 6. El conjunto de entradas RAS#, CAS#, WE# y CS# definen el comando a ejecutarse por la lógica de control de la memoria. Así, las definiciones de las señales son las siguientes:

- CK, CK#: representan las señales de reloj diferenciales. Las señales de control y dirección son muestreadas en el cruce del flanco positivo de CK y el flanco negativo de CK#.

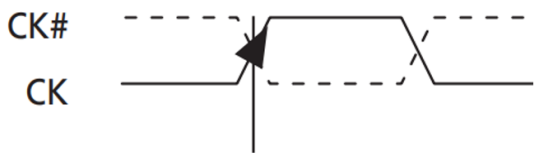


Figura 2.14. Señal de reloj diferencial CK [11].

- RAS#, CAS# y WE#: definen el comando que será ejecutado por la lógica de control de la memoria.
- CS#: representa la línea de selección de la memoria, que al ser registrada en bajo habilita el decodificador de comandos y lo deshabilita al registrarse en alto, haciendo que todos los comandos ingresados sean enmascarados, permitiendo así la activación selectiva de memorias en sistemas con múltiples unidades.
- CKE: habilita o deshabilita las señales de reloj y la circuitería interna de la memoria DRAM.
- A[n:0]: son entradas de datos que de acuerdo al proceso a ejecutar pueden representar parámetros de configuración, dirección de fila, dirección de columna, tamaño de ráfaga o tipo de operación.
- BA[n:0]: son entradas que de acuerdo al proceso representan el banco sobre el cual se aplica un comando o el registro interno a configurar.
- DQ[n:0]: es el bus de datos bidireccional para la memoria.
- DQS, DQS#: es una señal de sincronización diferencial y bidireccional para las operaciones de lectura y escritura. En una lectura actúa como salida y está alineada al flanco de subida de los datos leídos. En una escritura actúa como entrada y está centrada con los datos escritos. Para el caso de memorias con configuración x16, se emplean dos pares de esta señal, UDQS y UDQS# para el byte superior y LDQS y LDQS# para el byte inferior.
- DM: es una señal de entrada que al ser registrada en alto permite enmascarar datos al momento de realizar ráfagas de escritura, evitando que los datos en esa zona de memoria sean modificados y permitiendo realizar operaciones menores al tamaño de ráfaga disponible. Para memorias con configuración x16, se emplea UDM para el byte superior y LDM para el byte inferior.
- RESET#: es una señal asíncrona y activa en bajo para el reinicio de toda la lógica de la memoria SDRAM.

- ODT: en alto habilita la resistencia de terminación interna de la SDRAM y en bajo la deshabilita.
- ZQ: es una referencia externa conectada a una resistencia para cuando se realiza la calibración de la memoria.

Diagrama simplificado de funcionamiento

El diagrama simplificado de la memoria SDRAM DDR3 se muestra en la Figura 9. El primer estado comienza a partir del encendido del dispositivo, para después pasar al proceso de reinicio, en el cual se asegura un inicio desde cero en la memoria. Posteriormente se ejecuta la secuencia de inicialización, en la cual se configuran los parámetros de funcionamiento de la memoria a través de los registros de configuración internos. Posterior a esto se lleva a cabo el proceso de calibración, mediante el cual se establecen los parámetros de las resistencias de acondicionamiento de señal. Al finalizar este proceso la memoria se encuentra en estado inactivo, en el cual es posible ejecutar operaciones como autorrefresco, refresco, suspensión con precarga, calibración o activación de una fila para realizar operaciones de escritura y lectura sobre esa misma fila. Cuando se requiera acceder a otra fila, esta debe ser previamente precargada y volver al estado inactivo antes de iniciar con otro proceso de escritura o lectura.

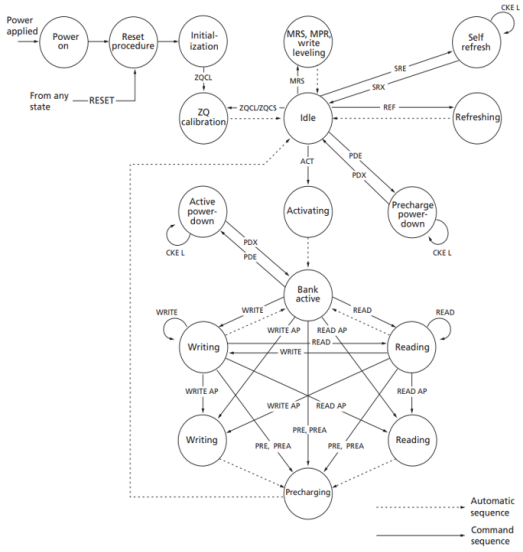


Figura 2.15. Diagrama de estados simplificado [11].

Descripción de los comandos de la memoria DDR3

Function		Symbol	CKE Prev. Cycle	CKE Next Cycle	CS#	RAS#	CAS#	WE#	BA [2:0]	An	A12	A10	A[11, 9:0]	Notes
MODE REGISTER SET		MRS	H	H	L	L	L	L	BA	OP code				
REFRESH		REF	H	H	L	L	L	H	V	V	V	V	V	
Self refresh entry		SRE	H	L	L	L	L	H	V	V	V	V	V	6
Self refresh exit		SRX	L	H	H	V	V	V	V	V	V	V	V	6, 7
					L	H	H	H						
Single-bank PRECHARGE		PRE	H	H	L	L	H	L	BA	V	V	L	V	
PRECHARGE all banks		PREA	H	H	L	L	H	L	V		V	H	V	
Bank ACTIVATE		ACT	H	H	L	L	H	H	BA	Row address (RA)				
WRITE	BL8MRS, BC4MRS	WR	H	H	L	H	L	L	BA	RFU	V	L	CA	8
	BC4OTF	WRS4	H	H	L	H	L	L	BA	RFU	L	L	CA	8
	BL8OTF	WRS8	H	H	L	H	L	L	BA	RFU	H	L	CA	8
WRITE with auto precharge	BL8MRS, BC4MRS	WRAP	H	H	L	H	L	L	BA	RFU	V	H	CA	8
	BC4OTF	WRAPS4	H	H	L	H	L	L	BA	RFU	L	H	CA	8
	BL8OTF	WRAPS8	H	H	L	H	L	L	BA	RFU	H	H	CA	8
READ	BL8MRS, BC4MRS	RD	H	H	L	H	L	H	BA	RFU	V	L	CA	8
	BC4OTF	RDS4	H	H	L	H	L	H	BA	RFU	L	L	CA	8
	BL8OTF	RDS8	H	H	L	H	L	H	BA	RFU	H	L	CA	8
READ with auto precharge	BL8MRS, BC4MRS	RDAP	H	H	L	H	L	H	BA	RFU	V	H	CA	8
	BC4OTF	RDAPS4	H	H	L	H	L	H	BA	RFU	L	H	CA	8
	BL8OTF	RDAPS8	H	H	L	H	L	H	BA	RFU	H	H	CA	8
NO OPERATION		NOP	H	H		H	H	H	V	V	V	V	V	9
Device DESELECTED		DES	H	H	H	X	X	X	X	X	X	X	X	10
Power-down entry		PDE	H	L	L	H	H	H	V	V	V	V	V	6
					H	V	V	V						
Power-down exit		PDX	L	H	L	H	H	H	V	V	V	V	V	6, 11
					H	V	V	V						
ZQ CALIBRATION LONG		ZQCL	H	H	L	H	H	L	X	X	X	H	X	12
ZQ CALIBRATION SHORT		ZQCS	H	H	L	H	H	L	X	X	X	L	X	

Notes: 1. Commands are defined by the states of CS#, RAS#, CAS#, WE#, and CKE at the rising edge of the clock. The MSB of BA, RA, and CA are device-, density-, and configuration-dependent.

Figura 2.16. Tabla de comandos de una memoria DDR3 [11].

Deseleccionar (Deselect)

Evita la ejecución de nuevos comandos en el controlador de la memoria DRAM, mientras que aquellas operaciones en progreso no se verán afectadas.

No operación (No operation)

Previene que se registren comandos no deseados durante los estados de espera o de inactividad. Las operaciones en progreso no se ven afectadas.

Calibración ZQ larga y ZQ corta (Long ZQ calibration, Short ZQ calibration)

El comando de calibración larga es usado para llevar a cabo la calibración de las impedancias en los pines DQn durante la etapa de inicialización y secuencia de reinicio. Al solicitarse se activan los sistemas de calibración de la DRAM para obtener valores de calibración que son transferidos a los pines DQn. Este comando puede ser solicitado en cualquier momento por el controlador de acuerdo a las condiciones del

ambiente. Por otra parte, la calibración corta es usada para llevar a cabo calibraciones periódicas ocasionadas por pequeñas variaciones en los voltajes o temperaturas.

Activar (Activate)

Es usado para abrir (o activar) una fila (row) en un banco (bank) para un acceso subsecuente. Los valores en las entradas BA[2:0] seleccionan el banco, mientras que los valores en las entradas A[n:0] seleccionan la fila. Esta fila permanece abierta para ser accedida hasta que un comando de precarga es solicitado para ese o todos los bancos. Es necesario ejecutar un comando de precarga para cerrar el banco antes de acceder a otra fila del mismo.

Lectura (Read)

Este comando es usado para iniciar una ráfaga de lecturas sobre una fila activa de un banco. La dirección provista en las entradas A[2:0] seleccionan la dirección de la columna de inicio en la fila activa, dependiendo además de la longitud y tipo de ráfaga seleccionada. Una vez solicitado este comando no se debe interrumpir la operación. El valor en la entrada A10 determina si se usará la auto precarga del banco. Si se selecciona auto precarga, la fila a la que se está accediendo será precargada al final de la ráfaga de lectura, en caso contrario la fila permanecerá abierta para accesos subsecuentes. Adicionalmente la entrada A12 determina si se usa el modo BC4 o BL8 durante la lectura, de acuerdo con el valor programado en el registro de configuración.

Escritura (Write)

Este comando es usado para iniciar una ráfaga de escritura hacia una fila activa. El valor en las entradas BA[2:0] seleccionan el banco, mientras que A10 selecciona si se usará auto precarga y el valor en A12 determina el modo BC4 o BL8 de acuerdo a la configuración de los registros.

Precarga (Precharge)

Es usado para desactivar la fila abierta en un banco en particular o en todos los bancos. La entrada A10 determina si se precargará uno o todos los bancos, mientras que las entradas BA[n:0] determinarán el banco a precargar o serán tratadas como no importa para el caso de precarga en todos los bancos. Una vez que un banco ha sido precargado, este se encontrará en estado inactivo, por lo que deberá de activarse previamente para ejecutar operaciones de lectura y escritura en ese banco.

El comando se interpreta como NOP cuando el banco seleccionado o todos los bancos se encuentran en estado inactivo o si la fila del banco se encuentra previamente en proceso de precarga.

Refresco (Refresh)

Es usado durante la operación normal de la DRAM y al ser un comando no persistente debe ejecutarse cada vez que se requiera un refresco de la información. El direccionamiento es generado por el controlador de refresco interno, por lo que las entradas de dirección son tomadas como no importa durante esta operación.

Auto refresco (Self refresh)

Este comando sirve para retener los datos en la DRAM sin la intervención del controlador y sin requerir las señales de reloj externas.

2.5.1. Registros de configuración o modo (Mode registers)

Registro de configuración 0 (Mode Register 0)

Se usa para definir varios modos de operación de la memoria, como el tamaño de ráfaga, tipo de ráfaga, latencia CAS, modo de operación, reinicio del DLL, tiempo de recuperación de escritura y el modo inactivo de precarga.

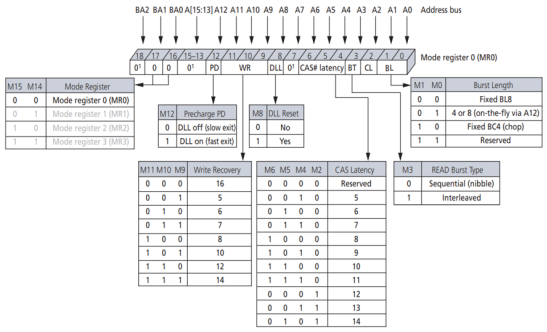


Figura 2.17. Registro de modo 0 [11].

Longitud de ráfaga (Burst length)

Debido a que las operaciones de lectura y escritura de la memoria son orientadas a ráfagas, es posible definir el tamaño de palabras a operar durante estos procesos,

ya sea en ajuste fijo a 4 palabras (chop) u 8 palabras (fixed) o bien en ajuste sobre la marcha, donde el tamaño de 4 y 8 se selecciona por medio de la entrada de dirección A12.

Tipo de ráfaga (Burst type)

Define el orden de acceso de una ráfaga para ser en modo secuencial o en modo intercalado. El orden de acceso en una ráfaga está determinado por la longitud de ráfaga, el tipo de ráfaga y la dirección de la columna de inicio.

Burst Length	READ/ WRITE	Starting Column Address (A[2, 1, 0])	Burst Type = Sequential (Decimal)	Burst Type = Interleaved (Decimal)	Notes
4 (chop)	READ	0 0 0	0, 1, 2, 3, Z, Z, Z, Z	0, 1, 2, 3, Z, Z, Z, Z	1, 2
		0 0 1	1, 2, 3, 0, Z, Z, Z, Z	1, 0, 3, 2, Z, Z, Z, Z	1, 2
		0 1 0	2, 3, 0, 1, Z, Z, Z, Z	2, 3, 0, 1, Z, Z, Z, Z	1, 2
		0 1 1	3, 0, 1, 2, Z, Z, Z, Z	3, 2, 1, 0, Z, Z, Z, Z	1, 2
		1 0 0	4, 5, 6, 7, Z, Z, Z, Z	4, 5, 6, 7, Z, Z, Z, Z	1, 2
		1 0 1	5, 6, 7, 4, Z, Z, Z, Z	5, 4, 7, 6, Z, Z, Z, Z	1, 2
		1 1 0	6, 7, 4, 5, Z, Z, Z, Z	6, 7, 4, 5, Z, Z, Z, Z	1, 2
		1 1 1	7, 4, 5, 6, Z, Z, Z, Z	7, 6, 5, 4, Z, Z, Z, Z	1, 2
	WRITE	0 V V	0, 1, 2, 3, X, X, X, X	0, 1, 2, 3, X, X, X, X	1, 3, 4
		1 V V	4, 5, 6, 7, X, X, X, X	4, 5, 6, 7, X, X, X, X	1, 3, 4
8 (fixed)	READ	0 0 0	0, 1, 2, 3, 4, 5, 6, 7	0, 1, 2, 3, 4, 5, 6, 7	1
		0 0 1	1, 2, 3, 0, 5, 6, 7, 4	1, 0, 3, 2, 5, 4, 7, 6	1
		0 1 0	2, 3, 0, 1, 6, 7, 4, 5	2, 3, 0, 1, 6, 7, 4, 5	1
		0 1 1	3, 0, 1, 2, 7, 4, 5, 6	3, 2, 1, 0, 7, 6, 5, 4	1
		1 0 0	4, 5, 6, 7, 0, 1, 2, 3	4, 5, 6, 7, 0, 1, 2, 3	1
		1 0 1	5, 6, 7, 4, 1, 2, 3, 0	5, 4, 7, 6, 1, 0, 3, 2	1
		1 1 0	6, 7, 4, 5, 2, 3, 0, 1	6, 7, 4, 5, 2, 3, 0, 1	1
		1 1 1	7, 4, 5, 6, 3, 0, 1, 2	7, 6, 5, 4, 3, 2, 1, 0	1
	WRITE	V V V	0, 1, 2, 3, 4, 5, 6, 7	0, 1, 2, 3, 4, 5, 6, 7	1, 3

Figura 2.18. Orden de las ráfagas de datos [11].

Reinicio del DLL (Reset DLL)

Permite reiniciar el DLL al escribir un valor de 1 en el campo de bit, mismo que será limpiado automáticamente una vez que se haya iniciado la función de reinicio. Cada vez que se inicie la función, CKE debe estar en alto y el reloj se debe de mantener estable por 512 ciclos de reloj (t_{DLLK}) antes de la solicitud de un comando de lectura, con el fin de permitir la sincronización del reloj interno con el reloj externo. Si esta sincronización no se realiza correctamente puede conducir a salidas con especificaciones inválidas de tiempo.

Recuperación de escritura (Write recovery)

Permite establecer el tiempo mínimo o máximo de retardo en ciclos de reloj (t_{CK}) que conforma el tiempo en que la operación de escritura será completada (t_{WR}) y esté lista para iniciar una operación de precarga.

Modo inactivo de precarga (Precharge power-down)

El bit de configuración solo es válido mientras el modo inactivo de precarga se esté usando.

- Cuando se establece en 0, el DLL se apaga durante el modo inactivo de precarga, disminuyendo el consumo energético, pero teniendo en consideración que se debe satisfacer un tiempo t_{XPDLL} al salir.
- Cuando se establece en 1, el DLL continúa encendido durante el modo inactivo de precarga, permitiendo así una salida más rápida del mismo, satisfaciendo un tiempo de t_{XP}.

Latencia CAS (CAS latency)

Es el retardo en ciclos de reloj (t_{CK}) entre el comando interno de lectura y la disponibilidad del primer bit de datos a la salida. Está definido como CL. Registro de configuración 1 (Mode Register 1) Controla funciones adicionales y características que no están disponibles en otros registros, como la desactivación de la salida, TDQS, habilitación del DLL, valores de ODT, nivelación de escritura, posted CAS additive latency y drive strength. Este registro debe ser cargado cuando todos los bancos están inactivos y no hay transferencias en progreso.

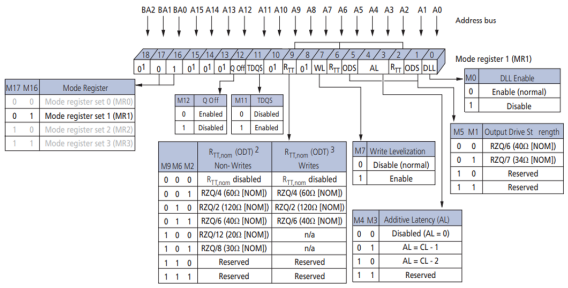


Figura 2.19. Registro de modo 1 [11].

Activación/desactivación del DLL

El DLL puede ser activado o desactivado durante el comando de carga de configuración (MRS). Debe estar activado para la operación normal, además, se requiere que esté activado durante la etapa de inicialización y al regresar a la operación normal, después de haber sido desactivado para propósitos de evaluación y depuración. Se debe reiniciar el DLL posterior a su activación.

Output drive strength

La memoria emplea un buffer de salida con impedancia programable.

Activación/desactivación de la salida

Cuando se encuentra en 0, todas las salidas (DQ, DQS, DQS#) se encuentran en modo normal de operación, si se establece en 1, todas estas salidas se ponen en alta impedancia.

Nivelación de escritura (Write leveling)

Este proceso es usado durante la inicialización para alinear la señal DQS con la señal de reloj de la memoria.

Posted CAS additive latency (AL)

Es una característica soportada para hacer el bus de datos y comandos eficientes para los anchos de banda de la memoria DDR3.

Registro de configuración 2 (Mode Register 2)

Controla funciones adicionales como la latencia de escritura CAS (CWL, por sus siglas en inglés), auto refresco (ASR, por sus siglas en inglés), temperatura de auto refresco (SRT, por sus siglas en inglés) y ODT dinámico.

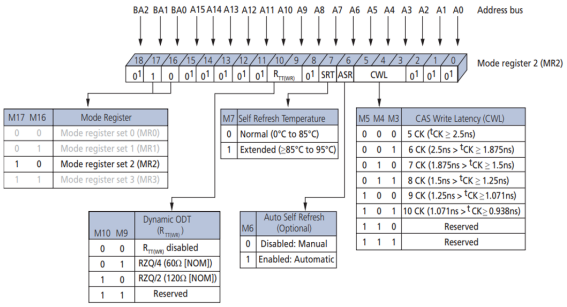


Figura 2.20. Registro de modo 2 [11].

Latencia de escritura CAS (CWL)

Es el retardo en ciclos de reloj (t_{CK}) a partir del procesamiento interno del comando de escritura hasta la captura del primer dato de entrada. Debe ser establecido

correctamente de acuerdo a la frecuencia de operación de la memoria. La latencia de escritura general (WL) será entonces igual a CWL + AL.

Registro de configuración 3 (Mode Register 3)

Se le conoce también como el registro multipropósito (MPR, por sus siglas en inglés). Es usado para producir una salida de una secuencia de bits para la calibración del sistema. Al escribir en el campo de bit correspondiente se puede activar o desactivar su función. Cuando está desactivado, la memoria opera en modo normal, mientras que, al activarlo, la memoria opera en un modo especial en el cual solo se permiten los comandos de lectura y lectura con autoprecarga, así como un subsecuente comando MRS para desactivarlo. La función de reinicio también es válida durante este modo.

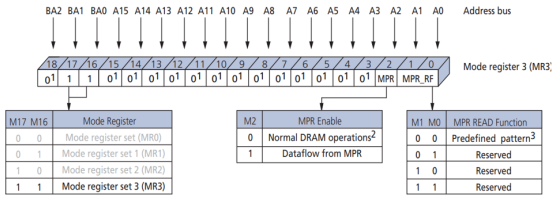


Figura 2.21. Registro de modo 3 [11].

Al estar activada la función MPR, al ejecutar un comando de lectura se obtiene un patrón de bits a la salida, que corresponde a la secuencia 0, 1, 0, 1, 0, 1, 0, 1 para el caso de una ráfaga de tamaño 8 (BL8) y 0, 1, 0, 1 para una ráfaga de tamaño 4 (BC4), en la que el orden de ráfaga no altera el patrón de salida.

MR3[2]	MR3[1:0]	Function	Burst Length	Read A[2:0]	Burst Order and Data Pattern
1	00	READ predefined pattern for system calibration	BL8	000	Burst order: 0, 1, 2, 3, 4, 5, 6, 7 Predefined pattern: 0, 1, 0, 1, 0, 1, 0, 1
			BC4	000	Burst order: 0, 1, 2, 3 Predefined pattern: 0, 1, 0, 1
			BC4	100	Burst order: 4, 5, 6, 7 Predefined pattern: 0, 1, 0, 1
1	01	RFU	N/A	N/A	N/A
			N/A	N/A	N/A
			N/A	N/A	N/A
1	10	RFU	N/A	N/A	N/A
			N/A	N/A	N/A
			N/A	N/A	N/A
1	11	RFU	N/A	N/A	N/A
			N/A	N/A	N/A
			N/A	N/A	N/A

Figura 2.22. Registro de modo 3 - tabla [11].

Modo deshabilitado del DLL (DLL disable mode)

Es una característica que permite deshabilitar el DLL de la memoria, haciendo que la relación entre los tiempos de sincronía se vea afectada, además de permitir frecuencias de operación fuera del estándar, definidas por t_{CK}. Entre las consideraciones al usar este modo se encuentran las siguientes:

- La DRAM soporta solamente un valor de latencia CAS (CL = 6) y un valor de latencia CAS de escritura (CWL = 6).
- La relación entre la señal de reloj y la señal de strobe (t_{DQSCK}) se ve afectada, pero no así la relación de la señal de strobe con los propios datos (t_{DQSQ}, t_{QH}).
- En la operación normal (DLL habilitado), t_{DQSCK} comienza a partir del flanco de subida una vez transcurridos (AL + CL) ciclos de reloj a partir del comando de lectura, mientras que con el DLL deshabilitado lo hace en un ciclo de reloj menos (AL + CL - 1), además de que el valor de t_{DQSCK} puede ser más grande que t_{CK}.
- La característica de ODT no está soportada cuando se deshabilita el DLL, por lo que las resistencias ODT se deben deshabilitar previamente a través de los registros de configuración.

Cambio de la frecuencia del reloj de entrada (Input clock frequency change)

Es el proceso que permite cambiar la frecuencia de operación del reloj de la memoria hacia otra válida y estable. Esta frecuencia solo se permite cambiar bajo dos condiciones: en el modo de auto refresco y bajo el modo inactivo de precarga, siendo ilegal hacerlo en cualquier otro modo.

Nivelación de escritura (Write leveling)

Es un proceso mediante el cual es posible ajustar la relación de la señal strobe DQS (DQS, DQS#) respecto a la señal de reloj CK para realizar las operaciones de escritura. Durante este proceso, DQS funciona como entrada mientras que DQ como salida, permitiendo así el ajuste de la señal DQS de acuerdo al nivel de desfasamiento. Se requiere que el controlador tenga ajuste de retardo en la señal DQS para alinea el flanco de subida a la señal de reloj CK.

Para realizar el proceso de nivelación, la memoria envía el estado de la señal de reloj a través del bus DQ y se muestrea con cada flanco de subida de DQS. El controlador entonces retrasa la señal de strobe DQS hasta que una transición de 0 a 1 es detectada. Este proceso debe realizarse como consecuencia del desfase entre las señales como consecuencia de la topología fly-by.

Cambio de voltaje (Voltage change/Initialization)

Es una característica que permite cambiar el voltaje entre los niveles de 1.35 V (DDR3L) y 1.5 V (DDR3) del estándar. La secuencia requerida para realizar el cambio se especifica en la hoja de datos.

Modo de temperatura extendido (Extended temperature usage)

Es una característica que le permite a la memoria operar en un rango superior al estándar (85°C), logrando alcanzar una temperatura de hasta 95°C, teniendo como consecuencia una modificación en el ciclo de refresco de la memoria, el cual ha de ser dos veces más frecuente.

2.6. Arreglo de compuertas lógicas programables en campo (FPGA)

Los FPGA son dispositivos electrónicos que poseen internamente un arreglo de bloques funcionales, elementos lógicos y conexiones programables, que tienen la capacidad de ser conectados y configurados a través de una memoria de configuración. Xilinx creó el primer FPGA comercial (XC2064) en 1985, como una nueva propuesta tecnológica frente a los dispositivos lógicos programables que existian, al contar con compuertas y conexiones programables. Este dispositivo contaba con 64 bloques lógicos configurables (CLB, por sus siglas en inglés) y con dos tablas de búsqueda (LUT, por sus siglas en inglés) de tres entradas.

Actualmente en el mercado, los principales fabricantes de FPGA son Xilinx (Propiedad de AMD), Intel (con la adquisición de Altera), Lattice Semiconductor y Microchip. Cada uno de estos fabricantes han definido características adicionales a cada uno de sus productos con la finalidad de abastecer los requerimientos del mercado y de aplicaciones específicas.

La arquitectura de cada fabricante contiene características que han sido agregadas para brindar distintas funcionalidades y potenciar su uso en áreas específicas. Para el caso de Xilinx, a lo largo del desarrollo de sus productos ha implementado características que han permitido mejorar el rendimiento de los mismos.

Diseño de circuitos secuenciales en un FPGA

Un circuito secuencial en electrónica digital es un sistema conformado por una lógica combinacional de entrada y una de salida, así como elementos de memoria que permiten almacenar el estado del circuito. De manera general, los circuitos secuenciales se pueden agrupar en dos principales categorías: síncronos, cuando dependen de una señal de reloj externa para cambiar de estado y asíncronos, cuando no están sujetos a esta condición. Este tipo de circuitos además conforman una máquina de estados finita, lo que permite que se puedan clasificar en una máquina de tipo Mealy o una máquina de tipo Moore.

En una máquina de estados Mealy, la lógica combinacional para la salida depende del estado actual en los elementos de memoria y de las variables de entrada, mientras que, en una máquina de estados tipo Moore, la lógica de salida dependerá únicamente del estado actual almacenado en la memoria, además de que en una máquina Moore, existe una única salida para cada estado, mientras que en una máquina Mealy, pueden existir distintas salidas para un estado en base a la combinación de este con las entradas.

En base a las dos categorías de máquinas de estados, es importante notar que el hecho de que la salida de una máquina Mealy dependa también de la entrada le da una característica de funcionamiento a la salida asíncrono, puesto que las variables de entrada pueden cambiar en cualquier momento y modificar la salida, mientras que para el caso de la máquina Moore, al depender únicamente del estado almacenado, la salida cambiará de manera síncrona con este.

El diseño de un circuito secuencial involucra varios aspectos que deben ser considerados por el diseñador y que se deben documentar correctamente. Entre algunos de estos aspectos se destaca el uso de diagramas de estado, diagramas de los circuitos, diagramas de tiempo y especificaciones de tiempos. Las especificaciones de tiempo tienen un papel importante en el desempeño del circuito, debido a que conforman los requerimientos necesarios que se deben cumplir para su correcto funcionamiento,

como la frecuencia máxima de operación, requerimientos de los tiempos de ajuste y de espera respecto al reloj del sistema, las duraciones mínimas de pulsos, tiempos de transición, entre otros.

Codificación de estados

Cuando se diseñan máquinas de estados síncronas, existen diversas técnicas que se pueden usar en la codificación correspondiente a la asignación de estados. La elección de uno u otro método dependerá de los requisitos del diseño, particularmente en cuanto a características de velocidad, área y consumo energético, ya que la elección de una metodología tendrá impacto en el desempeño de la máquina. En el caso de la descripción de una máquina de estados de forma funcional a través de un lenguaje de descripción de hardware, la elección de la técnica más óptima será realizada durante el proceso de síntesis. Existen tres técnicas principales para codificar los estados: binaria o secuencial, one-hot y Gray.

En la codificación binaria, la asignación de estados se realiza a través de una secuencia binaria, en donde la relación entre el número de variables de estado (q) y el número de estados (n) está dada por la ecuación: $q = log_2(n)$. De esta forma, el número de flip-flops usados es igual al número de variables de estado (q). Esta técnica minimiza el número de variables de estado (flip-flops) requeridos para representar los estados de una máquina secuencial, sin embargo, requiere de mayor lógica combinacional para decodificar cada estado y produce más de un cambio de bit en cada transición de estado.

En la codificación one-hot, solo un bit de la variable de estado es ”1” o ”hot”, mientras que los demás bits de estado son "0". De esta forma, un flip-flop es usado por cada estado en la máquina, es decir, para "n.^{es}tados se usan "n"flip-flops, de manera que la decodificación de estados se simplifica. Está técnica supone una buena opción para generar una máquina secuencial optimizada en términos de rendimiento, al requerir pocos niveles de lógica entre los flip-flops, potenciando así la frecuencia de operación que se puede alcanzar y reduciendo también el consumo energético.

En la codificación Gray, la asignación de estados se hace siguiendo la secuencia del código Gray, lo cual hace que los códigos para cada estado solo difieran de un bit, haciendo que solo un flip flop cambie en la transición consecutiva de estados. En esta codificación, la relación entre el número de variables de estado (q) y el número

de estados (n) está dada también por la ecuación: $q = log_2(n)$. Esto muestra que la decodificación de estados requiere de una gran cantidad de lógica al igual que en la codificación binaria, pero minimiza la cantidad de bits que cambian en cada transición de estado, convirtiéndola en una buena opción para diseños con enfoque en el bajo consumo de energía.

Los elementos de memoria principales en la construcción de un circuito secuencial síncrono son los flip-flops. Estos elementos difieren de los latches en la forma en la que cambian su estado, ya que mientras que en los latch se hace de manera asíncrona por el cambio de los niveles en las entradas, en los flip-flops se hace de manera síncrona en base a la detección de un flanco de subida o de bajada de una señal de reloj externa.

Existen cuatro tipos de flip-flops: flip-flop D, T, JK y SR. Cada uno de estos cuenta con una característica de funcionamiento y entradas características, por lo que para hacer uso de ellos se requiere conocer de su tabla de verdad y su tabla de excitación. La tabla de verdad de un flip-flop muestra la salida, o estado siguiente correspondiente a cada una de las combinaciones de las entradas, mientras que la tabla de excitación muestra cuales deben ser las entradas para lograr el cambio de estado deseado.

Tiempos de ajuste

El tiempo requerido durante el cual las señales de entrada de un flip-flop deben permanecer estables previo al flanco de reloj se le conoce como tiempo de ajuste (Setup time, en inglés), mientras que el tiempo de espera (Hold time, en inglés), será el tiempo durante el cual estas entradas deberán permanecer estables posterior al flanco de reloj. Adicionalmente, se debe considerar el tiempo de propagación del propio flip-flop (t_p), que representa el tiempo desde el cual se detecto el flanco de reloj y hasta que se produjo el cambio en las salidas del flip-flop.

Metaestabilidad

La metaestabilidad es un comportamiento que se presenta durante una transición de estados estables (niveles lógicos definidos) en un circuito digital. En este estado de inestabilidad, conocido como estado metaestable, las salidas se encuentran en un nivel lógico indefinido, pudiendo permanecer en este estado por un tiempo ya determinado hasta estabilizarse o incluso permanecer en el de manera indefinida.

Desfase de reloj (Clock skew)

Es un fenómeno que se presenta en un sistema síncrono, en el cual el flanco de la señal de reloj que activa a los flip-flops del sistema se ve afectada por retardos ocasionados por la transmisión de la misma a través de rutas de conexión de diferentes longitudes, por las cuales viaja. Esto se ve como un desfase de la señal de reloj para la activación de cada uno de los flip-flops, lo cual podría ocasionar que estos tomen valores erróneos del dato de entrada o que se incumplan los tiempos de ajuste y espera de los flip-flops.

Variación del reloj (Clock jitter)

Es la fluctuación aleatoria de frecuecia o fase de una señal causada por la distorsión de la señal a través de su recorrido y las variaciones inducidas por el propio transmisor de la señal, ocasionando así que en los circuitos síncronos se presenten flancos de reloj en instantes variables de tiempo. Actualmente los FPGA incluyen bloques especializados para reducir los efectos de este fenómeno, como es el caso del uso de los lazos de seguimiento de fase (PLL, por sus siglas en inglés). Un circuito PLL es un sistema con retroalimentación empleado para generar una señal de salida de amplitud fija y coincidente con la señal de entrada. Sus aplicaciones van desde el uso como filtros (lineales y no lineales) y como demoduladores.

Cruce de dominios de reloj (CDC, por sus siglas en inglés)

Un diseño de un circuito digital pertenece a un dominio de reloj cuando todos los elementos que lo conforman funcionan de manera síncrona a una fuente de reloj. Se produce un cruce de dominios de reloj cuando.

2.6.1. Técnicas para los cruces de dominios de reloj

2.6.2. Arquitectura de los FPGA serie 7 de Xilinx

Clock capable inputs

Los relojes externos de usuario deben conectarse al FPGA a través de los pares diferenciales de pines de reloj llamadas entradas con capacidad de reloj (CC, clock-capable pin, en inglés). Este tipo de pines proveen un acceso dedicado y de alta velocidad a los recursos de reloj internos tanto regionales como globales. Estos pines usan un enrutamiento dedicado y deben ser usados para las entradas de reloj

a manera de garantizar que se cumplan los requerimientos de tiempo. Las entradas/salidas de propósito general con interconexiones globales no deben ser usadas para las señales de reloj.

2.7. Protocolo AXI

El protocolo AXI AMBA soporta diseños de sistemas que demanden un alto rendimiento y funcionamiento en altas frecuencias.

El protocolo AXI:

- Es adecuado para diseños que requieran un alto ancho de banda y baja latencia.
- Provee operación en alta frecuencia sin usar enlaces complejos.
- Cumple los requerimientos de interfaz de un amplio rango de componentes.
- Es adecuado para controladores de memoria con una alta latencia de acceso inicial.
- Provee flexibilidad en la implementación de interconexión de diversas arquitecturas.
- Es retrocompatible con las interfaces existentes AHB y APB.

Las características clave del protocolo AXI son:

- Fases separadas de dirección, control y datos
- Soporte de transferencias de datos no alineados a través del uso de byte strobes
- Usa transacciones basadas en ráfagas con solo la emisión de la dirección de inicio
- Canales separados para los datos de escritura y de lectura, lo cual permite tener Accesos Directos a Memoria (DMA, por sus siglas en inglés) a un bajo costo.
- Soporte para emitir múltiples direcciones pendientes
- Soporte para completar transacciones fuera de orden
- Permite una fácil adición de etapas de registros para cumplir con los requisitos de tiempos

El protocolo también contempla el uso de extensiones opcionales para modos de operación en bajo consumo, además de que incluye también la especificación para AXI4-Lite, con el fin de proporcionar una comunicación simple de control basada en interfaces estilo registro dentro de los componentes.

2.8. Arquitectura de AXI

El protocolo AXI está basado en ráfagas (burst based) y tiene definidos los siguientes canales independientes de transacción:

- Dirección de lectura
- Datos de lectura
- Dirección de escritura
- Datos de escritura
- Respuesta de escritura

Un canal de dirección contiene información de control que describe la naturaleza de los datos a ser transferidos. Los datos son transferidos entre el maestro y el esclavo usando cualquiera de los siguientes casos:

- Un canal de escritura de datos para tranferir datos del maestro al esclavo. En una transacción de escritura, el esclavo usa la respuesta del canal de escritura para señalar la finalización de la transferencia al maestro.
- Un canal de datos de lectura para transferir datos del esclavo al maestro.

La arquitectura de AXI define canales independientes para cada una de las transacciones. Estas se pueden agrupar en procesos de escritura y lectura:

Proceso de escritura

- **Canal de dirección**
  - Awaddr (Dirección de escritura)
  - Awprot (Tipo de protección)
  - Awready (Dirección de escritura lista)
  - Awvalid (Dirección de escritura válida)
- **Canal de datos**
  - Wdata (Datos de escritura)
  - Wstrb (Indicadores de escritura)
  - Wready (Escritura lista)
  - Wvalid (Escritura válida)
- **Canal de respuesta**
  - Bresp (Respuesta de escritura)
  - Bready (Respuesta de escritura lista)

- Bvalid (Respuesta de escritura válida)

Proceso de lectura

- **Canal de dirección**
  - Arprot (Tipo de protección)
  - Araddr (Dirección de lectura)
  - Arready (Dirección de lectura lista)
  - Arvalid (Dirección de lectura válida)
- **Canal de datos**
  - Rdata (Datos de lectura)
  - Rresp (Respuesta de lectura)
  - Rready (Lectura lista)
  - Rvalid (Lectura válida)

Señales globales

- Aclk (Fuente de reloj)
- Aresetn (Fuente de reinicio)

2.9. Protocolo AMBA AXI4-Stream

Es un protocolo que permite crear una interfaz para la transferencia de datos de manera unidireccional entre componentes. El generador de datos se le conoce como maestro, mientras que al receptor de datos se le conoce como esclavo. Está desarrollado y especificado por ARM [12]. Las señales que son usadas por el protocolo se muestran en la tabla 2.1.

Nombre	Fuente	Descripción
ACLK	Fuente de reloj	Señal de reloj global. Todas las señales se muestrean en el flanco de subida de ACLK.
ARESETn	Fuente de reinicio	Señal de reinicio global. Se activa en nivel bajo.
TVALID	Maestro	Transmisión válida. Indica que el maestro está conduciendo una transferencia válida. Una transferencia toma lugar cuando tanto la señal TVALID y TREADY son válidas (Activas en alto).
TREADY	Maestro	Transmisión lista. Indica que el esclavo puede aceptar una transferencia en el ciclo actual.
TDATA	Maestro	Datos. Es la carga primaria usada para proveer los datos que pasan a través de la interfaz. Su ancho de bus se define por el número de bits correspondientes a un número entero de bytes.
TSTRB	Maestro	Es el identificador de byte que indica si el contenido del byte asociado a TDATA es procesado como un byte de datos o de posición. Su ancho de bus está definido por el número de bytes en TDATA.
TKEEP	Maestro	Es el identificador de byte que indica si el contenido del byte asociado a TDATA es procesado como parte del stream de datos. Aquellos bytes que tengan el identificador TKEEP inactivo son considerados como bytes nulos y pueden ser removidos del stream de datos. Su ancho de bus está definido por el número de bytes en TDATA.
TLAST	Maestro	Indica el límite de un paquete.
TID	Maestro	Es el identificador para cada uno de los stream de datos. Tiene un ancho de bus máximo de 8 bits.
TDEST	Maestro	Provee información de enrutamiento para el stream de datos. Tiene un ancho de bus máximo de 4 bits.
TUSER	Maestro	Es información definda por el usuario que puede ser transferida en conjunto con el stream de datos. Su ancho de bus es un múltiplo entero del ancho de la interface en bytes.

Tabla 2.1. Señales usadas en AXI4-Stream

La señal de reloj **ACLK** es única en la comunicación de los componentes. Todas las señales de entrada son muestreadas en el flanco de subida, mientras que todos los cambios en las señales de salida deben ocurrir después de este. Por otro lado, la señal de reinicio puede ser activada de manera asíncrona, pero su desactivación debe ocurrir posterior a un flanco de subida de **ACLK**. Durante el reinicio **TVALID** debe permanecer en bajo, mientras que las otras señales pueden tener cualquier valor.

Una interfaz maestra debe activar la señal **TVALID** solamente en el ciclo de reloj que es posterior a aquél en donde la señal **ARESETn** fue desactivada en alto, en ambos casos dichas señales cambian también posterior al flanco de reloj. Esto se muestra en la figura 2.23.

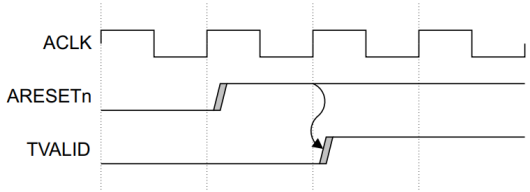


Figura 2.23. Salida del reinicio [12].

En la especificación de este protocolo, se definen los siguientes tipos de bytes:

- **Byte de dato**. Contiene la información principal válida que es transmitida entre la fuente y destino.
- **Byte de posición**. Un byte que indica la posición relativa de los bytes de datos dentro del *stream*. Este es un marcador de posición que no forma parte de la información principal.
- **Byte nulo**. Un byte que no contiene ni información principal ni de posición dentro del *stream*.

Los siguientes son algunos de los términos adoptados respecto a un *stream* de datos:

- **Transferencia**. Un único flujo de datos a través de la interfaz AXI4-stream, definido a través de un único proceso de *handshake* de **TVALID** y **TREADY**, el cual tiene un tamaño igual al ancho de bus en bytes de **TDATA**.
- **Packet**. Un grupo de bytes que son transportados de manera conjunta a través de la interfaz AXI4-stream. Un paquete es similar a una ráfaga o *burst* de AXI4. Un paquete puede estar conformado por una transferencia o múltiples transferencias.

- **Frame**. Representa el nivel más alto de agrupamiento de bytes en AXI4-stream. Un *frame* contiene un número entero de *packets*, por lo cual puede ser de un gran tamaño de bytes.
- **Stream de datos**. El transporte de datos de una fuente a un destino. Así un *stream* de datos puede ser:
  - Una serie de transferencias de bytes individuales.
  - Una serie de transferencias de *packets*.

Señalización de las transferencias

Las transmisiones en este protocolo se dan a través del uso de un proceso de *handshake*, en el cual intervienen las señales **TREADY** y **TVALID**. Este mecanismo de control de dos vías hace posible que tanto el maestro como el esclavo tengan control sobre la tasa a la cual se transmiten los datos a través de la interfaz. Para que una transferencia ocurra, tanto **TVALID** como **TREADY** se deben encontrar en activo alto. Una vez que **TVALID** se pone en activo alto por parte del maestro, deberá permanecer así hasta que el *handshake* ocurra por parte del esclavo, por lo cual, es el maestro quien debe iniciar el proceso de *handshake*. El maestro no tiene permitido esperar por la señal de **TREADY** antes de activar la señal de **TVALID**, mientras que el esclavo puede esperar por **TVALID** activando antes **TREADY**. **TREADY** se puede desactivar antes de que la señal **TVALID** se active.

En la figura 2.24 el maestro presenta los datos e información de control y pone a **TVALID** en activo alto. Una vez que el maestro ha activado **TVALID**, los datos e información de control del maestro deben permanecer sin cambios hasta que el esclavo active la señal **TREADY**, indicando que puede aceptar los datos e información de control. En este caso, la transferencia toma lugar una vez que el esclavo activa en alto a **TREADY**, indicandose a través de la flecha en la figura.

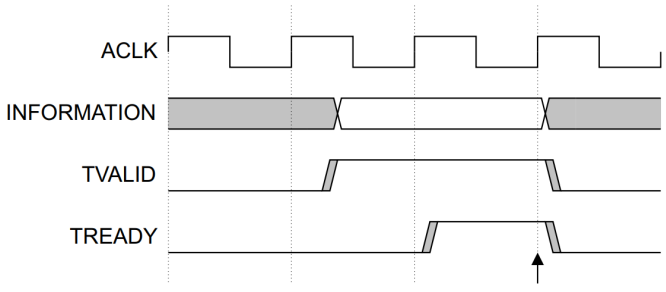


Figura 2.24. TVALID antes de TREADY [12].

En la figura 2.25 el esclavo activa en alto la señal **TREADY** antes de que los datos y la información de control sean válidos. Esto indica que el destinatario puede aceptar los datos y la información de control en un solo ciclo de **ACLK**. La transferencia toma lugar una vez que el maestro activa en alto a **TVALID**, indicado por la flecha en la figura.

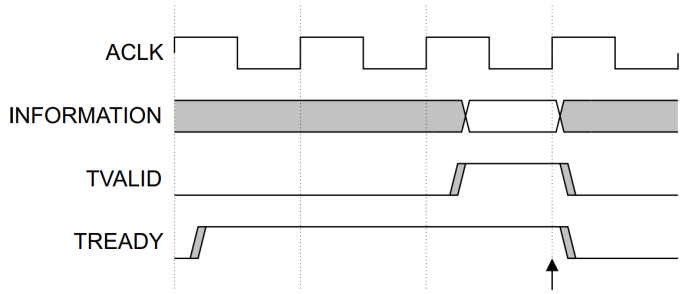


Figura 2.25. TREADY antes de TVALID [12].

En la figura 2.26 el maestro activa en alto **TVALID** al igual que el esclavo lo hace con **TREADY**, esto en el mismo ciclo de **ACLK** durante el cual la transferencia tiene lugar.

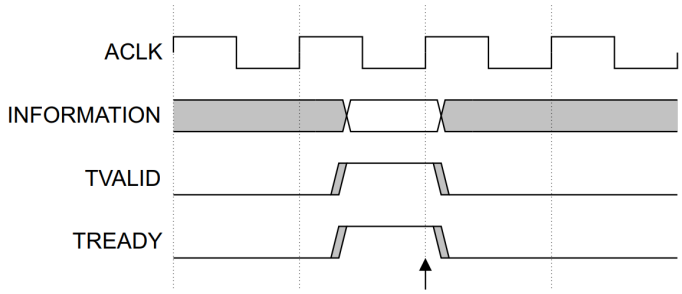


Figura 2.26. TVALID y TREADY [12].

El *stream* de datos puede tomar distintas formas en su modo de transmisión. En la figura 2.27 se muestra un ejemplo de un *stream* alineado contiguo, en donde cada columna vertical representa los bytes en una sola transferencia para un ancho de bus de 4 bytes, ordenadas por tiempo de izquierda a derecha, mientras que en la figura 2.28 se muestra un *stream* de datos desalineado. El introducir bytes de posición en el *stream* provoca un desalinemiento de los datos. En un *stream* de datos puede también incluir bytes nulos como en la figura 2.29, siendo que en este ejemplo

se transmite la misma información en ambos casos, debido a que los bytes nulos no representan información y pueden ser desechados.

D-03	D-07	D-0B	D-0F	D-13
D-02	D-06	D-0A	D-0E	D-12
D-01	D-05	D-09	D-0D	D-11
D-00	D-04	D-08	D-0C	D-10

Figura 2.27. Stream de datos alineado [12].

D-03	D-07	D-0B	D-0F	Position
D-02	D-06	D-0A	D-0E	Position
D-01	D-05	D-09	D-0D	D-11
D-00	D-04	D-08	D-0C	D-10

Figura 2.28. Stream de datos desalineado [12].

Null	Null	D-07	D-0A	Null	D-0F
D-01	Null	D-06	D-09	Null	D-0E
Null	D-03	D-05	D-08	D-0C	Null
D-00	D-02	D-04	Null	D-0B	D-0D

D-02	D-06	Null	D-0B	Null	Null
Null	D-05	Null	D-0A	D-0E	D-0F
D-01	D-04	Null	D-09	D-0D	Null
D-00	D-03	D-07	D-08	D-0C	Null

Figura 2.29. Stream de datos con bytes nulos [12].

El procolo define calificadores de bytes, los cuales sirven para indicar características específicas de cada uno de los bytes que se transmiten a través del uso de las señales **TKEEP** y **TSTRB**, en donde cada una tiene un bit asociado a cada uno de los bytes de la carga útil de datos.

TKEEP	**TSTRB**	**Tipo de dato**	**Descripción**
Alto	Alto	Byte de dato	El byte asociado contiene información válida que debe ser transmitida entre la fuente y destino.
Alto	Bajo	Byte de posición	El byte asociado indica la posición relativa de los bytes de datos en el *stream* de datos, pero no contiene valores de datos relevantes.
Bajo	Bajo	Byte nulo	El byte asociado no contiene información y puede ser removido del *stream*.
Bajo	Alto	Reservado	No debe ser usado.

Tabla 2.2. Combinaciones de **TKEEP** y **TSTRB**

En la especificación se define que la forma de señalar la transferencia de un *packet* es por medio de las señales **TID**, **TDEST** y **TLAST**. Así los usos de **TLAST** son los siguientes:

- Cuando se desactiva (en bajo), **TLAST** indica que otra transferencia puede seguir y por lo tanto es aceptable retrasar la transferencia actual para propósitos de redimensionamiento o fusión.
- Cuando se activa (en alto), **TLAST** puede ser usado por un destinatario para indicar un límite de un *packet*. Su activación también puede usarse para indicar un punto eficiente para hacer un cambio de arbitraje en un link compartido.

Las señales **TID** y **TDEST** tienen como función señalizar la fuente y destino de los datos. **TID** provee un identificador de *stream* y puede ser usado para diferenciar entre múltiples *streams* de datos que están siendo transferidos a través de la misma interfaz. Por otra parte, **TDEST** provee un enrutamiento más robusto de la información para el *stream* de datos.

Aquellas transferencias que tienen los mismos valores de **TID** y **TDEST** pertenecen al mismo *stream*. No está permitido fusionar transferencias que pertenecen a distintos *streams*. El intercalado de transferencias de diferentes *streams* está permitido y no se limita a las delimitaciones de **TLAST**.

Los componentes de interconexión pueden manipular las señales de **TID** y **TDEST**:

- El interconector puede generar señales adicionales **TID**, permitiendo así que dos *streams* idénticos puedan ser distinguidos.
- Un interconector puede generar o manipular las señales **TDEST** para proveer información de enrutamiento a un *stream*.
- Cualquier manipulación de **TID** o **TDEST** debe asegurar que dos diferentes *streams* permanezcan únicos.

La señales **TUSER** se usan cuando la interfaz requiere información adicional proporcionada por el usuario. Esta señalización puede ser usada para los bytes de datos, transferencias, *packets* o *frames*.

Señales opcionales
Las señales **TKEEP** y **TSTRB** son opcionales y no son requeridas para todos los tipos de *streams* de datos.

Las reglas establecidas para los valores por defecto son:

- Cuando **TKEEP** está ausente, su valor por defecto es considerar todos sus bits en alto.
- Cuando **TSTRB** está ausente, su valor por defecto es considerar **TSTRB** = **TKEEP**.
- Cuando **TSTRB** y **TKEEP** están ausentes, su valor por defecto es considerar a todos sus bits en alto.

La funcionalidad de este protocolo se da en permitir que exista un flujo continuo de datos a alta velocidad entre dos elementos, de manera que un bloque de datos se transmite en cada pulso de reloj. Para reducir la sobrecarga, no se emplea direccionamiento, además de que la comunicación es de punto a punto. En su implementación tiene mucha flexibilidad, al poder omitir varias señales que no sean requeridas en la implementación.

Un conjunto de señales para llevar a cabo transmisiones basicas son las siguientes:

- ACLK
- TDATA
- TREADY
- TLAST

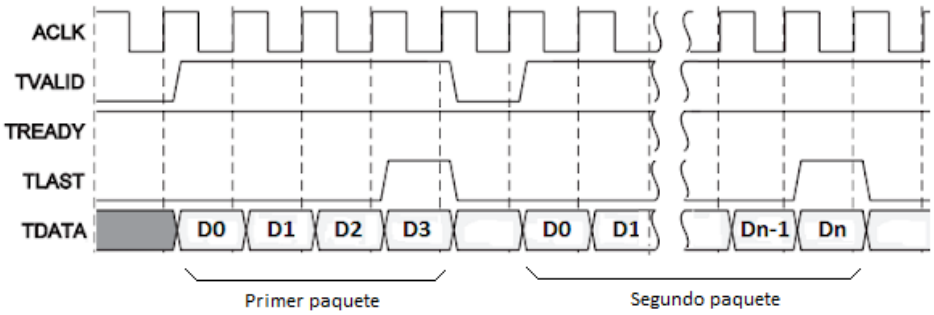


Figura 2.30. Transferencia usando AXI4-stream

Capítulo 3

Estado del arte

Se han usado cámaras en diversas misiones llevadas a cabo por satélites pequeños, particularmente para este trabajo son de interés aquellos basados en el estándar CubeSat. Cada una de estas misiones ha estado orientada a propósitos diferentes y han empleado distintos medios para adaptar el sistema de la cámara a la computadora principal del satélite. A continuación, se presenta una recopilación de algunas misiones llevadas a cabo por distintas organizaciones y universidades.

3.1. Misiones de satélites CubeSat con sistemas de cámara a bordo

3.1.1. AAU-CubeSat

Es un satélite CubeSat de tamaño 1U desarrollado por la Universidad de Aalborg en Dinamarca, el cual comenzó a ser desarrollado en 2001 para ser lanzado en septiembre del 2003 en una órbita de 820 km a 98.7°. Fue desarrollado con fines educativos, enfocado en brindar experiencia práctica a los estudiantes en un proyecto interdisciplinario de ingeniería enfocado en el ámbito espacial. El satélite estuvo en funcionamiento por alrededor de dos meses debido a que la batería comenzó a deteriorarse rápidamente, impidiendo así su correcta operación, además de que el transmisor también presentó intermitencias debido a fallas que no se pudieron determinar, causando así que solo una cantidad limitada de datos fuera adquirida.

Se empleó una cámara como carga útil en el satélite, con fines de muestra de tecnología y evaluación de la misión. La computadora principal tiene una interfaz di-

recta para el control y transmisión de datos del sensor de imagen, empleando además una interfaz DMA para llevar los datos directamente a la memoria del sistema.

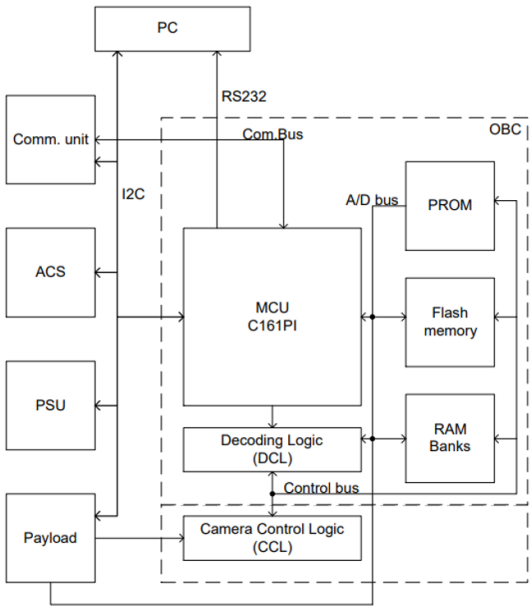


Figura 3.1. Diagrama de bloques de AAU-CubeSat [13].

Característica	Parámetro
Sensor de imagen	MCM20027 CMOS
Tamaño de pixel	6.0 × 6.0 µm
Máxima resolución	(1280 × 1024) 1.3 MP
Fotogramas por segundo en resolución completa	10 FPS
Formatos de salida	RGB de 24 o 32 bits
Controlador	Incluido en la computadora principal con interfaz DMA
Comunicación para comandos de control	I2C
Comunicación para recepción de datos	Interfaz MIPI-CPI

Tabla 3.1. Características de la carga útil del AAU-CubeSat.

3.1.2. SwissCube

Es un proyecto de CubeSat 1U desarrollado en Suiza por el Centro Espacial de la Escuela Politécnica Federal de Lausanne en conjunto con otras universidades suizas. Fue lanzado el 23 de septiembre de 2009 en una órbita a 720 km y 98.3°, reportándose que, en 2019, a 10 años de su lanzamiento todavía se encontraba operando. Fue construido con fines educativos y técnicos para obtener datos de regreso, así como probar tecnología a base de componentes COTS.

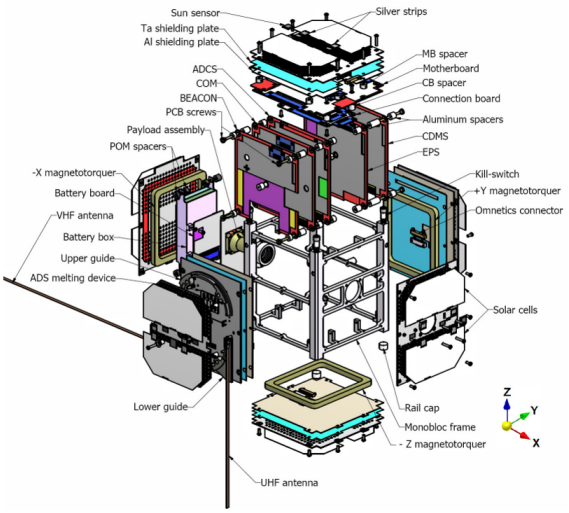


Figura 3.2. Estructura de los subsistemas del SwissCube [14].

Característica	Parámetro
Sensor de imagen	MT9V032 CMOS
Tamaño de pixel	6.0 × 6.0 µm
Máxima resolución	VGA (752 × 480)
Fotogramas por segundo en resolución completa	60 FPS
Formatos de salida	RGB de 24 o 32 bits
Controlador	MCU MSP430F1611
Comunicación para comandos de control	I2C
Comunicación para recepción de datos	I2C

Tabla 3.2. Características de la carga útil del SwissCube

3.1.3. VZLUSAT-2

Es un proyecto de CubeSat de 3U llevado a cabo por la Universidad de Bohemia Occidental en República Checa, teniendo como objetivo la verificación de tecnología a usar en futuras misiones. Está equipado con una carga útil primaria compuesta por una cámara para observación de la tierra con una resolución de 30 a 50 m GSD, mientras que las secundarias se componen de un monitor de radiación orbital, un detector de ráfagas de rayos gamma y un detector de rayos X. El satélite fue lanzado el 13 de enero de 2022.

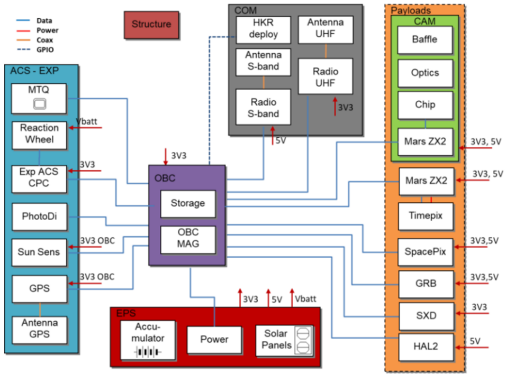


Figura 3.3. Estructura de los subsistemas del VZLUSAT-2 [15].

La carga útil primaria se compone de la cámara industrial MQ013CG-ON de Ximea, con un sistema óptico modificado, que tiene como objetivo servir como muestra tecnológica de percepción remota con el uso de componentes COTS. Para comunicarse con la computadora principal se emplea el protocolo USB.

Característica	Parámetro
Sensor de imagen	NOIP1SN1300A CMOS
Tamaño de pixel	4.8 × 4.8 µm
Máxima resolución	(1280 × 1024) 1.3 MP
Fotogramas por segundo en resolución completa	210 FPS
Formatos de salida	Mono de 8 y 12 bits
Controlador	No especificado
Comunicación para comandos de control	USB 3.1 (10 Gbps)
Comunicación para recepción de datos	USB 3.1 (10 Gbps)

Tabla 3.3. Características de la carga útil del VZLUSAT-2

3.1.4. TUMnanoSAT

Es un proyecto desarrollado por la Universidad Técnica de Moldavia con fines educativos, de investigación y desarrollo enfocado en brindar experiencia práctica hacia sus estudiantes en el campo de la exploración espacial, fomentar la investigación y presentar una oportunidad de desarrollo tecnológico para la institución. El satélite de tamaño 1U fue lanzado el 15 de julio de 2022, en una órbita a 400 km y 51.6°.

Entre los objetivos de la misión y experimentos se encuentran:

- Mostrar a los estudiantes los subsistemas que conforman un satélite, el proceso de diseño del mismo y del lanzamiento.
- Estudiar el comportamiento y la confiabilidad de los sensores basados en nano y microalambres en las condiciones del espacio.
- Realizar pruebas con los sensores del subsistema de posición del satélite (magnetómetros, giroscopios, sensores solares) con el fin de optimizar los algoritmos del control de posición.
- Establecer un subsistema de comunicación efectiva (satélite-estación terrestre) con la posibilidad de modificar el rango de la tasa de comunicación para asegurar una alta confiabilidad.
- Prueba de la operación de los componentes electrónicos COTS en condiciones de radiación.
- Captura de imágenes de la tierra con la cámara a bordo del satélite.

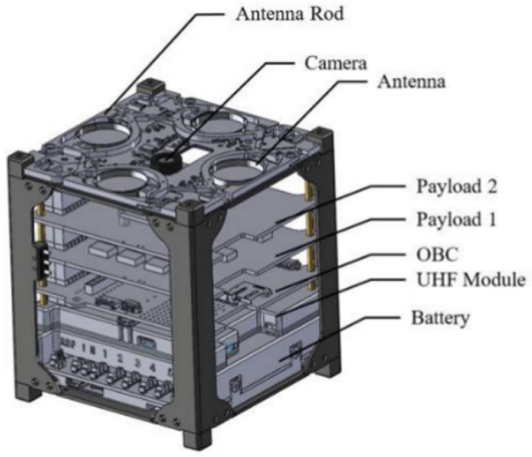


Figura 3.4. Estructura del satélite TUMnanoSAT [16].

Para el sistema de carga útil, conformado por una cámara COTS, se empleó el módulo µCAM-II de la compañía 4D Systems, el cual posee un diseño que le permite ser acoplado a un sistema embebido, empleando una interfaz serial para recibir comandos de control y transmisión de datos. Entre los componentes se encuentran principalmente el sensor de imagen POA030R de Pixel Plus en conjunto con el SoC SPWVID328 de SpeedPixel para el control del sensor. No se detalla información referente a un método de protección contra la radiación en módulo tanto a nivel software como a nivel hardware.

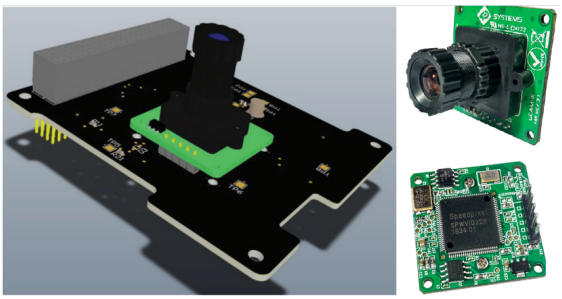


Figura 3.5. Módulo de imagen uCAM-II [17].

Característica	Parámetro
Sensor de imagen	POA030R CMOS
Tamaño de pixel	5.55 × 5.55 µm
Máxima resolución	VGA (640 × 480) 0.3 MP
Fotogramas por segundo en resolución completa	30 FPS
Formatos de salida	CCIR656 YCbCr422 RGB565 9 bits en mono 9 bits en RGB Bayer
Controlador	SoC SPWVID328
Comunicación para comandos de control	UART
Comunicación para recepción de datos	UART (Hasta 3.68 Mbps)

Tabla 3.4. Principales características del módulo µCAM-II

3.1.5. BeaverCube

Es un satélite CubeSat desarrollado por el Instituto Tecnológico de Massachusetts con fines educativos y de investigación. Emplea un sistema de cámaras comerciales para recolectar datos del clima de la tierra, enfocándose en obtener imágenes para oceanografía, así como la prueba de un sistema de propulsión por electrospray que será usado para desplazar al satélite. Tiene un tamaño de 3U y fue lanzado el 15 de julio de 2022, en una órbita de 400 km a 51.6°.

Entre las tareas principales de la misión se encuentran:

- Capturar exitosamente imágenes infrarrojas y visibles de la tierra.
- Tomar datos de la temperatura de la parte superior de las nubes y la superficie oceánica.
- Transmitir y recibir datos y comandos desde la terrestre del MIT.
- Realizar la calibración de las cámaras por calibración vicarious y cruzada.
- Obtener los datos de geolocalización de las imágenes tomadas.
- Mostrar la operación de un sistema de propulsión basado en ionización por electrospray.

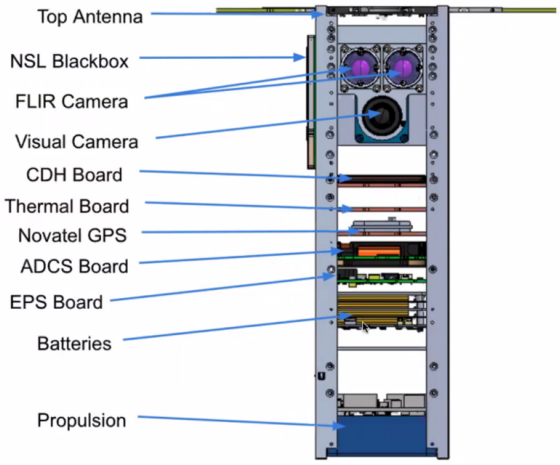


Figura 3.6. Estructura del BeaverCube [18].

La carga útil del satélite está compuesta por tres cámaras comerciales, dos de espectro infrarrojo y una de espectro visible. Estas se encuentran montadas en una sola estructura y tienen interfaz USB para comunicarse con la computadora principal. Para las cámaras del espectro infrarrojo se usó el modelo Boson 320 de la

compañía FLIR, mientras que para la cámara de espectro visible se usó el modelo mvBlueFOX-IGC 200w de Matrix Vision. No se implementó algún sistema de protección contra radiación en el hardware de las cámaras.

Característica	Parámetro
Sensor de imagen	MT9V034 CMOS
Tamaño de pixel	6.0 × 6.0 µm
Máxima resolución	(752 × 480) 0.4 MP
Fotogramas por segundo en resolución completa	60 FPS
Formatos de salida	RGB Bayer de 8 y 10 bits Formato mono de 8 y 10 bits
Controlador	No especificado
Comunicación para comandos de control	USB 2.0 (Hasta 480 Mbps)
Comunicación para recepción de datos	USB 2.0 (Hasta 480 Mbps)

Tabla 3.5. Principales características de la cámara de espectro visible mvBlueFOX-IGC 200w [19].

3.1.6. Ex-Alta 2

Es un proyecto canadiense de un satélite enfocado en promover e involucrar a los estudiantes en el desarrollo y la investigación espacial. Se trata de un CubeSat de tamaño 3U que busca contribuir al desarrollo del proyecto canadiense CubeSat, diseñado, construido y probado principalmente por estudiantes de la Universidad de Alberta en Edmonton, Canadá. Para este satélite aún no se ha definido una fecha de lanzamiento.

Entre los objetivos de la misión se encuentran:

- Evaluar y monitorear los incendios forestales a través de un sistema de imagen a bordo desarrollado en la Universidad de Alberta.
- Continuar con el trabajo desarrollado en Ex-Alta 1, al proponerse como una plataforma de desarrollo para futuros proyectos de CubeSat.
- Promover el desarrollo de tecnología espacial entre los estudiantes de la universidad a través del proyecto llevado a cabo.

Para el sistema de la carga útil se ha desarrollado un sistema de imagen denominado “Iris” para ser integrado en el satélite, con el cual se planea reunir imágenes

multiespectrales para fines de investigación y monitoreo de incendios forestales. El sistema ha sido construido empleando componentes COTS, desde el sistema óptico, estructural y electrónico. Este sistema a su vez emplea un controlador denominado "Electra" conformado por un pSoC de Altera para realizar las funciones de control y almacenamiento de datos de los sensores de imagen G11478-512WB y CMV4000.

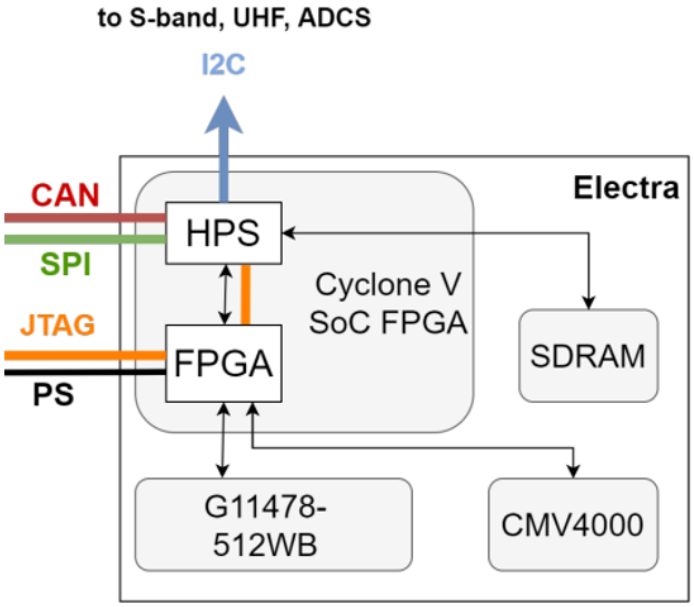


Figura 3.7. Diagrama de bloques del controlador Electra [20].

Las interfaces que incluye el controlador son:

- Bus CAN para la recepción de comandos desde la computadora principal.
- SPI para la transmisión de imágenes hacia la computadora principal.
- I2C para un modo de recuperación implementado.
- JTAG para realizar la configuración y pruebas.
- Bus para el esquema de configuración del FPGA.

La computadora Electra se apagará cuando no se encuentre en uso. Antes de una comenzar con la captura de imágenes, la computadora se encenderá y reprogramará por medio de la computadora principal, para posteriormente recibir los comandos por medio del bus CAN. En el proceso de captura de imagen, los datos del sensor son almacenados de manera temporal en la memoria SDRAM de la computadora

Electra, por medio de un comando a través de CAN se puede solicitar la transferencia de estos datos hacia la computadora principal, por medio del protocolo SPI.

Se incluye protección por hardware en Electra contra fenómenos SEL (Single Event Latch-Up, por sus siglas en inglés), el cual consiste en agregar sensores de corriente para monitorear que no se exceda un límite de esta, teniendo otro circuito para cortar la energía por aproximadamente 10 segundos en caso de que esto suceda para eliminar la condición de latch-up. Así mismo, cuenta con temporizadores watch-dog para monitorear el funcionamiento de la computadora.

Característica	Parámetro
Sensor de imagen	CMOS CMV4000
Tamaño de pixel	5.5 × 5.5 µm
Máxima resolución	(2048 × 2048) 4 MP
Fotogramas por segundo en resolución completa	180 FPS
Formatos de salida	12 y 10 bits por píxel
Controlador	SoC FPGA
Comunicación para comandos de control	CAN
Comunicación para recepción de datos	SPI

Tabla 3.6. Principales características de la carga útil

3.2. Módulos comerciales de sistemas de percepción remota de imágenes para satélites CubeSat

Debido a la popularidad adquirida por las plataformas de satélites pequeños, algunas compañías dedicadas al sector espacial han incluido en su catálogo de productos módulos de carga útil con cámara para CubeSats. Estos productos presentan un diseño en el cual se basan para dar garantía de que son aptos para ser usados en entornos espaciales. Al ser módulos comerciales, su diseño de arquitectura interna no está disponible para el usuario, además de que se encuentra protegido por un marco

legal de propiedad intelectual, así como algunos otros documentos del dispositivo, los cuales muchas veces requieren de un proceso de aprobación para acceder a ellos. Es por eso que se realizó una recopilación en esta sección de aquellos que proporcionan algunos datos relevantes sobre las características que ofrecen.

3.2.1. Módulo de carga útil para CubeSat IM200 de AAC Clyde Space

Es una cámara basada en la plataforma del seguidor de estrellas ST200, con una interfaz USB 2.0 dedicada, con capacidad para capturar hasta 5 fotogramas por segundo en alta resolución. El buffer interno le permite almacenar hasta 25 imágenes en alta resolución, con capacidad para ser comprimidas en formato JPEG. La potencia del procesador permite realizar el procesamiento de las imágenes adquiridas en tiempo real, lo cual permite que el módulo sea usado como una cámara rastreadora de objetivos, una cámara de acoplamiento o como cámara de inspección.

El módulo IM200 es de propósito general, su factor de forma reducido y masa le otorgan flexibilidad para ser usado en un amplio número de aplicaciones, sin embargo, está orientado a poder ser usado en plataformas satelitales pequeñas, teniendo antecedentes de misiones en órbitas LEO al ser sucesor del rastreador de estrellas ST200, el cual se ha empleado en múltiples misiones desde 2015.

Figura 3.8. Módulo de imagen IM200 [21].

Característica	Parámetro
Sensor y resolución máxima	CMOS (2048 × 1944) 4 MP
Profundidad de pixel	No especificado
Fotogramas por segundo en resolución completa	5 FPS
Almacenamiento interno	Hasta 25 imágenes en alta resolución
Comunicación para control y datos	UART (Comandos) USB 2.0 (Datos)
Consumo energético	0.7 W
Voltaje de alimentación	3.65 V
Tolerancia a la radiación	9 krad
Dimensiones	29 × 29 × 70.7 mm
Masa	59 g

Tabla 3.7. Principales características del IM200

3.2.2. Módulo de carga útil para CubeSat C3D de XCAM

El XCAM C3D es un sistema de imagen que ha sido puesto a prueba en las misiones UKube-1 y AlSat Nano de la Agencia Espacial del Reino Unido. La configuración del módulo ofrece flexibilidad para operar con hasta 3 cámaras por unidad de computadora. Se incluyen las opciones de imagen en campo lejano y cercano, tanto para color como para la escala de grises. Adicionalmente el fabricante ofrece la opción de proporcionar ajustes para adaptarse a los requerimientos del usuario.

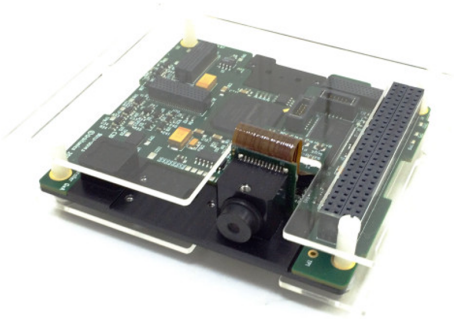


Figura 3.9. Módulo de imagen C3D [22].

Característica	Parámetro
Sensor y resolución máxima	EV76C560 CMOS (1280 × 1024) 1.3 MP
Profundidad de pixel	8 bits
Fotogramas por segundo en resolución completa	No especificado
Almacenamiento interno	16 MB SDRAM 8 MB Flash
Comunicación para control y datos	I2C (Comandos) SPI (Datos)
Consumo energético	0.845 W
Voltaje de alimentación	5 V 3.3 V
Tolerancia a la radiación	No especificado
Dimensiones	95 × 91 × 27
Masa	85 g

Tabla 3.8. Principales características del C3D

3.2.3. Módulo de carga útil para CubeSat ECAM-IR1 de Malin Space Science Systems

Es un sistema de cámara infrarroja compacta de longitud de onda larga (LWIR, por sus siglas en inglés) que puede usarse como una herramienta de diagnóstico para pruebas de vuelos, detección de objetos con emisiones térmicas o bien hacer observaciones en ambientes oscuros.

El módulo emplea un sensor infrarrojo con salidas de 12 bits por píxel, transmitidos hacia su unidad de procesamiento a 100 Mbits/s por medio de una interfaz serial. La imagen o video es procesada y comprimida en tiempo real, para posteriormente pasarse a una memoria de almacenamiento no volátil. El video se puede comprimir siguiendo el proceso para JPEG o Huffman, siendo una tasa ajustable de video de hasta 50 Hz con opción de ventana reprogramable.

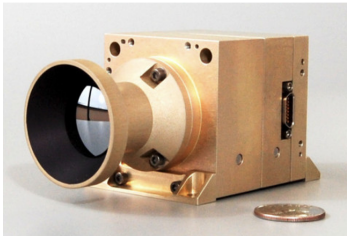


Figura 3.10. Módulo de imagen ECAM-IR1 [23].

Característica	Parámetro
Sensor y resolución máxima	384 × 288
Profundidad de pixel	12 bits
Fotogramas por segundo en resolución completa	50 FPS
Almacenamiento interno	No especificado
Comunicación para control y datos	Spacewire
Consumo energético	8.75 W
Voltaje de alimentación	5 V
Tolerancia a la radiación	Para uso en GEO
Dimensiones	78 × 58 × 63
Masa	330 g

Tabla 3.9. Principales características de ECAM-IR1

3.2.4. Módulo de carga útil para CubeSat SpectraCAM de Redwire Space

Es un módulo de imagen orientado a misiones en LEO, pudiendo ser usado como cámara de acoplamiento, navegación, inspección o percepción remota. Es una plataforma de bajo costo a base de componentes COTS, pudiendo ser empleada para misiones de carácter civil, comercial y de defensa. Su baja sensibilidad a la luz y amplio rango dinámico hacen que el sistema sea adecuado para imágenes con alto contraste, encontrado en aplicaciones orbitales con alta presencia de luz solar.

Al ser de bajo costo y fácil de integrar, se convierte en una opción para las plataformas de satélites pequeños, además, el sensor de imagen CMOS le permite tener un mejor rendimiento en ausencia de luz. Incluye filtros electrónicos que hacen posible la captura de imágenes con un bajo nivel de ruido, permitiendo que las imágenes sean claras y nítidas para mostrar detalles con alta precisión. El sistema está construido y diseñado también con un enfoque en el bajo consumo de energía, masa reducida y alta confiabilidad en su funcionamiento.

Figura 3.11. Módulo de imagen SpectraCAM [24].

Característica	Parámetro
Sensor de imagen	CMOS Modelo no especificado
Tamaño de pixel	No especificado
Máxima resolución	(2592 × 1944) 5 MP
Fotogramas por segundo en resolución completa	10 FPS
Formatos de salida	Imagen con profundidad de 12 bits
Capacidad de almacenamiento	No especificado
Controlador	No especificado
Comunicación para comandos de control	Ethernet
Comunicación para recepción de datos	Ethernet

Tabla 3.10. Principales características de SpectraCAM

3.2.5. Módulo de carga útil para CubeSat HyperScape100 de Simera Sense

Es un módulo de imagen enfocado en aplicaciones de percepción remota para pequeños satélites. Está basado en una unidad de control, un sensor de imagen CMOS y un sistema óptico con un filtro variable para el rango espectral visible e infrarrojo, diseñado además con los requerimientos necesarios de un sistema de aplicación espacial, manteniendo su funcionamiento a través de un amplio intervalo de temperaturas. Su factor de forma le permite poder integrarse en estructuras CubeSat de tamaño 3U o mayores.

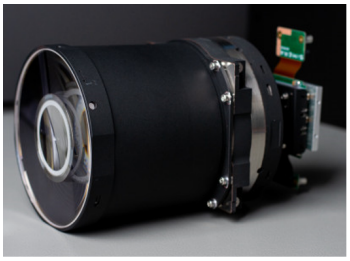


Figura 3.12. Módulo de imagen HyperScape100 [25].

Característica	Parámetro
Resolución máxima	4K (4096 píxeles horizontales)
Profundidad de pixel	12 bits
Fotogramas por segundo en resolución completa	No especificado
Almacenamiento interno	128 GB
Comunicación para datos y control	Spacewire, I2C, SPI
Consumo energético	7 W
Tolerancia a la radiación	Hasta 15 krad
Tamaño	98 × 98 × 176 mm
Masa	1.1 kg

Tabla 3.11. Principales características de HyperScape100

3.2.6. Chamaleon Imager de DragonFly Aerospace

Es un sistema de imagen desarrollado para ser empleado como un sistema de carga útil en plataformas CubeSat. Está compuesto por un sistema óptico, electrónico y estructural. El sistema electrónico se encarga de controlar la adquisición y almacenamiento de imágenes, así como de llevar a cabo la comunicación con la computadora de a bordo.

Figura 3.13. Módulo de imagen Chamaleon imager MS [26].

Característica	Parámetro
Resolución máxima	No especificado
Profundidad de pixel	8 o 16 bits
Fotogramas por segundo en resolución completa	No especificado
Almacenamiento interno	128 GB
Comunicación para datos y control	CAN, LVDS, SPI, I2C, RS422
Consumo energético	10 W
Tolerancia a la radiación	30 krad
Tamaño	100 × 100 × 215 mm
Masa	1.6 kg

Tabla 3.12. Principales características del Chamaleon Imager MS

3.3. Tablas comparativas de las características de los módulos de imagen

A partir de la recopilación de algunas de las misiones llevadas a cabo por CubeSats se realiza una comparativa de los componentes claves que conforman su carga útil empleada. No todas las misiones con sistemas de percepción remota proporcionan detalles acerca de las características de sus sistemas, o no de manera detallada, por lo cual en algunas secciones no se tienen datos para comparar.

CubeSat	AAU-CubeSat	SwissCube	VZLUSAT-2	TUMnanoSAT	BeaverCube	ExAlta-2
Sensor de imagen	CMOS MCM20027	CMOS MT9V032	CMOS NOIP1SN1300A	CMOS POA030R	CMOS MT9V034	CMOS CMV4000
Resolución	(1280 × 1024) 1.3 MP	VGA (752 × 480)	(1280 × 1024) 1.3 MP	VGA (640 × 480) 0.3 MP	(752 × 480) 0.4 MP	(2048×2048) 4 MP
FPS	10 FPS	60 FPS	210 FPS	30 FPS	60 FPS	180 FPS
Formatos de dato	RGB de 24 o 32 bits	RGB de 24 o 32 bits	Mono de 8 y 12 bits	CCIR656 YCbCr422 RGB565 9 bits en mono 9 bits en RGB Bayer	RGB Bayer de 8 y 10 bits Formato mono de 8 y 10 bits	12 y 10 bits por píxel
Buffer y/o almacenamiento adicional	Memoria RAM de OBC	Memoria RAM de OBC	Cámara industrial	Memoria SDRAM	Cámara industrial	Memoria SDRAM DDR
Controlador	MCU con DMA	MCU	Cámara industrial	SoC	Cámara industrial	FPGA + SoC
Interfaz de control y de datos	I2C/CPI	I2C	USB 3.1 (10 Gbps)	UART (Hasta 3.68 Mbps)	USB 2.0 (Hasta 480 Mbps)	CAN/SPI
Tolerancia a fallas	No	No	No	No	No	Si

Tabla 3.13. Principales características de los módulos de carga útil empleados en CubeSats

Módulos de carga útil comerciales disponibles

Módulo	IM200	C3D	ECAM-IR1	SpectraCAM	HyperScape100	Chamaleon Imager
Sensor de imagen	CMOS	CMOS EV76C560	CMOS	CMOS	--	--
Resolución	(2048 × 1944) 4 MP	(1280 × 1024) 1.3 MP	384 × 288	(2592 × 1944) 5 MP	4K (4096 H)	--
FPS	5 FPS	--	50 FPS	10 FPS	--	--
Formatos de dato	--	8 bits	12 bits	12 bits	12 bits	8 o 16 bits
Buffer y/o almacenamiento adicional	Capacidad para 25 imágenes	16 MB SDRAM 8 MB Flash	--	--	128 GB	128 GB
Interfaz de control y de datos	UART USB 2.0	I2C SPI	Spacewire	Ethernet	Spacewire, I2C, SPI	LVDS, SPI, I2C, CAN, RS422
Voltaje de alimentación	3.65 V	5 V 3.3 V	5 V	--	--	5 V
Consumo energético	0.7 W	0.845 W	8.75 W	--	7 W	10 W
Tolerancia a la radiación	9 krad	--	Uso en órbita GEO	--	15 krad	30 krad

Tabla 3.14. Principales características de los módulos de carga útil comerciales

3.4. Conclusiones sobre las misiones llevadas a cabo y los módulos comerciales

A partir del análisis del estado del arte, se encontró que se han realizado varias misiones satelitales bajo el estándar CubeSat llevadas a cabo principalmente por universidades en las cuales se han empleado cámaras en el sistema de carga útil. Entre las características de estos sistemas se encontró que los componentes empleados son principalmente componentes COTS, bajo los cuales no se incluye o detalla la implementación de técnicas de tolerancia a fallas para garantizar la fiabilidad de su uso en la misión, siendo varias de ellas misiones de demostración de tecnología, por lo cual no incluyen sistemas de protección más avanzados. En todas las misiones analizadas se observa que se emplea un sensor de imagen de tecnología CMOS, esto debido a que esta tecnología es más barata que su contraparte de CCD, además de presentar un consumo energético menor y es menos susceptible al ruido en la imagen.

Capítulo 4

Diseño conceptual

El objetivo del desarrollo de este trabajo es el diseño de una computadora de a bordo que será usada en un sistema de percepción remota de imágenes. En base al análisis del estado del arte se hace una lista de prerequisitos que serán evaluados con las posibilidades tecnológicas con las que se cuenta, a manera de generar posteriormente el concepto de diseño a desarrollar.

4.1. Metodología

Para llevar a cabo el desarrollo de un producto en ingeniería es necesario contar con una metodología de diseño, a fin de garantizar un producto funcional y que cumpla con los requerimientos propuestos. Una metodología para lograr esto es la propuesta por Ulrich.

Metodología de diseño de Ulrich

Es una metodología de diseño que se seleccionó por emplearse en el desarrollo de los proyectos en el laboratorio de la Facultad de Ingeniería LIESE. El método de Karl T. Ulrich abarca todas las etapas que intervienen en el desarrollo de un producto, desde la planeación hasta la venta y producción.

Las principales etapas que conforman esta metodología son:

1. Planeación
2. Desarrollo del concepto
3. Diseño a nivel sistema
4. Diseño de detalle

5. Pruebas y refinamiento
6. Producción

Planeación

En esta etapa se comienza con la aprobación del proyecto, revisión del estado del arte y definición de los requerimientos. Los resultados que se obtienen de esta fase son: la misión del proyecto, objetivo del producto, las metas de desempeño y limitaciones.

Diseño conceptual

En esta etapa se definen el conjunto de especificaciones del producto, se lleva a cabo un análisis de los productos de la competencia y una justificación económica del proyecto. Adicionalmente se selecciona un modelo de propuesta para tener un prototipo.

Diseño a nivel sistema

En esta etapa se define la arquitectura del producto y se detalla la estructura del mismo al describir los subsistemas y componentes que lo conforman. Así, de esta etapa se obtiene un diagrama a bloques del producto, especificaciones funcionales de cada uno de los subsistemas del producto y un diagrama de flujo preliminar del proceso para el desarrollo final.

Diseño de detalle

Esta etapa permite generar una especificación completa de los elementos y sistemas que conforman al producto, las partes a ser adquiridas con los proveedores, un plan de proceso y las herramientas para la fabricación.

Pruebas y refinamiento

Etapa que comprende la construcción de versiones de preproducción del producto con el fin de tener prototipos que serán evaluados para determinar su desempeño y verificar que cumpla con los requerimientos.

Producción

Se inicia el proceso de contruir las unidades del producto de manera sistematizada, en base al trabajo llevado a cabo en las etapas anteriores.

4.1.1. Planeación

El desarrollo de este proyecto se basó en los requerimientos iniciales para el desarrollo del concepto del sistema de percepción remota del proyecto CONDOR y adicionalmente se realizó una investigación del estado del arte de sistemas de percepción remota de imágenes, empleados en misiones de satélites bajo el estándar CubeSat y disponibles comercialemente para este fin, enfocándose en la arquitectura de la computadora de a bordo que se incluyó como parte de este sistema, a fin de generar una propuesta de diseño.

Siguiendo la metodología de diseño de Ulrich, se presenta el desarrollo de cada una de las etapas. Para la etapa de planeación del proyecto se ha de definir la misión y visión del mismo. La misión establece el propósito y objetivo inmediato del proyecto, mientras que la visión declara el objetivo ideal, sin dar detalles de la tecnología empleada o de funciones específicas.

Misión
La misión del proyecto consiste en diseñar una computadora que será empleada en un sistema de percepción remota de imágenes para un satélite del estándar CubeSat, la cual será implementada en un FPGA.

Visión
Diseñar la computadora de a bordo para un sistema de percepción remota de imágenes que cuente con las capacidades tecnológicas para competir frente a las opciones disponibles en el mercado y pueda ser integrada en satélites del estándar CubeSat.

Objetivo
El objetivo del proyecto es generar el diseño que permita la construcción de la computadora para un sistema de percepción remota de imágenes que pueda ser usada en un satélite del estándar CubeSat.

A partir de esto se puede resumir la misión del proyecto en la siguiente tabla:

Tabla 4.1. Declaración de la misión

Declaración de la misión: computadora de a bordo para un sistema de percepción remota de imágenes	
Descripción del producto	Computadora de a bordo en FPGA que permite la captura de imágenes desde un sensor CMOS
Propuesta de valor	Contrucción en un FPA GA con posibilidad de implementar técnicas de tolerancia a fallas a nivel hardware.
Mercados objetivo	Institutos y universidades
Suposiciones y restricciones	▪ Construcción empleando componentes COTS ▪ Emplear un sensor CMOS y un FPGA

Requisitos generales

- El sistema debe permitir capturar imágenes en alta resolución del espectro visible
- La computadora se debe implementar en un FPGA
- Diseño versatil y escalable
- Estándar CubeSat
- Con posibilidad de ser tolerante a fallas

Metas de desempeño

- Capturar imágenes del espectro visible con una resolución de hasta 5 MP
- Diseño que permita sear tolerante a fallas
- Bajo costo al emplear componentes COTS

Limitaciones

- Se debe emplear un FPGA Xilinx/AMD de la serie 7
- Usar un sensor de imagen CMOS

4.2. Metodología de diseño

4.2.1. Desarrollo del concepto

Siguiendo con la metodología, para la etapa del desarrollo del concepto, se han de definir un conjunto de especificaciones para el producto, basándose en los requisitos generales y en los resultados obtenidos de la investigación del estado del arte, a fin de definir la forma, función y características del producto.

Identificación de las necesidades del cliente

La identificación de las necesidades del cliente tiene como objetivo servir como una base para establecer las especificaciones objetivo del producto y poder generar propuestas iniciales de concepto para el diseño. Para el caso de este proyecto, esto se resume en la siguiente tabla:

Necesidades generales

- Almacenar imágenes adquiridas por un sensor CMOS
- Capacidad para recibir los datos del sensor de alta velocidad
- Usar componentes COTS para su construcción
- Implementar en un FPGA
- Compatibilidad con las interfaces de sensores CMOS
- Interfaces de comunicación para datos y control desde la computadora principal
- Posibilidad de incluir técnicas de tolerancia a fallas para el ambiente espacial
- Posibilidad de implementar la computadora en el estándar CubeSat

Especificaciones objetivo

A partir de las necesidades identificadas, se genera una lista de especificaciones de nivel técnico con el fin de poder cuantificar esas necesidades en el desarrollo del proyecto. El diseño del producto deberá contemplar estas métricas con el fin de generarlo acorde a las necesidades identificadas del cliente.

Especificaciones del producto

- Compatibilidad con sensores CMOS bajo la interfaz CPI
- Resolución del sensor de imagen de 0.3 MP hasta 5 MP
- Velocidad de toma de imagen de 10 FPS hasta 60 FPS

- Tamaño de palabra del sensor de 8 hasta 32 bits por pixel
- Interfaces de comunicación para datos y control desde la computadora principal
- Formato en base al estándar CubeSat

Generación de conceptos

Este proceso tiene como finalidad generar prototipos del producto en base a las especificaciones, definir en forma simplificada la estructura del mismo y sus principales componentes para su operación, teniendo como finalidad cumplir con el objetivo principal del proyecto. Para esto, teniendo en cuenta el objetivo inicial de construir una computadora de a bordo para un sistema de percepción remota de imágenes (SPRI), se muestra un diagrama básico de este sistema (Figura 4.1) y su interfaz con los principales componentes, que son la computadora principal, el sensor de imagen y el banco de memorias.

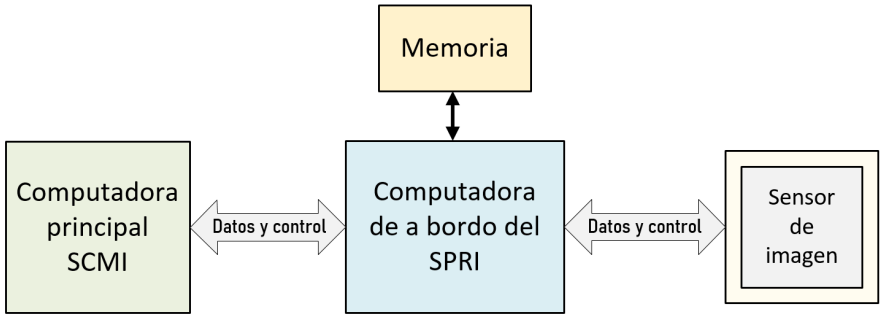


Figura 4.1. Diagrama de bloques conceptual del sistema.

El diagrama muestra a la computadora principal que se encargará de comunicarse con la computadora del SPRI para configurar e intercambiar datos. La computadora del SPRI puede operar de manera autónoma y simultánea con la computadora principal, teniendo así la capacidad de ejecutar una rutina de control independiente para la captura de imágenes. Debido a que se contempla el uso de un sensor de imagen de alta resolución, el almacenar una sola imagen requiere de una cantidad considerable de almacenamiento que esta dentro del orden de los MB, por lo que no sería posible almacenarla por ejemplo en la memoria RAM de un microcontrolador, la cual es del orden de KB, incluso los bloques de memoria de un FPGA apenas alcanzan unos cuantos MB. Es por esto que se contempla el uso de un banco de memoria que cuente con la capacidad necesaria para almacenar una cantidad óptima de imágenes. Adicionalmente es importante considerar que este banco de memoria debe poseer una

velocidad de transferencia suficiente para responder a la velocidad a la que los datos son generados por el sensor de imagen.

En la figura 4.2 se presenta un diagrama a bloques de la estructura de la computadora del SPRI. Este diagrama se crea siguiendo la metodología "top-down" y representa el nivel de abstracción más alto del sistema. El diseño base sigue el modelo clásico de una computadora, compuesto de una unidad de procesamiento, unidades de memoria y puertos de entrada y salida.

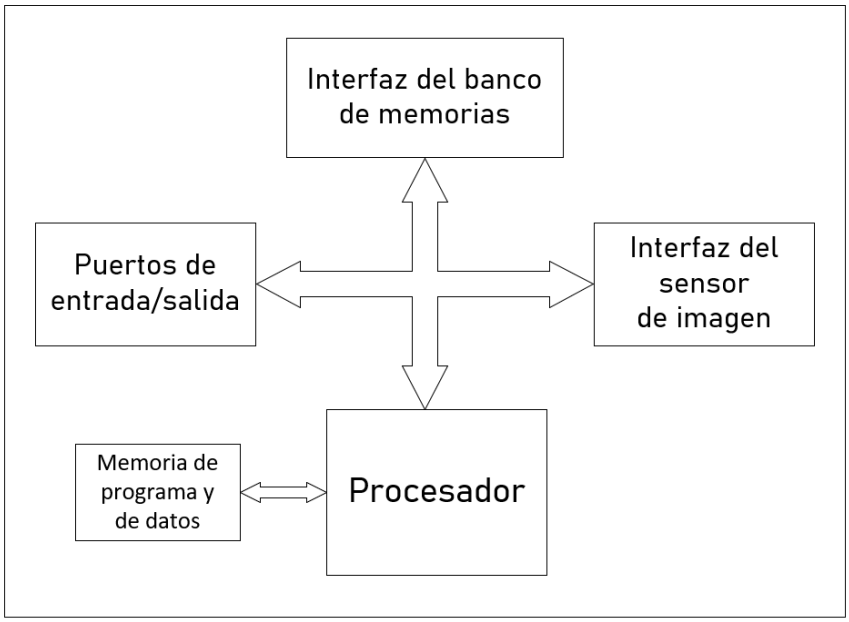


Figura 4.2. Diagrama de bloques conceptual con la definición de las interfaces.

4.2.2. Diseño a nivel sistema

A partir de la selección del concepto final del producto es posible comenzar con la generación de su arquitectura, la cual contempla la definición de sus componentes principales y susbsistemas. Como referencia de arquitectura para la computadora, se selecciona un modelo base de una computadora con arquitectura de memoria Harvard, por ser un modelo que se encuentra presente en la mayoría de computadoras actuales, así como por las ventaja de rendimiento que ofrece sobre otros tipos de arquitecturas como la Von Neumman.

En la figura 4.3 se muestra la arquitectura base propuesta, en la cual se muestran los componentes principales: procesador, memorias de programa y de datos, periféricos de entrada/salida, la interfaz sensor-memoria, módulo de configuración y depuración, así como los principales buses del sistema.

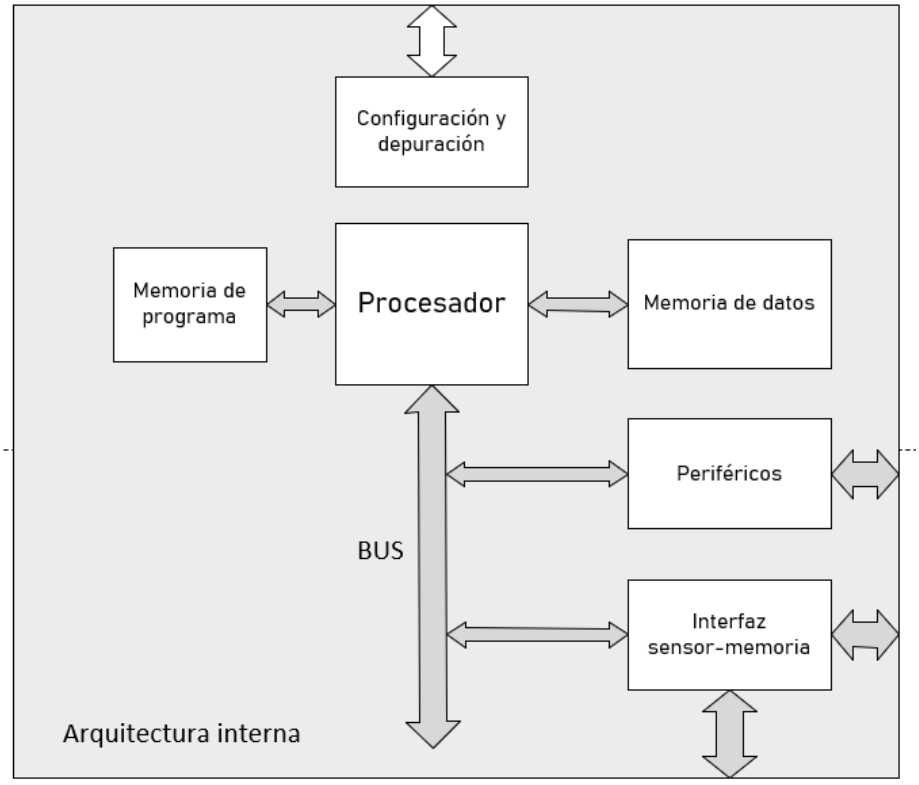


Figura 4.3. Arquitectura interna base del sistema.

De manera general, los componentes realizan las siguientes funciones:

- El módulo de **configuración y depuración** es el componente de hardware interno que permite la configuración del FPGA, así como también abarca aquellos componentes que forman parte del diseño y tienen como propósito depurar y configurar los componentes del mismo.

- El **procesador** será el componente encargado de gestionar el funcionamiento general de la computadora, lleva a cabo la ejecución de las tareas definidas por el código almacenado en la memoria de programa. Tiene la capacidad para

realizar cálculos, realizar tranferencia de datos y responder a las interrupciones de los componentes del sistema.

- La **memoria de programa** contiene el código del programa principal, el cual a su vez define las tareas a realizarse, las configuraciones de los periféricos, así como las rutinas de interrupción. En algunos casos también se considera el uso de un bootloader que forma parte de esta memoria, el cual permite la actualización del código de programa de una manera más simple.
- La **memoria de datos** es la encargada de funcionar como un medio de almacenamiento temporal para datos, variables y resultados del programa en ejecución.
- Los **periféricos** son los componentes que otorgan funcionalidades específicas al sistema como la comunicación con una interfaz externa a través de un puerto serial o paralelo, llevar valores de cuenta, generar señales PWM, procesar señales analógicas o implementar protocolos de comunicación específicos para comunicarse con un dispositivo.
- La interfaz **sensor-memoria** proporciona la via de transferencia de datos desde el sensor de imagen a la memoria de almacenamiento, que funcionará como buffer de datos. Esta interfaz tiene implementado el mecanismo para comunicarse tanto con el sensor de imagen como con la memoria.
- El **bus** representa el conjunto de conexiones que comunican al procesador con los periféricos del sistema, permitiendo así su configuración y una vía de comunicación bidireccional para transmisión y recepción de datos.

Con los componentes y subsistemas identificados, es posible generar el conjunto de especificaciones funcionales para cada uno de estos elementos.

Procesador

El procesador es una de las unidades principales del sistema, ya que se encargará de ejecutar las tareas principales y de la administración de resursos y periféricos que conforman a la computadora. Para esto se plantean algunas de las especificaciones con las que deberá contar este componente, obtenidas en base al estado del arte:

- Procesador RISC de 32 bits
- Capacidad de múltiples fuentes de interrupción

- Sintetizable en un FPGA (Núcleo soft) de Xilinx
- Interfaz de bus estándar para conectar periféricos
- Amplia disponibilidad de herramientas de desarrollo
- Ofrece herramientas de depuración
- Experiencia de su uso en proyectos de investigación

Memoria de programa y de datos

La memoria de programa contiene las tareas que deben ejecutarse, mientras que la memoria de datos funciona como almacenamiento temporal de los datos obtenidos de la ejecución de estas tareas. No hay especificaciones técnicas precisas con las que deben cumplir estos elementos, por lo que se presentan algunas características de funcionalidad que serían deseables en estos componentes:

- Tamaño de palabra de 32 bits
- Tamaño recomendable de 32K
- Facilidad para conectarse a los buses de instrucciones y datos del procesador.
- Frecuencia de operación acorde con el rango de frecuencia de operación del procesador

Periféricos

Estos componentes definen varias de las funcionalidades que estarán presentes en la computadora, permitiendo además la comunicación con otros sistemas. Para la selección de los periféricos que deberán incluirse se recurre a usar los principales presentes en los sistemas de percepción remota revisados en el estado del arte. Principalmente su selección se toma con base en los usados en microcontroladores. La lista de periféricos a incluir en el diseño son:

- GPIO
- UART
- SPI
- I2C
- Temporizador/Generador PWM
- Watchdog

En la selección de los periféricos no se consideró usar aquellos como CAN, USB, Ethernet o SpaceWire como se encontró en el estado del arte, debido a que estos módulos para el caso de su uso en FPGA requieren de licencias comerciales para su

uso. El uso de un DMA tampoco se tomo en cuenta por la poca documentación encontrada de los módulos para usarse en FPGA al momento de definir la arquitectura. Por lo tanto, los periféricos que se consideran para el diseño de la computadora son los siguientes:

- Puerto paralelo de entrada/salida (GPIO, por sus siglas en inglés)
- Puerto serial UART
- Puerto serial SPI
- Puerto serial I2C
- Temporizador/Generador de PWM
- Watchdog

En base a los periféricos seleccionados, se definen algunas de las principales características de funcionalidad y de especificaciones objetivo para el diseño. Debido a que en base a la aplicación específica se pueden necesitar más o menos unidades de alguno de los periféricos, solo se muestra el diseño con una muestra de cada periférico, sin embargo, se tiene en cuenta que existe la posibilidad de incluir o incluso quitar unidades de estos.

Puerto GPIO

- Tamaño estándar de 8 bits.
- Configuración individual de bits como entrada/salida.
- Generación de interrupciones por detección de flanco/nivel.
- Compatible con la interfaz de bus del procesador seleccionado.

Puerto serial UART

- Modo de transmisión y recepción de datos.
- Trama configurable para tamaño de dato, bit de paridad y bit de stop.
- Tasa de baudios estándar configurable, por ejemplo para 9600, 115200, 460800 o 921600 bauds.
- Generación de interrupciones para transmisión, recepción de datos y estado.
- Compatible con la interfaz de bus del procesador seleccionado.

Puerto serial SPI

- Modo estándar con líneas SCK, MOSI, MISO y de selección de esclavo.
- Configuración en modo esclavo y maestro.

- Tamaño de dato de 8 bits.
- Tasa de transmisión de datos configurable.
- Generación de interrupciones para transmisión, recepción de datos y estado.
- Compatible con la interfaz de bus del procesador seleccionado.

Puerto serial I2C

- Configuración en modo esclavo y maestro.
- Modo de direccionamiento en 7 y 10 bits.
- Tasa de transmisión de datos en modo estándar (100 kbps), rápido (400 kbps) y rápido plus (1 Mbps).
- Generación de interrupciones para transmisión, recepción de datos y estado.
- Compatible con la interfaz de bus del procesador seleccionado.

Temporizador

- Tamaño de 32/16 bits.
- Cuenta hacia arriba/abajo.
- Modo captura.
- Generación de interrupciones para eventos de cuenta.
- Compatible con la interfaz de bus del procesador seleccionado.

Generador de PWM

- Configuración de modo normal e invertido.
- Configuración de la frecuencia y el ciclo de trabajo.
- Compatible con la interfaz de bus del procesador seleccionado.

Watchdog

- Configuración del tiempo de disparo de evento.
- Generación de interrupciones para eventos de cuenta
- Compatible con la interfaz de bus del procesador seleccionado.

Interfaz sensor-memoria

Este es uno de los bloques más importantes del sistema, ya que se encarga de la tarea principal que es la adquisición de los datos del sensor de imagen y del almacenamiento de estos en la memoria. Este módulo engloba una estructura mediante la cual se puedan guardar los datos del sensor de imagen en la memoria, a la vez que

pueden también ser leídos por el procesador para ser procesados o enviados a otro periférico. El módulo permite realizar la captura de imágenes al tiempo que las guarda en el banco de memorias. Se definen las principales especificaciones funcionales para el módulo y en su diseño se detallarán algunas más específicas.

- Este bloque engloba la interfaz para el sensor de imagen y para el banco de memoria.
- Permite la configuración y control de las acciones por medio de la interfaz del bus para periféricos.
- Permite la lectura y/o escritura de datos en el banco de memoria.

Bus de los periféricos

El bus en la computadora permitirá que el procesador tenga comunicación con los periféricos y seguirá un estándar de comunicación igual para todos estos, el cual permita la comunicación bidireccional entre el procesador y el periférico que está siendo accedido. Debido a que este tiene que ser un bus que siga un estándar de comunicación disponible para los buses, estará condicionado por el tipo de bus usado por el procesador.

Módulo de configuración y depuración

Como se considera el uso de un FPGA, estos dispositivos contienen ya interfaces por las cuales se puede configurar el hardware interno, no obstante sería deseable también agregar componentes adicionales que proporcionen la capacidad de poder depurar componentes del sistema, como es el caso del procesador y que también permita configurar la memoria de programa para tener la capacidad de actualizar el código de esta.

Con base en las especificaciones mencionadas, el diagrama de la arquitectura interna ahora se presenta en la figura 4.4

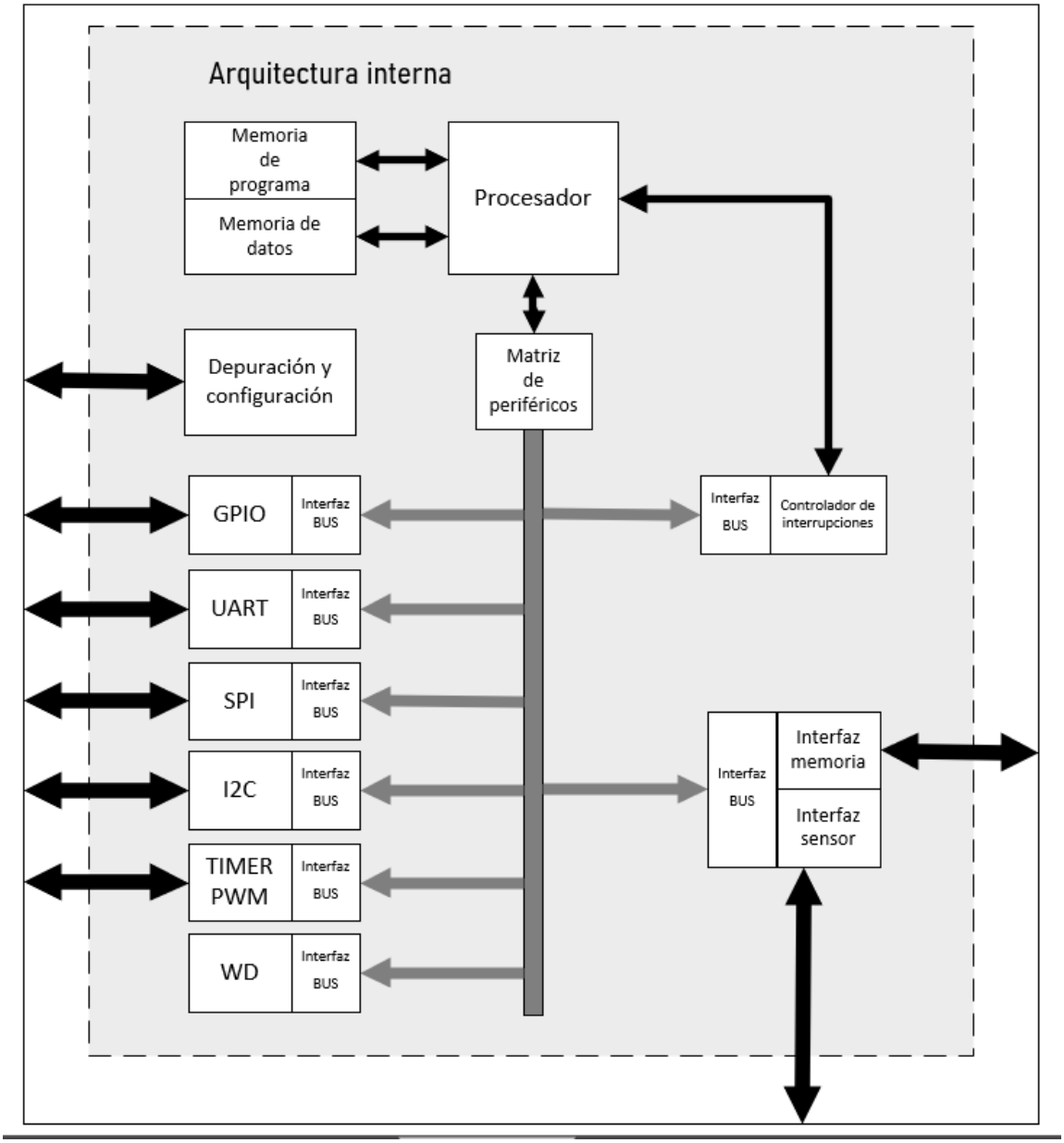


Figura 4.4. Arquitectura interna base del sistema 2.

Capítulo 5

Diseño de detalle

El diseño de detalle contempla la selección de los componentes a emplearse en el proyecto con base en las especificaciones definidas previamente a través del diseño conceptual y diseño a nivel sistema del producto. En esta etapa se incluye la especificación de los componentes seleccionados detallando sus principales características y diagramas de estos.

5.0.1. Selección de los componentes

Para la selección de los componentes se toma en cuenta que cumplan con las especificaciones definidas en el diseño a nivel sistema, pudiendo en algunos casos llegar a ofrecer características adicionales que podrían considerarse incluir en el diseño o bien descartarlas para su implementación. Existen además casos donde la selección de un componente condiciona la selección de otros, por lo que se debe tener en consideración también la jerarquía de los componentes que conforman el sistema y priorizar aquellos que se consideran importantes para este.

Selección del procesador

El procesador es un componente fundamental en la arquitectura de una computadora, encargandose de la ejecución de las tareas y del control de los periféricos. Para la selección del procesador se tomó en consideración el estado del arte llevado a cabo de los principales procesadores soft disponibles y tomando también en cuenta las especificaciones definidas como parte del diseño a nivel sistema. Considerando esto, el procesador a emplearse es el procesador soft MicroBlaze ofrecido por Xilinx, el cual de acuerdo con la comparación realizada en la tabla 5.1 cumple con las características especificadas e incluso ofrece algunas adicionales.

Tabla 5.1. Características del procesador seleccionado

Especificación	Característica del componente
Procesador RISC de 32 bits	MicroBlaze está basado en la arquitectura RISC y permite ser configurado para 32 y 64 bits.
Capacidad de múltiples fuentes de interrupción	Soporta una fuente de interrupción externa, sin embargo, se puede ampliar mediante el uso de un controlador de interrupciones.
Sintetizable en un FPGA de Xilinx	MicroBlaze es un núcleo de procesamiento soft que Xilinx ofrece como parte del catálogo de componentes de propiedad intelectual (IP) para sus FPGA.
Interfaz de bus estándar para conectar periféricos	Incluye la interfaz AMBA AXI4 para la conexión con periféricos, la cual es establecida como una interfaz usada en la industria.
Disponibilidad de herramientas de desarrollo	Xilinx ofrece el entorno de desarrollo integrado Vivado para soluciones de hardware y el entorno de desarrollo Vitis para soluciones de software.
Herramientas de depuración	Existen diversas herramientas para simular y realizar depuración en hardware y software.
Experiencia de uso en proyectos con fines de investigación	Existen artículos en los cuales se evidencia el uso de MicroBlaze como núcleo de procesamiento.

Selección de la memoria de programa y de datos

Para la selección de estos componentes se toma en cuenta que al tratarse de un diseño en FPGA se deben de emplear recursos de almacenamiento que estén disponibles en este. Como solución a la memoria de programa y de datos Xilinx ofrece la integración de un bloque IP que permite la generación de controladores de memoria que emplean los recursos de memoria disponibles en el FPGA.

Selección de la interfaz para el bus de los periféricos

La selección de la interfaz a emplearse para comunicarse con los periféricos estará condicionada a la empleada por el procesador, misma que a su vez también condicionará a que los periféricos usen esta interfaz. El procesador seleccionado en este caso incluye la interfaz AXI4 en modo Full, con la cual ofrece la posibilidad de conectarse a una gran variedad de sistemas y módulos IP del catálogo de Xilinx. Esta interfaz además está estandarizada para su uso en la interconexión de sistemas y está especificada por ARM [12].

5.0.2. Selección de los periféricos

Los periféricos conforman los componentes que agregan funcionalidades específicas al sistema, ya sea para comunicación o control de procesos. La selección de la interfaz a emplear condiciona a que los mismos también la empleen a manera de que puedan ser conectados al bus del procesador. Xilinx ofrece algunos bloques IP de uso libre para ser usados en el diseño de plataformas de hardware en su entorno de desarrollo Vivado, por lo cual en base a su catálogo de disponibilidad se seleccionaron estos componentes a emplear.

La lista que conforman los periféricos a seleccionar son los siguientes:

- GPIO
- UART
- SPI
- I2C
- Temporizador
- Watchdog

Selección del módulo de interfaz paralela GPIO

El GPIO representa una interfaz paralela con la cual la computadora puede tener comunicación con el mundo exterior. Para el módulo GPIO requerido se seleccionó el módulo de propiedad intelectual AXI GPIO v2.0, cuyas características principales se comparan con las especificaciones en la tabla 5.2.

Tabla 5.2. Características del módulo GPIO seleccionado

Especificación	Característica del componente
Tamaño de interfaz paralela de 8 bits	El tamaño de la interfaz paralela es configurable para un ancho de bus de hasta 32 bits.
Configuración individual de bits como entrada/salida	Cada bit se puede configurar individualmente como entrada o salida a través del valor de bit de su registro asociado.
Generación de interrupciones por flanco/nivel	El módulo tiene la capacidad de generar interrupciones al detectar un cambio de nivel.
Compatibilidad con la interfaz del bus del procesador	El módulo emplea la interfaz AXI4-Lite.

Selección del módulo de comunicación serial UART

Es una interfaz serial que permite la comunicación de la computadora, estando presente en varios dispositivos como los microcontroladores, módulos de comunicación y computadoras personales. Como parte de las opciones disponibles de Xilinx, se seleccionó el módulo AXI UART Lite v2.0, con base en las especificaciones mostradas en la tabla 5.3.

Tabla 5.3. Características del módulo UART seleccionado

Especificación	Característica del componente
Modo de transmisión y recepción de datos	El módulo tiene la capacidad de funcionar en modo transmisor y receptor.
Trama configurable para tamaño de dato y bit de paridad	El tamaño de dato, bit de paridad y bit de stop es configurable, sin embargo, esta configuración es por hardware y no se puede realizar por software.

Tasa de baudios configurable	La tasa de baudios es configurable pero solamente en hardware.
Generación de interrupciones	El módulo tiene la capacidad de generar interrupciones al transmitir y recibir datos.
Compatible con la interfaz de bus del procesador seleccionado	Emplea la interfaz AXI4-Lite.

Selección del módulo de comunicación serial SPI

SPI es una interfaz serial presente en los sistemas embebidos para transmitir datos. El módulo seleccionado que incluye esta interfaz de acuerdo con el catálogo de IPs de Xilinx es el módulo AXI Quad SPI v3.2, del cual se comparan algunas de sus características en la tabla 5.4.

Tabla 5.4. Características del módulo SPI seleccionado

Especificación	Característica del componente
Modo estándar con líneas SCK, MOSI, MISO y de selección de esclavo	El módulo para funcionar en los modos: estándar SPI, dual SPI y Quad SPI. El modo estándar incluye las líneas SCK, MOSI, MISO e incluye una interfaz configurable para la selección de múltiples esclavos.
Configuración en modo esclavo y maestro	El módulo puede funcionar en modo maestro y esclavo.
Tamaño de dato de 8 bits	El tamaño de datos es configurable desde 8 hasta 16 bits.
Tasa de transmisión de datos configurable	La velocidad de transferencia de datos es configurable por medio de hardware.

Generación de interrupciones para transmisión, recepción de datos y estado	El módulo puede generar interrupciones.
Compatible con la interfaz de bus del procesador seleccionado	Tiene la opción de poder seleccionar emplear la interfaz AXI4 en modo Full o modo Lite al momento de configurarse.

Selección del módulo de comunicación serial I2C

I2C es una interfaz de bus serial que puede ser usada para la creación de redes en sistemas embebidos. De acuerdo con las opciones disponibles de Xilinx, la IP de I2C está identificada como AXI IIC Bus Interface v2.1 y la comparación con las especificaciones se resume en la tabla 5.5.

Tabla 5.5. Características del módulo I2C seleccionado

Especificación	Característica del componente
Configuración en modo esclavo y maestro	El módulo puede operar en modo maestro y esclavo.
Modo de direccionamiento en 7 y 10 bits	Ofrece el modo de direccionamiento para 7 y 10 bits.
Tasa de transmisión de datos en modo *standard*, *fast* y *fast plus*	El módulo incluye los tres modos de transmisión.
Generación de interrupciones para transmisión, recepción de datos y estado	La generación de interrupciones ocurre al enviar o recibir datos, así como al detectarse errores en la operación.

Compatible con la interfaz de bus del procesador seleccionado	Emplea la interfaz AXI4-Lite para configurar los registros.

Selección del módulo temporizador

Es un módulo que permite llevar a cabo eventos de cuenta en la computadora. Para esta opción se seleccionó la IP de Xilinx AXI Timer v2.0, cuyas características se comparan con las especificaciones definidas en la tabla 5.6.

Tabla 5.6. Características del módulo temporizador seleccionado

Especificación	Característica del componente
Tamaño de 32/16 bits	El módulo incluye dos contadores de 32 bits cada uno.
Cuenta hacia arriba/abajo	Se puede seleccionar el modo de cuenta hacia arriba o hacia abajo.
Modo captura	Ofrece el modo de captura.
Generación de interrupciones para eventos de cuenta	Tiene la capacidad de generar interrupciones por eventos de tiempo.
Compatible con la interfaz de bus del procesador seleccionado	Emplea la interfaz AXI4-Lite para su configuración.

Selección del módulo watchdog

El watchdog es un módulo temporizador que tiene la capacidad de reiniciar a la computadora en caso de que esta se haya bloqueado. La IP AXI Timebase Watchdog Timer v3.0 de Xilinx ofrece esta funcionalidad y cumple con las especificaciones de acuerdo a la tabla 5.7.

Tabla 5.7. Características del módulo watchdog seleccionado

Especificación	Característica del componente
Configuración del tiempo de disparo de evento	El tiempo de disparo de eventos se puede configurar a través de los registros.
Generación de interrupciones para eventos de cuenta	Permite generar interrupciones para eventos de cuenta y reiniciar el sistema si es el caso.
Compatible con la interfaz de bus del procesador seleccionado	Emplea la interfaz AXI4-Lite para la configuración de los registros.

5.0.3. Selección del hardware

Para la selección del FPGA se deben tomar en cuenta características específicas de acuerdo con las necesidades del hardware que se requieren en el diseño. Para este caso, tenemos que considerar factores como la cantidad de recursos lógicos empleados por el hardware a configurar, la cantidad de pines necesarios para el diseño a implementar, si se requieren bloques especiales de hardware, especificaciones eléctricas de su funcionamiento y de consumo energético, así como el precio del dispositivo. Debido al concepto sobre el cual estaba basado este proyecto, el hardware que se iba a emplear ya estaba definido, por lo cual se trabajará en base a este hardware.

Con base en lo anterior, el hardware con el que se cuenta será el siguiente:

- Tarjeta de desarrollo Nexys Video con un FPGA Artix de la serie 7 de Xilinx [27].
- Módulo de memoria SDRAM DDR3 integrado en la tarjeta de desarrollo [11].
- Sensor de imagen MT9P031 de 5 MP [8].

Adicionalmente con el propósito de aumentar la capacidad de desarrollo y depuración se emplean los siguientes componentes:

- Tarjeta de desarrollo Basys 3 [28].
- Sensor de imagen OV7670 de 0.3 MP [29].

5.1. Especificaciones de los componentes

Las especificaciones de los componentes forman parte de la etapa de diseño de detalle. En esta sección se describen algunas de las principales características técnicas y operativas de los componentes a ser empleados en el proyecto, así como diagramas de bloques donde se muestra su estructura interna y sus principales interfaces de entrada/salida.

Especificaciones del procesador

MicroBlaze es un núcleo de procesamiento tipo soft diseñado por Xilinx, cuya arquitectura del set de instrucciones está basada en la arquitectura RISC-DLX. Se desarrolló entre el año 2000 y 2001, presentandose oficialmente en el año 2002. A partir de entonces se han lanzado periódicamente versiones de MicroBlaze, agregando nuevas características y corrigiendo errores, siendo la versión 11.0 la última lanzada hasta la fecha.

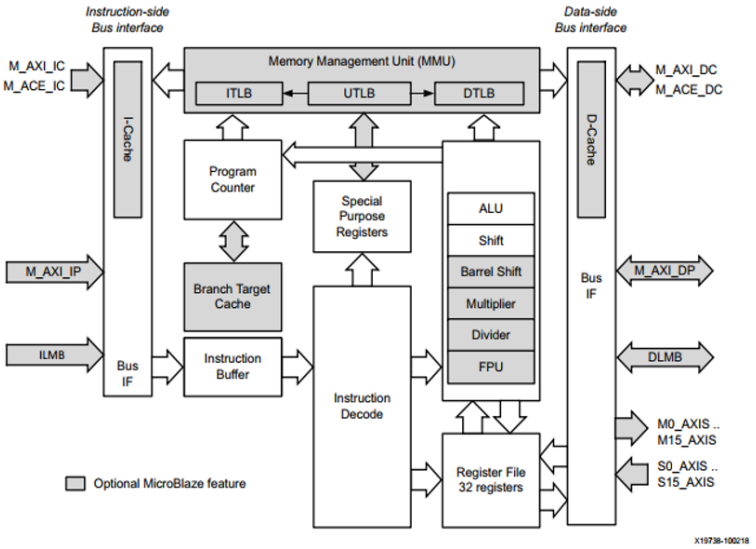


Figura 5.1. Arquitectura interna de Microblaze [30].

Características principales

- 32/64 bits.
- 32 registros de propósito general

- Interfaces de buses AXI4 y FSL.
- Profundidad de pipeline de 3/5/8 etapas.
- Frecuencia de operación de hasta 700 Mhz en algunas FPGA.
- Métodos de tolerancia a fallas por medio de bit de paridad, memoria ECC y scrubbing.

Características adicionales

- Unidad artimética de punto flotante (FPU).
- Divisor por hardware.
- Hardware de desplazamiento combinacional.
- Caché de datos e instrucciones.
- Soporte de interrupciones en modo normal (Una sola fuente) y modo avanzado (múltiples fuentes de interrupción).
- Soporte de excepciones por hardware.
- Protección de sobrecarga de stack

Ventajas

- Es un procesador soft con características avanzadas para usarse como el núcleo de procesamiento de un microcontrolador o ejecutar un sistema operativo.
- Compatible con la familia de FPGA's de Xilinx.
- Posee documentación acerca de la estructura del procesador y sus características.
- Foros de soporte.
- Módulos IP para integrarse en conjunto con el procesador.
- Posee un entorno de desarrollo integrado (IDE Vitis) para programar y depurar el procesador.

5.1.1. Especificaciones de los periféricos

De acuerdo con los periféricos seleccionados, se presentan algunas de las especificaciones técnicas y funcionales con las que cuentan estos componentes.

GPIO

El módulo IP AXI GPIO de Xilinx proporciona una interfaz paralela de entrada/salida de propósito general de hasta dos canales e implementa una interfaz AXI para acceder a los registros de configuración, además de incluir la generación de interrupciones programables.

Entre sus principales características se encuentran:

- Interfaz AXI4-Lite para acceso a registros.
- Configuración para usar uno o dos canales para la interfaz.
- El tamaño de la interfaz se puede configurar para 1 bit y hasta 32 bits.
- Cada bit puede configurarse de manera individual como entrada o salida.
- Cada canal de la interfaz se puede configurar por separado.
- Permite la generación de interrupciones al detectar una transición en cualquiera de sus entradas.

En la figura 5.2 se muestra el diagrama de bloques de su estructura interna.

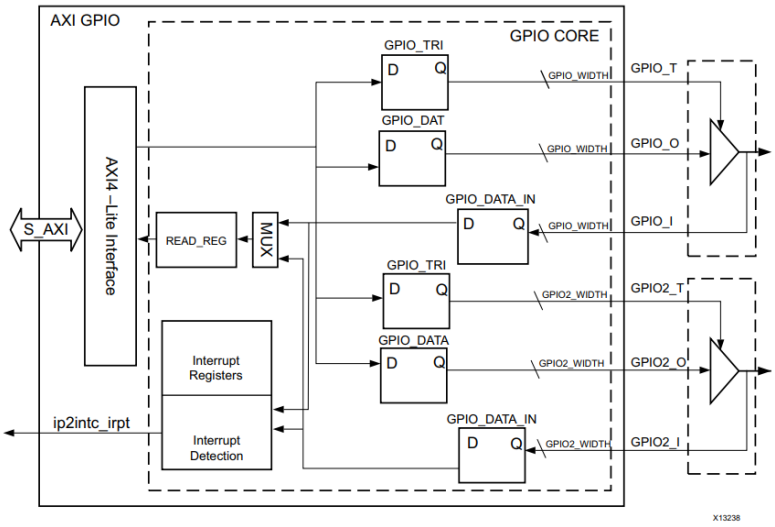


Figura 5.2. Diagrama de bloques interno del AXI GPIO [31].

Las tablas 5.8 y 5.9 muestran las interfaces y señales presentes en el módulo.

Tabla 5.8. Interfaces de AXI GPIO

Interfaz	**Descripción**
S_AXI	Interfaz AXI4-Lite del módulo.
GPIO	Interfaz para el canal GPIOn del módulo.

Tabla 5.9. Señales de AXI GPIO

Señal	Tipo	Descripción
s_axi_aclk	Entrada	Señal de reloj para la interfaz AXI.
s_axi_aresetn	Entrada	Señal de reset de AXI activa en bajo.

En la tabla 5.10 se muestran los registros de este módulo.

Tabla 5.10. Registros de AXI GPIO

Desplazamiento del espacio de direcciones[2]	Nombre del registro	Tipo de acceso	Valor por defecto	Descripción
0x0000	GPIO_DATA	R/W	0x0	Registro de datos del canal 1.
0x0004	GPIO_TRI	R/W	0x0	Registro de control triestado del canal 1.
0x0008	GPIO2_DATA	R/W	0x0	Registro de datos del canal 2.
0x000C	GPIO2_TRI	R/W	0x0	Registro de control triestado del canal 2.
0x011C	GIER	R/W	0x0	Registro de habilitación global de las interrupciones.
0x0128	IP_IER	R/W	0x0	Registro de habilitación de las interrupciones de la IP.
0x0120	IP_ISR	R/TOW[1]	0x0	Registro de estado de las interrupciones.

Notas:
1 Toggle-On-Write (TOW) invierte el estado de un bit cuando se escribe un valor de 1 al bit correspondiente.
2 Relativo a la dirección asignada C_BASEADDR.

5.1.2. UART: AXI UART Lite v2.0

El módulo UART Lite de Xilinx proporciona un medio de mínimos recursos para la transferencia de datos seriales a través del protocolo UART y controlado por medio de una interfaz AXI.

Entre sus principales características se encuentran:

- Interfaz AXI4-Lite para el acceso a los registros de configuración.
- Full duplex
- FIFOs de transmisión y recepción con capacidad de 16 caracteres.
- Número configurable de bits de datos (5-8) por caracter.
- Bit de paridad configurable (Par, impar o ninguno).
- Tasa de baudios configurable.

En la figura 5.3 se muestra el diagrama de bloques interno del módulo. El bloque de interfaz AXI permite el acceso hacia los registros del módulo, destinados a usarse en la configuración, control, transferencia y recepción de datos. El bloque de control se encarga de gestionar la transmisión y recepción de los datos, la generación de la tasa de baudios y el control de las interrupciones.

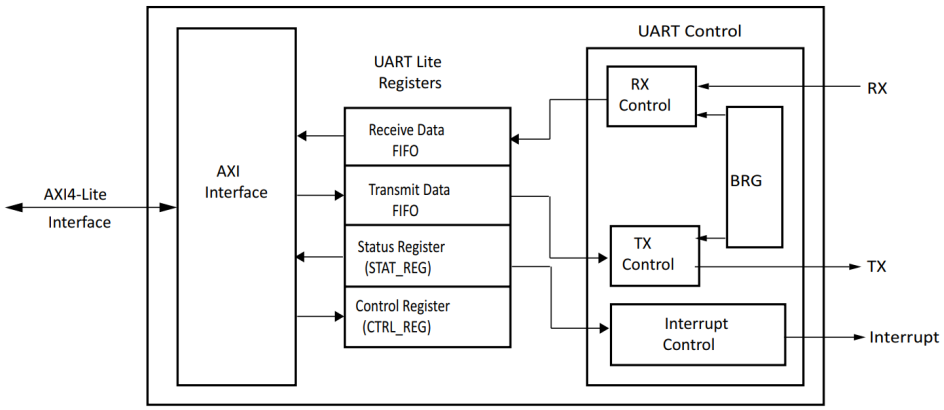


Figura 5.3. Diagrama de bloques interno de AXI UART Lite [32].

La tabla 5.11 muestra los registros que contiene este bloque.

Tabla 5.11. Registros de AXI DMA para el modo DMA simple

Desplazamiento del espacio de direcciones[1]	**Nombre**	**Descripción**
0x00	Rx FIFO	Registro para la FIFO de recepción de datos.
0x04	Tx FIFO	Registro para la FIFO de transmisión de datos.
0x08	STAT_REG	Registro de estado.
0x0C	CTRL_REG	Registro de control.

Notas:
1 Relativo a la dirección asignada C_BASEADDR.

En su intefaz de configuración este módulo solo permite configurar la tasa de baudios, el tamaño del dato y el tipo de paridad.

5.1.3. SPI: AXI Quad SPI v3.2

Este módulo permite la comunicación serial a través del protocolo SPI y con acceso a su interfaz a través de AXI4.

Entre sus principales características se encuentran:

- Acceso al módulo a través de la interfaz AXI4-Full o AXI4-Lite.
- Interfaz AXI4 configurable para operaciones en modo de ráfaga a través de los registros de transmisión y recepción de datos.
- Modo de operación XIP.
- Modo de operación configurable para SPI en modo estándar, dual o quad.
- Fase y polaridad de reloj programables.
- Tamaño de transacción configurable para 8, 16 y 32 bits.

En la figura 5.4 se muestra el diagrama de bloques de la estructura interna del bloque. La interfaz AXI4 se comunica con la interfaz IPIC, la cual se encarga de procesar las operaciones con el módulo de registros, la recepción y transmisión de datos a través de las FIFO y de configurar el controlador de interrupciones. Los contadores de ocupación, transmisión y recepción (OCC, Tx y Rx) se comunican con los registros. Existe también un bloque CDC que permite realizar el cruce de los dominios de

reloj entre AXI y SPI, así como también el módulo que contiene la lógica para la transmisión y recepción de SPI.

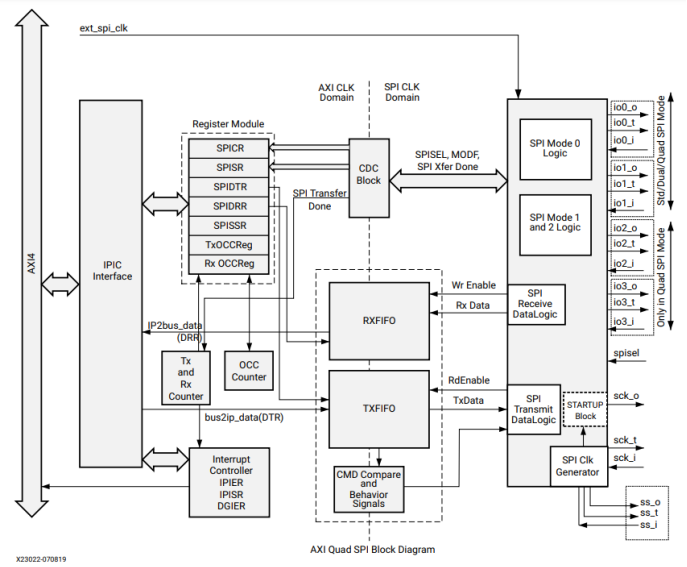


Figura 5.4. Diagrama de bloques interno de AXI Quad SPI [33].

En la tabla 5.12 se muestran los registros que contiene este módulo.

Tabla 5.12. Registros de AXI Quad SPI

Desplazamiento del espacio de direcciones[1]	**Nombre del registro**	**Tipo de acceso**	**Valor por defecto**	**Descripción**
0x40	SRR	W	N/A	Registro de reinicio por software.
0x60	SPICR	R/W	0x0180	Registro de control.
0x64	SPISR	R	0x00A5	Registro de estado.
0x68	SPI DTR	W	0x0000	Registro de transmisión de datos. Un único registro o con funcionalidad de FIFO.

0x6C	SPI DRR	R	N/A	Registro de recepción de datos. Un único registro o con funcionalidad de FIFO.
0x70	SPISSR	R/W	0xFFFF	Registro de selección de esclavo activo en bajo.
0x74	SPITFOR[3]	R	0x0000	Registro de ocupación de la FIFO de transmisión.
0x78	SPIRFOR[3]	R/W	0x0000	Registro de ocupación de la FIFO de recepción.
0x1C	DGIER	R/W	0x0000	Registro de habilitación global de interrupciones.
0x20	IPISR	R/TOW[2]	0x0000	Registro de estado de interrupción.
0x28	IPIER	R/W	0x0000	Registro de habilitación de interrupción.

Notas:
1 Relativo a la dirección asignada C_BASEADDR.
2 Toggle-On-Write (TOW) invierte el estado de un bit cuando se escribe un valor de 1 al bit correspondiente.
3 Existe solamente cuando la profundidad de la FIFO se configura en 16 o 256.

Principales opciones de configuración en la interfaz

- **XIP mode**. Esta opción habilita la característica *eXecute In Place*, destinada a ser usada en aplicaciones que requieran el acceso a código de *booteo* que reside en una memoria flash SPI. El código es directamente executado de la memoria no volátil, evitando tener que mover el código a la memoria RAM del sistema para su ejecución.
- **Performance mode**. Esta opción permite habilitar la interfaz AXI4-Full, ya que la interfaz que se usa por defecto es AXI4-Lite.
- **Mode**. Selecciona el modo estándar, dual o quad de SPI.
- **Transaction width**. Define el tamaño de las transacciones a 8, 16 o 32 bits.
- **Frequency Ratio**. Es un valor divisor que va desde 2 hasta 2048. La frecuencia resultante para el reloj del SPI es el cociente de la frecuencia asociada a `ext_spi_clk` dividida por el valor seleccionado. En el modo dual o quad del modo heredado y mejorado, el divisor está restringido a 2. en el modo XIP el valor está restringido a 2 para los modos estándar, dual y quad.
- **No. of slaves**. Define el número de esclavos que pueden ser seleccionados a

través del registro de selección de esclavo.
- **Enable master mode**. Cuando se selecciona, el módulo funciona como maestro, mientras que si se deselecciona, el módulo funciona como esclavo.
- **Byte level interrupt enable**. Permite generar interrupciones por nivel de bytes y no por nivel de transacción.
- **Enable FIFO**. Habilita el uso de una FIFO para la transmisión y recepción de datos.
- **Enable startup primitive**. Permite que la primitiva STARTUPE2 sea generada, la cual sirve como interfaz hacia las señales de reset globales asíncronas (GSR, por sus siglas en inglés), la ruta global dedicada de triestado (GTS, por sus siglas en inglés) y algunos otros recursos de configuración del FPGA.

5.1.4. I2C: AXI IIC Bus Interface

Es un módulo que permite la comunicación serial entre dispositivos a través del protocolo I2C. Emplea una interfaz AXI para acceder a sus registros de configuración.

Entre sus principales características se encuentran:

- Cumple con el estándar del protocolo I^2C
- El acceso a los registros de control y configuración es a través de la interfaz AXI4-Lite.
- Puede operar en modo maestro y en modo esclavo.
- Permite la operación en modo multimaestro.
- Modo de operación estándar, *fast mode* y *fast mode plus*.
- Direccionamiento de 7 o 10 bits.
- FIFOs de transmisión y recepción con capacidad para 16 elementos.
- Puerto de propósito general de 1 a 8 bits.

En la figura 5.5 se muestra el diagrama de bloques de su arquitectura interna. El bloque de interfaz AXI4-Lite se encarga de comunicarse con los registros de configuración, control y datos ubicados en el bloque de interfaz de registros (REG Interface). El bloque de reinicio por software (soft reset) permite reiniciar el módulo por software a través del acceso por registros. El bloque (Interrupt control) gestiona el control de las interrupciones. Las FIFO se encargan de almacenar temporalmente los datos que se están transmitiendo o recibiendo, mientras que el bloque (IIC control) contiene la lógica necesaria para llevar a cabo estas operaciones.

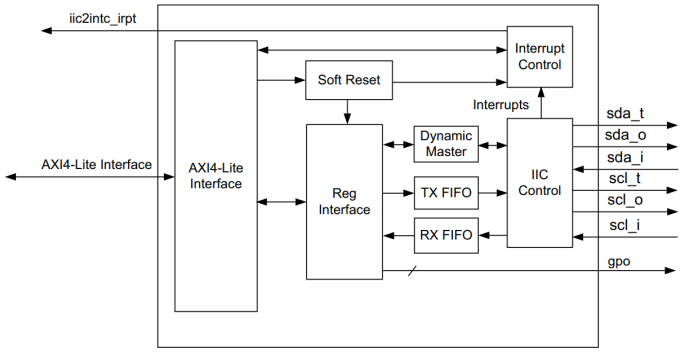


Figura 5.5. Diagrama de bloques del AXI IIC [34].

En la tabla 5.13 se muestran los registros del módulo.

Tabla 5.13. Registros de AXI IIC

Desplazamiento del espacio de direcciones[1]	**Nombre**	**Descripción**
0x001C	GIE	Registro de habilitación global de las interrupciones.
0x0020	ISR	Registro de estado de las interrupciones.
0x0028	IER	Registro de habilitación de interrupciones.
0x0040	SOFTR	Registro de reinicio por software.
0x0100	CR	Registro de control.
0x0104	SR	Registro de estado.
0x0108	TX_FIFO	Registro de la FIFO de transmisión.
0x010C	RX_FIFO	Registro de la FIFO de recepción.
0x0110	ADR	Registro de dirección de esclavo.
0x0114	TX_FIFO_OCY	Registro de ocupación de la FIFO de transmisión.
0x0118	RX_FIFO_OCY	Registro de ocupación de la FIFO de recepción.
0x011C	TEN_ADR	Registro de la dirección de esclavo de 10 bits.
0x0120	RX_FIFO_PIRQ	Registro de interrupción de la profundidad programable de la FIFO de recepción.

0x0124	GPO	Registro de salida de propósito general.
0x0128	TSUSTA	Registro de parámetro de tiempo.
0x012C	TSUSTO	Registro de parámetro de tiempo.
0x0130	THDSTA	Registro de parámetro de tiempo.
0x0134	TSUDAT	Registro de parámetro de tiempo.
0x0138	TBUF	Registro de parámetro de tiempo.
0x013C	THIGH	Registro de parámetro de tiempo.
0x0140	TLOW	Registro de parámetro de tiempo.
0x0144	THDDAT	Registro de parámetro de tiempo.

Notas:
1 Relativo a la dirección asignada C_BASEADDR.

5.1.5. Temporizador

El módulo AXI Timer de Xilinx proporciona una solución para implementar un bloque con capacidad para generar eventos de cuenta, función de captura y función de PWM.

Entre sus principales características se encuentran:

- Interfaz de acceso a través de AXI4-Lite.
- Dos temporizadores programables por intervalos con capacidad de interrupción, generación de eventos y captura.
- Tamaño configurable del contador.
- Una salida de modulación de ancho de pulso (PWM, por sus siglas en inglés).
- Operación en cascada de los temporizadores en los modos de captura y generación.
- Entrada de congelamiento para detener a los temporizadores durante depuración por software.

Se muestra el diagrama de su arquitectura interna en la figura.

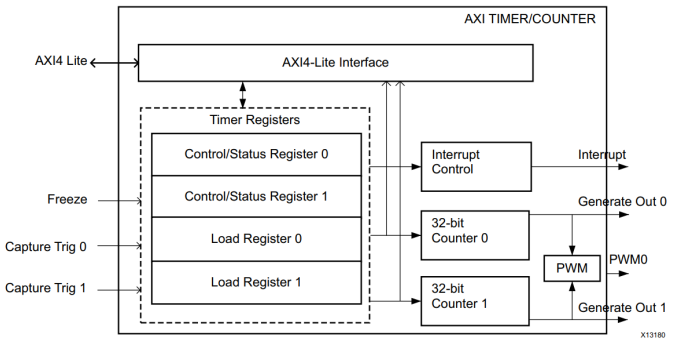


Figure 1-1: **Block Diagram of AXI Timer**

Figura 5.6. Diagrama de bloques del AXI IIC [35].

La tabla con sus registros correspondientes se muestra a continuación:

Tabla 5.14. Registros de AXI Timer

Desplazamiento del espacio de direcciones[1]	**Nombre**	**Descripción**
0x00	TCSR0	Registro de control y estado del temporizador 0.
0x04	TLR0	Registro de carga del temporizador 0.
0x08	TCR0	Registro de cuenta del temporizador 0.
0x10	TCSR1	Registro de control y estado del temporizador 1.
0x14	TLR1	Registro de carga del temporizador 1.
0x18	TCR1	Registro de cuenta del temporizador 1.

Notas:
1 Relativo a la dirección asignada C_BASEADDR.

5.1.6. Watchdog: AXI Timebase watchdog timer v3.0

El temporizador watchdog. Ofrece dos modos de operación: el modo *window watchdog timer* (WWDT) y el modo *legacy watchdog timer* (WDT).

- Acceso mediante la interfaz AXI4-Lite.
- Tamaño programable del temporizador.

- Periodo de *timeout* e interrupción seleccionable.
- Habilitación del watchdog configurable: habilita una vez o habilita repetidamente.
- Contador de base de tiempo de 32 bits de libre funcionamiento con registro de control de interrupción dual en *rollover*.
- Modo de ventana configurable con las siguientes características:
  - Capacidad de ventana del watchdog.
  - Monitoreo de la secuencia del programa.
  - Temporizador de segunda secuencia.
  - Contador de falla.

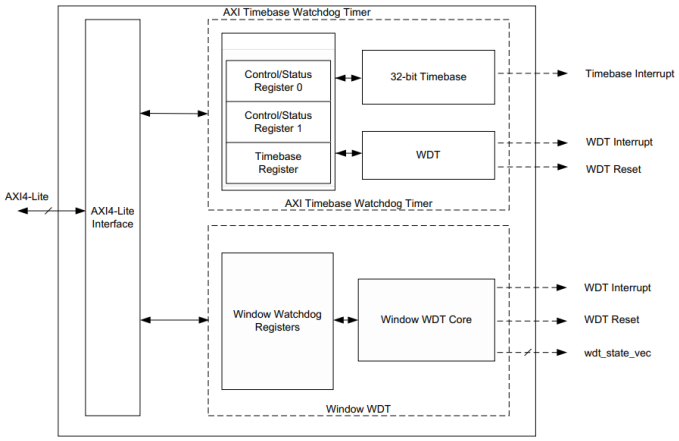


Figura 5.7. Diagrama de bloques del watchdog [36].

Tabla 5.15. Registros del AXI Watchdog

Desplazamiento del espacio de direcciones[1]	**Nombre del registro**	**Tipo de acceso**	**Valor por defecto**	**Descripción**
0x00[1]	TWCSR0	R/W	0x00	Registro 0 de control/estado.

0x04[1]	TWCSR1	W[2]	0x00	Registro 1 de control/estado, el estado está reflejado en TWCSR0 para lectura.
0x08[1]	TBR	R[3]	0x00	Registro de base de tiempo.
0x0C[5]	MWR	R/W	0x00	Cuando WDT es habilitado, este registro es usado como el registro maestro de control de escritura. Cuando está deshabilitado, este es usado para programar el tamaño del temporizador watchdog.
0x10[4]	ESR	R/W	0x00	Registro de habilitación y estado.
0x14[4]	FCR	R/W	0x00	Registro de control de función.
0x18[4]	FWR	R/W	0x00	Registro de configuración de la primera ventana.
0x1C[4]	SWR	R/W	0x00	Registro de configuración de la segunda ventana.
0x20[4]	TSR0	R/W	0x00	Registro 0 de firma de tarea.
0x24[4]	TSR1	R/W	0x00	Registro 1 de firma de tarea.
0x28[4]	STR	R	0x00	Registro de segunda secuencia del temporizador.

Notas:
1 Presente cuando la ventana de habilitación WDT = 0. En este caso, el espacio de registros de 0x0C a 0x30 está reservado.
2 La lectura de este registro regresa valores indefinidos.
3 La escritura en este registro no tiene efecto.
4 Presente cuando la ventana de habilitación WDT = 1. En este caso, el espacio de registros 0x00 a 0x08 está reservado.
5 El registro 0x0C está disponible en dos modos, sin embargo, la funcionalidad es diferente basada en el modo de selección.

WDT usa una arquitectura de expiración dual. Después de que se da una expiración del intervalo de *timeout*, una interrupción se genera y el bit de estado WDT se pone en alto en el registro de estado. Si el bit de estado no es limpiado (al escribir un uno al bit de estado) antes de la siguiente expiración del intervalo de *timeout*, un reinicio por WDT es generado.

Cuando se genera el reinicio por WDT, se activa el bit de estado del reinicio por WDT en el registro de estado. Este bit es usado para determinar si el último reinicio

del sistema fue debido a al reinicio por WDT o no.

El watchdog solo puede deshabilitarse escribiendo dos distintas direcciones, reduciendo la posibilidad de inadvertidamente deshabilitar el watchdog en el código de la aplicación. Una vez que el WDT expire, puede ser reiniciado solamente a través de `s_axi_aresetn`.

Operación del WDT

El intervalo de *timeout* del watchdog es configurado por un parámetro que es igual a $2^{\text{Tamaño del WDT}}$ ciclos de reloj, donde el tamaño del WDT es un entero en el rango de 8 a 31. El intervalo del WDT se configura en la generación del bloque y no puede ser modificado de manera dinámica a través de los registros de control. En la figura 5.8 se muestra un diagrama de estados de su operación.

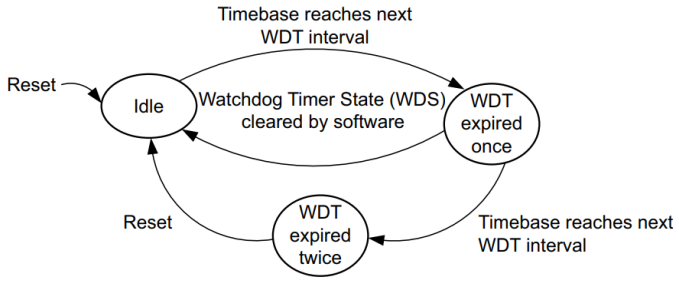


Figura 5.8. Diagrama de estados del WDT [36].

En la interfaz de configuración la opción **WDT Enable Behavior** permite seleccionar el modo de habilitación del watchdog, para habilitarse una vez y no poder deshabilitarse o para poder habilitarse/deshabilitarse por software.

5.1.7. Controlador de interrupciones

El controlador de interrupciones recibe múltiples entradas de interrupciones de los distintos periféricos y las combina en una sola fuente de interrupción para el procesador del sistema. Los registros usados para almacenar las direcciones de los vectores de interrupción, estados, verificación y validación son accedidos a través de la interfaz AXI4-Lite.

Entre sus principales características se encuentran:

- Acceso a los registros de configuración a través de la interfaz AXI4-Lite.
- Modo de interrupción rápida.
- Soporta hasta 32 interrupciones. Puede ponerse en cascada para aumentar el número de interrupciones.
- Interfaz de interrupción en modo de una sola vía o en modo de bus.
- La prioridad entre las solicitudes de interrupciones son determinadas en base a la posición del vector. El bit menos significativo (bit 0) tiene la prioridad más alta.
- El registro de habilitación de interrupción permite habilitar las entradas de interrupción de manera selectiva.
- Registro de habilitación maestra para habilitar las solicitudes de interrupción a la salida.
- Cada entrada puede ser configurada para ser sensible por nivel o por flanco.
- El pin de salida de solicitud de interrupción puede ser configurado para flanco o nivel de salida.
- Capacidad de generar interrupciones por software.
- Soporte para interrupciones anidadas.

En la figura 5.9 se muestra el diagrama de bloques interno del controlador, en el cual se presentan los cuatro principales bloques. El bloque de detección de interrupciones (Int Det) se encarga de detectar las entradas de interrupción, pudiendo ser configurado el modo de detección por nivel o flanco para cada entrada. El bloque de registros (Regs block) contiene los registros de control y estado. El bloque de generación de interrupción (Irq Gen) se encarga de la salida de la solicitud de interrupción y por último el bloque de interfaz (AXI interface) proporciona el acceso a los registros haciendo uso de la interfaz AXI4-Lite.
En la tabla 5.16 se muestran los registros del bloque para su configuración.

Tabla 5.16. Registros del controlador de interrupciones AXI

Desplazamiento del espacio de direcciones[1]	**Nombre**	**Descripción**
0x00	ISR	Registro de estado de interrupción.
0x04	IPR	Registro de interrupción pendiente.
0x08	IER	Registro de habilitación de interrupción.

0x0C	IAR	Registro de reconocimiento de interrupción.
0x10	SIE	Establecer habilitación de interrupción.
0x14	CIE	Borrar habilitación de interrupción.
0x18	IVR	Registro del vector de interrupción.
0x1C	MER	Registro habilitador maestro.
0x20	IMR	Registro de modo de interrupción.
0x24	ILR	Registro de nivel de interrupción.
0x100 a 0x17C	IVAR	Registro de dirección del vector de interrupción.
0x200 a 0x2FC	IVEAR	Registro de extensión de dirección del vector de interrupción.

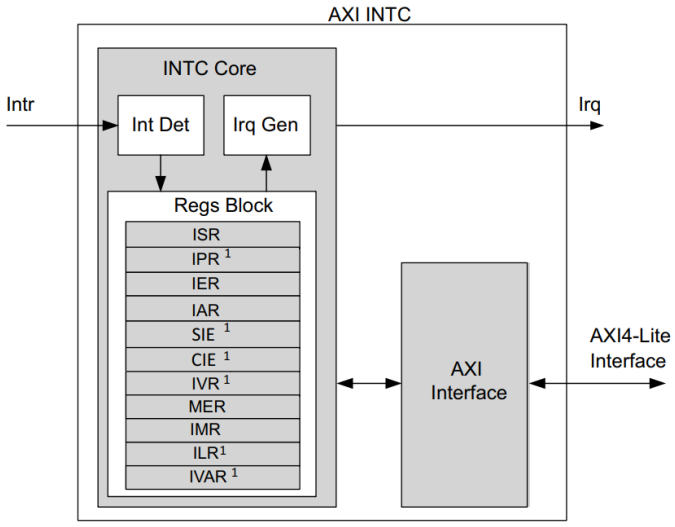
Notas:
1 Relativo a la dirección asignada C_BASEADDR.

Interrupciones anidadas

El núcleo provee la capacidad de manejar el anidamiento de interrupciones, implementando un registro de nivel de interrupción. Este puede ser usado por el software para prevenir que interrupciones con más baja prioridad ocurran cuando se está atendiendo una interrupción, permitiendo así que las interrupciones sean habilitadas durante la atención de una interrupción para inmediatamente tomar una interrupción de más alta prioridad. A través del software se debe guardar y restaurar el registro de nivel de interrupción y dirección de retorno.

Debido a que el procesador en el modo de interrupción rápida salta directamente al único vector de dirección de interrupción para atender a una interrupción en particular, la rutina de servicio de interrupción debe guardar y restaurar el registro de nivel de interrupción y dirección dirección de retorno.

Para conectar las entradas de interrupción en este módulo se requiere del uso de un bloque concatenador, el cual permite la conexión de varios buses y señales en un único bus.



1. Estos registros son opcionales.

Figura 5.9. Diagrama de bloques del controlador de interrupciones [37].

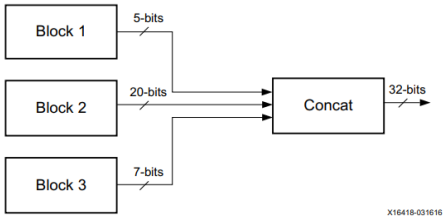


Figura 5.10. Bloque concatenador [38].

5.1.8. DMA

Inicialmente no se consideró emplear un módulo DMA, sin embargo, debido a la necesidad de llevar los datos del sensor de imagen a la memoria de una manera más eficiente sin la intervención del procesador, se hace necesaria la inclusión de este bloque. El AXI DMA de Xilinx es un bloque IP que provee acceso directo a memoria entre memoria y periféricos que usan AXI4-stream. Adicionalmente ofrece capacidad de funcionalidad *scatter/gather* que permite liberar tareas de transferencia de datos de la unidad central de procesamiento (CPU, por sus siglas en inglés). Los registros de inicialización, reporte de estado y control son accedidos a través de una interfaz AXI4-Lite esclava.

Características principales:

- Cumple con la especificación AXI4
- Soporte opcional del modo [scatter/gather].
- Soporte para un ancho de bus de datos de 32, 64, 128, 256, 512 y 1024 bits en AXI4.
- Soporte para un ancho de bus de datos de 8, 16, 32, 64, 128, 256, 512 y 1024 bits en AXI4-Stream.
- Soporte opcional keyhole.
- Soporte opcional de realineamiento de datos para flujos de datos con un ancho de bus de hasta 512 bits.
- Flujos de estado y control opcionales.
- Soporte opcional de Micro DMA.
- Soporte para un direccionamiento de hasta 64 bits.

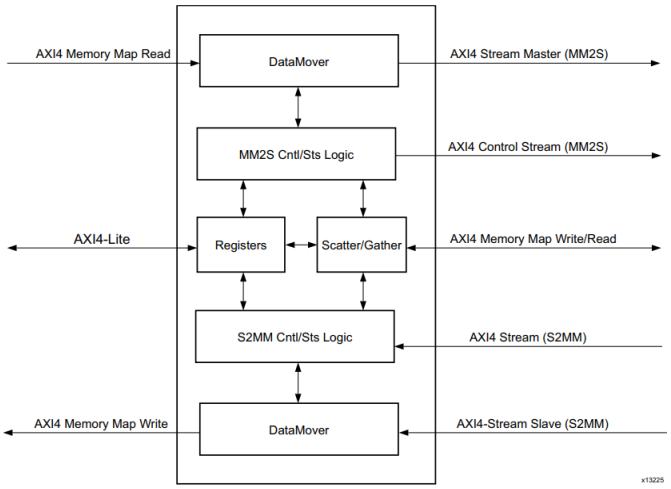


Figura 5.11. Diagrama de bloques del módulo AXI DMA [39].

El movimiento de datos primarios en alta velocidad entre la memoria del sistema y el *stream* objetivo es a través de los módulos *AXI4 Read Master to AXI4 memory-mapped to stream (MM2S) Master* y *AXI stream to memory-mapped (S2MM) Slave to AXI4 Write Master*. En el modo *scatter/gather* el DMA permite usar hasta 16 canales múltiples para movimiento de datos tanto para MM2S como para S2MM.

El canal MM2S y el canal S2MM operan de manera independiente.

Los registros para el modo directo se muestran en la tabla.

Tabla 5.17. Registros de AXI DMA para el modo DMA simple

Desplazamiento del espacio de direcciones[1]	**Nombre**	**Descripción**
0x00	MM2S_DMACR	Registro de control del DMA para el canal MM2S![2]
0x04	MM2S_DMASR	Registro de estado del DMA para el canal MM2S.
0x18	MM2S_SA	Registro para la parte baja de 32 bits de la dirección destino para el canal MM2S.
0x1C	MM2S_SA_MSB	Registro para la parte alta de 32 bits de la dirección destino para el canal MM2S.
0x28	MM2S_LENGTH	Tamaño de la transferencia en bytes para el canal MM2S.
0x30	S2MM_DMACR	Registro de control del DMA para el canal S2MM![3]
0x34	S2MM_DMASR	Registro de estado del DMA para el canal S2MM.
0x48	S2MM_DA	Registro para la parte baja de 32 bits de la dirección destino para el canal S2MM.
0x4C	S2MM_DA_MSB	Registro para la parte alta de 32 bits de la dirección destino para el canal S2MM.
0x58	S2MM_LENGTH	Tamaño de la transferencia en bytes para el canal S2MM.

Notas:
1 Relativo a la dirección asignada C_BASEADDR.
2 *Memory Mapped to Stream.*
3 *Stream to Memory Mapped.*

5.1.9. MIG DDR3

Xilinx proporciona herramientas para la generación de interfaces avanzadas de memoria, como lo son SDRAM DDR2, DDR3, QDR II+, RLDRAM II, RLDRAM 3 y LPDDR2. Para la serie 7 de FPGAs se ofrece la herramienta generadora de interfaz de memoria (MIG, por sus siglas en inglés) en su versión 4.2 [40], que permite configurar un controlador de memoria que se adapta a las especificaciones de las interfaces DDR2 o DDR3 y a las especificaciones y limitaciones del dispositivo FPGA seleccionado.

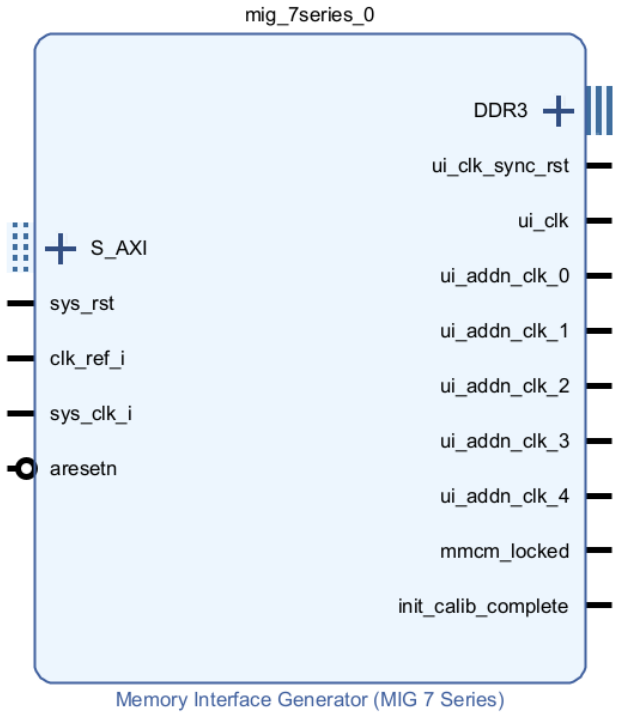


Figura 5.12. Diagrama de los puertos del controlador MIG.

S_AXI: Interfaz AXI4 esclava. Permite mapear las transacciones AXI4 hacia la interfaz UI del controlador, obteniendo así protocolo de bus estándar para comunicarse con el controlador de memoria. Todas las señales de la interfaz AXI esclava son síncronas al reloj **ui_clk**

sys_clk: Es la entrada de reloj del sistema para la interfaz de memoria y está tipicamente conectada a fuente de reloj externa con bajo jitter. La entrada debe estar en la misma columna que la interfaz de memoria.

clk_ref: Esta es la entrada para la frecuencia de referencia para el control del IDELAY. Puede ser manejada de manera interna o externa.

sys_rst: es la entrada del reset asíncrono del sistema, la cual puede ser generada internamente o manejada a través de un pin.

init_calib_complete: Esta salida indica que la inicialización y calibración de la

memoria se ha completado y que la interfaz está lista para ser usada.

aresetn: Entrada de reset hacia el AXI shim, la cual debe ser síncrona con la lógica del FPGA.

ui_clk_sync_rst: Es el reset del UI, el cual es síncrono con ui_clk

ui_clk. Esta es la salida de reloj del UI. Esta tiene que ser un la mitad (2:1) o un cuarto (4:1) de la frecuencia de reloj que va hacia la SDRAM. Se determina en base al modo seleccionado en la configuración

ui_addn_clk. Son los relojes adicionales que se pueden generar a través del mismo MMCM de la memoria

mmcm_locked. Indica que el generador de reloj, ya sea PLL o MMCM se ha iniciado.

Las restricciones para el uso de este bloque en el hardware del FPGA son las siguientes:

- La interfaz se puede extender a través de un máximo de tres bancos consecutivos.
- Los bancos usados para la interfaz deben residir en la misma columna del FPGA.
- Los bancos usados en la interfaz deben ser del tipo alto rendimiento (HP, por sus siglas en inglés) o de alto rango (HR, por sus siglas en inglés).
- Si el dispositivo emplea *stacked silicon interconnect technology*, los bancos empleados para la interfaz deben tener la misma región SLR.
- Los pines relacionados a un conjunto *strobe* deben residir en el mismo grupo de byte.
- Los pares de *strobe* DQS deben estar alojados en los pares de pines con función DQS.
- Un carril de byte del FPGA no debe contener pines relacionados a dos diferentes conjuntos de *strobe*.
- El reloj del sistema solo puede ser conducido a través de los pines con capacidad de reloj (SRCC o MRCC), los cuales deben estar localizados en la columna de los bancos de memoria. Lo mismo aplica para el caso del reloj de referencia.

Debido a que el controlador de memoria tiene un único canal de dirección y en la

interfaz AXI4 se emplean canales independientes para las direcciones de escritura y lectura, se ofrecen distintos esquemas de arbitraje para llevar a cabo el acceso para los canales de dirección de escritura y lectura, los cuales son los siguientes:

- **Time Division Multiplexing (TDM)**. En este modo se le da igual prioridad a los canales de dirección de escritura y lectura, alternando el acceso con cada ciclo de reloj. Las solicitudes de lectura o escritura del maestro AXI no tienen relación con la concesión de los accesos, es decir, si se presenta una solicitud de escritura o lectura, esta es atendida de manera alternada sin importar que no existan solicitudes de la otra operación.
- **Round-Robin**. En este modo se le da igual prioridad a los canales de dirección de escritura y lectura. El acceso otorgado a los canales de escritura o lectura depende del último acceso otorgado a la solicitud del maestro AXI. De esta manera si la última operación atendida fue de escritura, la solicitud de lectura tiene precedencia sobre una de lectura. Un caso contrario ocurre para cuando la última operación atendida fue de lectura, mientras que ocurren peticiones simultáneas de la escritura y lectura, el canal de escritura tiene precedencia.
- **Read priority (RD_PRI_REG)**. Las solicitudes del canal de dirección de escritura son procesadas cuando alguna de las siguientes condiciones ocurre:
  - No hay solicitudes pendientes del canal de dirección de lectura.
  - Se alcanza el límite de *starve* de 256 para el canal de lectura, el cual solo se verifica al final de la ráfaga de datos.
  - Se alcanza un valor de espera para la lectura de 16.
- **Read priority with Starve Limit (RD_PRI_REG_STARVE_LIMIT)**. El canal de lectura siempre tiene prioridad en este modo, por lo que las solicitudes del canal de dirección de escritura son procesadas una vez que no hay solicitudes pendientes del canal de dirección de lectura o una vez que el límite de *starve* del canal de lectura es alcanzado.
- **Write Priority (WRITE_PRIORITY, WRITE_PRIORITY_REG)**. El canal de direción de escritura tiene siempre prioridad en este modo, de manera que las solicitudes del canal de dirección de lectura son procesadas cuando no solicitudes pendientes del canal de dirección de escritura.

Los siguientes componentes derivan del uso de las IP seleccionadas y son requeridos para el funcionamiento del sistema, por lo cual también se detallan sus especificaciones.

5.1.10. SmartConnect v1.0

Es un bloque que permite conectar uno o más dispositivos maestros AXI por mapeo de memoria (*memory-mapped*) a uno o más dispositivos esclavos AXI por mapeo de memoria.

Entre sus principales características se encuentran:

- Hasta 16 interfaces esclavas y hasta 16 interfaces maestras por bloque.
- Se pueden conectar bloques en cascada para ampliar el número de maestros y esclavos.
- Cada interfaz maestra o esclava se puede conectar a un maestro o esclavo con interfaz de tipo AXI3, AXI4 o AXI4-Lite.
- Tamaño de la interfaz de datos de AXI4 y AXI3 de 32, 64, 128, 256, 512 y 1024 bits. Para AXI4-Lite es de 32 y 64 bits.
- Las transacciones con distinto ancho de bus son convertidas automáticamente por el bloque.
- Soporta transacciones en múltiples dominios de reloj.
- Ancho del bus de direcciones de hasta 64 bits.
- Resincronización de la entrada de reinicio interna.

5.1.11. Processor system reset

Es un módulo que permite llevar a cabo una secuencia de reinicio de sistemas basados en el uso de un procesador. Se encarga de gestionar las distintas condiciones de reinicio a la entrada, ya sean externas o internas para poder generar un mecanismo apropiado de reinicio a la salida.
Entre sus principales características se encuentran:

- Soporta la entrada de una señal de reinicio asíncrona que es sincronizada con el reloj del sistema.
- Las entradas de reinicio externo y auxiliar se pueden configurar para ser activas en alto o bajo.
- Ancho de pulso mínimo seleccionable para que una entrada de reinicio sea reconocida.
- Entrada para el bloqueo por DCM.
- Generación de *Power On Reset*
- Secuencia de reinicio definida para la estructura de bus, periféricos y procesador.

La secuencia de reinicio que sigue el módulo es la siguiente:

- Las estructuras de bus (Interconexiones y puentes) salen del estado de reinicio primero.
- Los periféricos (UART, SPI, I2C, etc.) salen del estado de reinicio 16 ciclos de reloj más tarde.
- El procesador sale del estado de reinicio 16 ciclos de reloj después de los periféricos.

Las señales presentes en este bloque son las siguientes:

Tabla 5.18. Señales del módulo Processor System Reset

Señal	Tipo	Descripción
peripheral_reset	Salida	Señal de reinicio para todos los periféricos que están conectados a cualquier bus que sea síncrono con la señal de reloj `slowest_sync_clk`. Activa en alto.
peripheral_aresetn	Salida	Señal de reinicio para los periféricos conectados a una interconexión que es síncrona con la señal de reloj `slowest_sync_clk`. Activa en bajo.

5.1.12. Microblaze debug module (MDM)

Este módulo habilita la depuración del procesador MicroBlaze a través de JTAG, haciendo posible el uso de herramientas de depuración por software, como es el caso del entorno de desarrollo de software Vitis.

Entre sus principales características se encuentran:

- Soporte para herramientas de depuración por software basadas en JTAG.
- Soporte para depurar hasta 32 procesadores MicroBlaze.
- Permite el control síncrono de varios procesadores.
- Basado en la lógica de escaneo de bordes (BSCAN) de los FPGA de Xilinx.

En la figura 5.13 se muestra el diagrama de bloques interno del bloque IP. El acceso JTAG se consigue a través del uso de las primitivas BSCAN, presentes en las FPGA de Xilinx.

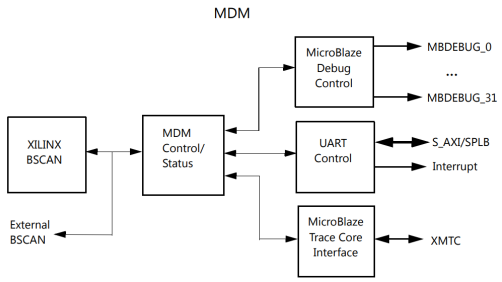


Figura 5.13. Diagrama de bloques interno del módulo MDM [41].

5.2. Arquitectura de la computadora

En esta sección se detallan las principales interfaces empleadas en la arquitectura, así como la distribución de los dominios de reloj y la distribución de las entradas y salidas de reinicio de los diferentes componentes.

5.2.1. Interfaces principales

La figura 5.14 muestra algunas de las principales interfaces para los principales componentes que se encuentran en el sistema. El procesador emplea la interfaz LMB para comunicarse con la memoria de datos y con la memoria de programa.
La interfaz AXI4 permite la comunicación del procesador con la matriz de buses definida por el módulo de interconexión, el cual a su vez también emplea una interfaz AXI4 para comunicarse con el módulo de memoria MIG DDR3, mientras los otros periféricos emplean una interfaz AXI4-Lite. El controlador de interrupciones es controlado por medio de la interfaz AXI4-Lite y a su vez tiene un bus para el control de las interrupciones del procesador.
El sensor de imagen transmite las señales de sincronización de video y de datos al módulo capturador de imagen, el cual se encargará de convertir esta interfaz CPI a una interfaz AXI4-Stream, la cual está conectada a un bloque de cruce de reloj para pasar del dominio de reloj del sensor de imagen a el dominio de reloj del procesador.

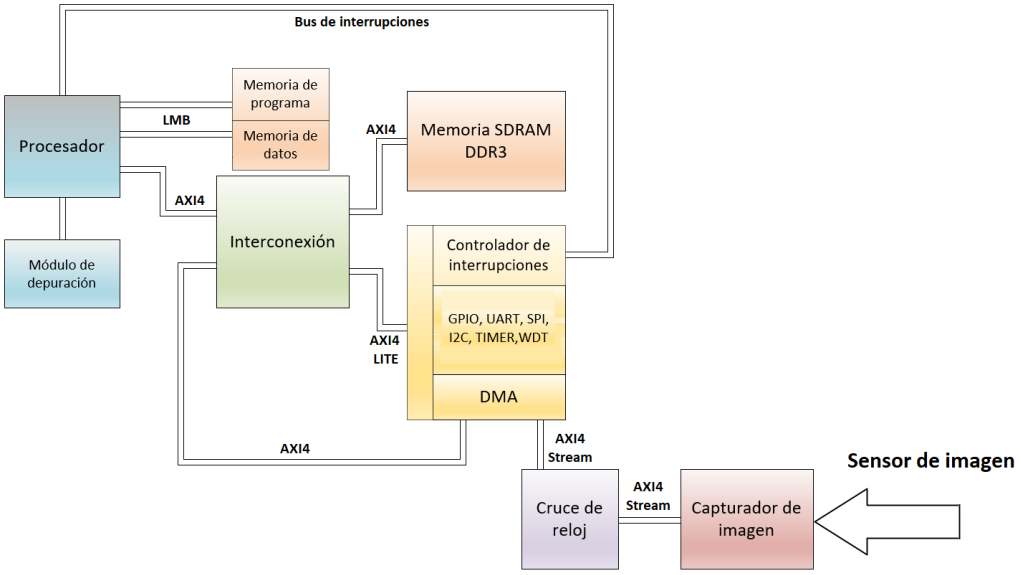


Figura 5.14. Principales interfaces.

5.2.2. Distribución de los dominios de reloj

Los relojes que estarán disponibles se definen en base a cada uno de los requerimientos de los bloques. Existen tres principales dominios de reloj definidos en el sistema: el dominio de reloj del procesador, el dominio de reloj de la memoria SDRAM DDR3 y el dominio de reloj del sensor de imagen. La señal de reloj ui-clk del módulo MIG DDR3 se usa como el reloj base para generar las señales de reloj adicionales al sistema. La configuración está en 4:1, por lo que si el reloj es de 400 MHz, `ui-clk` será de 100 Mhz.

Se considera lo siguiente para las señales de reloj:

- La señal de reloj `ui-clk` se usará como entrada a otro bloque generador de reloj para generar las señales de reloj adicionales del sistema, en este caso la del procesador, sensor de imagen y para el AXI QSPI.
- El procesador, el controlador de interrupciones y los periféricos funcionan bajo el mismo dominio de reloj.
- El sensor CMOS necesita de una señal de reloj externa para funcionar, la cual será generada por el bloque generador de reloj.

- Se usará un módulo axi (AXI STREAM CLOCK CONVERTER) que permitirá realizar el cruce de dominio de reloj del sensor de imagen al reloj del procesador.
- AXI QSPI necesita una señal de reloj adicional para funcionar, misma que también será generada por medio del bloque generador de reloj.

La distribución de las señales de reloj se muestra en la figura 5.15.

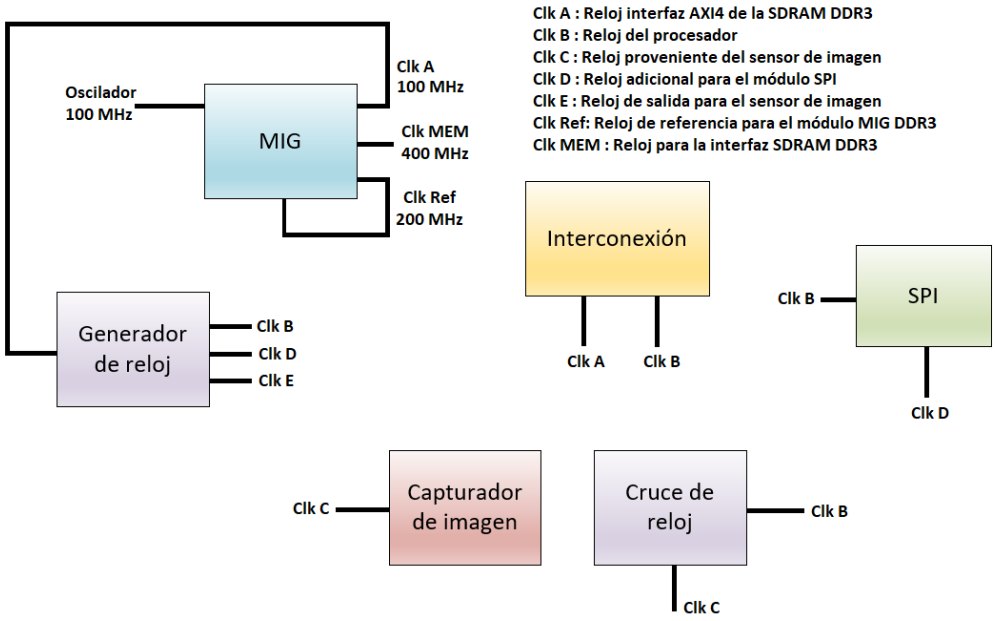


Figura 5.15. Distribución de los dominios de reloj.

5.2.3. Distribución de las señales de reinicio

Las fuentes de las señales de reinicio principales son la externa, la del watchdog y la del módulo de depuración. La señal de reset externa se conecta al MIG DDR3, el cual genera la señal de reloj principal a partir del oscilador. Esta señal también está conectada a los módulos secuenciadores de reinicio, cada uno de los cuales funciona para un respectivo dominio de reloj.

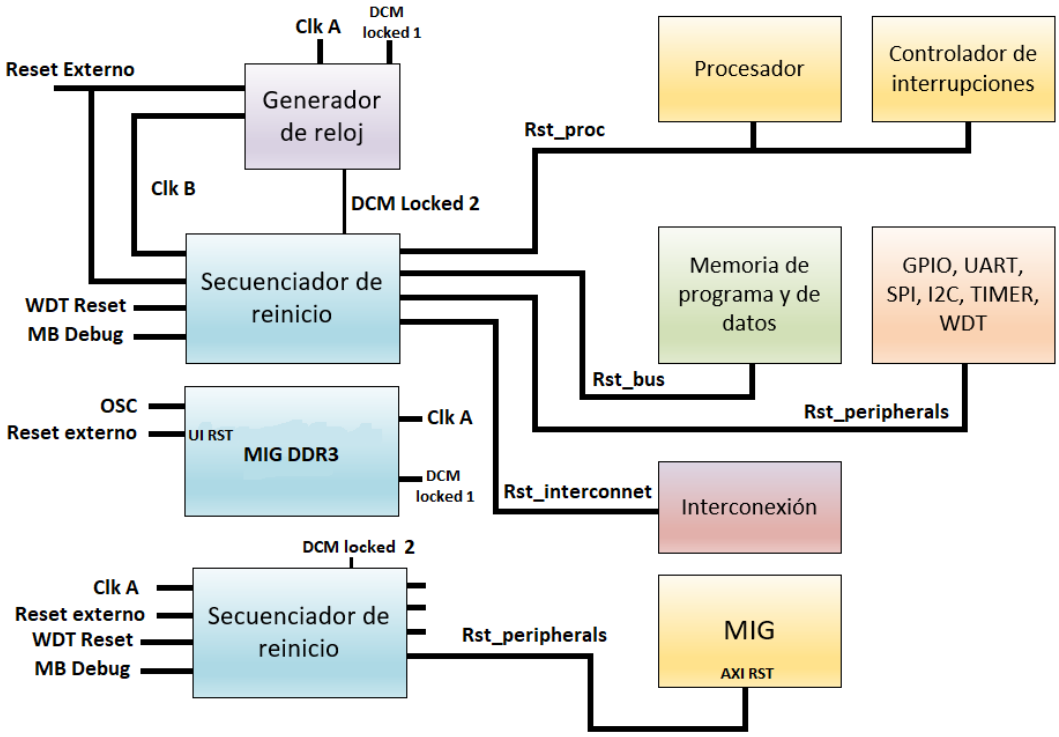


Figura 5.16. Distribución de las señales de reinicio.

Capítulo 6

Implementación

La etapa de implementación comprende la parte del proceso de diseño en la cual se comienza con la construcción del sistema, integrando los componentes que se definieron como parte del proceso de diseño de detalle. Para la implementación, con el fin de verificar la funcionalidad y contruir el hardware de la computadora se asignan y definen tareas a cada uno de los componentes de la computadora para verificar el funcionamiento de manera práctica. Para esto la asignación de tareas se lleva a cabo de acuerdo con la tabla 6.1. Como la tarjeta de desarrollo Nexys Video incluye algunos componentes conectados al FPGA, se hace uso de algunos de ellos para las pruebas con los periféricos.

Tabla 6.1. Asignación de tareas

Periférico o componente	**Función**	**Descripción**
AXI UART Lite	Envío y recepción de datos	Se encargará de recibir los comandos del sistema de control, así como de enviar la respuesta de los mismos y los datos de imagen cuando sean solicitados.
AXI GPIO 0	Puerto E/S multipropósito	Puerto de E/S que podrá ser usado para tareas de propósito general definidas por el usuario.

AXI GPIO 1	Entrada dip switches	Puerto de solo entrada que estará conectado a los dip switch incluidos en la tarjeta.
AXI GPIO 2	Leds de salida	Puerto de solo salida que estará conectado a los leds de la tarjeta con el propósito de mostrar información.
AXI GPIO 3	Control de display OLED	Puerto de solo salida que servirá para el control de la pantalla OLED disponible en la tarjeta.
AXI Timer 0	Eventos de cuenta del sistema	Temporizador que será usado para llevar a cabo tiempos de cuenta del sistema.
AXI Timer 1	Conteo de tiempo del sistema	Temporizador que será usado para llevar a cabo cuentas de tiempo, a fin de generar eventos como los retardos.
AXI IIC	Configuración del sensor Ov7670	Este módulo permitirá la comunicación por I2C con el sensor de imagen Ov7670, al permitir escribir y leer los registros para su configuración.
AXI QSPI	Pantalla OLED	El módulo permitirá la comunicación con la pantalla OLED por SPI para configurarla y poder mostrar información a través de ella.
AXI WDT	Watchdog del sistema	Se encargará de reiniciar el sistema en caso de que este no responda.
AXI DMA	Acceso directo a memoria	Es el módulo encargado de proporcionar el acceso directo a memoria, haciendo que el procesador no tenga que intervenir para mover los datos a esta.
MIG DDR3	Memoria de datos adicional	Será la memoria usada para almacenar los datos del sensor de imagen.

AXI Interrupt controller	Controlador de interrupciones del sistema	Gestiona las interrupciones de los periféricos y las envía al procesador.

6.0.1. Secuencia de implementación

Cada uno de los bloques IP fue probado de manera individual con el procesador MicroBlaze a modo de comprobar su funcionamiento y su configuración, posteriormente en base a la arquitectura del diseño a nivel sistema de la computadora se implementaron todos los bloques de los periféricos. El primero de los bloques que se implementó fue el controlador de memoria, debido a que es el elemento con mayor complejidad en su configuración y que además define como se conectarán los demás componentes. Posterior a este se se siguió con la implementación del procesador, el controlador de interrupciones, el DMA, posteriormente los periféricos y finalmente bloques adicionales como el secuenciador de reinicio del sistema y el módulo de depuración.

MIG DDR3

El primer bloque IP que se configuró para la construcción del sistema fue el controlador de memoria MIG DDR3, el cual requiere de una cuidadosa configuración de sus parámetros debido a que una vez que se genera no es posible modificarlo y para realizar cambios se tiene que realizar nuevamente toda la configuración del bloque. Los parámetros configurados para este bloque surgen de la documentación consultada, especificación de la interfaz DDR3, consultas en foros de soporte y una serie de pruebas llevadas a cabo de manera práctica.

Los siguientes son los principales parámetros configurados para este bloque, para la interfaz de memoria DDR3.

- Interfaz para memoria SDRAM DDR3.
- Periodo de reloj de 2500 ps (400 MHz) para la interfaz de la memoria.
- Relación de reloj del PHY respecto al controlador de 4:1 (100 MHz).
- Número de parte seleccionado: MT41K256M16XX-125.
- Voltaje de la memoria de 1.5 V.
- Ancho del bus de datos de 16 bits.
- Enmascaramiento de bytes habilitado.

- Ocho bancos de máquina.
- Ordenamiento de los comandos en modo normal (Los comandos se ordenan para obtener la máxima eficiencia posible).
- Interfaz AXI4-Full esclava de 32 bits.
- Esquema de arbitraje `WRITE_PRIORITY` para los canales de dirección de escritura y lectura.
- Periodo del reloj de entrada de 10000 ps (100 MHz).
- Generación de reloj adicional de 200 MHz.
- Tipo de ráfaga de lectura en modo secuencial.
- Impedancia para los buffers de salida igual a RZQ/6.
- Impedancia de terminación *On Die* igual a RZQ/6.
- Pin de selección de chip deshabilitado.
- Direccionamiento de memoria a 29 bits en modo [Banco][Fila][Columna].
- Modo *Single-ended* para el reloj de entrada.
- Reloj de referencia sin ruta dedicada externa.
- Polaridad de reinicio del sistema configurada para activa en bajo.
- Uso del voltaje de referencia.
- Reducción de la potencia de consumo activada.
- Generación de bloque XADC para monitoreo de la temperatura.

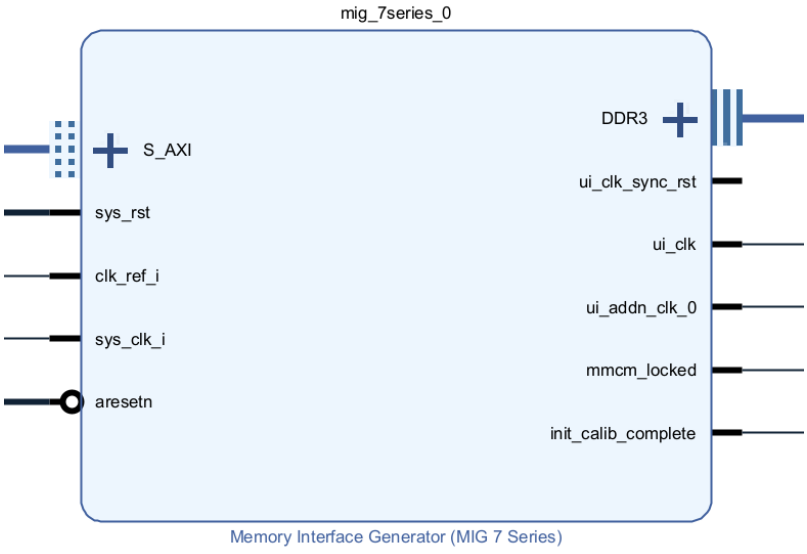


Figura 6.1. Implementación del controlador de memoria MIG SDRAM DDR3.

Procesador

Para el caso del procesador MicroBlaze, las opciones de configuración en la interfaz al implementarse fueron las siguientes:

- Modo de optimización para obtener el mejor rendimiento.
- Caché de instrucciones y datos deshabilitados.
- Unidad de punto flotante deshabilitada.
- Interrupciones en modo *fast*.
- Multiplicador entero y desplazador aritmético deshabilitado.
- Interfaz de depuración habilitada.
- Interfaz para el bus de memoria local y de instrucciones.
- Interfaz AXI4 habilitada.

De acuerdo con la figura 6.2 cada una de las interfaces y señales se conectan de la siguiente manera:

- La interfaz `INTERRUPT` del procesador al estar configurada en modo *fast* deberá ir conectada al controlador de interrupciones.
- La interfaz `DEBUG` deberá conectarse al módulo de depuración.
- La entrada `Clk` es el reloj del procesador y deberá ser síncrono con el acceso a la memoria de programa y de datos.
- **Reset** es la entrada para reiniciar al procesador activa en alto y se conectó al módulo de reinicio del sistema.
- La interfaz `DLMB` va hacia la memoria de datos, mientras que la interfaz `ILMB` hacia la memoria de instrucciones.
- `M_AXI_DP` es la interfaz maestra AXI4, que permitirá la comunicación con los periféricos del sistema. Esta requiere de un módulo de interconexión para poder comunicarse con varios esclavos.

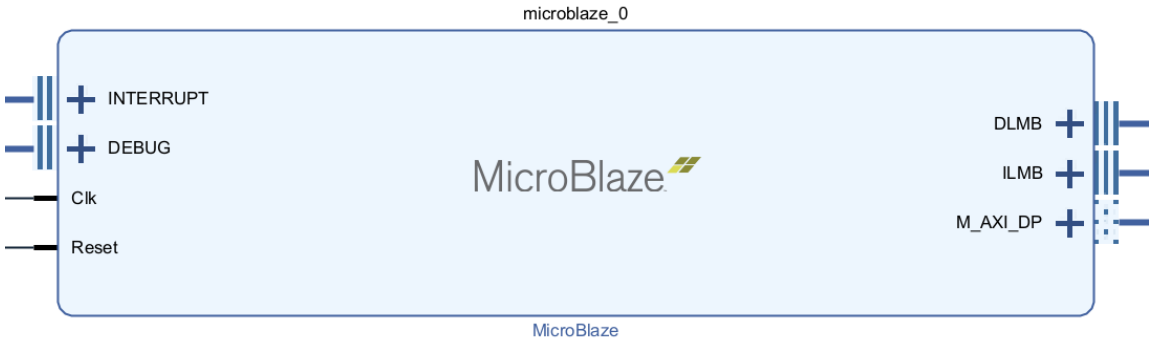


Figura 6.2. Implementación del procesador.

Para la memoria de programa y de datos se usaron los bloques mostrados en la figura 6.3. El bloque generador de memoria ofrece una interfaz que hace posible generar una memoria empleando los recursos de almacenamiento disponibles en el FPGA como los bloques BRAM. Este a su vez necesita de un bloque controlador que estará conectado también al módulo encargado de realizar la interfaz con el bus del procesador.

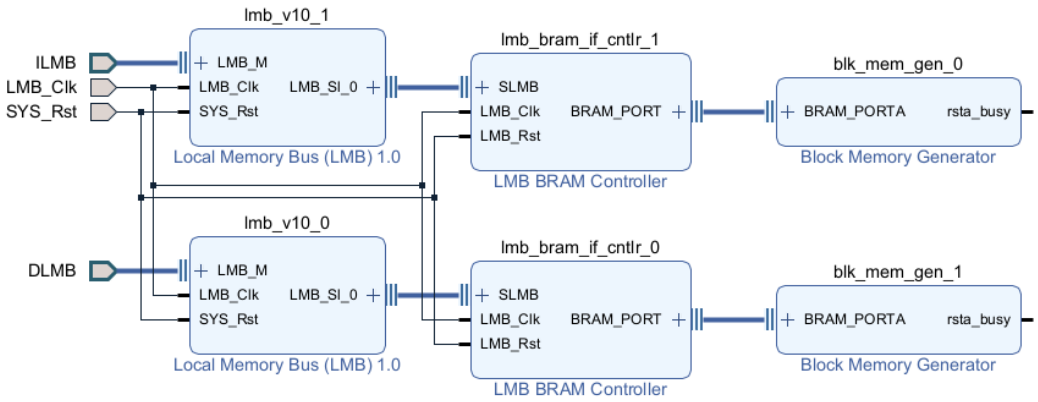


Figura 6.3. Estructura de la memoria de programa y de datos para el procesador.

Con la finalidad de gestionar mejor las conexiones, la memoria de datos y de programa fueron agrupadas en un solo bloque como se muestra en la figura 6.4.

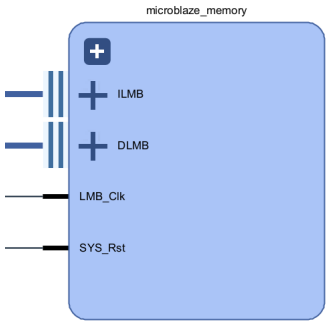


Figura 6.4. Bloque de la memoria de programa y de datos del procesador.

GPIO

Para el módulo AXI GPIO, las opciones de configuración fueron las siguientes:

- Tamaño de la interfaz de 8 bits.
- Doble canal habilitado.
- Valor de salida por defecto en 0x00.
- Valor por defecto del triestado en 0xFF.
- Capacidad para generar interrupciones.

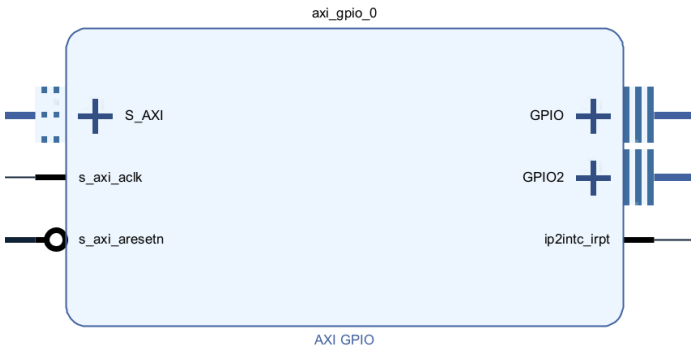


Figura 6.5. Implementación del GPIO.

UART

Para el módulo AXI UART Lite 2.0 las opciones de configuración fueron las siguientes:

- Tasa de baudios de 115200.
- Tamaño de datos de 8 bits.
- Sin bit de paridad

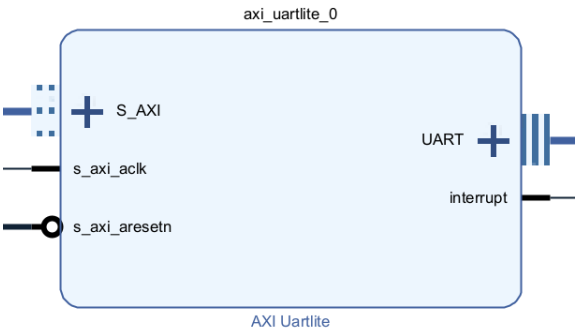


Figura 6.6. Implementación del UART.

SPI

Las opciones que se configuraron para el modulo fueron las siguientes:

- Modo XIP deshabilitado
- Modo performance deshabilitado
- Modo estándar seleccionado
- Tamaño de transacción de 8 bits
- Factor de división de frecuencia de 16
- Modo maestro
- Selección de un esclavo
- FIFO habilitada con profundidad de 16 elementos
- Señal de entrada externa de reloj de 16 Mhz

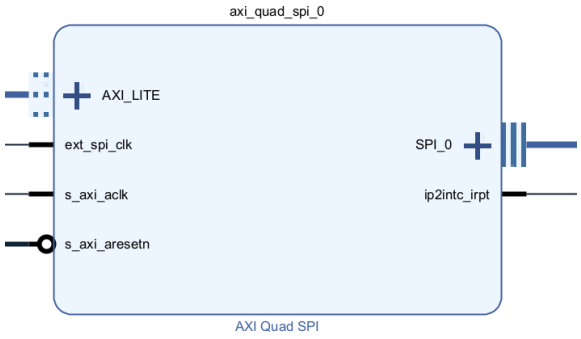


Figura 6.7. Implementación del SPI.

I2C

Las opciones de configuración empleadas para el I2C fueron las siguientes:

- Frecuencia de SCL a 100 kHz.
- Modo de direccionamiento de 7 bits.
- Retardo inercial de SCL y SDA a 0.
- Estado activo de SDA en 1.
- Puerto de salida de propósito general de 1 bit.

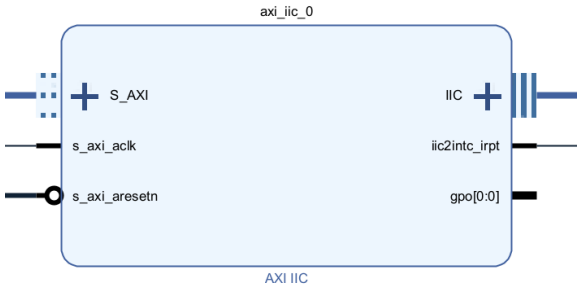


Figura 6.8. Implementación del I2C.

Temporizador

Las opciones habilitadas para el temporizador son las siguientes:

- Tamaño del contador de 32 bits.
- Estado activo del disparador de captura en activo alto para el temporizador 1.
- Estado activo de la señal de generación de salida en alto para el temporizador 1.
- Estado activo del disparador de captura en activo bajo para el temporizador 2.
- Estado activo de la señal de generación de salida en bajo para el temporizador 2.

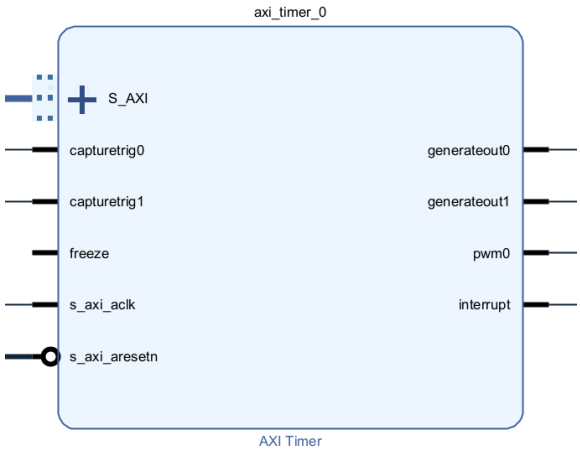


Figura 6.9. Implementación del temporizador.

Watchdog

El watchdog fue configurado en modo WDT temporizador con un tamaño de 27 bits y con la opción de poder activarse y desactivarse por software. El modo ventana watchodog no se habilitó.

En la figura 6.10 se muestran las interfaces y señales del módulo conectadas de la siguiente manera:

- `S_AXI` es la interfaz AXI4-Lite esclava que va al interconector.
- La entrada `freeze` se deshabilitó poniéndose en bajo.
- La entrada `s_axi_aclk` se conectó a la señal de reloj para los periféricos AXI.
- La entrada `s_axi_aresetn` activa en bajo se conectó a la señal de reinicio de los periféricos AXI.
- La señal de salida `timebase_interrupt` se conectó como señal de interrupción al controlador de interrupciones.
- La señal de salida `wdt_interrupt` se conectó también como una señal de interrupción al controlador de interrupciones.
- La señal de salida `wdt_reset` se conectó como entrada de reinicio auxiliar al módulo de reinicio de procesador.

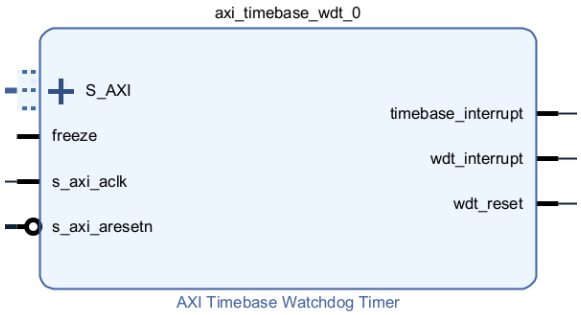


Figura 6.10. Implementación del watchdog.

DMA

El DMA se configuró con los siguientes parámetros:

- Modo *scatter/gather* deshabilitado.
- Registro de longitud del buffer con tamaño de 26 bits.
- Tamaño de la dirección de 32 bits.

- Canal de lectura deshabilitado.
- Canal de escritura con un ancho del bus de datos de 8 bits.
- Tamaño máximo de ráfaga de 256.

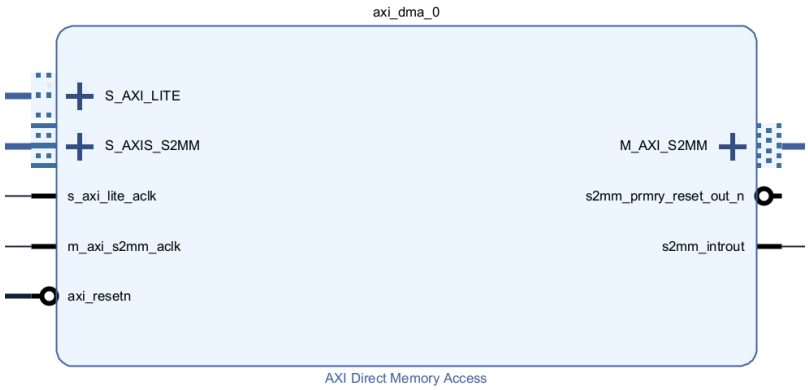


Figura 6.11. Implementación del DMA.

Controlador de interrupciones

La tarea asignada al controlador de interrupciones corresponde a realizar la gestión de las interrupciones de cada uno de los periféricos. Para esto es necesario realizar la siguiente configuración.

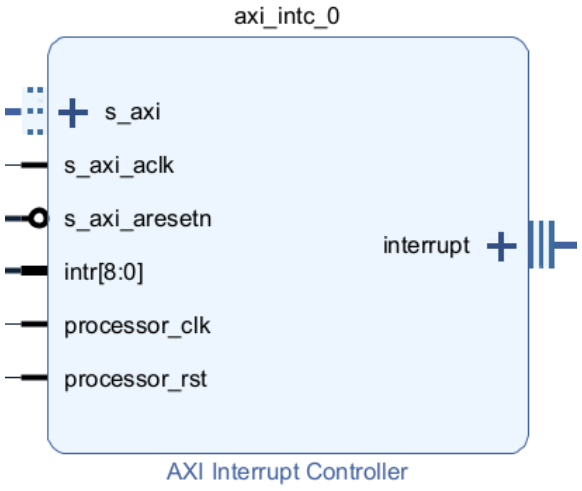


Figura 6.12. Implementación del controlador de interrupciones.

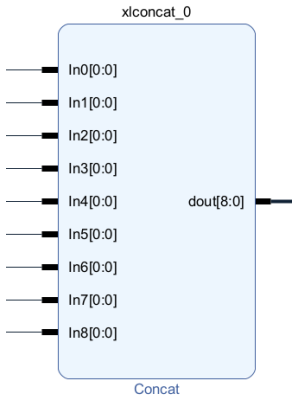


Figura 6.13. Concatenador de señales para el controlador de interrupciones.

Modulo secuenciador de reinicio (Processor system reset)

Este es el bloque encargado de llevar a cabo la secuencia de reinicio del sistema. Por cada dominio de reloj se implementa este bloque.

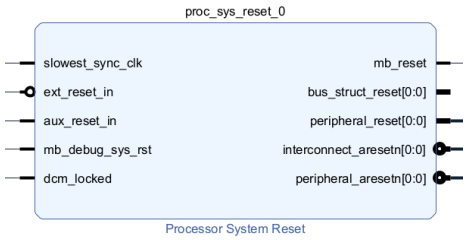


Figura 6.14. Implementación del módulo de reinicio.

Clock wizard

Es el bloque encargado de generar los relojes para el sensor CMOS, el SPI y el sistema.

Figura 6.15. Clocking wizard.

MDM

Es el módulo de depuración de MicroBlaze.

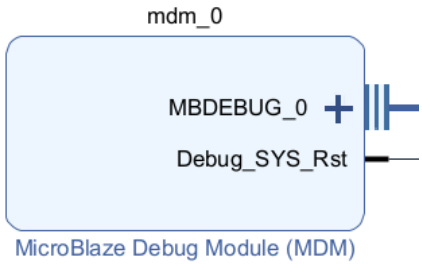


Figura 6.16. Módulo de depuración de MicroBlaze.

A fin de poder visualizar las imágenes y tener un medio de control para la computadora se desarrolló una interfaz gráfica con la cual se puedan visualizar datos.

Interrupciones

Se definieron nueve fuentes de interrupción en el sistema, a las cuales se les asignó la prioridad mostrada en la tabla 6.2. Esta asignación no sigue un modelo de prioridad para los componentes, teniendo como objetivo solamente el mostrar la funcionalidad del controlador de interrupciones al momento de implementarse.

Tabla 6.2. Prioridad de interrupciones

Prioridad	Periférico
0	WDT 0
1	WDT 1
2	DMA
3	UART Lite 0
4	Timer 0
5	Timer 1
6	GPIO 0
7	I2C 0
8	SPI 0

Asignación de direcciones de memoria

La asignación de la memoria se distribuye considerando a los componentes presentes en el sistema.

Tabla 6.3. Direcciones de memoria

Segmento	Tamaño	Dirección	Esclavo AXI	Periférico
Memoria de programa	128KB	0x0000 0000 0x0001 FFFF		
Memoria de datos	128KB	0x0000 0000 0x0001 FFFF		
	64KB	0x4000 0000 0x4000 FFFF	2	GPIO 0
	64KB	0x4001 0000 0x4001 FFFF	3	GPIO 1

	64KB	0x4002 0000 0x4002 FFFF	4	GPIO 2
	64KB	0x4003 0000 0x4003 FFFF	5	GPIO 3
	64KB	0x4100 0000 0x4100 FFFF	6	UART 0
	64KB	0x4200 0000 0x4200 FFFF	7	SPI 0
	64KB	0x4300 0000 0x4300 FFFF	8	I2C 0
	64KB	0x4400 0000 0x4400 FFFF	9	Timer 0
	64KB	0x4401 0000 0x4401 FFFF	10	Timer 1
	64KB	0x4500 0000 0x4500 FFFF	11	CMOS
	64KB	0x4600 0000 0x4600 FFFF	12	Watchdog
	64KB	0x4700 0000 0x4700 FFFF	13	DMA
	64KB	0x4800 0000 0x4800 FFFF	14	Controlador de interrupciones
Memoria adicional	64KB	0x8000 0000 0x9FFF FFFF	1	Memoria DDR3

Capítulo 7

Resultados

Una vez que se conectan los componentes se realiza la validación de las conexiones realizadas, a fin de identificar errores o advertencias. La validación del sistema se muestra en la figura y se comprueba que las conexiones fueron correctas.

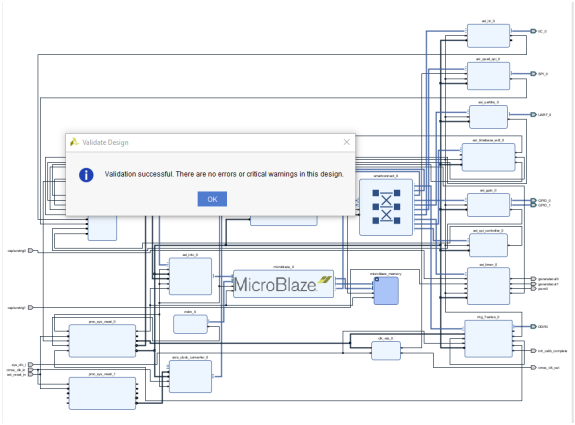


Figura 7.1. Validación de las conexiones sin errores.

El sistema para la prueba de envío de datos se muestra en la figura 7.3, este consiste del sensor de imagen, la tarjeta de desarrollo Nexys Video y Basys 3. La tarjeta Nexys Video alberga el sistema de procesamiento principal, en el cual reside el controlador de memoria. La basys se usó para contener al simulador de transferencias AXI4-Stream y posteriormente al controlador del sensor de imagen, está conectada de forma paralela a través de los puertos disponibles JA y JC de la Nexys Video. Esto se realizó así a fin de tener separado el controlador CMOS para propósitos de pruebas y depuración, evitando así la construcción de toda la arquitectura para solo

cambiar un módulo.

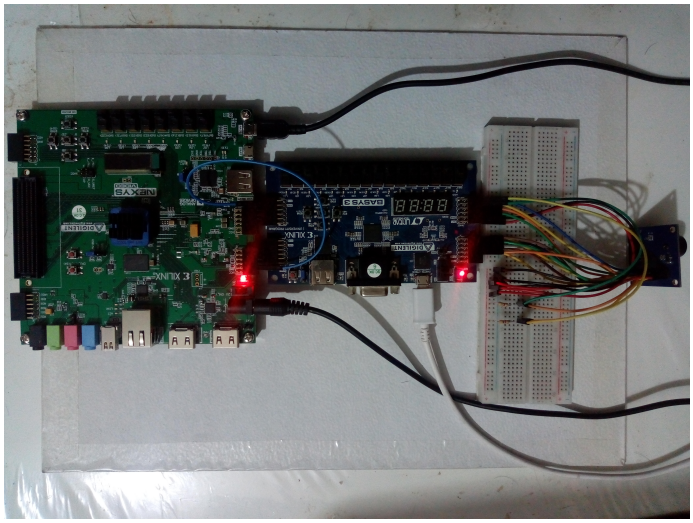


Figura 7.2. Sistema de prueba para el sensor de imagen.

7.0.1. Recursos lógicos y consumo energético

De acuerdo con los reportes obtenidos de la herramienta Vivado se tiene lo siguiente:

Utilization Post-Synthesis | **Post-Implementation**

Graph | **Table**

Resource	Utilization	Available	Utilization %
LUT	20166	133800	15.07
LUTRAM	2243	46200	4.85
FF	20753	269200	7.71
BRAM	18.50	365	5.07
IO	82	285	28.77
BUFG	8	32	25.00
MMCM	1	10	10.00
PLL	2	10	20.00

Power

Total On-Chip Power:	**1.076 W**
Junction Temperature:	**28,6 °C**
Thermal Margin:	56,4 °C (16,7 W)
Effective ϑJA:	3,3 °C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Implemented Power Report

Figura 7.3. Sistema de prueba para el sensor de imagen.

Para realizar una prueba de la transferencia de datos a memoria a través del DMA se creó un módulo de pruebas que genera transferencias AXI4-Streams.
El módulo simulador de transferencias emplea las siguientes señales:

- Señal de `Tvalid`
- Señal de `Tlast`
- Tamaño de `Tdata` de un byte
- Tkeep de un bit
- Tready por parte del esclavo
- Señal de reloj `clk` generada por el maestro.

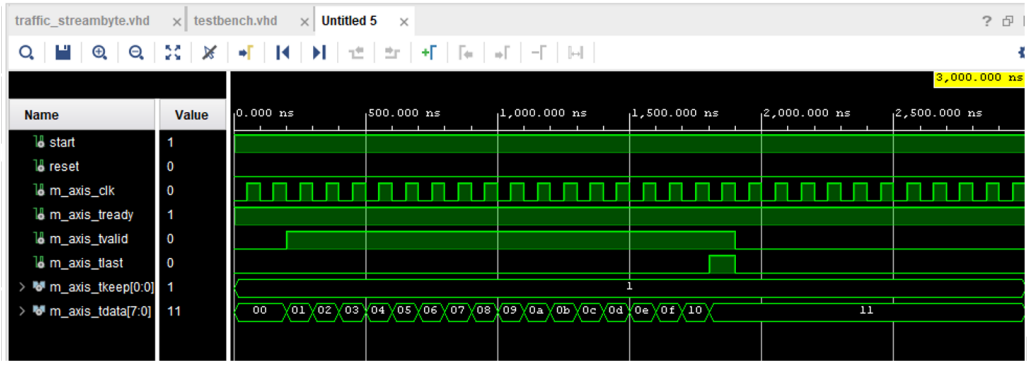


Figura 7.4. Simulación del funcionamiento del módulo de prueba para transferencias AXI4-Stream.

De acuerdo con la simulación presentada en la figura 7.4 la transmisión de datos comienza a partir de que se detecta la señal `Tvalid` en alto para un flanco de subida de reloj y finaliza una vez que se activa la señal `Tlast` en alto.

Este módulo de prueba se cargó en la tarjeta de desarrollo Basys 3 y se conectó a través del puerto de la tarjeta Nexys Video mostrado en la figura 7.5. La interfaz empleada en el diagrama de bloques de la computadora en Vivado se muestra en la figura 7.6. El reloj que se toma como de AXI4-Stream es `cmos_clk_in`, que es generado por medio del módulo maestro implementado en la Basys 3.

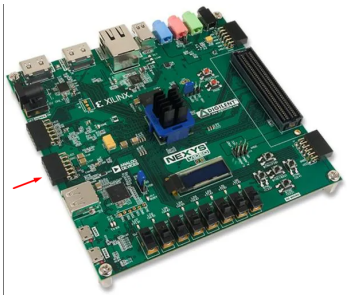


Figura 7.5. Puerto para la conexión de las señales.

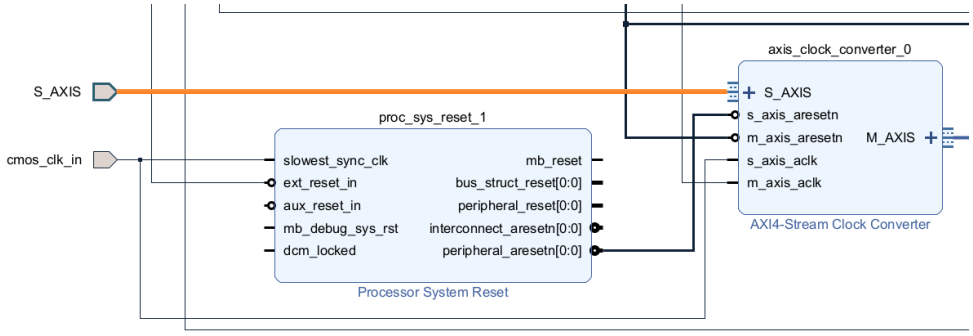


Figura 7.6. Interfaz AXI4-Stream en el diagrama de bloques.

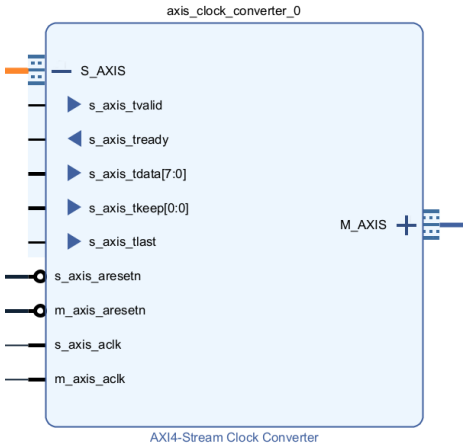


Figura 7.7. Señales AXI4-Stream del bloque AXI4-Stream clock converter.

Se creó un programa mediante el cual se realiza una transferencia de datos en AXI4-Stream, al leer los datos de memoria se comprueba que la secuencia de datos recibida concuerda con los datos producidos por el módulo simulador de transferencias AXI4-Stream, en este caso los primeros 4 bytes generados por el módulo corresponden a 0x01, 0x02, 0x03 y 0x04 de acuerdo con la figura 7.4. En la figura 7.8 se muestra que efectivamente esta secuencia se encuentra en la palabra de 4 bytes leída.

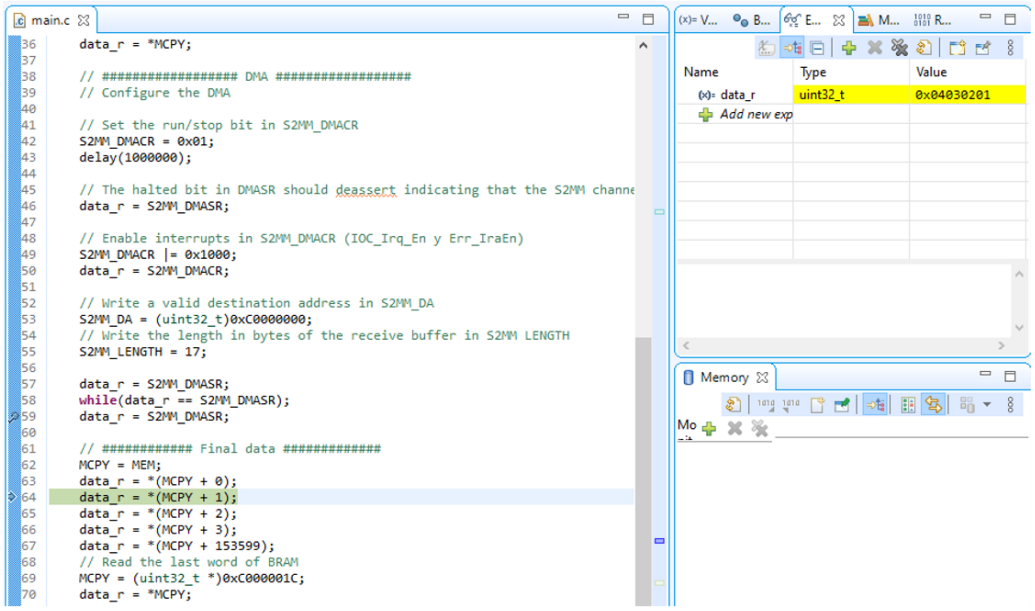


Figura 7.8. Simulación del funcionamiento del módulo de prueba para transferencias AXI4-Stream.

Para el funcionamiento del sensor de imagen, se configuró el módulo I2C para escribir a los registros de la cámara, posteriormente se configura el DMA y el UART, para una vez que se reciben los datos del sensor de imagen, mandarlos por el puerto UART de una computadora de escritorio hacia una interfaz gráfica. Todo el código empleado en la programación de los módulos se llevó a cabo a nivel de registros.

El módulo controlador de la cámara se integró posteriormente como parte de la arquitectura para no depender de la Basys 3, figura 7.9.

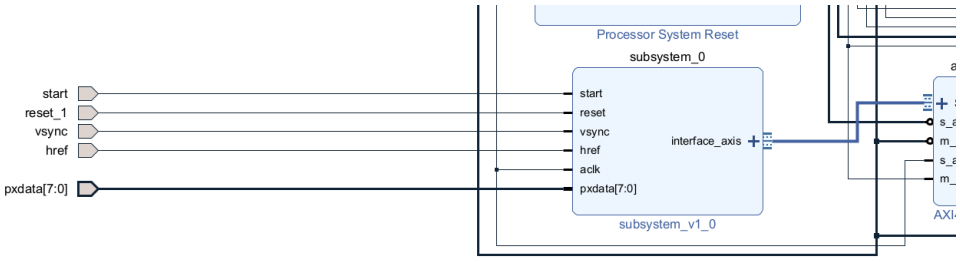


Figura 7.9. Controlador de la cámara implementado en la arquitectura.

Con un analizador de estados lógicos se comprueba que se transmiten las configuraciones por I2C al sensor de imagen.

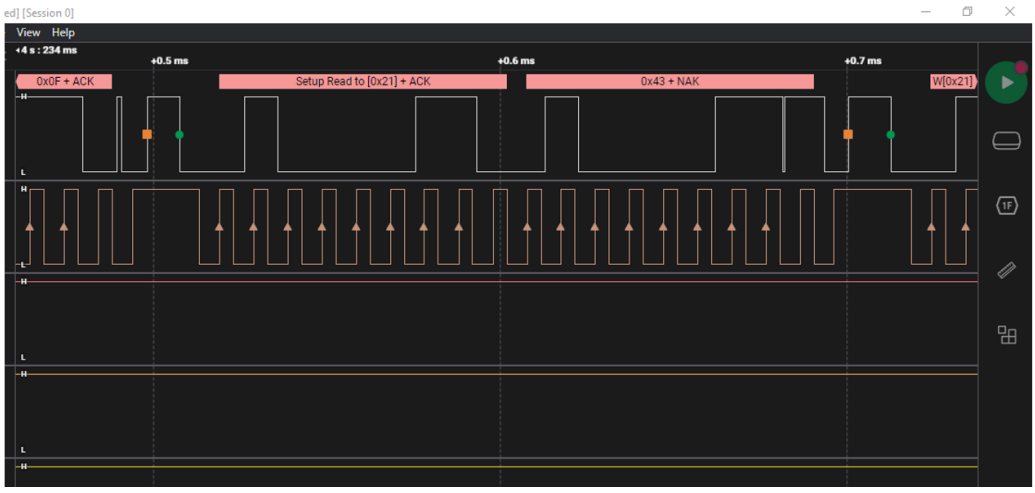


Figura 7.10. Muestra de la transferencia I2C para configurar el sensor de imagen Ov7670.

Captura de imágenes

Con la interfaz diseñada para la captura de imágenes se recibieron los datos del sensor a través de UART, mostrándose por medio de la imagen en el programa.

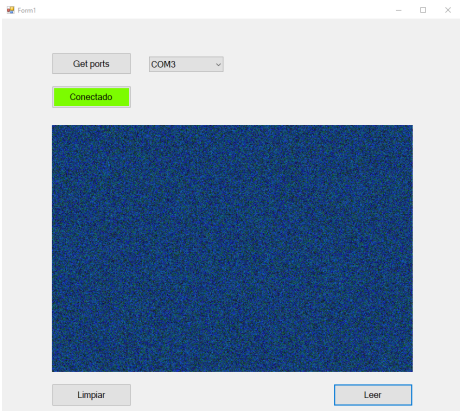


Figura 7.11. Sistema de prueba para el sensor de imagen.

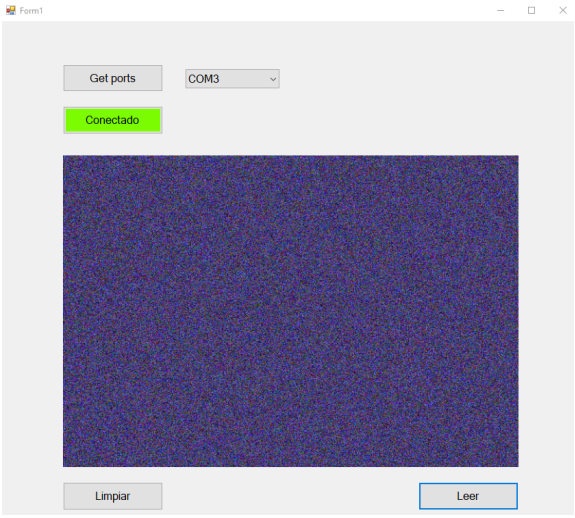


Figura 7.12. Sistema de prueba para el sensor de imagen.

Capítulo 8

Análisis de resultados

Los resultados preliminares obtenidos muestran en principio que los datos se pueden guardar en la memoria SDRAM, con la excepción de que para frecuencias mayores a 60 MHz se necesita emplear un conector y cableado que permita transferir datos a esa velocidad. La imagen no se logra ver través de la interfaz debido a que el formato en el cual se obtuvieron se dejó tal cual, por lo que hace falta procesamiento adicional de estos para poder visualizarse, además de que el sensor se probó en su configuración base, por lo cual necesita también ser configura para dar formato a los pixeles de salida. Al emplear un módulo de prueba para transferir datos por AXI4-Stream se verificó que estos se recibían a través del DMA hacia la memoria.

Capítulo 9

Conclusiones

El desarrollo de este trabajo se centró en la construcción de una arquitectura de computadora en un FPGA, mostrando el uso de bloques de propiedad intelectual como propuestas de solución para el sistema de procesamiento. La principal función de la computadora fue almacenar los datos de un sensor de imagen, empleando para ello una memoria de alta velocidad, en este caso una SDRAM DDR3, esto con la finalidad de tener un medio que tuviera una tasa de transferencia de datos lo suficientemente capaz para emplearse con un sensor como el MT9P031, el cual es capaz de requerir una velocidad de reloj de hasta 96 MHz y que cuente además con capacidad para almacenar la cantidad de información que se produce. El uso de los bloques IPs permitió extender la funcionalidad del sistema de procesamiento en el FPGA, agregando periféricos para que tuviera capacidad de comunicación con el mundo exterior y fuese capaz de transferir los datos almacenados en memoria. Con el uso del protocolo AXI4 en el sistema como interfaz de los periféricos se mostró que también existe la posibilidad de crear un bloque propio, el cual se adaptará a necesidades específicas de funcionamiento.

Se verificó el funcionamiento de la memoria SDRAM DDR3, de manera que se pudo comprobar que se podían guardar los datos en esta para posteriormente ser procesados o enviados. Como parte de investigación se encontró que los sistemas que funcionan en alta frecuencia necesitan de técnicas de diseño por medio de las cuales se puedan transferir los datos correctamente, involucrando procesos de sincronización con los datos que se reciben, a fin de reducir el efecto de *skew* en las señales, como el caso de la SDRAM DDR3, que emplea una secuencia de calibración para poder realizar una toma correcta de los datos. Para esto, un controlador como el que se empleó requiere de características especiales al momento de implementarse en un FPGA, haciendo uso de recursos especiales disponibles como parte de los recursos

lógicos del FPGA.

Otro aspecto que se encontró a tomar en cuenta es que cuando se tienen sistemas que funcionan a distintas frecuencias y deben de transferirse datos entre ellos requieren de emplear técnicas de cruce de dominios de reloj, siendo el uso de FIFOs asíncronas una solución adecuada para la transferencia de datos en alta velocidad de distintos dominios de reloj. El enrutamiento de las conexiónes en el hardware del FPGA se vuelve también fundamental en el diseño a fin de cumplir con los tiempos de propagación y evitar condiciones de metastabilidad.

Con la construcción e implementación de la arquitectura de la computadora queda en evidencia que es posible llevar a cabo la construcción de una computadora de a bordo para la captura de imágenes, a emplearse en un sistema de percepción remota. Adicionalmente, debido a las capacidades de potencia y tamaño, es factible su integración bajo un formato PC-104 para integrarse bajo el estándar CubeSat.

Capítulo 10

Trabajo a futuro

Como trabajo a futuro se plantea construir la interfaz para el sensor de imagen MT9P031, a manera de transferir a la máxima tasa de datos, empleando para esto el conector FMC de la tarjeta Nexys Video. El diseño de la interfaz es requerido debido a que la cantidad de pines de los cuales se disponen en la tarjeta son limitados y además son de propósito general, es decir, no están destinados a ser usados para transferir datos a máxima velocidad.

Por otra parte, teniendo la implementación de la arquitectura de la computadora es posible llevar a cabo un análisis de los principales puntos de falla, a manera de aplicar alguna técnica de tolerancia a fallas a nivel de software y principalmente a nivel de hardware empleando características de las FPGA como la reconfiguración parcial, así mismo, es posible realizar también inyección de fallas en la memoria de configuración del FPGA, a fin de evaluar su desempeño frente a condiciones de eventos SEE ocasionados por la radiación espacial.

Apéndice A

Software

A.1. Programa principal en C para la computadora

Se presenta el código fuente en C del programa principal empleado como prueba de control de la computadora. Debido a la extensión de los códigos empleados para cada una de las funciones no se incluyeron como apéndice de este trabajo, sin embargo, pueden ser consultadas en este repositorio [Archivos].

```
/*
 *
 *
 * Autor: Israel Pérez Vicente
 * Institución: Universidad Nacional Autónoma de México
 *        Facultad de Ingeniería
 *
 * Este código corresponde al programa de prueba desarrollado
 * para emplearse en el trabajo de tesis "Diseño de una com-
 * putadora de a bordo en FPGA para un sistema de percepción
 * remota de imágenes de un satélite bajo el estándar CubeSat"
 *
 *
 * */

#include <stdint.h>
#include <string.h>

#include "mb_interface.h"
#include "axigpio.h"
#include "axiuartlite.h"
#include "axispi.h"
#include "axiiic.h"
```

```
24 #include "timer.h"
25 #include "dma.h"
26 #include "axiintc.h"
27
28 #include "Ov7670.h"
29 #include "oled.h"
30 #include "utils.h"
31
32 // Tamaño en bytes de la imagen
33 #define FRAME_BYTE_SIZE 614400
34 // Número de columnas del OLED
35 #define OLED_NUM_COLS 127
36 // Tamaño de transferencia para barra de progreso
37 #define FRAME_CHUNCK  (int)FRAME_BYTE_SIZE/OLED_NUM_COLS
38
39 /* Declaración de las funciones */
40 void initCom(void);
41 void processFrame(void);
42 void processTimeout(void);
43 void sendDataFrame(uint32_t address);
44
45 /*  Variables */
46 uint8_t countLeds;
47 double time = 0, timeA = 0;
48
49 /* Función principal */
50 void main()
51 {
52   initCom();
53   while(1)
54   {
55     processFrame();
56     processTimeout();
57   }
58 }
59
60 /* Función que se encarga de la inicialización
61  * de los periféricos y configuración  inicial
62  * de los módulos empleados.
63  */
64 void initCom(void)
65 {
66   // Inicializa el watchdog
67   // T_WDT = (2^n)/FREQ_SYS
68   // (2^28)/48 MHz = 5.59 s
69   wdtSetup(28);
```

```
70
71  // Habilita el timer 1 para los retardos
72  setInterrupt(TIMER1_INT, intHandlerTimer1);
73  microblaze_enable_interrupts();
74
75  // UART
76  enableUart0();
77  setInterrupt(UARTL0_INT, intHandlerUart0);
78
79  // DMA
80  initDMA();
81  setInterrupt(DMA_INT, dmaIntHandler);
82
83  // Habilita I2C
84  initI2C0();
85
86  // Reinicia el módulo Ov7670
87  writeRegister(0x12, 0x80);
88  // Retardo de 100 ms
89  delay(100);
90
91  // Establece la configuración inicial del módulo Ov7670
92  setRegisters(regsConfig);
93
94  // Inicia el SPI
95  initSPI();
96
97  // Enciende y configura el display OLED
98  displayOn();
99  displayConfig();
100 clearDisplay();
101
102 // Habilita la interrupción del timer 0
103 setInterrupt(TIMER0_INT, intHandlerTimer0);
104 writeGpio2(0x09);
105
106 // Habilita la interrupción del WDT
107 setInterrupt(WDT_INT2, wdtIntHandler2);
108
109 // Establece en 250ms el periodo del timer 0
110 setTimer0ReloadMode(12000000);
111
112 // Imprime mensaje
113 displayPrint(0, 0, "Sistema listo\n");
114
115 // Limpia bandera para evitar reinicio por watchdog
```

```
116   clearResetWDT();
117 }
118
119 /* Función que procesa el envío de los datos de imagen.
120  * Por medio de la interrupción del DMA se actualiza
121  * el estado de la transferencia, con lo cual se evalua
122  * la acción a ejecutar.
123  */
124 void processFrame(void)
125 {
126   switch(transferDMAstatus())
127   {
128   case WAITING_TRANSFER:
129     break;
130   case TRANSFER_COMPLETE:
131     // Mensaje de transferencia correcta y envío
132     clearDisplay();
133     displayPrint(0, 0, "Transferencia\0");
134     displayPrintTemp(1, 0, "completeta!\0", 1000);
135     clearDisplay();
136     displayPrint(0, 0, "Enviando ...\0");
137
138     // Envia los datos de imagen
139     sendDataFrame(MEMDDR3);
140
141     // Sistema listo
142     clearDisplay();
143     clearTransferStatusDMA();
144     displayPrint(0, 0, "Sistema listo\0");
145     break;
146   case TRANSFER_ERROR:
147     // Mensaje de error en la captura
148     clearDisplay();
149     displayPrint(0, 0, "Error en la\0");
150     displayPrintTemp(1, 0, "transferencia!\0", 1000);
151
152     // Reinicia el DMA
153     initDMA();
154
155     // Sistema listo
156     displayPrint(0, 0, "Sistema listo\0");
157     break;
158   }
159 }
160
161 /* Conmuta el estado de los leds para indicar
```

```
162  * que el sistema se encuentra operativo.
163  * Se lee el puerto 1 y su valor se conmuta
164  * hacia el puerto 2 al ocurrir dos eventos
165  * de fin de cuenta del timer 0.
166  */
167 void processTimeout(void)
168 {
169   if (timeOutTimer0())
170   {
171     // 2 cuentas = 500ms para los Leds
172     if (countLeds == 2)
173     {
174       countLeds = 0;
175       writeGpio2(readGpio1() ^ 0xFF);
176     }
177     else
178     {
179       countLeds++;
180     }
181     restartTimer0();
182   }
183 }
184
185 void sendDataFrame(uint32_t address)
186 {
187   // Variables
188   uint32_t i, chunck = 0;
189   uint8_t *MEM = (uint8_t *)address;
190   uint8_t data = 0;
191
192   // Dirección inicial para imprimir la barra de progreso
193   setDisplayAddress(1, 0);
194
195   // Lee los datos de memoria y los transfiere
196   for (i = 0; i < FRAME_BYTE_SIZE; i++)
197   {
198     // Lee el contenido del puntero
199     data = *(MEM + i);
200
201     // Transmite el dato por UART
202     transmitByte(data);
203
204     // Imprime una barra de progreso en el display OLED
205     if (chunck >= FRAME_CHUNCK)
206     {
207       displayWrite(0xFF);
```

```
208         chunck = 0;
209
210         // Limpia bandera para evitar reinicio por watchdog
211         clearResetWDT();
212       }
213       else
214       {
215         chunck++;
216       }
217     }
218
219     // Mensaje de completado
220     clearDisplay();
221     displayPrint(0, 0, "Completado\0");
222     displayPrintTemp(1, 0, "100 %", 2000);
223
224     // Limpia bandera para evitar reinicio por watchdog
225     clearResetWDT();
226 }
```

Programa principal en C para el SPRI

A.2. Código en VHDL para el transmisor de pruebas

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3   use ieee.std_logic_unsigned.all;
4
5   entity streambytegen is
6   port (
7   start           : in std_logic; -- Inicio de la transaccion
8   -- axi stream ports
9   m_axis_clk      : in  std_logic;
10  m_axis_tdata    : out std_logic_vector(7 downto 0);
11  m_axis_tkeep    : out std_logic_vector(0 downto 0);
12  m_axis_tlast    : out std_logic;
13  m_axis_tready   : in  std_logic;
14  m_axis_tvalid   : out std_logic
15  );
16  end streambytegen;
17
18  architecture rtl of streambytegen is
```

```
19
20  -- Estados
21  type    state is (IDLE, SEND_STREAM, STOP);
22  signal  sm_state : state := IDLE;
23
24  -- Seniales AXI Stream
25  signal tvalid       : std_logic := '0';
26  signal tlast        : std_logic := '0';
27  signal data           : std_logic_vector(7 downto 0) := (others => '0
    ');
28  signal packet_len_cnt : std_logic_vector(19 downto 0)  := (others =>
    '0');
29
30  begin
31  -- I/O conexiones
32  m_axis_tkeep(0) <= '1';
33  m_axis_tvalid   <= tvalid;
34  m_axis_tlast    <= tlast;
35  data            <= packet_len_cnt(7 downto 0);
36  m_axis_tdata    <= data;
37
38  -- Maquina de estados
39  sm_pr : process(m_axis_clk)
40  begin
41  if (rising_edge (m_axis_clk)) then
42  case (sm_state) is
43  when IDLE =>
44  tvalid            <= '0';
45  tlast             <= '0';
46  packet_len_cnt    <= (others => '0');
47  -- Comienza el envio
48  if (start = '1') then
49  sm_state <= SEND_STREAM;
50  end if;
51  when SEND_STREAM  =>
52  tvalid            <= '1';
53  packet_len_cnt    <= packet_len_cnt + 1;
54  if (packet_len_cnt = x"98000") then
55  tlast           <= '1';
56  sm_state        <= STOP;
57  end if;
58  when STOP =>
59  tvalid <= '0';
60  tlast  <= '0';
61  sm_state <= STOP;
62  when others =>
```

```
63  sm_state <= IDLE;
64  end case;
65  end if;
66  end process sm_pr;
67  end rtl;
68
69
```

A.3. Código en VHDL para el capturador de imagen

El siguiente código corresponde a la descripción de hardware del módulo capturador de imagen empleado.

```
1  ----------------------------------------------------------------------------
2  ------------------------------ Habilitador de captura
     ---------------------------------
3  library ieee;
4  use ieee.std_logic_1164.all;
5
6  entity framenable is
7  port
8  (
9  start, reset, vsync, clk: in std_logic;
10 enable_out: out std_logic:='0';
11 -- Seniales de depuracion
12 status: out std_logic_vector(3 downto 0)
13 );
14 end framenable;
15
16 architecture rtl of framenable is
17
18 -- Seniales para la maquina de estados
19 type state is (IDLE, WAITVS, CAPTURE, STOP);
20 signal sm_state: state:= IDLE;
21 signal next_state: state:= IDLE;
22
23 signal ffd1, ffd2: std_logic := '0';
24
25 begin
26 FF1: process(clk)
27 begin
28 if rising_Edge(clk) then
29 ffd1 <= vsync;
30 end if;
```

```
31  end process;
32
33  FF2: process(clk)
34  begin
35  if rising_Edge(clk) then
36  ffd2 <= ffd1;
37  end if;
38  end process;
39
40  -- Cambio de estado o reset
41  process(clk) is
42  begin
43  if falling_edge(clk) then
44  if (reset = '0') then
45  sm_state <= IDLE;
46  else
47  sm_state <= next_state;
48  end if;
49  end if;
50  end process;
51
52  -- Proceso combinacional de las entradas y el estado actual
53  process(sm_state, start, ffd1, ffd2) is
54  begin
55  --next_state <= sm_state;
56  case (sm_state) is
57  when IDLE=>
58  if (start = '1') then
59  next_state <= WAITVS;
60  else
61  next_state <= IDLE;
62  end if;
63  when WAITVS =>
64  if (ffd1 = '0' and ffd2 = '1') then -- Detecta falling edge
65  next_state <= CAPTURE;
66  else
67  next_state <= WAITVS;
68  end if;
69  when CAPTURE =>
70  if (ffd1 = '1' and ffd2 = '0') then -- Detecta rising edge
71  next_state <= STOP;
72  else
73  next_state <= CAPTURE;
74  end if;
75  when STOP =>
76  next_state <= STOP;
```

```
77  when others=>
78  next_state <= IDLE;
79  end case;
80  end process;
81
82  -- Proceso combinacional del cambio de estado
83  process(sm_state) is
84  begin
85  case sm_state is
86  when WAITVS =>
87  enable_out <= '0';
88  status <= x"A";
89  when CAPTURE =>
90  enable_out <= '1';
91  status <= x"5";
92  when STOP =>
93  enable_out <= '0';
94  status <= x"2";
95  when others =>
96  enable_out <= '0';
97  status <= x"8";
98  end case;
99  end process;
100
101 end rtl;
102
103 ------------------------------------------------------------------------------
104 ---------------------------------- Sistema
      ------------------------------------
105 library ieee;
106 use ieee.std_logic_1164.all;
107
108 entity cmos_module_v1 is
109 port
110 (
111 -- Seniales de entrada
112 start, reset, vsync, href, clk: in std_logic;
113 pxdata: in std_logic_vector(7 downto 0);
114 -- Seniales AXI4 stream
115 Tready: in std_logic;
116 Tvalid, Tlast, Tkeep: out std_logic;
117 Tdata: out std_logic_vector(7 downto 0);
118 -- Seniales de depuracion
119 status: out std_logic_vector(3 downto 0)
120 );
121 end cmos_module_v1;
```

```
122
123 architecture structural of cmos_module_v1 is
124
125 component framenable is
126 port
127 (
128 start, reset, vsync, clk: in std_logic;
129 enable_out: out std_logic;
130 -- Seniales de depuracion
131 status: out std_logic_vector(3 downto 0)
132 );
133 end component;
134
135 signal enable_s: std_logic;
136 signal reg_A, reg_B: std_logic_vector(9 downto 0) := "0000000000";
137
138 begin
139
140 -- I/O asignacion
141 Tvalid <= reg_B(9);
142 Tlast <= reg_B(8);
143 Tkeep <= reg_B(9);
144 Tdata <= reg_B(7 downto 0);
145
146 -- Captura href, vsync, flip y pxdata en un flanco de subida
147 CPT1: process(clk)
148 begin
149 if rising_edge(clk) then
150 reg_A(9) <= (href or vsync) and enable_s;
151 reg_A(8) <= vsync and enable_s;
152 reg_A(7 downto 0) <= pxdata;
153 end if;
154 end process;
155
156 -- Recaptura en flanco de bajada para generar las salidas a AXI
157 CPT2: process(clk)
158 begin
159 if falling_edge(clk) then
160 reg_B(9) <= reg_A(9);
161 reg_B(8) <= reg_A(8);
162 reg_B(7 downto 0) <= reg_A(7 downto 0);
163 end if;
164 end process;
165
166 UNIT1: framenable port map
167 (
```

```
168   start => start,
169   reset => reset,
170   vsync => vsync,
171   clk => clk,
172   enable_out => enable_s,
173   status => status
174   );
175   end structural;
176
```

Bibliografía

[1] Erik Kulu. Nanosats database. https://www.nanosats.eu/figures.

[2] Juan Andres Perez Celis, Saul de la Rosa Nieves, Carlos Romo Fuentes, Saul Daniel Santillan Gutierrez, and Alvar Saenz-Otero. Methodology for designing highly reliable fault tolerance space systems based on cots devices. In *2013 IEEE International Systems Conference (SysCon)*, pages 591–594, 2013.

[3] Jose Alberto Ramirez Aguilar, Saul De La Rosa Nieves, Emilio Sanchez Medina, Saul Daniel Santillan Gutierrez, Carlos Romo Fuentes, Jorge Alfredo Ferrer Perez, and Oleg M. Brekhov. Satellite condor unam-mai: Technical scientific cooperation. In *2013 6th International Conference on Recent Advances in Space Technologies (RAST)*, pages 1037–1040, 2013.

[4] F. Siegle, Tanya Vladimirova, J. Ilstad, and O. Emam. Mitigation of Radiation Effects in SRAM-Based FPGAs for Space Applications. 2 2016.

[5] Melanie Berg and Kenneth LaBel. Introduction to fpga devices and the challenges for critical application-a user's perspective. In *Hardened Electronics and Radiation Technology (HEART) 2015 Conference*, number GSFC-E-DAA-TN22657, 2015.

[6] Ensafatef. Satellite subsystems. https://hackaday.io/project/166568-a-cubesat-comm-design-for-in-space-assembly/details.

[7] Gabor Holtzer. How does a camera work? https://expertphotography.com/how-does-a-camera-work/.

[8] Onsemi. *1/2.5-Inch 5 Mp CMOS Digital Image Sensor*, 2006. Rev. 11.

[9] J. L. Hennessy. *Computer architecture: a quantitative approach*. Morgan Kaufmann, Cambridge, 2019.

[10] Bruce Jacob, Spencer W. Ng, and David T. Wang. *Memory Systems*. Morgan Kaufmann, San Francisco, 2008.

[11] Micron Technology. *4Gb: x4, x8, x16 DDR3L SDRAM Description*, 2011. Rev. I 9/13 EN.

[12] ARM. *AMBA 4 AXI4-Stream Protocol*, 2010. Rev. 1.0.

[13] Kristian Rasmus Peter Thomas, Allan. Designing on board computer and payload for the aau cubesat. 2002.

[14] Steiner Choueiri George Roethlisberger Scheidegger Peter-Contesse Noca, Jordan and Borgeaud. Lessons learned from the first swiss pico-satellite: Swisscube. In *23nd Annual AIAA/USU Conference on Small Satellites*, 2009.

[15] Vladimír Dániel, P. Svoboda, M. Junas, M. Sabol, J. Cagáň, Matěj Stejskal, J. Dudas, L. Mikulickova, A. Marek, R. Pavlica, P. Sedláčková, and David Jech. Development of cubesat with cots camera enabling eo with high gsd. page 31, 09 2020.

[16] Tumnanosat. https://nanosat.utm.md/.

[17] ucam-ii. https://nz.element14.com/4d-systems/ucam-ii/micro-cam-module-4d-display-module/dp/2451220.

[18] Paula do Vale Pereira, Maddie Garcia, Madeleine Schroeder, Humberto Caldelas, Charles Lindsay, Alex Choi, Kaila Pfrang, Amelia Gagnon, Patrick Mckeen, Jim Clark, Thomas Murphy, Christian Haughwout, and Kerri Cahoy. Beavercube: Coastal imaging with vis/lwir cubesats. 08 2020.

[19] mvbluefox. https://www.matrix-vision.com/en/products/MP11708989.

[20] Ball Swanson Saive, Droog and Chao. Design of an imaging payload for earth observation from a nanosatellite. In *35th Annual Small Satellite Conference*, 2021.

[21] Im-200. https://www.aac-clyde.space/what-we-do/space-products-components/payloads/im200.

[22] C3d. https://www.xcam.co.uk/c3d-cubesat-camera.

[23] Ecam-ir1. https://www.satnow.com/products/satellite-cameras/malin-space-science-systems/113-1293-ecam-ir1.

[24] Spectra cam. https://redwirespace.com/products/spectracam/.

[25] hyperscape100. https://satsearch.co/products/simera-sense-hyperscape100.

[26] Chamaleon imager. https://dragonflyaerospace.com/products/chameleon/.

[27] Digilent. *Nexys Video FPGA Board Reference Manual*, 2020. Rev. A.

[28] Digilent. *Basys 3 FPGA Board Reference Manual*, 2019. Rev 1.0.

[29] OmniVision. *CMOS VGA (640x480) OV7670/OV7171*, 2006. Rev. 1.4.

[30] AMD/Xilinx. *MicroBlaze Processor Reference*, 2018. v2018.2.

[31] Xilinx/AMD. *AXI GPIO v2.0 LogiCORE IP Product Guide*, 2016. Rev. 2.0.

[32] Xilinx/AMD. *AXI UART Lite v2.0 LogiCORE IP Product Guide*, 2017. Rev. 2.0.

[33] Xilinx/AMD. *AXI Quad SPI v3.2 LogiCORE IP Product Guide*, 2022. Rev. 3.2.

[34] Xilinx/AMD. *AXI IIC Bus Interface v2.1 LogiCORE IP Product Guide*, 2021. Rev. 2.1.

[35] Xilinx/AMD. *AXI Timer v2.0 LogiCORE IP Product Guide*, 2016. Rev. 2.0.

[36] Xilinx/AMD. *AXI Timebase Watchdog Timer v3.0 LogiCORE IP Product Guide*, 2017. Rev. 3.0.

[37] Xilinx/AMD. *AXI Interrupt Controller (INTC) v4.1 LogiCORE IP Product Guide*, 2021. Rev. 4.1.

[38] Xilinx/AMD. *LogiCORE IP Concat (v2.1)*, 2016. Rev. 1.0.

[39] Xilinx/AMD. *AXI DMA v7.1 LogiCORE IP Product Guide*, 2022. Rev. 7.1.

[40] Xilinx/AMD. *Zynq-7000 SoC and 7 Series Devices Memory Interface Solutions v4.2 User Guide*, 2018. Rev. 4.2.

[41] Xilinx/AMD. *MicroBlaze Debug Module v3.2 Logicore IP Product Guide*, 2021. Rev. 3.2.