UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

PROGRAMA DE MAESTRÍA Y DOCTORADO EN INGENIERÍA
ELÉCTRICA – INSTRUMENTACIÓN

PROPUESTA DE UN ESQUEMA DE INSTRUMENTACIÓN PARA EL SUBSISTEMA HOUSEKEEPING DEL TELESCOPIO ESPACIAL JEM-EUSO Y ANÁLISIS DE VIABILIDAD PARA SU IMPLEMENTACIÓN EN UN FPGA VIRTEX-5

TESIS
QUE PARA OPTAR POR EL GRADO DE:
MAESTRA EN INGENIERÍA

PRESENTA:
NELLY AGLAÉ GÓMEZ BENÍTEZ

TUTOR PRINCIPAL
DR. GUSTAVO ADOLFO MEDINA TANCO
INSTITUTO DE CIENCIAS NUCLEARES, UNAM

MÉXICO, D. F. DICIEMBRE 2014

JURADO ASIGNADO:

Presidente:	Dr. De la Rosa Nieves Saúl
Secretario:	Dr. Vicente Vivas Esau
Vocal:	Dr. Medina Tanco Gustavo Adolfo
1 er. Suplente:	Dr. Mendoza Bárcenas Mario Alberto
2 d o. Suplente:	Dr. Arias Estrada Miguel Octavio

Lugar o lugares donde se realizó la tesis: Centro de Ciencias Aplicadas y Desarrollo Tecnológico, UNAM.

TUTOR DE TESIS:

Gustavo Adolfo Medina Tanco

FIRMA

Dedicatoria

A mi madre y a mi padre, a Dios y a nuestra Santísima Madre.

"Aunque tuviera el don de la profecía y conociera todos los misterios y toda la ciencia, aunque tuviera toda la fe, una fe capaz de trasladar montañas, si no tengo amor, no soy nada"

(Cor1. 13,2)

Agradecimientos

*A **Dios** por su amor infinito, por estar conmigo a pesar de todo, por haberme dado esta oportunidad y por haberme dado la capacidad de realizar este trabajo. Te amo.*

*A la mi madrecita del cielo la **Virgen Santísima**, por consolarme en los momentos más difíciles, por su ternura y su apoyo en estos dos años y en mi vida. Te amo.*

*A **Patricio** mi padre, por todos sus consejos y por oírme siempre. A **Paula** mi hermosa madre por su cálido amor que me levantó en los momentos más difíciles. A mi hermana **Gabriela** por hacerme reír siempre y contagiarme su alegría. A mi hermano **Victor Hugo** por su amor, sus chistes, por mostrarme que la vida es sencilla y alegre. A mi hermanita **Viridiana** por darme un amor tan puro, por todas las obras de arte que me han alegrado, por su detalles de amor. Gracias por estar conmigo siempre, LOS AMO.*

Al Dr. Gustavo por darme la oportunidad de participar en este proyecto tan interesante y emocionante, por su guía y por su apoyo.

Al Dr. Mario Alberto, por su gran apoyo, por todos sus consejos, por su amistad, no lo habría logrado sin tu ayuda :)

*A **Fray Juan** por su amistad y por el gran apoyo espiritual. A mis amigas misioneras: **Lorella, Giuliana, Rosie, Nucha, Regina** y **Nadia** por animarme siempre y acompañarme en el camino. A mis amigos y compañeros **Adrián**, **César**, **Sócrates**, **Fidel**, **Greta**, **William**, **Óscar**, **Dany**, **Javis**, **Gerardo** y **Martín**, y a todos mis amigos de Morelia que se han preocupado por mí durante todo este tiempo, gracias por todo su apoyo.*

*A **Javi**, por su amistad, y por animarme en momentos difíciles.*

*A **Juan Carlos** por su inestimable apoyo, por su paciencia, por su amistad y por confiar siempre en mí.*

A mis sinodales, y a mis profesores, por compartir sus conocimientos conmigo.

Al CONACyT a traves de Red CyTE, Red FAE, CB-SEP, PAPIIT-UNAM, AEB y al ICN-UNAM.

¡Gracias a todos!

Contenido

Capítulo 1
Antecedentes

En este capítulo se proporciona una descripción general del proyecto, de modo que el lector pueda identificar el origen del problema que ocupa esta tesis, hasta familiarizarse con algunos términos que se utilizarán durante el desarrollo de este trabajo.

1.1 JEM-EUSO

Los rayos cósmicos cubren un amplio rango de energías que va desde 10^9 hasta 3 X 10^{20} eV. Los rayos cuya energía se encuentra arriba de 10^{19} eV, se considera que son rayos cósmicos de ultra alta energía y de estos rayos se han detectado muy pocos, incluso se sabe que existen rayos cósmicos más energéticos pero aún no se han podido detectar ni analizar. Esto se debe a que el flujo de rayos cósmicos a las más altas energías actualmente es de una partícula por cada 1000 km^2 al año, por lo que el área efectiva de detección es un parámetro crítico para poder estudiarlos y hasta ahora, la mayor área de detección disponible es la del Observatorio Pierre Auger (OA) ubicado en Malargüe, Argentina, el cual posee un área de detección de 3000 km^2. En la práctica, es poco factible construir detectores con área mucho mayor que la del OA sobre la superficie terrestre, como para explorar adecuadamente energías mayores que $10^{19.8}$ eV.

Por este motivo, se ha proyectado el desarrollo del telescopio espacial JEM-EUSO (*Extreme Universe Space Observatory on the Japanese Experiment Module*), el cual utilizará la atmósfera terrestre como un detector gigante de rayos cósmicos de alta energía y que se instalará en la Estación Espacial Internacional (ISS). De este modo, a diferencia del detector terrestre más grande, tendrá un área proyectada de más de 1 millón de km^2 y será capaz de detectar rayos cósmicos de más de 10^{20}eV.

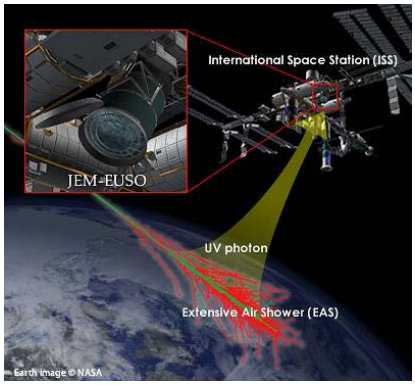


Figura 1.1. Imagen artística del observatorio JEM-EUSO en la Estación Espacial Internacional [1].

JEM-EUSO observará estas partículas por medio de los fenómenos luminosos que se generan cuando estas partículas cruzan la atmósfera. Lo anterior sucede debido al fenómeno de las cascadas atmosféricas extensas (EAS Extensive Air Shower), que consiste en la producción de una cascada de partículas secundarias cuando una partícula de energía extrema, procedente del

espacio, choca con un núcleo de la atmósfera. Las partículas cargadas, excitan al N_2 atmosférico el cual emite por fluorescencia radiación ultravioleta (UV) cuya longitud de onda está entre 300 y 400 nm. JEM-EUSO detectará estas emisiones y adquirirá imágenes de la fluorescencia UV durante varios microsegundos, para reproducir un mapa tridimensional de las trazas que dejan las EAS a su paso y eventualmente, determinar la dirección de incidencia de la partícula primaria, su energía y el perfil espacial de la deposición de energía en la atmósfera, que codifica información indirecta sobre la identidad del primario [12]

[13].

El telescopio JEM-EUSO será lanzado en el año 2018 y se planea que tenga una vida aproximada de 3 a 5 años, orbitará cada 90 minutos alrededor de la tierra a una altura entre 300-400 km a bordo de la Estación Espacial Internacional y tendrá un peso aproximado de dos toneladas.

Este proyecto liderado por la agencia espacial de Japón (JAXA) y la agencia espacial Rusa (ROSCOSMOS), cuenta con la colaboración de las agencias espaciales de Europa (ESA) y Estados Unidos (NASA) entre otras, además de contar con 75 grupos de investigación de 16 países entre los cuales se tiene una participación activa de México a través del Instituto de Ciencias Nucleares (ICN) de la UNAM. México es el país responsable de la construcción de dos sistemas diferentes para la caracterización y verificación tanto de la superficie óptica y los algoritmos de disparo, así como del sistema óptico del telescopio como un todo; además, también es responsable del diseño, construcción e implementación del sistema de monitoreo del telescopio, del cual se hablará en el siguiente apartado.

1.2 Housekeeping

La principal participación de México a través del Instituto de Ciencias Nucleares (ICN) de la UNAM a nivel del desarrollo de hardware y software dentro del proyecto JEM-EUSO, consiste en el desarrollo de un subsistema de adquisición y procesamiento de datos, así como de generación de comandos de alarma y control denominado *Housekeeping* (HK), el cual interactúa con todos los demás subsistemas del telescopio JEM-EUSO y cuyas tareas son el monitoreo de todos los subsistemas que lo componen y la atención ante la ocurrencia de fallas en alguno de ellos.

1.2.1 Interacción del Housekeeping con los subsistemas del JEM-EUSO

El diseño del HK debe satisfacer los requerimientos de interfaces físicas y de comunicaciones con el resto de los subsistemas asociados, que son siete; por lo que la descripción de cada uno de ellos y su interacción con el HK se aborda en los siguientes párrafos.

1.2.1.1 PDM

Las siglas de este subsistema corresponden al módulo foto detector (PDM: *Photo Detector Module*) y es el subsistema que conforma la superficie detectora del telescopio JEM-EUSO. En total se tendrán 140 tarjetas PDM y a su vez, cada una de estas estará formada por 9 células elementales (EC: *Elementary Cell*) y 36 Tubos fotomultiplicadores multi-ánodo (MAPT: *Multi Anode Photo Multipliers Tubes*), que son los sensores que se pueden observar en la Figura 1.2. La

interacción de este subsistema con el HK se lleva a cabo mediante el protocolo SPI diferencial, mediante el cual se monitorean datos de temperaturas, voltajes y corrientes.

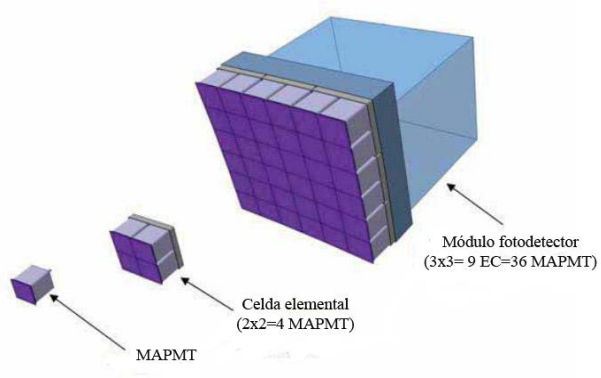


Figura 1.2. Estructura del subsistema PDM.

1.2.1.2 CCB

Este subsistema se encarga del análisis de datos provenientes del subsistema PDM, así como de la discriminación de los datos que no aportan información relevante. Se le llama tarjeta de control de grupo (CCB: *Control Cluster Board)*, dado que cada CCB está conectada a grupos de PDMs. La función del HK es la de monitorear los datos de voltajes, temperaturas y corrientes proporcionados por esta tarjeta a través del protocolo SPI diferencial**.** Para el JEM-EUSO se considera manejar un total de 20 CCBs**.**

1.2.1.3 GPS

El sistema de posicionamiento global (GPS: *Global Positioning System*) en el JEM-EUSO, es el encargado de proveer una estampa de tiempo UTC (Coordinated Universal Time) con una precisión de 1μs, así como de proveer datos de posicionamiento con una precisión de 10m desde tierra hasta 42 km sobre el nivel del mar. El HK se encargará de tomar datos de temperatura y voltaje del subsistema, a través de un protocolo diferencial SPI, con el fin de verificar el estado del mismo.

1.2.1.4 CLKB

Las siglas de este subsistema significan tarjeta de reloj (CLKB: *Clock Board*); es el encargado de generar y distribuir un par de bases de tiempo de 40 MHz y 400 KHz para la sincronización de eventos en todos los dispositivos del JEM-EUSO. Contiene interfaces de comunicación con los subsistemas CCB y GPS. El HK, se encarga de monitorear datos de temperaturas, voltajes y corrientes proporcionados por esta tarjeta a través del protocolo SPI diferencial, con el fin de detectar posibles fallas en este subsistema.

1.2.1.5 LVPS

Las siglas de este subsistema significan fuente de alimentación de bajo voltaje (LVPS: *Low Voltage Power Supply*) y es el encargado de alimentar, apagar y encender a los subsistemas del JEM-EUSO incluyendo al HK [6]. También contiene circuitos electrónicos que monitorean voltaje y corriente,

cuyos valores son enviados al *HK* para verificar que sus valores se encuentren dentro de un rango aceptable. Las tareas en las que interactúa con el HK son:

- Envía al HK señales de monitoreo de voltaje y corriente de todos los subsistemas que alimenta. Todas las señales monitoreadas se envían en forma analógica, las cuales son digitalizadas para que puedan ser leídas por el dispositivo lógico del HK.
- Contiene un sistema de encendido/apagado electromecánico de todos los subsistemas, el cual es controlado por el *HK,* a través de una interfaz de comando de alto nivel (HLC: *High Level Command*).
- Envía señales que indican el correcto estado de encendido/apagado a través de la lectura de una señal que se ha denominado "*Contact Closure*" (abreviada CC), la cual se encontrará, en estado alto para indicar encendido, y estado bajo para indicar apagado.

1.2.1.6 CPU

Este subsistema es la unidad central de procesamiento (CPU: *Central Processing Unit*) del JEM-EUSO que será una computadora, cuya placa madre contará con un procesador Intel Atom® y que estará encargada de llevar a cabo las siguientes tareas: decidir cuándo encender o apagar cualquier subsistema a través de comandos enviados al HK, realizar tareas de calibración periódicas del instrumento, comenzar la adquisición y almacenamiento de datos, solicitar información al HK para monitorear los subsistemas alimentados por la LVPS y leer alarmas provenientes de otros subsistemas, entre otras. La comunicación del HK con este subsistema es por medio del protocolo RS-422.

1.2.1.7 Sensores de temperatura

La interacción de estos dispositivos con el HK consiste en medir la temperatura del instrumento a partir de sensores distribuidos en el mismo. Una primera aproximación de la cantidad de sensores que se manejarán en el JEM-EUSO es de 400.

En la Figura 1.3, se puede observar la interacción del HK con los siete subsistemas descritos anteriormente, y de los cuales: cuatro comparten el protocolo de comunicación SPI diferencial, uno utiliza el protocolo serial RS-422 y dos de ellos señales analógicas.

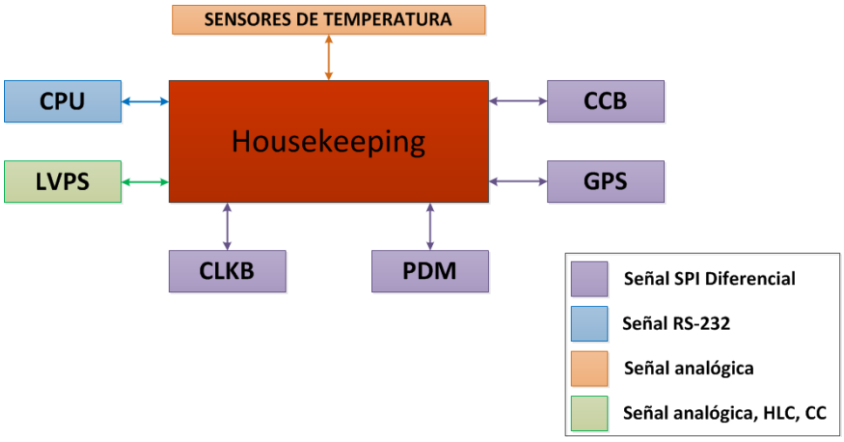


Figura 1.3. Interacción del Housekeeping con los subsistemas del JEM-EUSO.

1.3 Requerimientos

1.3.1.1 Requerimientos de funcionalidad del HK

En general las tareas que el HK debe realizar son:

- Monitoreo de corrientes (I), voltajes (V) y temperaturas (T) de todos los subsistemas, a través de los datos analógicos proporcionados por la LVPS y los datos digitales adquiridos por el protocolo SPI.
- Adquisición de señales de sensores de temperatura, distribuidos en todo el instrumento.
- Recepción de alarmas de todos los subsistemas.
- Generación de sus propias alarmas por medio de la revisión del rango de valores de los datos recibidos, las cuales podrán ser atendidas por otros subsistemas a bordo o bien desde la estación terrestre.
- Envío de señales de reinicio y reprogramación: *Reset* y *Program_B*, a los subsistemas que lo requieran.
- Atención de solicitudes de procesos desde la CPU mediante interrupciones.
- Encendido y apagado de los subsistemas mediante comandos.
- Verificación del estado de encendido/apagado de cada subsistema.
- Activación de sistemas de respaldo.
- Control para la acción del mecanismo de apertura y cierre de tapa y del mecanismo de inclinación.

En resumen, cualquier anomalía en la operación de los subsistemas que sea detectada por el HK es reportada como alarma para la CPU. En principio, el HK no realiza ninguna acción al recibir estas alarmas, en espera de recibir algún comando desde la CPU. En las circunstancias más críticas, puede apagar un subsistema y emitir una alarma para evitar posibles daños, mientras se espera la orden de la CPU, ya sea para reiniciar el subsistema o reemplazarlo. Los comandos recibidos y ejecutados por el HK son generados ya sea por la CPU o por la estación terrestre. Básicamente HK es un sistema auxiliar de distribución de comandos, monitoreo del estado de los subsistemas del telescopio y un generador de alarmas [1].

Cabe mencionar que el control de las tareas se define como de control lento, ya que se espera que el HK tenga un tiempo de respuesta que va de 1 a unos pocos segundos.

1.3.1.2 Requerimientos eléctricos y físicos

De acuerdo al Purple Book, el cual es el resumen del estudio de la fase-A del proyecto JEM-EUSO [**1**], los requerimientos actuales son:

- Las dimensiones del HK deben corresponder a una medida eurocard doble extendida[1] (200×233 mm).

[1] Eurocard es un estándar europeo de medidas para circuitos impresos. Entre las medidas de este estándar se encuentra la medida *Doble Extended Eurocard* que equivale a una tarjeta de 200x233 mm [**1**].

- El consumo de potencia sebe ser máximo de 40 W.
- Es altamente recomendable que los circuitos integrados del HK sean de grado 1[2].
- Debe tener una masa total de 85 kg, donde 70 kg corresponden al peso de los cables.

1.4 Objetivos

1.4.1.1 Objetivo general.

- Proponer un esquema de instrumentación basado en una plataforma FPGA, para el diseño e implementación de arquitecturas de cómputo reconfigurables, que alberguen la lógica del HK.

1.4.1.2 Objetivos específicos.

- Analizar diferentes tipos de esquemas de instrumentación el HK basados en FPGA.
- Transferir la lógica operativa del prototipo del sistema de monitoreo HK versión 2 [6], a una plataforma de desarrollo FPGA, evolucionando el paradigma de diseño e integración de HK.
- Analizar la viabilidad de la implementación del sistema de cómputo que alberga la lógica del HK en un FPGA Virtex®-5.
- Optimizar la lógica implementada, aprovechando la capacidad del FPGA para diseñar e integrar arquitecturas de cómputo polimórficas, para realizar tareas de adquisición, preprocesamiento y comunicación de datos.
- Validar operativamente el sistema de cómputo embebido por medio de la interacción con bancos de prueba, los cuales emulan la integración con el resto de los subsistemas que componen la instrumentación del telescopio JEM-EUSO.

1.5 Justificación

Actualmente se tiene una primera versión del prototipo del HK basado en un microcontrolador, cuyo diseño fue desarrollo para interactuar con un sistema mínimo del JEM-EUSO, llamado EUSO-BALLOON [50]. No es resulta sencillo escalar de manera óptima esta primera versión del HK para que interactúe con el sistema completo del JEM-EUSO debido principalmente a limitaciones en el número de terminales de entrada/salida y a la rigidez de su arquitectura. Por lo tanto, la siguiente etapa, y la cual es coincidente con el objetivo de esta tesis, es el diseño de una propuesta de arquitectura computacional para el HK basada en un FPGA, con el fin de evaluar la viabilidad de la implementación de su lógica embebida en este dispositivo. En esta tesis, la implementación se hará utilizando la tarjeta de desarrollo comercial XUPV5-LX110T, cuyo dispositivo principal es un FPGA Virtex-5 de Xilinx®.

Cabe mencionar que ya se ha realizado un experimento previo utilizando un FPGA a bordo de la plataforma Pixqui [5], el cual fue un vuelo suborbital que alcanzó los 32 km de altura, y cuyo objetivo fue el de familiarizarse con el uso de estos dispositivos así como visualizar su

[2] Componentes de calificación espacial o equivalentes [32].

comportamiento en un vuelo suborbital. Dado que los resultados obtenidos fueron exitosos, se decidió continuar en este trabajo con el uso del FPGA Virtex-5 para implementar la lógica de funcionamiento del HK.

Es importante destacar que la implementación de la lógica utilizando una plataforma comercial, no solo será de utilidad para que se continúe en etapas posteriores con el desarrollo de la misión JEM-EUSO, sino que también coadyuvará al crecimiento del conocimiento sobre el uso de FPGAs en aplicaciones de instrumentación espacial.

1.6 Prueba de un sistema Housekeeping basado en FPGA a bordo de "Pixqui" (proyecto AEMB-F1)

La AEM (Agencia Espacial Mexicana) logró un acuerdo de participación con la Agencia Espacial Estadounidense NASA (*National Aeronautic and Space Administration*), la cual colabora en el consorcio JEM-EUSO. El acuerdo permitió verificar experimentalmente los prototipos para JEM-EUSO fabricados en México, en una serie de vuelos con globos estratosféricos de la NASA. En México, el desarrollo de esta carga se denominó AEMB-F1, siglas que provienen de AEM Balloon-Flight one (Agencia Espacial Mexicana vuelo en globo uno) para posteriormente ser renombrado bajo el nombre de "Pixqui", que en lengua náhuatl significa guardián.

Estos vuelos se realizan a una altura cercana a los 30 km sobre el nivel del mar, en condiciones de vacío y presión muy parecidas a las del espacio exterior. Para el primer vuelo, realizado el 19 de Agosto de 2013, México participó con una carga de prueba en la góndola de la NASA, donde estuvo incluido un módulo basado en FPGA denominado HK.F.V1.

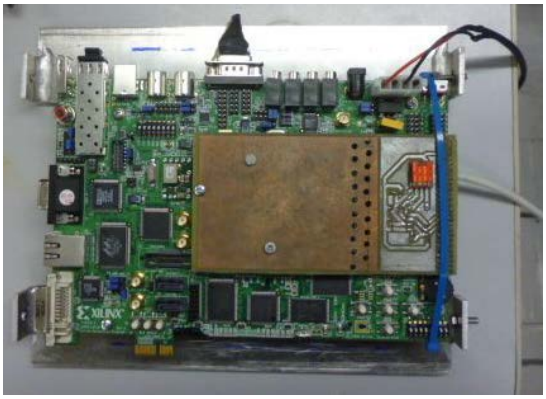


Figura 1.4. Fotografía del HK.F.V1.

El proyecto HK.F.V1 representó un experimento que permitió realizar un estudio previo sobre el uso de un FPGA y evaluar la viabilidad de la implementación de la lógica del HK en dicho dispositivo. En este experimento se probó la operación de un FPGA, integrado en una plataforma comercial, bajo condiciones estratosféricas. El experimento consistió en monitorear la temperatura ambiental a la que estaba sometido el FPGA y guardar los datos en una memoria EEPROM a bordo. Al regreso del vuelo, se comprobó que la tarjeta resistió las condiciones de presión, temperatura y humedad a las que estuvo sometido y que fue posible almacenar los datos

correspondientes a la temperatura. Los resultados que se obtuvieron permitieron respaldar la selección de un FPGA como plataforma de desarrollo, por lo que para la posterior implementación de la lógica del HK se ha decidido seguir trabajando con FPGAs.

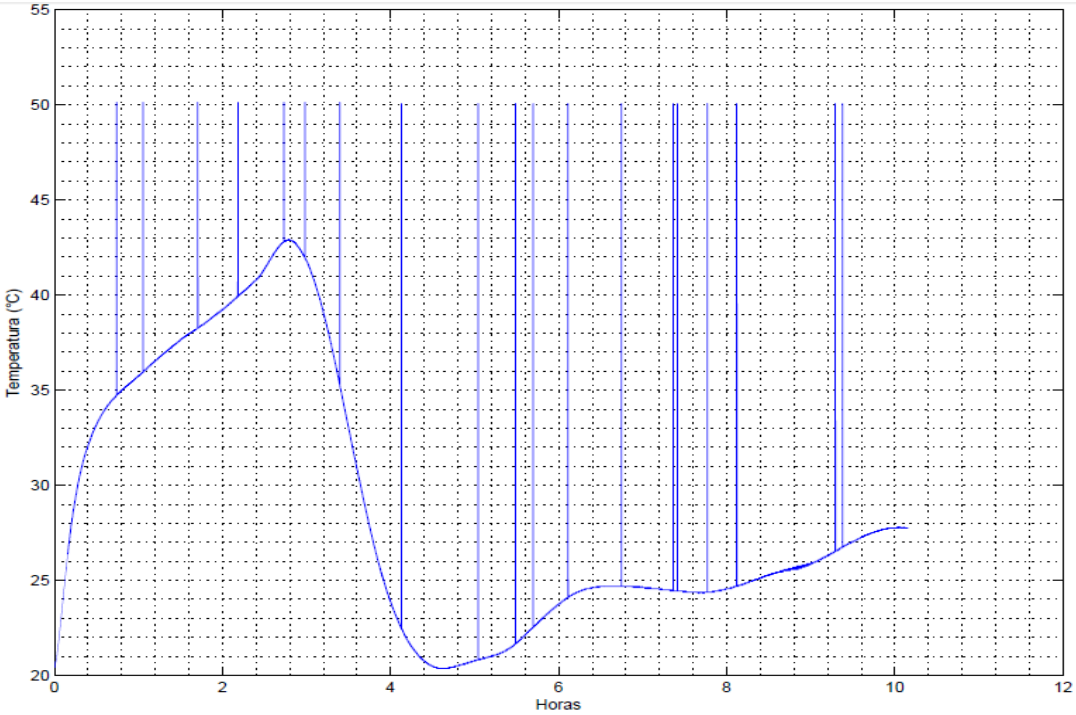


Figura 1.5. Temperaturas registradas por el sensor de temperatura del HK.F.V1 durante el vuelo de *Pixqui* en Fort Sumner, Texas, E.U.

Capítulo 2
Selección del dispositivo lógico principal de control

2.1 Tipos de dispositivos lógicos

No existe una clasificación universal de los circuitos integrados digitales, pero en general se pueden clasificar en dos grandes categorías de acuerdo a su arquitectura: fija o estándar y programable o a la medida, como se observa en la Figura 2.1 .

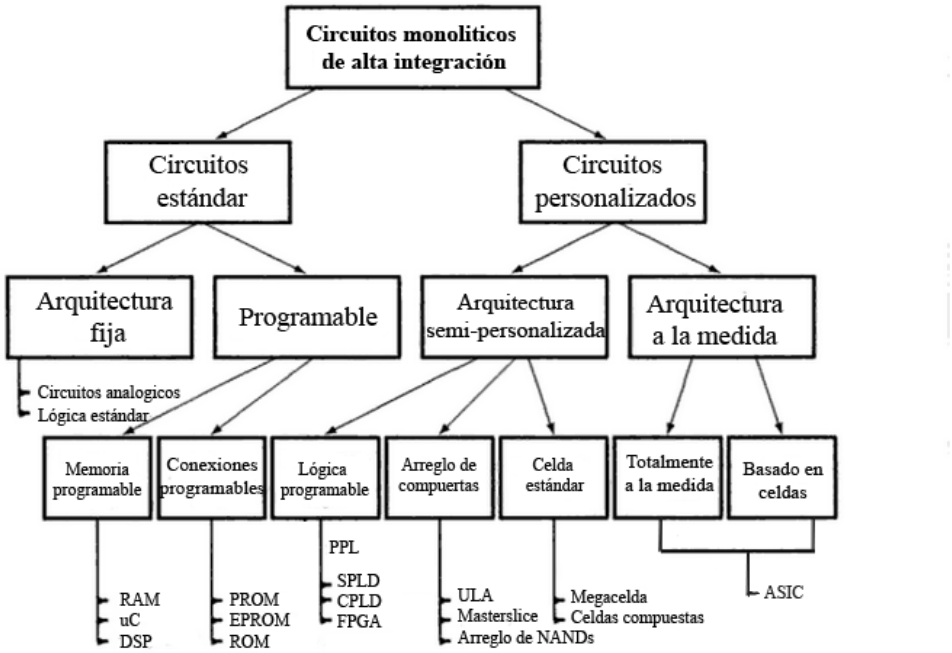


Figura 2.1. Clasificación de los circuitos integrados.

Mientras que un dispositivo de arquitectura fija tiene un funcionamiento predefinido desde que se construye, un dispositivo lógico programable puede ser configurado por el usuario para realizar una variedad de funciones lógicas. En cuanto a su estructura interna, los circuitos e interconexiones en un dispositivo de arquitectura fija, son permanentes y no pueden ser modificados una vez que el dispositivo ha sido manufacturado, como es el caso de los microcontroladores, los procesadores o las memorias. Al contrario de esto, los dispositivos lógicos programables como los ASICs, CPLDs y FPGAs, ofrecen al usuario la posibilidad de modificar y adaptar su arquitectura de acuerdo a la función o funciones que se requiera que ejecute el dispositivo. En el caso de los FPGAs por ejemplo, esta arquitectura puede incluso reconfigurarse en tiempo de ejecución[46].

Es importante tener en cuenta que el JEM-EUSO es un telescopio integrado por diversos subsistemas que se instalará en la Estación Espacial Internacional, y que para el desarrollo del HK, el dispositivo para la implementación de su lógica, deberá ser capaz de adaptarse a las

necesidades y a las condiciones de operación [1] del mismo. A continuación, se tiene una tabla que muestra las ventajas y desventajas de diferentes dispositivos, que se pueden considerar candidatos para convertirse en la unidad lógica principal del HK.

Dispositivo lógico programable	Ventajas	Desventajas
Microcontrolador	• Un solo chip contiene todos los elementos que conforman su arquitectura. • Bajo costo. • Reducción de espacio en el diseño. • Operatividad secuencial.	• Arquitectura fija (no reconfigurable). • No se adapta a una aplicación específica. • Los recursos de terminales de entrada/salida no son suficientes.
ASIC	• Se puede crear una aplicación a la medida. • Pueden implementarse diseños complejos. • Tienen un buen desempeño y un uso de recursos óptimo.	• No es reconfigurable. • Es costoso. • Requiere de una alta inversión de tiempo en su diseño.
CPLD	• Es reconfigurable. • Se pueden implementar una amplia variedad de funciones lógicas.	• Grandes diseños pueden resultar en un uso inadecuado de los recursos. • El número de recursos con los que cuenta son menos que en un FPGA. • Las terminales de entrada/salida no son suficientes.
FPGA	• Se pueden tener diseños complejos. • Tiempos de desarrollo cortos. • Arquitectura flexible, diseños a la medida de la aplicación. • Escalable, versátil. • Reconfigurable.	• Diseños menos complejos que con un ASIC. • El costo por unidad es grande.

Tabla 2.1. Ventajas y desventajas de diferentes dispositivos lógicos para la implementación de la lógica del HK.

2.2 Ventajas de la implementación del HK en un FPGA sobre otros dispositivos lógicos

Como se trató en el capítulo 1, el dispositivo que será la unidad lógica principal del HK debe tener una arquitectura que le permita adaptarse para interactuar con los diversos subsistemas del telescopio; es decir, que lo mejor es que sea un dispositivo cuya arquitectura sea flexible, de modo que se pueda tener un diseño completamente a medida de la aplicación.

Analizando las características de los dispositivos mencionados anteriormente, se ha llegado a la conclusión de que un FPGA es un dispositivo apto para explorar la viabilidad de su uso para la implementación de la lógica del HK. En la actualidad por ejemplo, existen microcontroladores que ofrecen una gran cantidad de recursos, cuya arquitectura fija no se compara con la posibilidad de diseñar una arquitectura a la medida de nuestra aplicación. Es por esto que se ha pensado en utilizar un dispositivo configurable como un ASIC o un FPGA. Sin embargo, el ASIC no se utiliza ya que no posee las características de flexibilidad que se esperan de la unidad lógica principal del HK.
Además de las características de flexibilidad y reconfiguración del FPGA, este también puede realizar tareas en modo paralelo o concurrente, es escalable, versátil, tiene una cantidad de recursos en terminales de entrada/salida grande en comparación con otros dispositivos, y además existen dispositivos análogos para aplicaciones espaciales.

Un FPGA posee las características idóneas para ser la unidad lógica del HK, entre otras:

- Es un dispositivo que permite un diseño a medida y que además puede rediseñarse en cualquier momento sin que esto requiera de invertir grandes cantidades de tiempo.
- Algunas familias de FPGAs tienen integrados poderosos procesadores (*hard processor cores*), y adicionalmente se le pueden embeber núcleos de propiedad intelectual (*IP cores*), por ejemplo, se pueden integrar procesadores (*soft processor cores*) al diseño [3], por lo que se puede tener un sistema que se ajuste a las necesidades del HK.
- Para el proyecto JEM-EUSO se tiene previsto que al final el HK interactúe con alrededor de 2000 sensores y relevadores [1], así como con otras señales. Por tal motivo, se necesita establecer las bases del subsistema final en un dispositivo escalable que sea capaz de interactuar con un número variable de equipos periféricos y señales, como es el caso de los FPGAs.
- El FPGA puede realizar tareas en modo paralelo o concurrente, lo que hará posible que el monitoreo de los subsistemas se realice en un tiempo menor que si no se contara con esta característica; además de que permite versatilidad y escalamiento en el diseño, lo cual lo hace adaptable a cambios en el diseño y funcionalidad.
- Posee una gran cantidad de terminales que se pueden configurar como entradas/salidas.
- Es necesario mencionar que diversas familias de FPGAs cuentan con modelos de calificación espacial, lo cual resulta atractivo para su uso como plataforma lógica de desarrollo del HK.

En conclusión, los FPGAs tienen la habilidad de combinar núcleos de procesadores embebidos con periféricos estándar y módulos de hardware con funciones personalizadas. Esto hace posible construir soluciones adecuadas y a la medida para resolver tareas específicas dentro de una aplicación, lo que ha convertido a los FPGAs en una opción a evaluar para el desarrollo del HK. Además, cabe recordar que la versión final del HK estará expuesta a condiciones espaciales y cuando se trabaja con un dispositivo de alta escala de integración como un FPGA, existe el riesgo

de que sea afectado por la radiación, pero con los FPGAs se tiene la opción de migrar el diseño de la lógica de una forma sencilla a un dispositivo con calificación espacial en caso de ser necesario.

2.3 Arquitectura de un FPGA

Un FPGA en general contiene un arreglo bidimensional de celdas lógicas (LC: *Logic Cell*) genéricas o elementos lógicos, además de tener interruptores o interconexiones programables rodeadas por elementos de entrada/salida, los cuales también son programables [15] [16]. El uso de estos dispositivos, permite implementar funciones y circuitos digitales sumamente complejos en un solo circuito integrado (IC: *Integrated Circuit*) [17]. En la Figura 2.2 **¡Error! No se encuentra el origen de la referencia.**se observa la arquitectura de un FPGA.

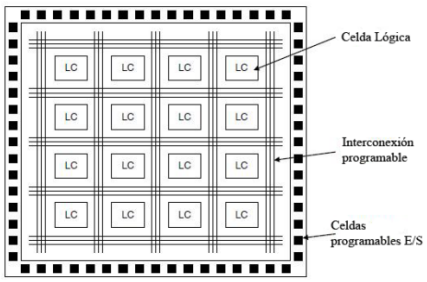


Figura 2.2. Arquitectura genérica de un FPGA.

Las celdas lógicas pueden configurarse para desempeñar una función, en tanto que los interruptores pueden programarse para realizar interconexiones entre las celdas lógicas [15] y a su vez estas conexiones están rodeadas por celdas programables como entradas/salidas. A partir de la configuración de estos elementos, es posible crear en el dispositivo una funcionalidad específica.

Los FPGAs en general están estructurados con tres principales tipos de bloques:

- Bloques lógicos configurables (CLB *Configurable Logic Block*). Están colocados en un arreglo matricial.
- Bloques de interconexión. Son canales de enrutamiento horizontales y verticales que conectan los bloques lógicos entre sí.
- Los bloques entrada/salida. Rodean a los dos elementos anteriores y permiten la comunicación del chip con elementos exteriores.

Adicionalmente, la arquitectura de un FPGA incluye bloques embebidos de alto nivel, tales como multiplicadores y bloques de memoria RAM. Estos bloques pueden ser programados para implementar, ya sea lógica secuencial o combinacional.

Cada fabricante de FPGAs tiene su propia arquitectura para sus dispositivos, pero en general entre los fabricantes la variación es mínima en comparación con la arquitectura propuesta inicialmente por Xilinx® [18].

2.3.1 Bloques lógicos reconfigurables

Un bloque lógico reconfigurable (CLB: *Configurabe Logic Block*), contiene la lógica programable de un FPGA. Este elemento por lo regular está formado por tablas de búsqueda ó LUTs (*look-up-table*), multiplexores, compuertas y flip-flops.

LUT: es una tabla de verdad almacenada en una SRAM, y proporciona las funciones de un circuito combinacional del bloque lógico. Se utiliza la sección de lógica combinacional, junto con varios multiplexores programables, para configurar las ecuaciones de entrada del flip-flop y la salida del bloque lógico [20].

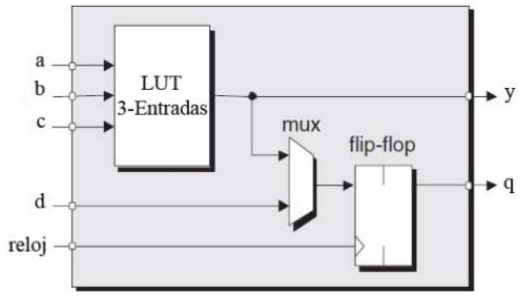


Figura 2.3. Elementos básicos de un bloque lógico programable (CLB).

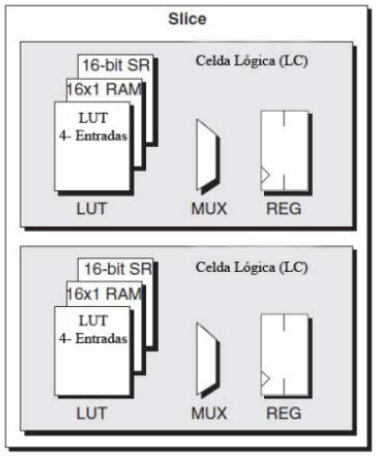


Figura 2.4. *Slice* con dos celdas lógicas.

Los elementos básicos de un CLB, en conjunto forman una celda lógica o LC y a su vez, dos LC forman lo que se conoce como un *Slice* (Figura 2.4), dependiendo del FPGA, este puede contener un número diferente de *slices* por CLB.

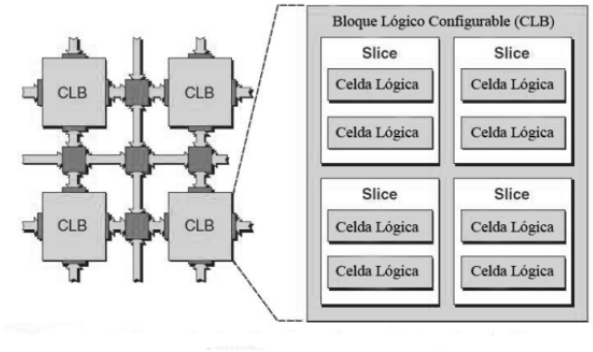


Figura 2.5. Arquitectura de un CLB con 4 *slices* con 2 LC.

2.3.2 Interconexiones programables

Todos los bloques internos en un FPGA están conectados por medio de cables con interruptores programables. Existen tres tipos de conexión para los componentes: conexiones locales, conexiones de interruptores y conexiones largas. En conjunto, las conexiones y los cables de conexión se conocen como recursos de enrutamiento. En la Figura 2.6 se muestran los tres tipos de conexiones.

El primer tipo de conexión, permite crear funciones lógicas complejas que un sólo CLB no sería capaz de realizar, pero varios CLB sí. El segundo recurso de enrutamiento son los interruptores, los cuales forman una matriz en donde los cables horizontales y verticales se interceptan. La matriz de interruptores da la posibilidad de conectar una matriz con otra, y esta a su vez con otra, y así sucesivamente según se necesite, los cuales eventualmente se conectan a los bloques lógicos, pudiendo de esta manera conectar bloques lógicos que se encuentran alejados entre sí. El tercer tipo de enrutamiento es por medio de conexiones largas, estas conexiones permiten conectar dos bloques lógicos que se encuentran a una distancia considerable, esto sin crear un retardo significativo.

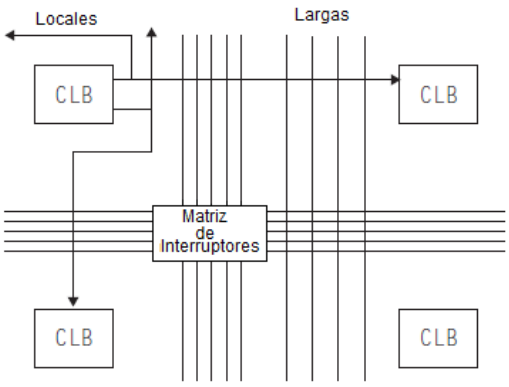


Figura 2.6. Interconexiones programables.

2.3.3 Bloques de entrada/salida

Estos bloques de entrada salida (IOB: *input/output blocks*) se encuentran en el perímetro del chip, y conectan los bloques lógicos con los pines externos del FPGA. Cada IOB puede ser usado para implementar varios estándares para señales en modo común, tales como LVCMOS (*Low Voltage*

Complementary Metal Oxide Semiconductor) que trabaja con un voltaje de 3.3 V o LVTTL (*Low Voltage Transistor-Transistor Logic*), que trabaja con un voltaje de 5 V. Además de que los IOBs pueden ser emparejados con un IOB adyacente para formar señales diferenciales LVDS (*Low-Voltage Differential Signaling*).

2.4 Elección del FPGA Virtex™-5 de Xilinx®

Si bien en esta tesis no ha sido uno de los objetivos el analizar cuál es el mejor FPGA para que sea la unidad lógica del HK, si se ha realizado una investigación sobre cuál cumplía con los requerimientos mínimos para una primera evaluación de la implementación de la lógica del HK en un FPGA.

Una de las principales razones por las que se ha elegido trabajar con los FPGAs de Xilinx®, es que durante la investigación se encontró que en diversas misiones dirigidas por NASA, se ha trabajado con esta compañía, por ejemplo en las misiones de exploración de Marte [48], o en la misión M-Cubed [49] en colaboración con la Universidad de Michigan. También se encontró que los FPGAs de grado espacial de Xilinx® son probados en los laboratorios del JPL (Jet Propulsion Laboratory) y que han demostrado ser altamente confiables para aplicaciones espaciales. Por lo tanto, ya que se busca trabajar con un dispositivo que en un futuro sea de grado espacial, se considera que esta es una buena opción.

Aunado a lo anterior, se tiene el hecho de que en la colaboración internacional, diversos grupos encargados del desarrollo de otros subsistemas, se encuentran trabajando con FPGAs de la familia Virtex™ de Xilinx®, y el trabajar con la misma línea de dispositivos facilitará en un futuro la tarea de integración con los demás subsistemas.

Finalmente, la selección del dispositivo comercial para esta tesis, fue el FPGA Virtex-5, ya que para esta etapa del proyecto se deseaba contar con una plataforma de desarrollo comercial sobre la cual se pudiera trabajar pero que al mismo tiempo se tuviera un análogo del dispositivo con calificación espacial, y entre las diversas opciones por motivos de costo y características de las FPGAs, se adquirió la plataforma XUPV5-LX110T, la cual cuenta con un FPGA Virtex-5 que posee su análogo con calificación espacial. Como se verá más adelante, esta FPGA es más eficiente que otras de la misma familia y además optimiza mejor el uso de los recursos por lo que tiene un consumo de potencia menor.

En conclusión, el FPGA Virtex™-5 de Xilinx® ha resultado ser un dispositivo que satisface las características mínimas requeridas para ser la unidad lógica del HK.

2.4.1 FPGAs de Xilinx®

Xilinx® es la industria líder en el desarrollo e innovación de dispositivos lógicos programables y fue la primer compañía en desarrollar un FPGA. Tiene más de 3,500 patentes y es conocida por sus logros, por ejemplo la introducción al mercado del modelo de manufactura *fabless*, el cual es un modelo en donde Xilinx se dedica exclusivamente al diseño y comercialización de su producto mientras que la manufactura se realiza en otro lugar. Actualmente produce FPGAs, SoCs (System

On Chip) y circuitos integrados 3D (*3D ICs*), proporcionando soluciones innovadoras para todo tipo de aplicaciones.

2.4.2 Familia Virtex™

Xilinx maneja varias familias de FPGAs, entre ellas la familia Virtex, a la cual se le han añadido bloques adicionales con el objetivo de hacer más eficiente el FPGA y mejorando el uso de los recursos con los que se cuenta; por ejemplo, FPGAs recientes como la Virtex® 4 ó 5 se les agrega un procesador embebido, reduciendo de esta forma el uso de recursos del FPGA y el consumo de potencia. Por lo regular se ocupan procesadores del tipo RISC (*Reduced Instruction Set Computer*), tales como el IBM PowerPC 405 o PowerPC 440. Para los FPGAs que no cuentan con un procesador es posible grabar uno haciendo uso de los *soft-core processors* que en el caso de Xilinx puede ser un PicoBlaze® o un MicroBlaze®.

Adicionalmente, la familia de FPGAs Virtex™, incluye terminales de entada/salida que tienen un alto desempeño y que se pueden configurar para proporcionar una amplia variedad de estándares, también tiene otras características que son programables como la corriente que manejaran estas terminales, así como el *slew rate*[3]. Como ya se ha mencionado, para este proyecto se ha decidido trabajar con esta familia, específicamente con Virtex®-5, de la cual se hablará en el siguiente apartado.

2.4.2.1 Virtex®-5

Al utilizar la arquitectura de columna basada en la segunda generación de bloque modular avanzado de silicón (ASMBL™: *Advanced Silicon Modular Block*), la familia Virtex-5 ofrece la posibilidad de elegir entre cinco plataformas distintas o subfamilias. Cada subfamilia esta optimizada en una característica diferente para que el usuario pueda elegir la que más se acomode a las necesidades del diseño. Adicionalmente, los FPGAs Virtex®-5, tienen la segunda generación de *slices* para los procesadores digitales de señal (DSP: Digital Signal Processing), selección del estándar de voltaje para la señales de entrada y salida con control digital de impedancia, entre otras características.

Las subfamilias se reconocen por las letras LXT, SXT, TXT y FXT. La subfamilia LXT, incluye dispositivos que tienen un gran número de bloques lógicos configurables en comparación con otros FPGAs que tienen un *hard core* embebido. Esto se debe a que los recursos son utilizados para la lógica configurable en vez de ser utilizados en un procesador embebido de fábrica. La subfamilia SXT, está optimizada para procesamiento de señales, ya que designa más recursos a la parte de procesamiento digital. Las subfamilias TXT/HXT, tienen mayores recursos asignados a la parte de transceptores seriales de alta velocidad, por lo que poseen una capacidad de interconexión con una ancho de banda grande, es decir, que esta subfamilia está optimizada para las comunicaciones. En cuanto a la subfamilia FXT, es la que tienen embebidos uno o más

[3] *Slew rate* (SR) es un efecto que representa la máxima tasa de cambio en el voltaje de salida cuando el voltaje de entrada cambia y cuyas unidades son V/μs.

procesadores (*hard-cores*) de fábrica, es decir que han sido físicamente construidos sobre la placa de silicio que conforma al FPGA.

En cuanto a los recursos disponibles, cada FPGA de la familia Virtex-5 está conformada como se explica a continuación.

LUTs

Virtex-5, está formada por LUTs de 6 entradas, las cuales pueden utilizarse como una sola LUT o dos LUTs de 5 entradas pero que comparten estas 6 entradas como se muestra en la Figura 2.7.

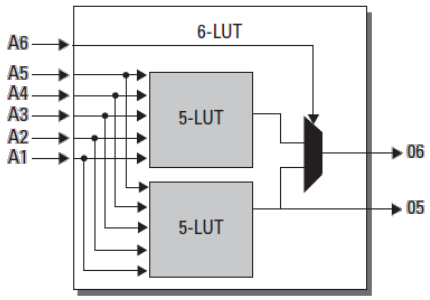


Figura 2.7. LUTs en Virtex-5.

Slices

Como ya se mencionó, cada celda lógica contiene LUTs y un elemento de almacenamiento (Flip-Flop). En el caso de esta familia, 4 LC conforman un slice.

CLBs

En la familia Virtex-5, un CLB contiene dos *slices* y la lógica necesaria para conectarse a *slices* vecinas. Cada *slice* está construida sobre una columna diferente de silicio y se conectan mediante una matriz de interruptores, como se muestra en la Figura 2.8.

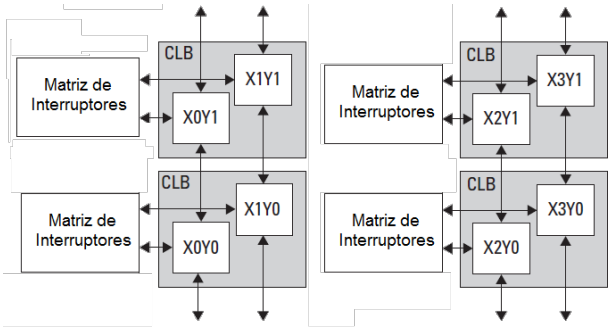


Figura 2.8. Diagrama de bloques de las interconexiones entre las *slices* en los CLBs y con la matriz de interruptores en Virtex-5.

Bloques de RAM (BRAMs)

Los bloques de RAM son memoria de acceso aleatorio agrupada en bloques de 36 Kbit. Proveen memoria dentro del chip, la cual puede ser utilizada como LUTs, almacenamiento local o buffer de datos (FIFOs).

Slices de DSP

Un uso común para los FPGAs es para el procesamiento digital de señales (DSP: *Digital Signal Processing*) por lo que para mejorar su rendimiento en esta aplicación, se incluyen slices especializadas conocidas como DSP48E (elementos DSP de 48 bits). Las *slices* de DSP incluyen 25x18 multiplicadores en complemento a dos, un acumulador de 48 bits, un sumador/restador y un operador lógico bit a bit (*bitwise*). Al añadir esta funcionalidad en un *slice*, se ahorran recursos en el FPGA.

Terminales de Entrada/Salida (IOB)

Los recursos de entrada/salida sirven para comunicar al FPGA con dispositivos externos, y se conectan a las terminales externas a través de un buffer. Estas entradas y salidas son configurables, de modo que pueden trabajar bajo diversos niveles lógicos de voltaje estándar (LVCMOS, SSTL, LVDS, etc.). También tienen la característica de tener un control digital de impedancia (DCI: *Digitally Controlled Impedance*) para eliminar resistencias parásitas en las terminales del dispositivo, lo cual reduce la degradación de señales. El DCI puede ajustar la impedancia de E/S para que se acople a la impedancia de la pista de entrada o salida.

Transceptores seriales de alta velocidad

Los FPGAs incluyen transceptores de comunicación serial de alta velocidad con baja latencia[4] y un ancho de banda de hasta 6.5 Gb/s. Estos transceptores son acoplados directamente al FPGA para proporcionar un acceso directo desde y hacia los *soft cores*. La Virtex-5 tiene dos tipos de transceptores: RocketIO GTX y GTP. Los primeros son capaces de manejar un ancho de banda grande, mientras que los segundos tienen un ancho de banda menor pero también requiere menos potencia.

Relojes

En los diseños con FPGA es común operar diferentes núcleos a diferentes frecuencias. En los FPGAs es posible generar un amplio rango de frecuencias de reloj a partir de una misma fuente. En la Virtex-5 existen alrededor de 24regiones de reloj y cada uno soporta 10 relojes distintos. Xilinx utiliza un administrador digital de relojes (DCM: *Digital Clock Managers*) el cual se encarga de generar las distintas frecuencias de trabajo y además es capaz de desfasar la señal 90, 128 270 grados.

PowerPC 440

El PowerPC™ 440 es un procesador RISC de 32 bits integrado como un núcleo *hard-core* a una de las subfamilias de Virtex®-5, es decir que está construido físicamente sobre la placa de silicio. Estos procesadores son capaces interactuar con otros núcleos, generalmente mediante interfaces y buses que provee la misma plataforma de desarrollo. De esta forma, el procesador gana

[4] Tiempo que tarda un dato en estar disponible desde que se realiza su petición. Cuanta menos latencia, mejor.

funcionalidades, pues es posible por ejemplo, conectarle un puerto de comunicaciones, una memoria adicional y algún otro núcleo de propósito específico. En comparación con un soft processor core, el PowerPC consume menos potencia, tiene una frecuencia más alta de operación y requiere menos espacio físico que si se implementara su equivalente en un FPGA.

Device	Configurable Logic Blocks (CLBs)			DSP48E Slices[2]	Block RAM Blocks			CMTs[4]	PowerPC Processor Blocks	Endpoint Blocks for PCI Express	Ethernet MACs[5]	Max RocketIO Transceivers[6]		Total I/O Banks[8]	Max User I/O[7]
	Array (Row x Col)	Virtex-5 Slices[1]	Max Distributed RAM (Kb)		18 Kb[3]	36 Kb	Max (Kb)					GTP	GTX		
XC5VLX30	80 x 30	4,800	320	32	64	32	1,152	2	N/A	N/A	N/A	N/A	N/A	13	400
XC5VLX50	120 x 30	7,200	480	48	96	48	1,728	6	N/A	N/A	N/A	N/A	N/A	17	560
XC5VLX85	120 x 54	12,960	840	48	192	96	3,456	6	N/A	N/A	N/A	N/A	N/A	17	560
XC5VLX110	160 x 54	17,280	1,120	64	256	128	4,608	6	N/A	N/A	N/A	N/A	N/A	23	800
XC5VLX155	160 x 76	24,320	1,640	128	384	192	6,912	6	N/A	N/A	N/A	N/A	N/A	23	800
XC5VLX220	160 x 108	34,560	2,280	128	384	192	6,912	6	N/A	N/A	N/A	N/A	N/A	23	800
XC5VLX330	240 x 108	51,840	3,420	192	576	288	10,368	6	N/A	N/A	N/A	N/A	N/A	33	1,200
XC5VLX20T	60 x 26	3,120	210	24	52	26	936	1	N/A	1	2	4	N/A	7	172
XC5VLX30T	80 x 30	4,800	320	32	72	36	1,296	2	N/A	1	4	8	N/A	12	360
XC5VLX50T	120 x 30	7,200	480	48	120	60	2,160	6	N/A	1	4	12	N/A	15	480
XC5VLX85T	120 x 54	12,960	840	48	216	108	3,888	6	N/A	1	4	12	N/A	15	480
XC5VLX110T	160 x 54	17,280	1,120	64	296	148	5,328	6	N/A	1	4	16	N/A	20	680
XC5VLX155T	160 x 76	24,320	1,640	128	424	212	7,632	6	N/A	1	4	16	N/A	20	680
XC5VLX220T	160 x 108	34,560	2,280	128	424	212	7,632	6	N/A	1	4	16	N/A	20	680
XC5VLX330T	240 x 108	51,840	3,420	192	648	324	11,664	6	N/A	1	4	24	N/A	27	960
XC5VSX35T	80 x 34	5,440	520	192	168	84	3,024	2	N/A	1	4	8	N/A	12	360
XC5VSX50T	120 x 34	8,160	780	288	264	132	4,752	6	N/A	1	4	12	N/A	15	480
XC5VSX95T	160 x 46	14,720	1,520	640	488	244	8,784	6	N/A	1	4	16	N/A	19	640
XC5VSX240T	240 x 78	37,440	4,200	1,056	1,032	516	18,576	6	N/A	1	4	24	N/A	27	960
XC5VTX150T	200 x 58	23,200	1,500	80	456	228	8,208	6	N/A	1	4	N/A	40	20	680
XC5VTX240T	240 x 78	37,440	2,400	96	648	324	11,664	6	N/A	1	4	N/A	48	20	680
XC5VFX30T	80 x 38	5,120	380	64	136	68	2,448	2	1	1	4	N/A	8	12	360
XC5VFX70T	160 x 38	11,200	820	128	296	148	5,328	6	1	3	4	N/A	16	19	640
XC5VFX100T	160 x 56	16,000	1,240	256	456	228	8,208	6	2	3	4	N/A	16	20	680
XC5VFX130T	200 x 56	20,480	1,580	320	596	298	10,728	6	2	3	6	N/A	20	24	840
XC5VFX200T	240 x 68	30,720	2,280	384	912	456	16,416	6	2	4	8	N/A	24	27	960

Figura 2.9. Subfamilias de FPGAs de la familia Virtex-5.

El FPGA que se utilizará para implementar la lógica del HK es el XC5VLX110T-FF1136, el cual como se observa en la Figura 2.9, posee 680 terminales de entrada/salida y es del tipo LX, lo que quiere decir que está optimizado para proveer recursos en el diseño de hardware.

Capítulo 3
Metodología de diseño de la arquitectura del Housekeeping

3.1 Sistemas centralizados y distribuidos

La arquitectura de un sistema puede ir desde un sistema local o centralizado hasta un sistema de alta confiabilidad basado en una arquitectura distribuida.

3.1.1.1 Sistema centralizado

Un sistema centralizado es un sistema en donde los sensores, actuadores, entre otros elementos, están conectados a una única unidad de procesamiento y control. Este tipo de arquitectura tiene la ventaja de tener una respuesta rápida frente a una posible falla. Además, en este tipo de arquitectura es difícil que se tengan pérdidas de comunicación, datos corruptos o cambio en el orden de los mensajes, ya que todos los elementos se encuentran en un mismo lugar y conectados al mismo procesador. Al mismo tiempo, dado que es un sistema centralizado se tiene la desventaja de que si el procesador tiene alguna falla, se producirá una falla total en el sistema.

3.1.1.2 Sistema distribuido

Un sistema distribuido, es aquel sistema con múltiples elementos de procesamiento autónomos que trabajan para alcanzar un mismo propósito; es decir, que estos elementos que forman parte del sistema trabajan en conjunto de modo que el sistema en su totalidad es capaz de realizar la función para la que ha sido diseñado.

Ventajas de un sistema distribuido [45]:

- Se tiene un sistema de mayor confiabilidad ya que se puede crear un sistema tolerante a fallas por redundancia.
- Se pueden mejorar el rendimiento del sistema al explotar las capacidades del paralelismo de un sistema distribuido.
- Distribución de las tareas de procesamiento entre los diferentes dispositivos que conforman el sistema distribuido.
- Facilidad para incrementar el crecimiento del sistema al añadir o aumentar el número de procesadores y buses de comunicación.

Dentro de los sistemas distribuidos existen dos tipos: los sistemas distribuidos estrechamente acoplados (tightly-coupled) y los sistemas distribuidos con acoplamiento débil (loosely-coupled). Los primeros, son aquellos en donde los elementos de procesamiento tienen acceso a una memoria común, mientras que en el segundo tipo no es así, lo cual proporciona la ventaja de que al no utilizar variables compartidas, la sincronización y comunicación en un sistema como este no se verán afectadas.

Existe otra clasificación para los sistemas distribuidos dependiendo de los elementos que se utilizan en el sistema. Un sistema homogéneo es aquel en donde todos los elementos de procesamiento son iguales mientras que un sistema heterogéneo es aquel en donde sus elementos de procesamiento son diferentes; los problemas que plantea este último sistema son que habrá diferentes representaciones del programa y datos.

3.2 Propuestas de arquitectura para el Housekeeping

El proceso de diseño del HK se comenzó proponiendo una arquitectura centralizada, la cual consiste, como ya se mencionó, en un sistema que contiene un elemento único de control y dado que en esta tesis se ha propuesto evaluar el desempeño de un FPGA para esta tarea, el primer sistema que se propuso fue uno cuya unidad central es este dispositivo lógico. En la Figura 3.1 se muestra un diagrama donde se observan todos los subsistemas del JEM-EUSO con los que tendrá que interactuar el subsistema HK, el cual se define por medio de una línea punteada.

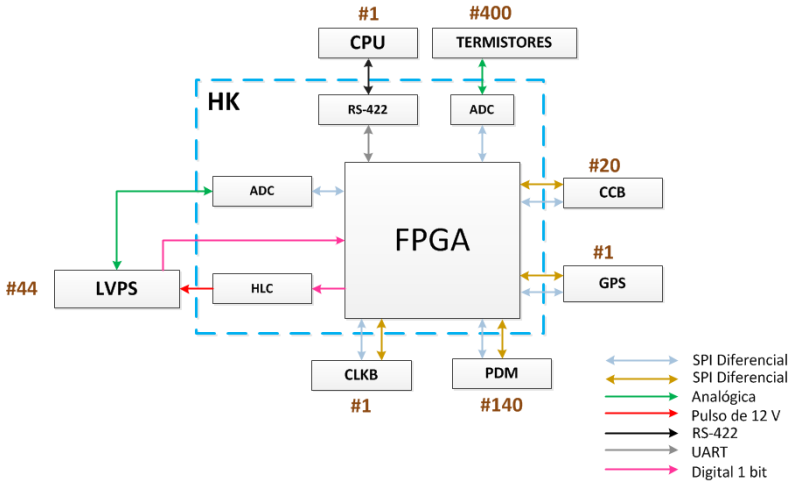


Figura 3.1. Diagrama de interacción entre el Housekeeping (HK) y los subsistemas del JEM-EUSO.

Como se observa, el FPGA necesitará de diversos elementos para llevar a cabo la tarea de HK como lo son multiplexores y transceptores. Así mismo, se observa que se requieren de diversos protocolos de comunicación como SPI y RS-422, para interactuar con cada subsistema, los cuales se pueden implementar a través de diferentes métodos, ya sea por software, hardware o núcleos embebidos en el mismo FPGA a través de un soft core. También se observa con números la cantidad de tarjetas que componen cada subsistema del JEM-EUSO.

3.2.1 Estimación de los recursos de terminales de entrada/salida que se requieren

Para tener una idea más clara sobre la interacción del HK con cada uno de los subsistemas, se analizarán a continuación cada uno de ellos, dando una breve descripción de su funcionamiento, así como un diagrama que muestra esta interacción, de modo que al finalizar de analizar cada uno de los subsistemas se tenga una idea aproximada a la realidad sobre los elementos básicos necesarios para la implementación de este subsistema.

3.2.1.1 LVPS

Como ya se describió en el primer capítulo, este subsistema maneja tres tipos de señales:

Nombre	Tipo de señal	Descripción	Rango de voltaje	Cantidad de pines por subsistema.	Total de pines por subsistema
Monitoreo	Entrada analógica	Se reciben voltajes que representan voltajes y corrientes de los subsistemas.	0V-5V	2 + 2 (retorno)	4
High Level Command (HLC)	Pulso digital de salida	Se envía un pulso de 12 V para encender o apagar los subsistemas.	0V-5V	1 + 1 a tierra	2
Contact Closure (CC)	Pulso digital de entrada	Señal que indica la correcta recepción de la señal de encendido o apagado de un subsistema.	0V-5V	1 + 1 (voltaje de referencia)	2
					TOTAL= 8 (contando GND)

Tabla 3.1. Señales para el HK provenientes del subsistema LVPS.

Los subsistemas con los que interactúa son:

- 1 CLKB
- 1 GPS
- 20 CCB
- 20 PDM
- 1 DST
- 1 CPU

Se necesitan un total de 352 terminales de las cuales 44 son GND, 44 son un voltaje de referencia, por ejemplo 3.3V, y las restantes 264 son señales de entrada o salida.

En la Figura 3.2 se tiene un diagrama de las tarjetas LVPS. Como se puede observar, se encuentran agrupadas en cuatro grupos diferentes para poder identificarlas de forma más sencilla: el primer grupo de tarjetas LVPS se denomina LVPS_DP1 y contiene a las tarjetas que alimentan los subsistemas GPS y CLKB; el segundo grupo, LVPS_DP2, contiene a las tarjetas que alimentan a CPU y DST; el siguiente grupo llamado LVPS_CCB, contiene 20 tarjetas de alimentación para 20 tarjetas del subsistema CCB; y por último, se tiene el grupo de tarjetas LVPS conformado por 20 tarjetas de alimentación para 20 tarjetas del subsistema PDM, y el cual se denomina LVPS_PDM. En total, se deben de monitorear estas 44 tarjetas LVPS.

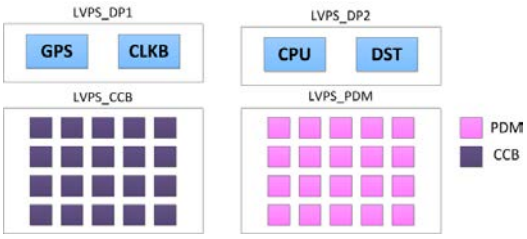


Figura 3.2. Tarjetas del subsistema LVPS con las que interactúa el HK.

3.2.1.2 PDM, CCB, CLKB y GPS

Hay cuatro subsistemas que se comunican por medio del protocolo SPI diferencial, los cuales son: PDM (Photo Detector Multiplier Board), CCB (Cluster Control Board), CLKB (Clock Board) y GPS. Estos cuatro subsistemas comparten por lo tanto la señales SCK, MOSI y MISO del protocolo SPI. En general tres de estos subsistemas además de enviar datos al HK también envían cada uno dos señales de alarma, una llamada ALARM y otra DONE.

3.2.1.3 Señales diferenciales

Una señal tipo *single-ended*, es una señal "individual o sencilla" contenida en un solo hilo, la cual crea un voltaje referenciado a un punto común en el circuito (normalmente una conexión a 0V).

Mientras tanto, una señal diferencial utiliza dos hilos que contienen señales que se complementan y la señal se obtiene por la diferencia de voltaje entre los dos hilos Figura 3.3. Las señales diferenciales se utilizan en circuitos de bajo voltaje ya que poseen robustez frente al ruido durante la transmisión de datos.

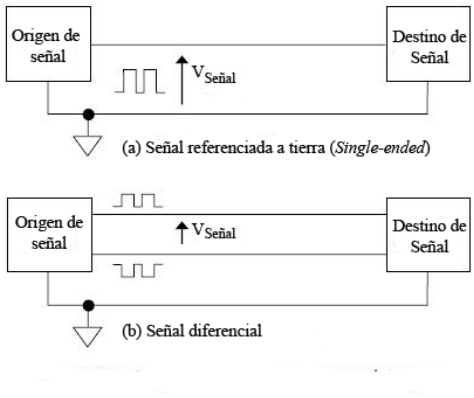


Figura 3.3. Señal *single-ended* (a) y señal diferencial (b).

3.2.1.4 Interacción del subsistema PDM con el HK

De acuerdo al Purple Book [**1**], la cantidad aproximada de PDMs es de 137, sin embargo para este diseño se manejarán un total de 140 PDMs debido a que en el mismo Purple Book se sugiere que el número de tarjetas de este subsistema puede aumentar hasta esta cantidad.

La interacción de este subsistema con el HK se lleva a cabo mediante el protocolo SPI diferencial y consiste por parte del HK en:

- Recibir y almacenar temperaturas y voltajes provenientes de PDM.
- Generar alarmas cuando los valores recibidos sobrepasen un umbral alto o bajo.
- Generar señales de RESET y PROGRAM_B en respuesta a ALARM y DONE.

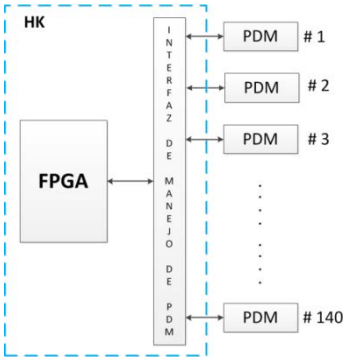


Figura 3.4. Representación de la cantidad de PDM e interfaz con el HK.

3.2.1.5 Interacción del subsistema CCB con el HK

En este caso se considerará manejar un total de 20 CCB suponiendo que cada uno tenga a su cargo un grupo de 7 PDMs como se observa en la Figura 3.5. En cuanto a su interacción con el subsistema HK, se lleva a cabo mediante un protocolo SPI diferencial y las acciones del HK en atención son este subsistema son:

- Recibir y almacenar temperaturas, corrientes y voltajes provenientes de CCB.
- Generar alarmas cuando los valores recibidos sobrepasen un umbral alto o bajo.
- Recibir valores de registros de CCB para verificar el funcionamiento del FPGA.
- Generar señales de RESET y PROGRAM_B en respuesta a ALARM y DONE.

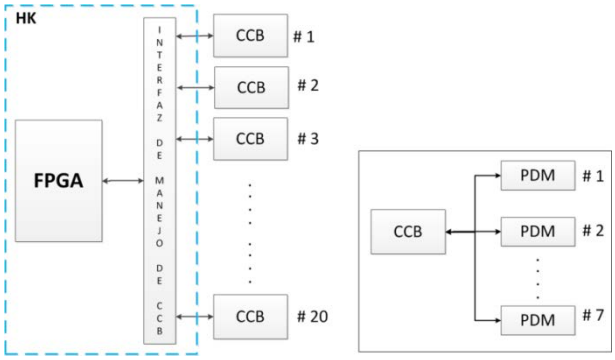


Figura 3.5. Representación de la interacción de HK con CCB.

Nombre	Tipo de señal	Descripción	Rango de voltaje	Cantidad de pines para cada unidad de subsistema
ALARM	Pulso digital diferencial de entrada	Esta señal esta normalmente en estado bajo.	LVDS	1 par diferencial
DONE	Pulso digital	Esta señal esta	LVDS	1 par diferencial

	diferencial de entrada	normalmente en estado alto.		
RESET	Pulso digital diferencial de salida	Esta señal envía un pulso de 3-5ms	LVDS	1 par diferencial
PROGRAM_B	Pulso digital diferencial de salida	Esta señal envía un pulso de 3-5ms	LVDS	1 par diferencial
SCK, MOSI, MISO	Señales SPI diferenciales	Estas señales se comparten con otros subsistemas.	LVDS	3 pares diferenciales
CS	Señal de selección de esclavo diferencial	Los subsistemas son activos en estado bajo.	LVDS	1 par diferencial

Tabla 3.2. Señales de los subsistemas CCB, PDM y CLKB con protocolo SPI.

3.2.1.6 Interacción del subsistema CLKB con el HK

Este subsistema es único en el JEM-EUSO, es decir que solo se debe interactuar con una tarjeta CLKB.

En cuanto a la interfaz entre CLKB y HK este último tiene las siguientes tareas:

- Recibir y almacenar temperaturas y voltajes provenientes de CLKB.
- Generar alarmas cuando los valores recibidos sobrepasen un umbral alto o bajo.
- Generar señales de RESET y PROGRAM_B en respuesta a ALARM y DONE.

En la Tabla *3.2.* Señales de los subsistemas CCB, PDM y CLKB con protocolo SPI.Tabla 3.2 se muestra la descripción de las señales en común que manejan estos tres subsistemas.

Nombre	**Cantidad de pines por unidad**	**Cantidad de pines para PDM**	**Cantidad de pines para CCB**	**Cantidad de pines para CLKB**	
ALARM	1 par dif.	140 pares dif.	20 pares dif.	1 par dif.	
DONE	1 par dif.	140 pares dif.	20 pares dif.	1 par dif.	**TOTAL**
RESET	1 par dif.	140 pares dif.	20 pares dif.	1 par dif.	
PROGRAM_B	1 par dif.	140 pares dif.	20 pares dif.	1 par dif.	**808 pares diferenciales**
SCK, MOSI, MISO	3 pares dif.	Compartidas	Compartidas	Compartidas	
CS	1 par dif.	140 pares dif.	20 pares dif.	1 par dif.	
TOTAL	5 pares por subsistema + 3 pares para SPI	700 pares diferenciales	100 pares dif.	5 pares dif.	

Tabla 3.3. Cantidad de pines para PDM, CCB y CLKB.

3.2.1.7 Interacción del subsistema GPS con HK

Este también es un subsistema único en el JEM-EUSO por lo que solo se tendrá interacción con **una tarjeta**, la cual como ya se mencionó maneja el **protocolo SPI diferencial,** ya que los dos sensores que contiene la tarjeta manejan este protocolo. Estos sensores son un ADS7868 y un ADT7301.

Las tareas que corresponden al HK en atención a este subsistema son:

- Leer temperatura (protocolo SPI diferencial)
- Leer una señal de voltaje (protocolo SPI diferencial).

Nombre	Cantidad de pines por unidad	Cantidad de pines para GPS	Total de pines para PDM, CCB, CLKB y GPS
SCK, MOSI, MISO	6 (compartidas)	-	**810 pares diferenciales**
CS	2 par diferencial	2 par diferencial	

Tabla 3.4. Cantidad de pines para GPS.

3.2.1.8 CPU

La interacción con este subsistema es por medio del protocolo RS-422 ya que es el protocolo que actualmente utiliza este subsistema. Debido a que se utilizara una tarjeta comercial que utiliza el protocolo RS-232, es necesario hacer la conversión al protocolo RS-422 por lo que se utilizará un transceptor que haga la conversión correspondiente.

Como primera solución se puede implementar un transceptor que convierta del protocolo RS232, el cual es el que maneja el FPGA, a protocolo R2422. De ser así el HK necesitaría además de los pines necesarios para el funcionamiento del transceptor 9 pines para este protocolo.

En cuanto a la interacción de CPU con el HK, las tareas de este último son:

- Recibir comandos para el encendido y apagado de los subsistemas a través de un protocolo serial.
- Enviar la lectura en forma periódica de todos los parámetros que el HK puede leer.
- Enviar información de alarmas de forma periódica.
- Recibir comandos de RESET y distribuirlos a los subsistemas o de reprogramación en caso de ser requerido.

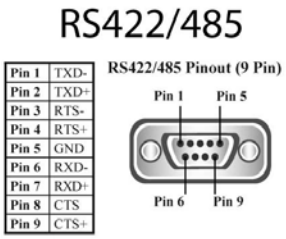


Figura 3.6. Pines para el protocolo RS422.

3.2.1.9 Sensores de temperatura

En una primera aproximación se tiene que la cantidad de sensores que se manejarán para este subsistema son 400. En el caso del HKV1 se utilizaron termistores para realizar esta tarea lo cual se observa en la Figura 3.7; suponiendo que esta fuera la configuración utilizada, sin aún utilizar el

multiplexor, se tendrían un total de 200 señales de las cuales 200 estarían conectadas a un voltaje de referencia y 200 serían las señales analógicas de entrada que tendrían que ir a un ADC.

Señal	Tipo	Cantidad de pines por sensor	Cantidad de pines en total
Vref	Voltaje de referencia analógico	1	400 pines: 200 a Vref 200 de entrada
Lectura	Voltaje analógico de entrada	1	

Tabla 3.5. Cantidad de pines para el subsistema de sensores de temperatura.

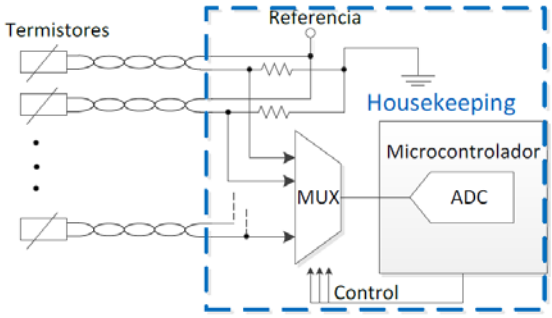


Figura 3.7. Diagrama de conexión para los termistores del HKV1. [6]

Subsistema	Cantidad de pines	TOTAL
LVPS	264 single-ended	**2,093 pines de entrada/salida**
PDM, CCB, CLKB, GPS	810 pares diferenciales	**Además:**
CPU	9 RS232	**244 a voltajes de referencia**
Sensores de temperatura	200 single-ended	**45 a tierra**

Tabla 3.6. Aproximación de la cantidad de pines necesarios del HK sin contar tierras.

Esta ha sido una primera aproximación de la cantidad de entradas y salidas que se requieren para la implementación del HK del JEM-EUSO. Cabe señalar que estas señales de entrada y salida aún no han sido procesadas, es decir que no han pasado por un convertidor ADC o un multiplexor, por lo que esta es la cantidad de entradas y salidas que se requieren para el HK mas no para el FPGA. Se analizara la forma de manejar estas señales en el siguiente tema durante la planeación del primer diseño.

3.2.2 Primer diseño

En un primer análisis se obtuvo que se requieren 264 pines de entrada/salida tipo *single-ended* y 810 pares de pines diferenciales de entrada/salida en caso de que se decidiera conectar cada subsistema de forma directa al HK, sin embargo se optó por utilizar multiplexores y expansores para esta primer propuesta con el fin de reducir el número de líneas de conexión necesarias y de esta forma facilitar el diseño del subsistema.

Por lo tanto para el primer diseño se consideró la creación de diferentes interfaces para poder comunicarse con cada uno de los subsistemas, atendiendo a los requerimientos de cada uno como se explica en el capítulo 1. A continuación de describe este diseño.

3.2.2.1 Interfaces con LVPS

Como ya se ha explicado con anterioridad, el HK tiene tres funciones principales en su interacción con el subsistema LVPS: encender y apagar los diversos subsistemas lo cual se conoce como HLC (*High Level Command*), recepción de señales de CC (*Contact Closure*) y adquisición de voltajes analógicos.

Para encender o apagar cada subsistema se requiere activar 44 relevadores, en la Figura 3.8 se puede observar el manejo de las 44 señales de HCL para realizar esta terea. Para este diseño se propone utilizar 3 demultiplexores 1:16.

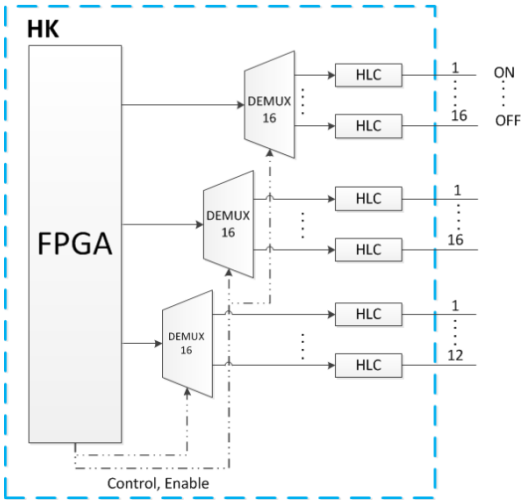


Figura 3.8. Interfaces para encender/apagar subsistemas.

Para la parte de adquisición de señales CC se tiene una interfaz con 2 multiplexores como se muestra en la Figura 3.9.

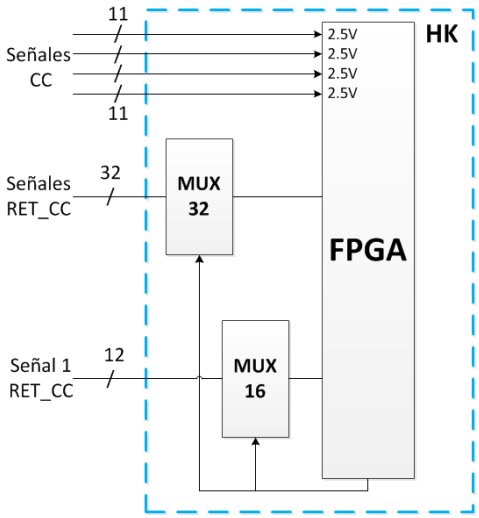


Figura 3.9. Interfaces para recibir las señales CC.

Por último, para la adquisición de señales analógicas las cuales representan tanto voltajes como corrientes, se tiene una interfaz compuesta de 6 multiplexores de 8 canales diferenciales , los cuales se conectan a 6 amplificadores de instrumentación para eliminar ruido y finalmente la señal va a un convertidor analógico digital donde finalmente pasa al FPGA para ser procesada como se observa en la Figura 3.10

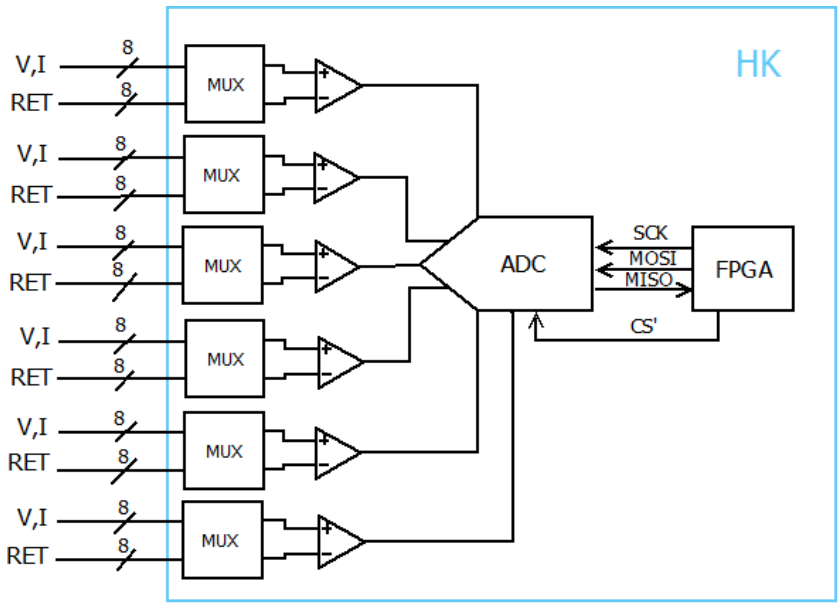


Figura 3.10. Interfaz para lecturas analógicas que corresponden a voltajes y corrientes.

3.2.2.2 Interfaz con CPU

El FPGA cuenta con un núcleo UART (*Universal Asynchronous Receiver-Transmitter*), que permite la comunicación serial entre el FPGA y CPU, que en este caso maneja el protocolo R2-422 cuyas

señales son de tipo diferencial, por lo que se requiere de un transceptor RS-422 entre el FPGA Y CPU como se muestra en la Figura 3.11.

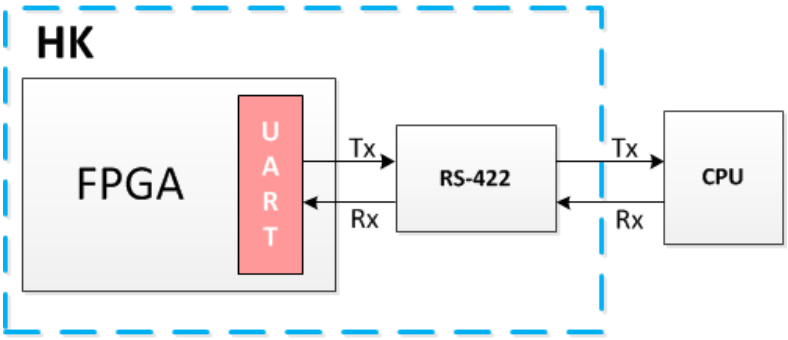


Figura 3.11. Interfaz para comunicación con CPU.

3.2.2.3 Interfaces con sensores de temperatura

Este diseño contempla el uso de termistores para medir la temperatura dentro del telescopio, los cuales se encontrarán distribuidos dentro de la estructura del JEM-EUSO. Para este diseño se contemplan un total de 400 termistores cuyas lecturas serán digitalizadas con un ADC y posteriormente leídas por el FPGA mediante el protocolo SPI y procesadas, el diagrama de esta interfaz se muestra en la Figura 3.12.

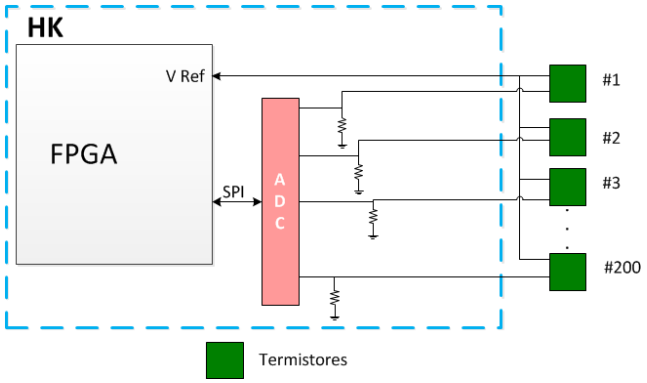


Figura 3.12. Interfaz con los termistores.

3.2.2.4 Interfaces con subsistemas SPI: CCB, PDM, CLKB, GPS

La interacción con los subsistemas CCB, PDM, CLKB y GPS se realiza a través del protocolo SPI el cual se empleara de forma diferencial con el objetivo de eliminar ruido. En este caso las señales SCK, MOSI, y MISO se toman directamente de las salidas diferenciales del FPGA y se comparten entre los subsistemas, mientras que las señales de selección de esclavo (*Chip Select*), se manejan como señales *single-ended* en el FPGA y después se convierten a modo diferencial con un transceptor LVDS, de modo que se pueda optimizar el número de pines utilizados en el FPGA. La conversión de estas señales se hace por medio de 21 decodificadores y 11 transceptores LVDS

para poder elegir entre los 162 dispositivos SPI que serán parte del JEM-EUSO como se observa en la Figura 3.13.

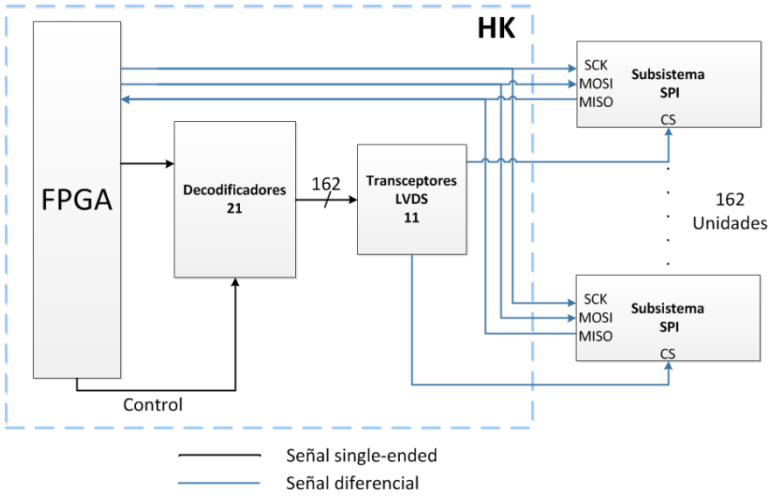


Figura 3.13. Interfaz con el subsistema PDM.

En cuanto a la recepción de las 322 alarmas se propone que se conviertan de señales diferenciales a señales *single-ended* mediante el uso de transceptores LVDS. Para el envío de los 322 pulsos en respuesta a las mismas, se propone el uso de multiplexores y convertidores LVDS y de esta forma optimizar el uso de los pines disponibles en un FPGA. El diagrama propuesto para la interacción con esta parte de los subsistemas se puede observar en la Figura 3.14.

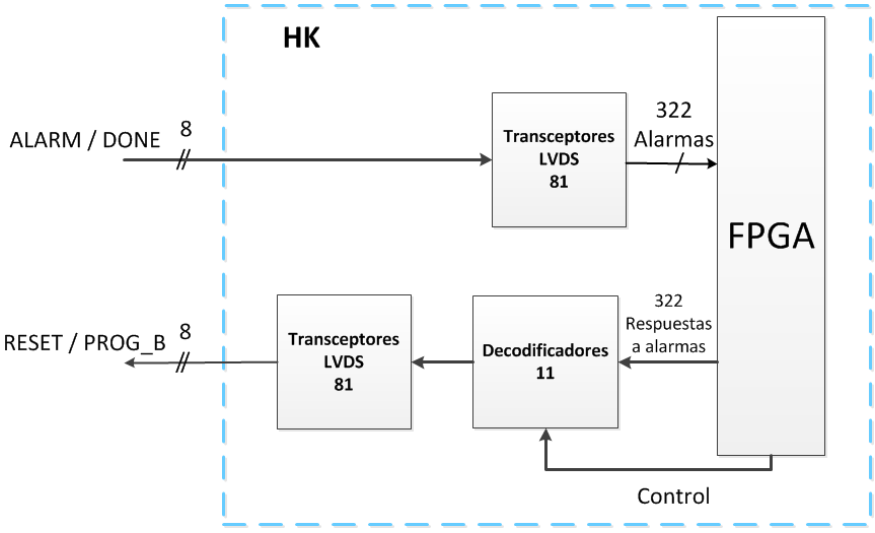


Figura 3.14. Interfaz para la recepción de alarmas y envío de *resets*.

El resultado de este nuevo análisis se puede observar en la Tabla 3.7 así como en la Figura 3.15 que muestra el ancho de los buses del HK.

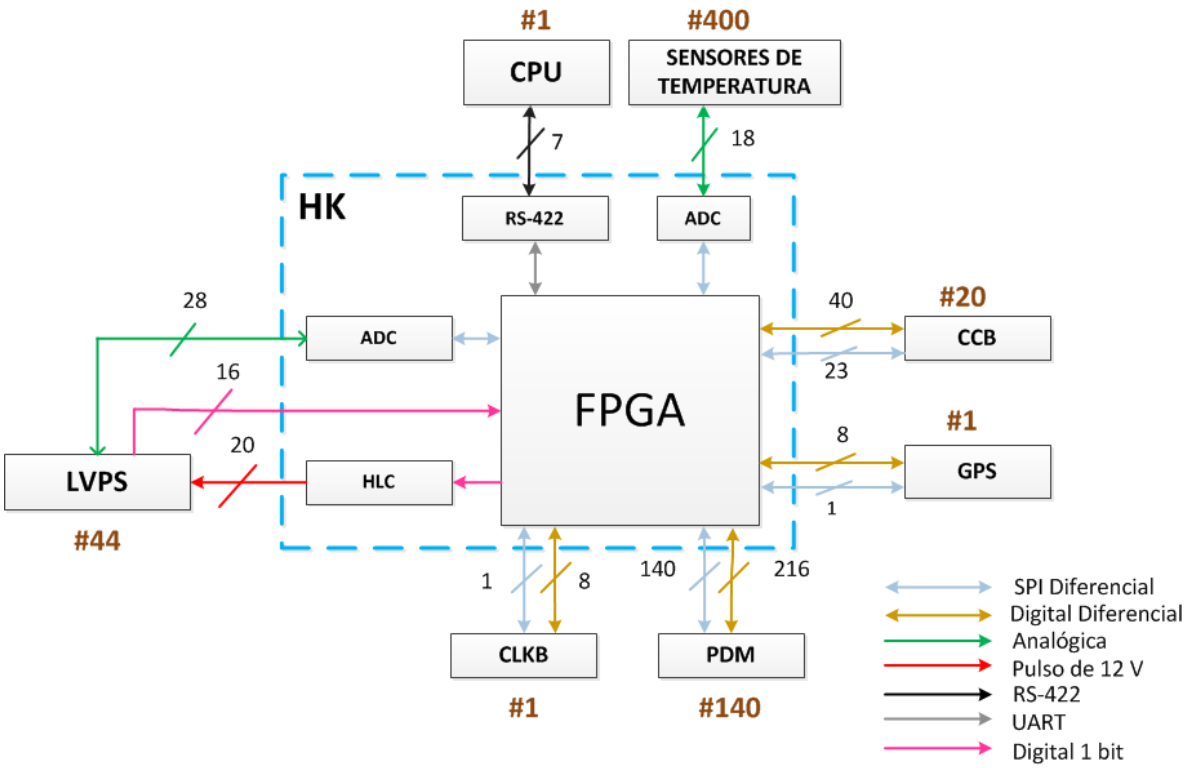


Figura 3.15. Diagrama que muestra el ancho de los buses de acuerdo a la tabla 3.7.

Estimación de los recursos de entrada/salida que requiere el Housekeeping						
Nombre del subsistema	**No. de tarjetas**	**Nombre de la señal**	**Descripción de la señal**	**Tipo de la entrada/salida**	**Cantidad de elementos de E/S**	**Detalles**
LVPS	44	Monitoreo de voltaje y corriente	2 señales por cada subsistema	*Single-ended*. SPI para el ADC, control del multiplexor	28	-
		Señal *Contact Closure*	1 señal por cada subsistema	*Single-ended*. Recive un estado alto o bajo.	16	Vref+control del expansor+lectura
		Encendido/apagado	2 señales por cada subsistema	*Single-ended.* Envía un pulso.	20	Control+encendido/apagado desde el FPGA
CPU	1	RS422	Señal diferencial para comunicación con CPU	*Single-ended*. Una línea de transmisión y otra de recepción.	2	Entradas para el integrado MAX3488
Comunicación RS-232	1	RS-232	Señal para comunicación con terminal de análisis	Single-ended. Una línea de transmisión y otra de recepción.	5	Entradas y control del integrado MAX3323E
Sensores de temperatura	400	Temperatura	Recibe un voltaje proporcional a la temperatura	Voltaje analógico	18	Señales para controlar 7 multiplexores
Subsistemas con interface SPI	20 CCB 140 PDM 1 CLKB 1 GPS	Comunicación SPI	162 esclavos (*Chip Select*). SPI diferencial	*Single-ended.* Envía una señal en estado bajo para habilitar un esclavo.	65	Control de expansores y convertidores LVDS
			SCK+MOSI+MISO	SPI diferencial. 3 pares de líneas son necesarias.	6	Salidas diferenciales del FPGA

		Señal *ALARM*	Recibe una señal en alto o bajo de cada uno de los 162 subsistemas	Entrada diferencial.	68	Control de expansores y transceptores LVDS
		Señal *DONE*	Recibe una señal en alto o bajo de cada uno de los 162 subsistemas	Entrada diferencial.	68	Control de expansores y transceptores LVDS
		Señal *RESET*	Envía un pulso para cada uno de los 162 subsistemas	Salida diferencial.	68	Control de expansores y transceptores LVDS
		Señal *PROGRAM_BIT*	Envía un pulso para cada uno de los 162 subsistemas	Salida diferencial.	68	Control de expansores y transceptores LVDS
*CCB *Control Cluster Board*					**Total de entradas/salidas 423**	
*PDM *Photo Detector Module*						
*CLKB *Clock Board*						
*GPS *Global Positioning system*						

Tabla 3.7. Estimación de los recursos entrada/salida para el Housekeeping

Con las interfaces descritas se cumplen las necesidades para la comunicación entre el HK y los subsistemas, pero se observan los siguientes problemas:

- El número de integrados que se necesitan es de aproximadamente 236 sin tomar en cuenta los conectores, resistencias, capacitores ni el FPGA. Se necesitarían aproximadamente 5.56 m^2 de área para la tarjeta, lo cual ya es un absurdo en cuanto a medidas se refiere.
- Luego se tiene que se necesitaría un área de aproximadamente 667 cm^2 para los 215 conectores que serán los que conectaran todas las señales de los subsistemas hacia el HK, por lo que se estima que en total la tarjeta ocuparía un área de al menos 5.6 m^2, lo cual es una proporción ilógica de implementar tomando en cuenta los requerimientos mencionados en el capítulo 1, donde se especifica un área de 200×233 mm.
- No todos los componentes que se plantean en los diseños tienen su componente análogo de grado espacial. Si los circuitos integrados se reemplazan por otros que hagan la misma función pero que sean de grado espacial, su número aumenta dado que por ejemplo se ocupan en este diseño multiplexores de 32 bits y solo existen en el mercado multiplexores de 8 bits, por lo que es fácil ver que la cantidad de elementos podría al menos triplicarse, lo cual a su vez aumentaría el área de la tarjeta, haciendo este diseño no viable de implementar.
- Un punto común de falla para el HK es el FPGA ya que al ser un sistema centralizado todo se concentra en éste.

Estas razones hacen que un diseño con arquitectura centralizada no sea viable, ya que a pesar de que la capacidad lógica del FPGA es suficiente existen restricciones físicas que no hacen viable un diseño de este estilo.

3.2.3 Segundo diseño

Debido a las limitaciones físicas que supone una arquitectura centralizada, se propone ahora un diseño para el HK basado en una arquitectura distribuida. En este diseño se ha decidido conservar al FPGA como la unidad lógica de control debido a sus características de flexibilidad y escalabilidad y dejar los procesos de control locales en una unidad lógica que no requiere flexibilidad como lo es un microcontrolador. Lo que se busca con este diseño, es minimizar el área que ocupa nuestro subsistema, por lo que para este nuevo esquema de instrumentación, es muy importante considerar el tamaño de cada una de las tarjetas que lo integrarán.

Dado lo anterior, el diseño de esta arquitectura se realizó bajo la consideración de que los componentes utilizados en las tarjetas que la integran, debían tener un componente análogo con calificación espacial, ya que esto implica que el área de las tarjetas es mayor en comparación con una tarjeta cuyos componentes fueran de grado comercial. Este es el peor caso y es por esto que se ha elegido trabajar bajo esta condición, con el objetivo de proponer un diseño que sea viable de implementar en un futuro, a pesar del área que los componentes puedan requerir. Esto no implica

que el diseño final no pueda tener componentes comerciales o que deba ser construido solo con componentes de grado 1, ya que esta es una decisión que se deberá tomar más adelante en base a un análisis detallado que no se realizará en este trabajo. Cabe mencionar que para este segundo diseño se ha eliminado el uso de expansores ya que no existen con calificación espacial y serían un elemento indispensable dentro del HK por lo que no se quisiera arriesgar el diseño al utilizar estos componentes.

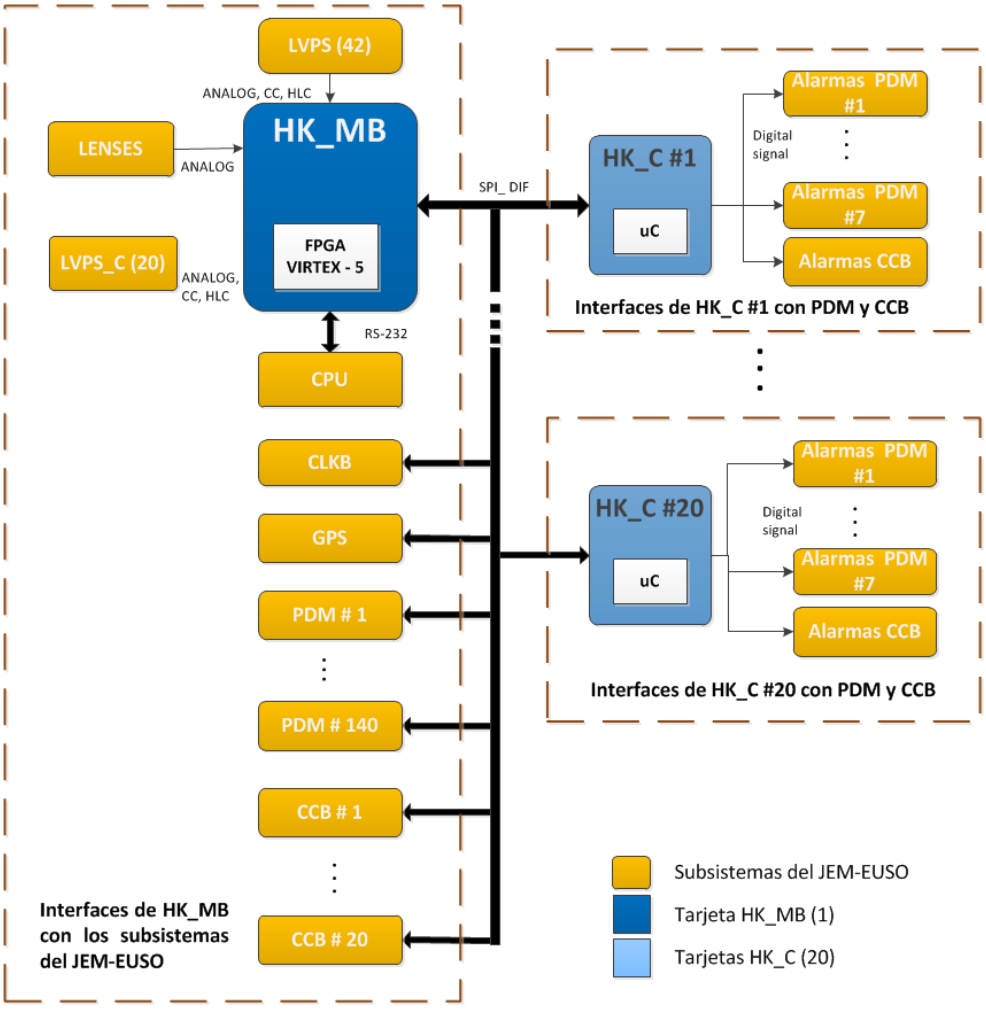


Figura 3.16. Segunda propuesta con arquitectura distribuida para el Housekeeping.

3.2.3.1 Tarjetas HK_C: primera propuesta

En este segundo diseño se propone que el HK esté formado de 20 tarjetas auxiliares que tengan un control local sobre las alarmas diferenciales: *ALARM* y *DONE* de los subsistemas PDM y CCB. Esta tarjeta se ha nombrado como *HK_C*, que significa *Housekeeping Cluster*, debido a que controlará *clusters* de subsistemas. Como ya se ha mencionado, la unidad de estas tarjetas se

propone que sea un microcontrolador, haría falta un análisis más para definir cuál sería el más conveniente, sin embrago no se abordará en esta tesis dado que en este momento se quiere enfocar el trabajo al análisis de la capacidad del FPGA para ser la unidad lógica principal.

3.2.3.2 Tarjetas HK_MB: primera propuesta

Se tendrá una tarjeta principal la cual se ha nombrado *HK_MB* que significa *Housekeeping Motherboard*, y la cual se encargará de la lectura de los subsistemas que utilizan el protocolo SPI diferencial, es decir, de la lectura de PDM, CCB, CLKB y GPS, además de encargarse de la lectura y control del subsistema LVPS, de los termistores y de la comunicación con CPU y obviamente del monitoreo de las 20 tarjetas HK_C. Aparece en este diseño la interacción con 20 tarjetas más las cuales pertenecen al subsistema LVPS, las cuales es necesario considerar dado que es necesario tener control sobre el encendido y apagado de las 20 tarjetas HK_C. Por último, la unidad lógica de esta tarjeta y por lo tanto del HK que se propone es un FPGA.

De este modo el FPGA seguiría teniendo el control de todos los subsistemas, incluso sería capaz de monitorear los datos de los subsistemas que se controlan por medio de las tarjetas HK_C. Este nuevo diseño reduce el espacio en las tarjetas debido a que en lugar de tener el diseño propuesto en la Figura 3.14 en una sola tarjeta, se dividiría entre 20 tarjetas.

Las ventajas de este diseño son:

- La reducción del espacio del HK como una sola área, lo cual hace más viable su implementación.
- La posibilidad de implementar un sistema con tolerancia a fallas en las tarjetas HK_C, ya que se tiene la ventaja de que serán iguales, lo que permite pensar en un sistema redundante.

Las desventajas de este diseño:

- El costo por la reducción de área es el aumento de la cantidad de tarjetas, lo cual podría traducirse por ejemplo, en un aumento en el peso del HK debido a los cables que se tendrán que utilizar para comunicar HK_MB con HK_C.
- La tarjeta HK_MB seguiría siendo demasiado grande ya que contendría al menos los 160 conectores DB25 para adquirir las señales de los subsistemas PDM y CCB.

3.2.4 Tercer diseño: arquitectura final

Finalmente se llegó a la conclusión de que un sistema distribuido es lo más conveniente, con una tarjeta principal HK_MB y 20 tarjetas de control local HK_C para los *clusters* de subsistemas. A diferencia del segundo diseño, en esta nueva arquitectura se aprovecha el hecho de tener 20 tarjetas con un microcontrolador para que estas se encarguen de monitorear no solo las alarmas de PDM y CCB, sino que también lean los datos que estos envían a través del protocolo SPI diferencial, de este modo el tamaño de la tarjeta HK_MB se reduce y el tamaño de HK_C aumenta pero no a un nivel que sea inviable su implementación, además la tarea de interactuar con el

subsistema LVPS también se comparte entre las 20 tarjetas; este diseño se puede observar en la Figura 3.17.

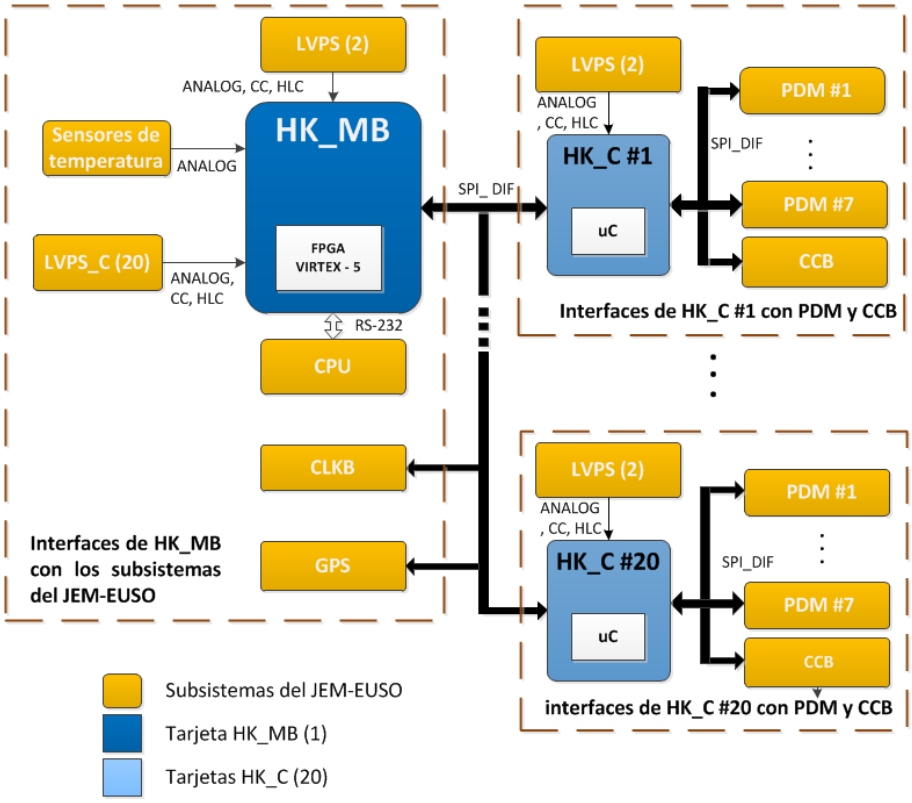


Figura 3.17. Tercera propuesta con arquitectura distribuida para el Housekeeping.

De este modo, los subsistemas CCB, PDM y LVPS son monitoreados por las tarjetas HK_C y aunque estas tarjetas pueden tomar decisiones locales sobre estos subsistemas, las tareas importantes siguen estando a cargo de la unidad principal de control, por ejemplo la atención a las alarmas de alta prioridad. Mientras tanto, la unidad lógica principal se encarga de monitorear a estas tarjetas para obtener los datos adquiridos y procesados por las mismas. También monitorea los subsistemas restantes: CLKB, GPS, termistores, LVPS y atiende los comandos del subsistema CPU.

Con esta propuesta las tarjetas HK_MB y HK_C trabajarán en conjunto como una sola unidad, de modo que los comandos de CPU llegarán siempre a la tarjeta HK_MB y ésta ya se encargará de distribuir las tareas a las tarjetas de control local. Más adelante se dará una descripción más detallada de estas tarjetas.

Ventajas de este diseño:

- Se tiene una reducción de espacio en la tarjeta HK_MB.
- Se tiene un sistema distribuido que facilita la atención a los subsistemas PDM y CCB.

- El FPGA comparte el procesamiento de la información del JEM-EUSO con 20 microcontroladores.

Desventajas de este diseño:

- Todos los componentes que se han considerado se basan en el requerimiento de poder encontrar un análogo espacial, es decir que en este diseño no se ha considerado aún la mezcla de componentes comerciales con componentes espaciales, lo cual hace que este diseño sea muy caro en caso de implementarse.
- Lo anterior implica que este no es un diseño con tolerancia a fallas, aunque existe la posibilidad de realizar esta implementación. Sin embargo, en este momento este diseño aun no es el más eficiente al que se puede llegar.

3.2.4.1 Descripción de la funcionalidad de las tarjetas HK_MB y HK_C

Como ya se mencionó en el punto 3.2.3, el HK estará formado por dos tipos de tarjetas, la tarjeta principal llamada HK_MB (*Housekeeping Motherboard*) y las 20 tarjetas adicionales llamadas HK_C (*Housekeeping Clusterborad*), las cuales en la arquitectura distribuida serán las tarjetas de control de grupos (clusters) de subsistemas. A continuación se proporciona una descripción más detallada de las tarjetas.

Tarjetas HK_C: segunda propuesta

Para esta tercer propuesta de la arquitectura se propone de la misma forma que en el diseño dos, usar 20 tarjetas auxiliares (HK_C) pero que ahora tengan más tareas que las anteriores, es decir que se encarguen de clusters de los subsistemas PDM, CCB y LVPS. De la misma forma se piensa en que la unidad lógica de estas tarjetas sea un microcontrolador dado que no se necesita de un sistema más complejo para implementar las tareas antes mencionadas.

Las tareas que realizará esta tarjeta son:

- Lectura de datos de PDM y CCB mediante el protocolo SPI diferencial.
- Control local sobre las alarmas diferenciales de *DONE* provenientes de los subsistemas PDM y CCB y las señales PROGRAM_BIT.
- Generación de alarmas de alta prioridad por la señal de *ALARM*.
- Envío de las señalas de *RESET* a PSM y CCB.
- Lecturas analógicas del subsistema LVPS.
- Encendido y apagado de PDM y CCB por comando de la tarjeta de control principal.
- Generación de alarmas de alta y bajar prioridad.
- Envío de todos los datos a la tarjeta HK_MB con el protocolo SPI diferencial.

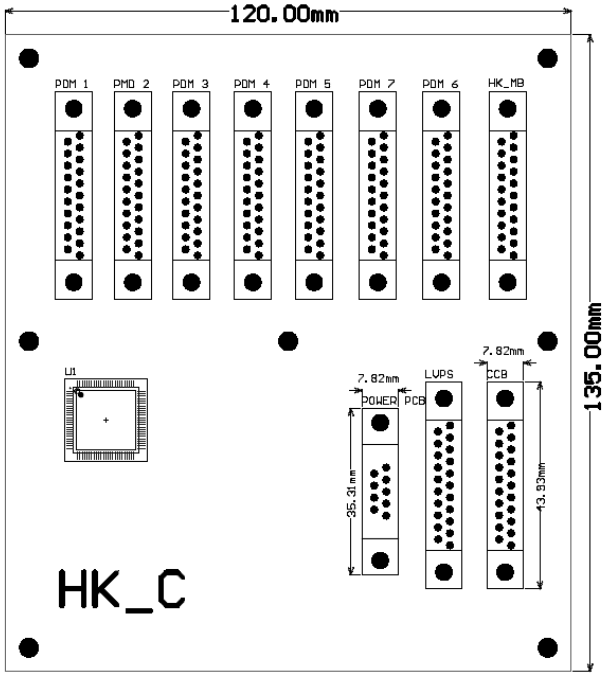


Figura 3.18. Distribución de los elementos de la tarjeta HK_C.

Tarjetas HK_MB: segunda propuesta

Las tareas de la tarjeta principal *HK_MB* se reducirán, ya que en vez de encargarse de la lectura de los 140 PDM y 20 CCB por SPI, solo se encargará de la lectura de CLKB, GPS y de las 20 tarjetas HK_C, las cuales son las que se encargarán de recolectar la información de PDM y CCB enviándole la información correspondiente al FPGA, de modo que este siga teniendo la visualización completa del subsistema. Además, como en el diseño anterior, HK_MB continuara a cargo de la lectura y control del subsistema LVPS, de los termistores y de la comunicación con CPU.

Las tareas que realizará esta tarjeta son:

- Atención a los comandos de CPU.
- Envío de alarmas a CPU.
- Lectura de los sensores de temperatura.
- Lectura por SPI diferencial de CLKB, GPS y las tarjetas HK_C.
- Atención a las señales ALRAM y DONE provenientes del CLKB.
- Encendido y apagado de subsistemas por comando de CPU.
- Lecturas analógicas del subsistema LVPS.
- Envío de comandos por SPI diferencial a las tarjetas HK_C.

En resumen, para este nuevo diseño se tiene lo siguiente:

Ventajas

- Este nuevo diseño reduce el espacio que ocupa HK_MB ya que en lugar de tener el diseño propuesto en la Figura 3.16 con 180 esclavos SPI se tienen solo 20 que reúnen la información.
- Se puede aprovechar el hecho de que las 20 tarjetas son iguales para implementar un sistema tolerante a fallas.
- Si se tiene alguna falla en alguna tarjeta solo se perdería un cluster de siete tarjetas PDM y una CCB con sus respectivas fuentes de alimentación.

Desventajas

- Inducción de ruido debido a la distancia que podría haber entre HK_MB y HK_C.
- Punto común de falla en el HK_MB.

Una vez que se ha decidido cómo estará conformado el HK, se ha hecho un diseño sencillo de las tarjetas HK_MB y HK_C en donde se incluyen los elementos más representativos del hardware y los conectores que se necesitarán para recibir las señales de los subsistemas, sobre todo este esquema se ha hecho para tener una idea aproximada de la medida de las tarjetas y que no sea un diseño inviable de implementar por su tamaño. Los esquemas se pueden observar en la Figura 3.18 y Figura 3.19.

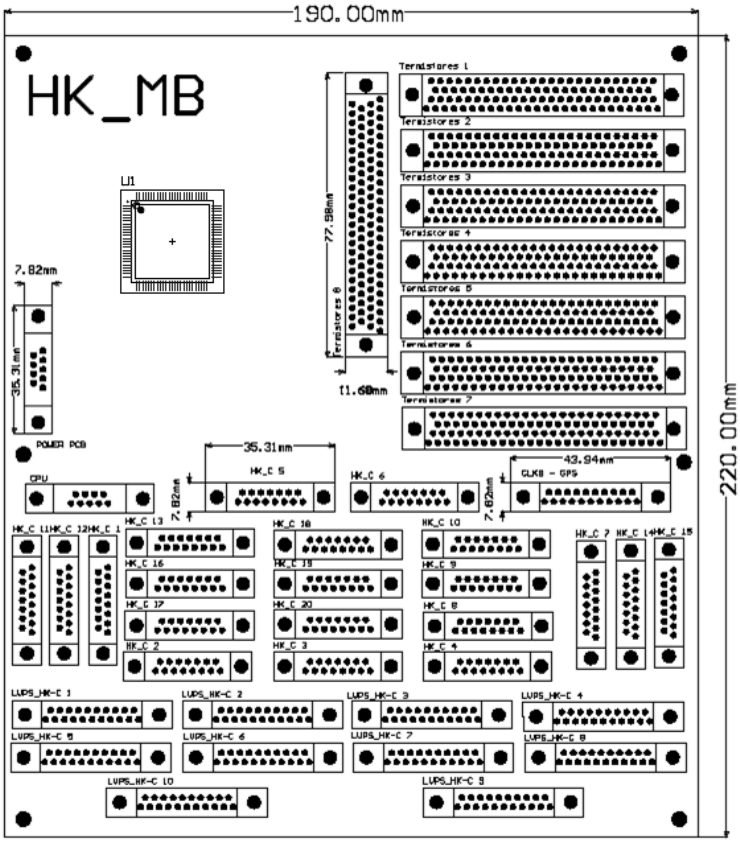


Figura 3.19. Distribución de elementos de la tarjeta HK_MB.

3.2.4.2 Sistema embebido

Ya que se ha propuesto el esquema de instrumentación para el HK, el segundo paso es implementar en el FPGA el sistema embebido para la arquitectura computacional y la lógica necesaria, para analizar la viabilidad de la Virtex-5 como unidad lógica del HK.

Qué es un sistema embebido

Se le llama sistema embebido a aquel dispositivo que ejecuta una tarea específica y que es controlado internamente por su propio microprocesador o microcontrolador, en vez de poseer un sistema de control externo [33]. Actualmente existe la posibilidad de incluir en un solo chip, el microprocesador con todos los elementos que necesita para realizar sus tareas como puertos de entrada/salida, buses, memoria, etc. Por lo que es posible tener un sistema embebido en un solo integrado, lo cual ahorra espacio, costo y consumo de potencia. Un sistema embebido en un FPGA se puede describir por medio de un lenguaje HDL, o también se puede utilizar una plataforma de desarrollo de sistemas embebidos.

VHDL

VHDL es un acrónimo que combina las siglas VHSIC (*Very High Speed Integrated Circuit*) y HDL (Hardware Description Language), y junto con el Verilog, es uno de los lenguajes que más se utilizan para describir la arquitectura de circuitos digitales sobre dispositivos configurables.

El código VHDL puede cumplir con uno de los tres estilos de descripción siguientes:

1. Una descripción estructural. Describe la estructura del circuito en términos de las compuertas lógicas utilizadas en la interconexión entre estas para formar un mapa de las conexiones o *netlist*.
2. Una descripción del flujo de datos. Describe la transferencia de datos de la entrada a la salida y entre las señales.
3. Una descripción del comportamiento. Describe el comportamiento del diseño en términos del comportamiento del circuito y el sistema utilizando algoritmos.

En VHDL, el diseño es creado inicialmente como la declaración de una entidad y un cuerpo para la arquitectura. La declaración de la entidad describe el diseño de las entradas y salidas e incluye parámetros que personalizan esta entidad. La entidad puede visualizarse como una caja negra con las conexiones de entadas y salidas visibles. En el cuerpo de la arquitectura se describe la forma interna de trabajo de la entidad por lo que aquí se contiene una combinación de descripción de estructura, de flujo de datos o de comportamiento, que describen la forma de trabajo de la entidad.

En el caso del HK, si se pensara en este subsistema como un núcleo cuya arquitectura se quiere describir en VHDL, resultaría en un sistema muy complejo por las tareas que este debe de realizar. Por ejemplo, se podría pensar en describir el comportamiento de este subsistema a través de la creación de una arquitectura basada en máquinas de estado, sin embrago, el resultado sería una

compleja máquina con una enorme cantidad de estados, lo cual le restaría flexibilidad al diseño ya que en el caso de que se quisiera hacer una modificación en el diseño, por ejemplo una reconfiguración en tiempo real, se necesitaría afectar a el diseño general por lo que el tiempo que requerido para realizar un cambio hace de ésta una opción ineficiente para implementar el HK en un FPGA por medio de VHDL.

Es por esto, que se ha decidido utilizar herramientas de alto nivel que permiten la creación de un sistema embebido más complejo, en donde se tendrá la oportunidad de tener un sistema flexible, que se puede modificar en cualquier momento sin que esto represente una pérdida en tiempo de diseño o complejidad en el mismo.

3.2.4.3 Entorno de desarrollo del sistema embebido

Para crear un sistema embebido en un FPGAse requiere diseñar hardware, software e integrarlos después para que puedan funcionar en conjunto como un solo sistema. En Xilinx® se utiliza el entorno de desarrollo ISE (*Integrated Software Environment*), el cual tiene una herramienta para diseño de sistemas embebidos llamada EDK (*Embedded Development Kit*), la cual a su vez posee dos herramientas para trabajar en el diseño de software y hardware:

XPS (Xilinx Platform Studio)

XPS brinda una interfaz gráfica de usuario (GUI: *Graphic User Interface*) que le facilita al diseñador de hardware construir, conectar y configurar sistemas embebidos basados en un procesador en poco tiempo [34].

Mediante XPS es posible realizar las siguientes tareas:

- Añadir núcleos del catálogo de Xilinx y editar sus parámetros, así como realizar sus conexiones.
- Añadir núcleos personalizados al sistema.
- Crear y modificar los archivos MHS (Microprocessor Hardware Specification) y MSS (Microprocessor Software Specification).
- Generar y visualizar un diagrama de bloques del sistema, así como el reporte del diseño.

SDK (Software Development Kit)

Es una interfaz gráfica para el usuario (GUI: Graphic User Interface) complementaria a XPS, que permite la creación de aplicaciones embebidas en cualquiera de los procesadores de Xilinx®. Algunas de sus características son las siguientes [35]:

- Tiene un editor de código C/C++, junto con un entorno para corrección de errores, depuración de código y compilación.
- Posee librerías personalizadas y controladores (drivers) para sus dispositivos.

Con la herramienta EDK, se tiene la capacidad de trabajar en el diseño de hardware y el diseño de software por separado, por lo que se puede trabajar de forma independiente sobre ambos diseños permitiendo finalmente la integración de ambos para conformar el sistema embebido.

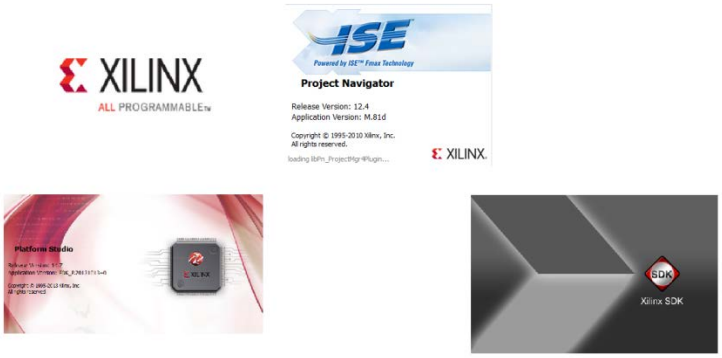


Figura 3.20. Herramientas para el desarrollo de sistemas embebidos de XILINX®.

Como ya se ha mencionado en el apartado 2.4, se ha decidido implementar la lógica de control principal del subsistema HK en un FPGA Virtex®-5. El sistema que se ha decidido implementar utiliza uno de los softprocesors que provee Xilinx®, en este caso un Microblaze™, como la unidad lógica central de este sistema embebido.

El HK lleva a cabo tareas cíclicas es decir hay tareas que requieren de una secuencia y dado el número de subsistemas con los que debe comunicarse y los procesos que se deben llevar a cabo (lecturas analógicas, generación de alarmas, respuesta a interrupciones, etc.), sería bastante complejo implementar el subsistema descrito en VHDL (*Very High Speed Integrated Circuit Hardware Description Language*) utilizando máquinas de estado para crear el HK en un solo núcleo, por lo que esta opción resulta ineficiente para cubrir las necesidades del subsistema, además de que le resta flexibilidad al mismo, lo cual es contrario a lo que se busca al utilizar un FPGA para esta tarea. Por esta razón es que se ha decidido utilizar un sistema embebido cuya estructura consiste en tener un microprocesador embebido en el cual se tiene la posibilidad de agregarle hasta 16 núcleos que lleven a cabo diversas tareas, desde enviar pulsos hasta generar interrupciones o interactuar mediante el uso de diversos protocolos de comunicación con dispositivos externos. Este microprocesador será el encargado de controlar la lógica de los procesos del HK.

Microprocesador embebido en el FPGA

Un microprocesador (μP), es un circuito integrado que es programable por medio de software, pero en el caso de un FPGA puede ser también un núcleo sintetizado en HDL que se puede agregar al diseño del sistema embebido en el FPGA. Las instrucciones que luego se le programen, serán las que definen las tareas que realizará el procesador.

Como ya se ha mencionado, se puede diseñar y embeber un microprocesador en un FPGA, el cual adquirirá el nombre genérico de núcleo. En este caso, cuando un núcleo es diseñado por una compañía o un desarrollador independiente al proyecto en donde se utiliza, se le llama *IP core* (Intelectual Property Core) [20]. Estos núcleos embebidos son optimizados y permiten a los FPGAs que los contienen, convertirse en un una solución muy adecuada al problema permitiendo también disminuir el área en las tarjetas electrónicas y su consumo de energía [18].

Xilinx® ofrece dos tipos de microprocesadores dependiendo de la familia a la que pertenezca el FPGA que se adquiera. Están los llamados *hard processor core* y los *soft processor core.*

Hard Processor Core

Es un bloque de hardware que está físicamente construido en el dispositivo como parte de la tarjeta de silicio, cuya funcionalidad se construye mediante transistores al manufacturarlo [15]. Los dispositivos resultantes son rápidos y sus resultados son precisos, pero los parámetros de su construcción no pueden modificarse, lo que los hace poco flexibles.

Soft Processor Core

En contraste con un *hard core*, un *soft core* se implementa con los recursos reconfigurables del FPGA, generalmente están descritos en HDL, por lo que requieren ser compilados por una herramienta de síntesis. Estos núcleos son altamente flexibles, porque se pueden modificar sus características, e incluso cambiar su código, pero al no estar diseñados para un dispositivo en específico son menos eficientes [21].

Microblaze™

El MicroBlaze™ es un microprocesador RISC de 32 bits que forma parte de la plataforma de desarrollo EDK (Embedded Development Kit) de Xilinx®. Este es un microprocesador embebido suave (SPC: *soft processor core*) con una arquitectura harvard, lo que significa que utiliza buses separados para acceso a los datos e instrucciones almacenados en los bloques de memoria RAM [23]; puede operar con arquitectura *pipeline*de 3 estados para optimizar espacio o puede operar en 5 estados para optimizar velocidad. También se pueden configurar sus propiedades como elegir el tamaño de la memoria caché para instrucciones y datos, configurar el uso de un administrador de memorias, utilizar unidad de punto flotante, entre otras. En cuanto a los recursos, ocupa de 716-1300 LUTs, de 299-1600 FFs, y de 1-32 bloques de RAM. El número total de instrucciones que soporta son 87, si se consideran diferentes las instrucciones que operan con valores inmediatos de

aquellas que realizan la misma operación con registros. Adicionalmente, cada instrucción se ha elegido para que el tamaño de la ALU también sea reducido.

Este SPC puede utilizarse en las familias Virtex® y Spartan-3™, pero se tiene la ventaja de que el un diseño puede fácilmente migrarse a dispositivos de nuevas familias, lo que hace que este procesador no se vuelva obsoleto [36].

Figura 3.21. Diagrama de bloques del microprocesador embebido MicroBlaze [47].

Buses del MicroBlaze™

MicroBlaze utiliza el estándar CoreConnect [43] creado por IBM, para conectar diferentes elementos en un circuito integrado. Un aspecto interesante es que CoreConnect permite reducir la carga capacitiva del bus, repartiéndola entre varios buses. Así, se consiguen mayores rendimientos, dado que los retardos de pistas globales son muy importantes en FPGAs. A continuación se describen los buses.

Bus PLB

El PLB (*Processor Local Bus*) es el bus que permite la conexión entre un procesador con otros núcleos. El PLB v4.6 que provee Xilinx es de 128 bits y provee de la infraestructura para conectar maestros y esclavos PLB a un PLB general, en el caso del Microblaze® es incluso posible conectar otro procesador idéntico. El bus incluye una unidad de control, un temporizador watchdog, direcciones de memoria separadas para la escritura y la lectura y una interfaz esclava opcional de control DCR (Device Control Register) que permite el acceso al bus de errores y estado de los registros[22]. En el PLB v4.6 se pueden conectar hasta 32 maestros y 16 esclavos.

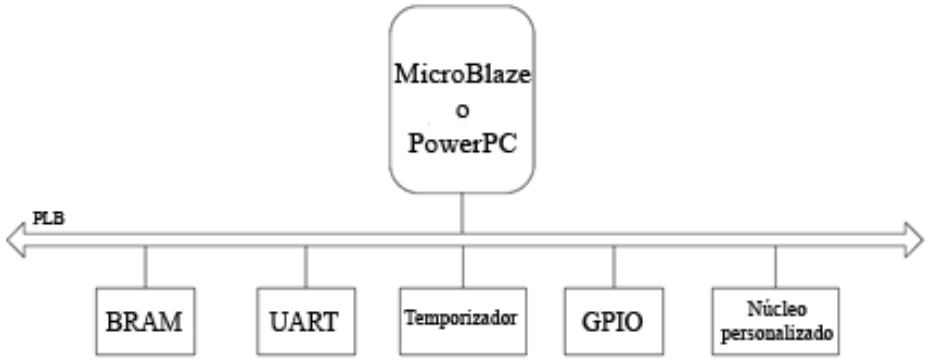


Figura 3.22. Comunicación de otros núcleos con el microprocesador a través del bus PLB.

Bus DCR

El DCR (*Device Control Register*) es un bus síncrono diseñado para una conexión tipo anillo de periféricos con ancho de banda muy bajo. Solo soporta un maestro y está diseñado para minimizar el uso de lógica interna [44].

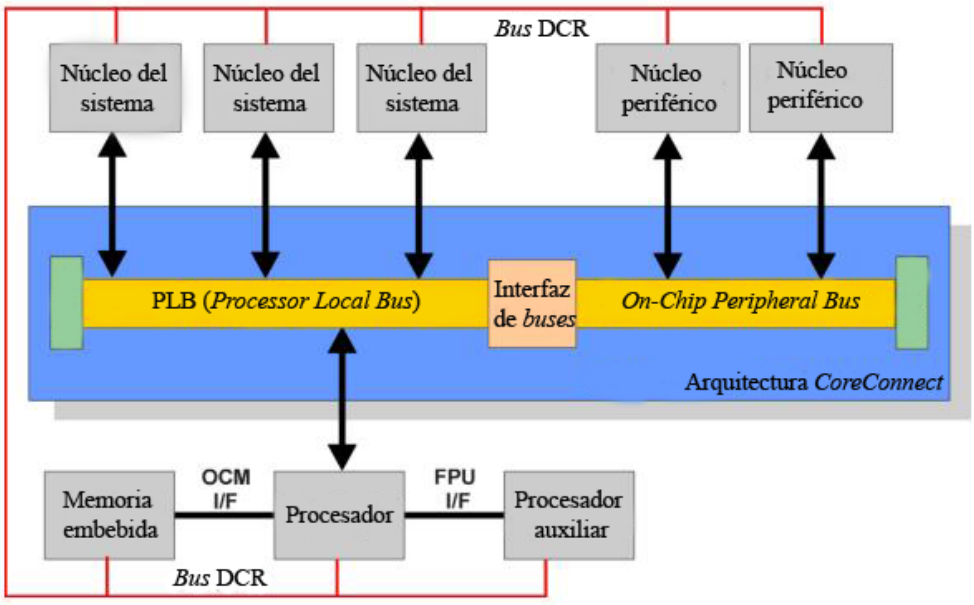


Figura 3.23. Diagrama de bloques del CoreConnect [43].

Bus LMB

El LMB (*Local Memory Bus*) es un bus síncrono de alta velocidad, utilizado para conectar periféricos y los bloques de memoria interna de la FPGA. Solo admite un maestro en su implementación para MicroBlaze y puede ser utilizado tanto para instrucciones como para datos.

Este bus es compatible con el PLB (Processor Local Bus) incluido en el estándar CoreConnect, pero a diferencia de éste, el LMB no admite varios maestros ni tamaños de palabra diferentes a 32 bits.

Bus OPB

El OPB (*On-chip Peripheral Bus*) es un bus síncrono utilizado para conectar periféricos con tiempos de acceso variables. Soporta varios maestros y la conectividad de muchos periféricos es sencilla gracias a su identificación por multiplexación distribuida. Soporta transferencias de tamaño de palabra dinámico.

El primer paso para realizar la implementación del sistema embebido en el FPGA fue identificar los núcleos que se iban a necesitar para integrar la arquitectura del sistema, lo cual, de acuerdo a las necesidades del diseño tres para el HK, se ha analizado en la Tabla 3.8.

Núcleos que requiere el sistema embebido para implementar el Housekeeping						
Nombre del subsistema	**Número de tarjetas**	**Nombre de la señal**	**Descripción de la señal**	**Tipo de la entrada/salida**	**Cantidad de elementos de E/S**	**Núcleos**
LVPS	22	Monitoreo de voltaje y corriente	2 señales por cada tarjeta, 44 señales en total	*Single-ended*. SPI para el ADC, control del multiplexor	4+18=22	GPIO#1, spi#1
		Señal *Contact Closure*	1 señal por cada subsistema	*Single-ended*. Recibe un estado alto o bajo.	22	GPIO#1 (Canal 2)
		Encendido/apagado	2 señales por cada subsistema	*Single-ended.* Envía un pulso.	44	GPIO#2 (Canal 1 y 2)
CPU	1	RS-422	Señal diferencial para comunicación con CPU	*Single-ended*. Una línea de transmisión y otra de recepción.	2	1 UARTLITE
Sensor temperatura	400	Temperatura	Recibe un voltaje proporcional a la temperatura	Señal digital I2C	12	6 núcleos I2C
Subsistemas con interface SPI	20 HK_C 1 CLKB 1 GPS	Comunicación SPI	22 esclavos (*Chip Select*). SPI diferencial	*Single-ended.* Envía una señal en estado bajo para habilitar un esclavo.	44 pines(22 diferenciales)	GPIO#3
			SCK+MOSI+MISO	SPI diferencial. 3 pares de líneas son necesarias.	6 pines (3 diferenciales)	SPI#2 (Diferencial)

		Señal *ALARM*	Recibe una señal en alto o bajo de cada uno de los 22 subsistemas	Entrada diferencial.	44 pines(22 diferenciales)	GPIO#4
		Señal *DONE*	Recibe una señal en alto o bajo de GPS o CLKB	Entrada diferencial.	4 pines (2 diferenciales)	GPIO#4
		Señal *RESET*	Envía un pulso para cada uno de los 22 subsistemas	Entrada diferencial.	44 pines(22 diferenciales)	GPIO#5
		Señal *PROGRAM_BIT*	Envía un pulso para cada uno de los 22 subsistemas	Entrada diferencial.	44 pines(22 diferenciales)	GPIO#5 (Canal 2)
*CCB *Control Cluster Board*					Total de núcleos esclavos del PLB	14 Núcleos
*PDM *Photo Detector Module*					Núcleos adicionales	INTERRUPT CONTROLLER
*CLKB *Clock Board*						WATCHDOG
*GPS *Global Positioning system*						I2C (EEPROM)
						TIMER

Tabla 3.8 . Núcleos necesarios para la arquitectura embebida.

Diseño en XPS

En esta herramienta es donde se diseña el hardware que se embeberá en el FPGA y que estará basado en el uso del Microblaze™, es decir que aquí se diseña la arquitectura de la unidad lógica principal del HK de acuerdo a los requerimientos del mismo, que en este caso ya se han visto en la Tabla 3.8. En esta plataforma se tiene un catálogo de núcleos multipropósito que se pueden utilizar pero también se pueden crear núcleos personalizados y anexarlos al proyecto.

Para este diseño se comenzó anexando un microprocesador Microblaze™ con su correspondiente bus de comunicación PLB (Processor Local Bus), el cual es el que le permitirá mantener una comunicación con los núcleos esclavos que a su vez se comunicarán con el entorno de trabajo del HK, para que exista la interacción FPGA con los subsistemas del telescopio.

En la Figura 3.24 se observa que se han añadido 5 núcleos I2C, los cuales servirán para leer los 400 termistores a través de un convertidor analógico-digital. Se ha decidido trabajar con este protocolo ya que ocupa menos líneas que el protocolo SPI.

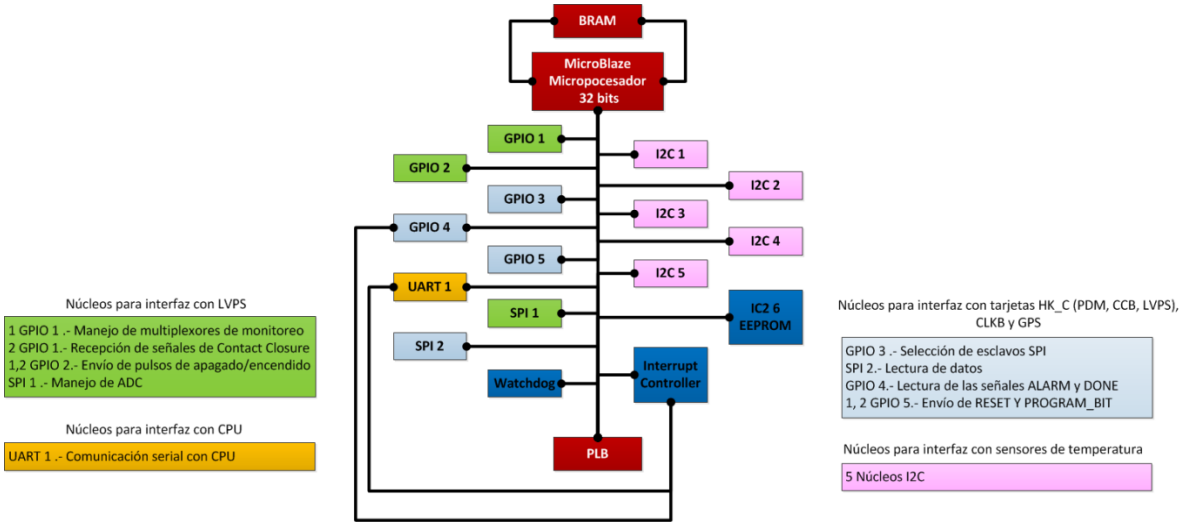


Figura 3.24. Arquitectura embebida diseñada para el Housekeeping.

En teoría el núcleo I2C de Xilinx® soporta de acuerdo a las especificaciones de Philips (referencia) un total de 128 esclavos, sin embargo, este protocolo se ve limitado por la capacitancia que se pueda generar en las líneas de transmisión, ya que la línea I2C soporta un máximo de 400pF de capacitancia para que su velocidad de transmisión no se vea afectada, por lo que se deberá cuidar que no se conecten más ADC de los que puede soportar este bus.

Diseño en SDK

En esta plataforma se desarrolla el software, ya sea en lenguaje C o C++, que le indicará al microprocesador la forma en la que debe interactuar con cada uno de los núcleos de la

arquitectura embebida para llevar a cabo la tarea de monitorear los subsistemas, generar alarmas, atender interrupciones, etc. que en conjunto crean lo que sería la lógica del HK. El código generado se puede observar en el anexo A de esta tesis. A continuación se muestran los diagramas de flujo de la lógica implementada en la arquitectura embebida.

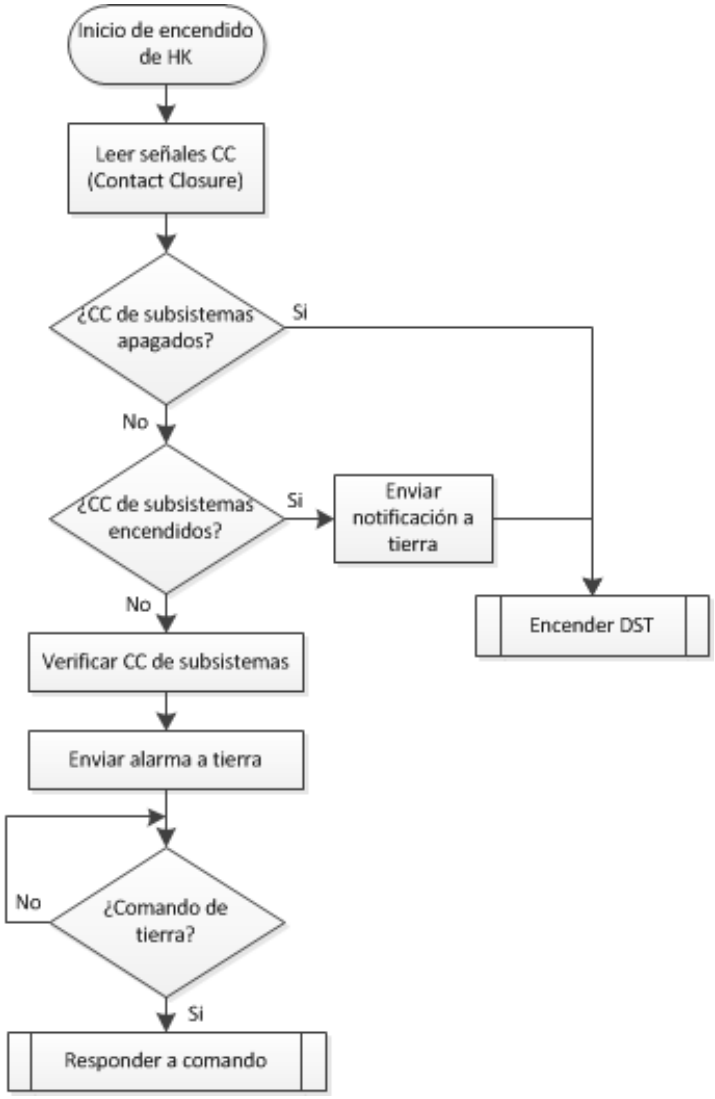


Figura 3.25. Rutina de inicio al encender el HK.

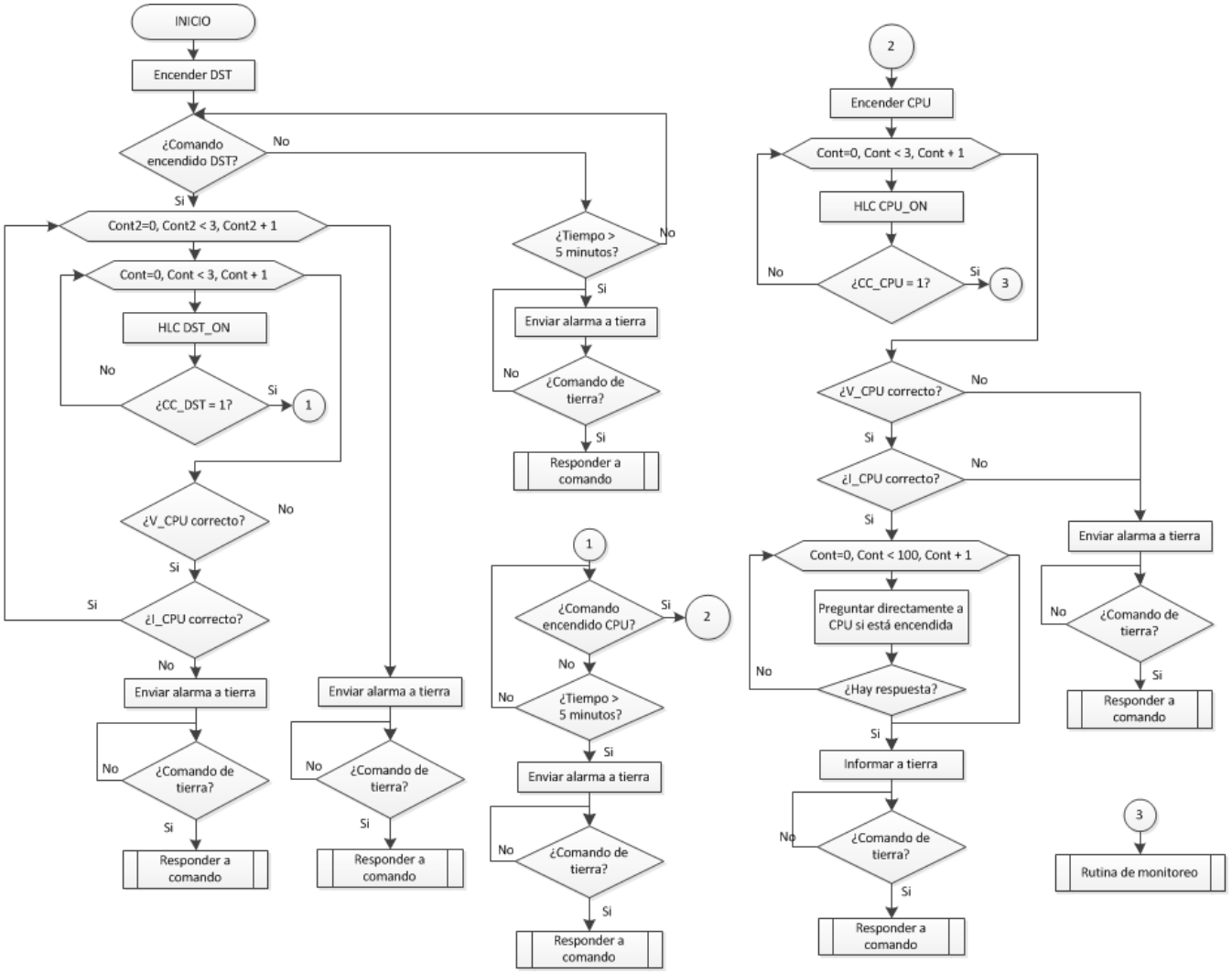


Figura 3.26. Diagrama de flujo de encendido de DTS y CPU.

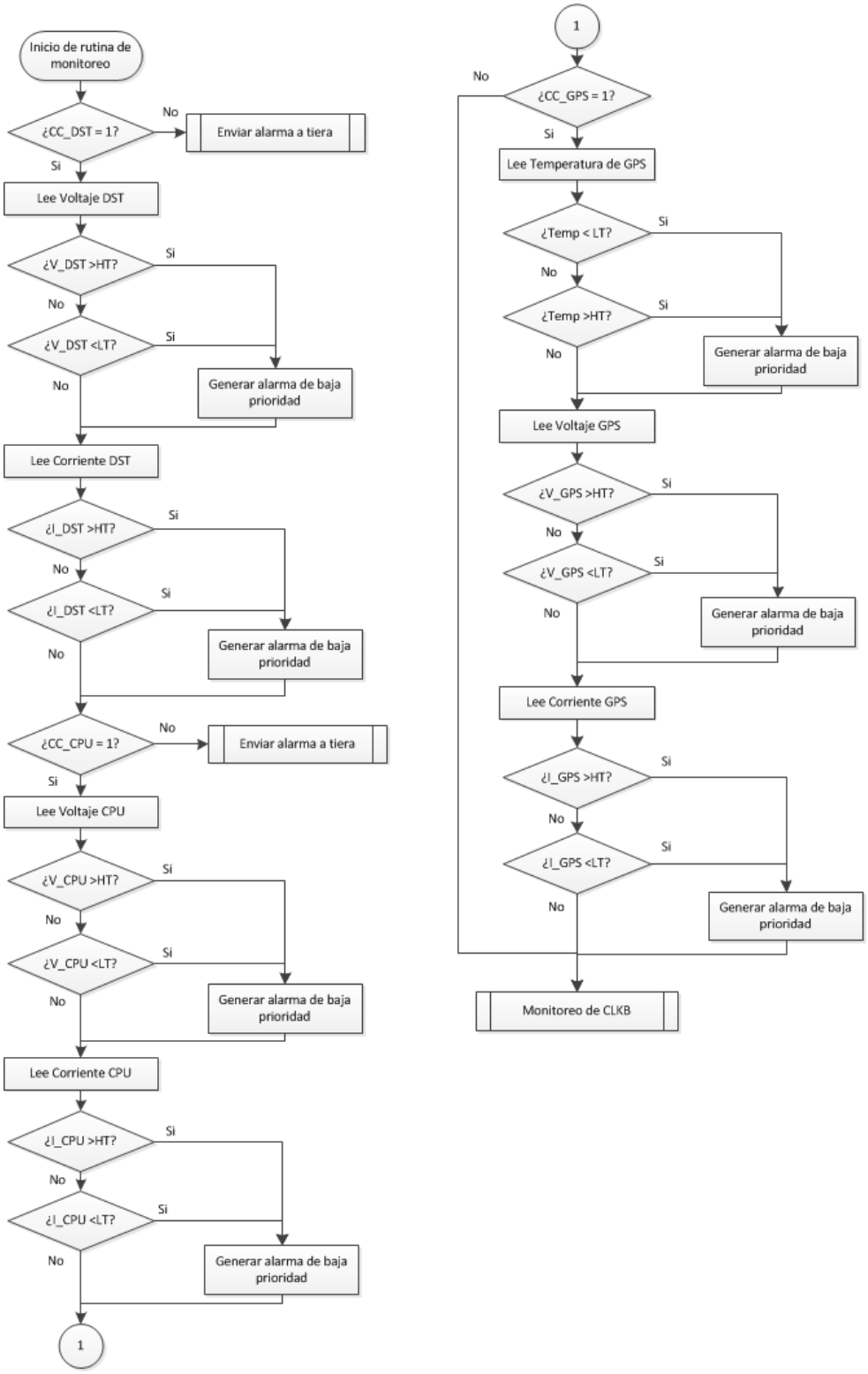


Figura 3.27. Diagrama de flujo del monitoreo de DST, CPU y GPS.

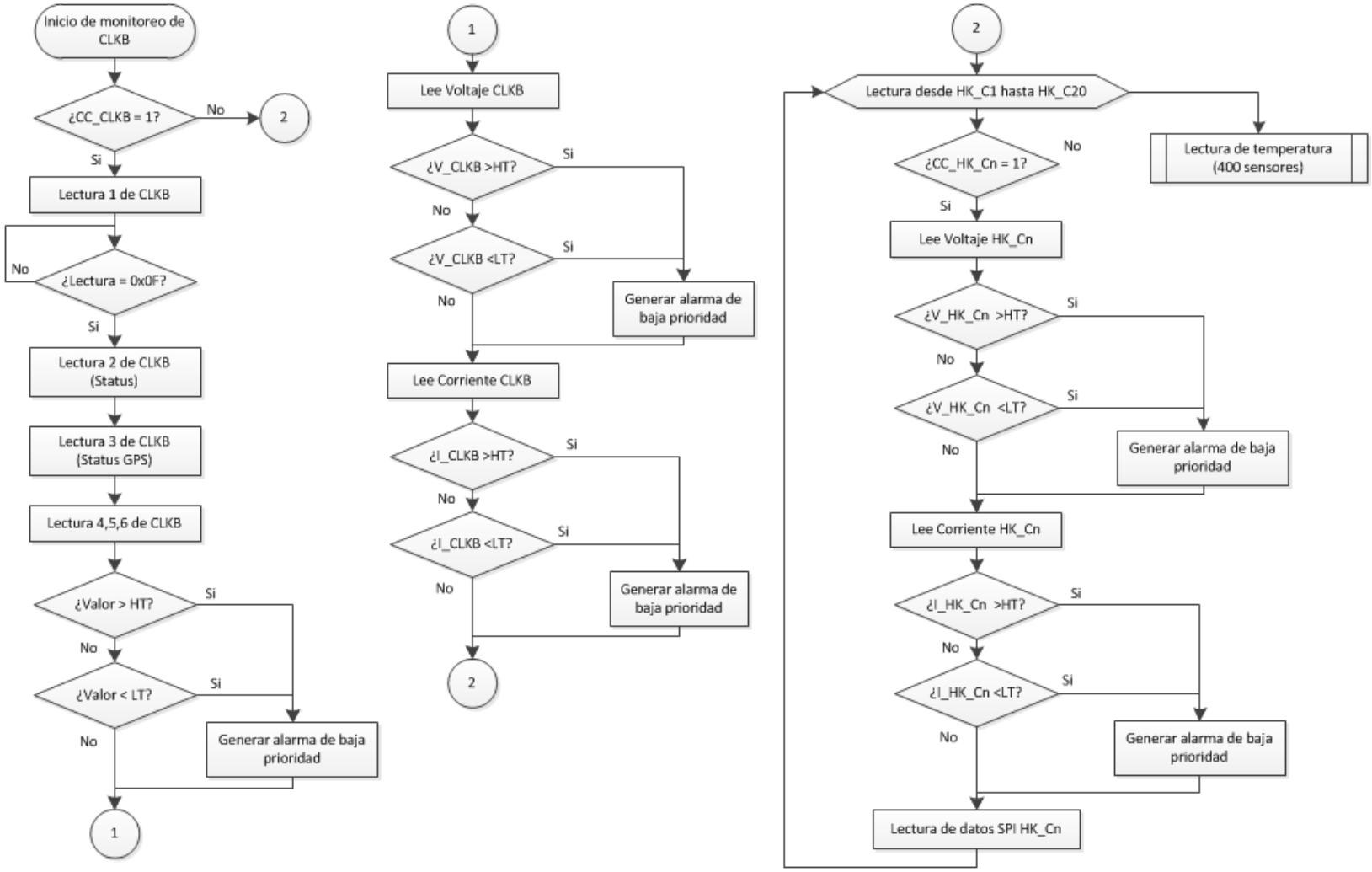


Figura 3.28. Diagrama de flujo de monitoreo de CLKB, HK_C y sensores de temperatura.

z

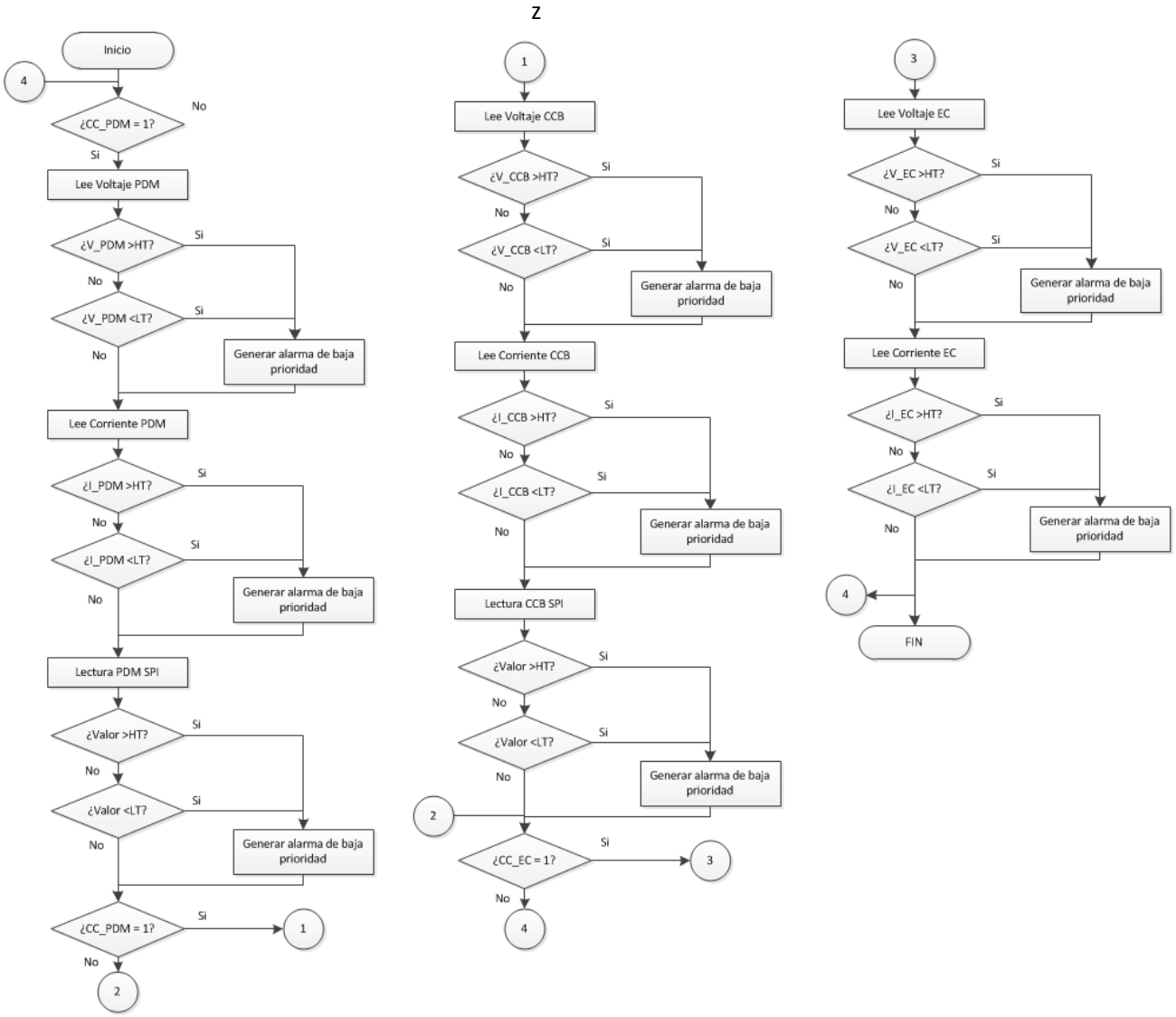


Figura 3.29. Diagrama de flujo de las tarjetas HK_C.

Capítulo 4
Estrategias de validación de la arquitectura propuesta

4.1 Metodología para realizar pruebas

Como se puede observar en la tabla del apéndice A, se proponen diversas pruebas para validar el funcionamiento del sistema propuesto para el HK. Sin embargo, se han elegido las tareas más representativas e importantes para simularse a través de la construcción de un sistema mínimo el cual se muestra en la Figura 4.1. Este sistema mínimo contiene todos los tipos de interfaces que se utilizarán en el HK, así como algunos subsistemas que se comunican a través de cada una de estas interfaces.

Específicamente, se ha elegido trabajar con:

CPU. De este sistema se recibirán diversos comandos para encender y apagar subsistemas así como para realizar lecturas de subsistemas, es uno de los subsistemas más importantes con los que el HK debe poder interactuar.

LVPS_CLKB y *LVPS_PDM*. Se ha elegido trabajar con dos subsistemas LVPS, uno que depende directamente de la tarjeta HK_MB y otro que depende de la tarjeta HK_C, para de este modo verificar la correcta interacción de estos subsistemas con ambas tarjetas, es decir, verificar su interacción con el HK.

PDM y CCB. Estos subsistemas se comunican con el HK mediante protocolo SPI diferencial. Específicamente, se comunican con la tarjeta HK_C y se ha elegido trabajar con estos dos esclavos para verificar la adquisición, procesamiento y envío de la información de estos dos subsistemas con los que deberá interactuar la tarjeta HK_C.

GPS. Este es un subsistema que es esclavo de la tarjeta HK_MB junto con CLKB y las tarjetas HK_C; en este caso se eligió este subsistema para validar que se pueden adquirir y procesar los datos de diversos esclavos del SPI diferencial de la tarjeta HK_MB.

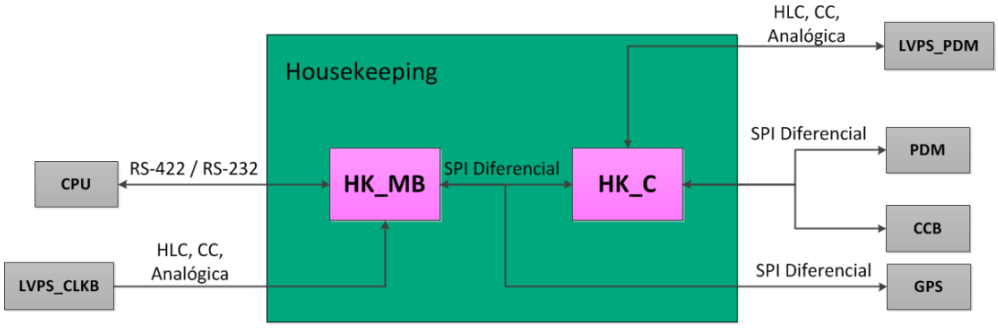


Figura 4.1. Sistema mínimo para realizar pruebas de funcionamiento de la arquitectura propuesta para el Housekeeping.

Con este sistema mínimo se puede comprobar la funcionalidad de la arquitectura propuesta de una forma cualitativa. Sin embargo, el objetivo de este trabajo es el analizar también las implicaciones de este diseño para el FPGA Virtex®-5 de forma cuantitativa, por lo que se han elegido algunos procesos a validar de una forma tanto cualitativa como cuantitativa. Para esto, lo primero que se hizo fue identificar las partes más importantes que representan a este subsistema, que son:

- *Comunicación con CPU*. Esta parte del subsistema HK es de vital importancia, ya que CPU es el subsistema encargado de controlar el telescopio en su totalidad, y el HK está subordinado a este. Es decir que si fallara la comunicación con CPU el HK no podría realizar sus tareas debido a que depende de los comandos que se le indiquen ejecutar.

- *Comunicación con HK_C*. Dentro de la misma arquitectura del HK, se ha considerado que la comunicación exitosa con entre HK_MB con sus 20 tarjetas HK_C es de vital importancia debido a que estas tarjetas contendrán la información de los detectores del telescopio, que son los subsistemas más importantes del JEM-EUSO, por lo que una falla en la comunicación con estas tarjetas significaría la pérdida de comunicación con una parte del telescopio.

- *Comunicación entre HK_C con PDM, CCB y LVPS*. Esta es la tercera tarea más importante, ya que como se ha mencionado en el punto anterior, cada tarjeta HK_C tiene a su cargo el monitoreo de un grupo de subsistemas y es importante validar que la comunicación con este grupo de subsistemas sea exitosa, ya que de otro modo el HK estaría perdiendo toda la información de los mismos.

A su vez, de las tareas anteriores se ha elegido estudiar las dos primeras, ya que en otros experimentos que se han llevado a cabo para el JEM-EUSO, se ha validado la lectura de los subsistemas PDM, CCB y LVPS con un microcontrolador; por lo que se prefiere, debido al tiempo para desarrollar esta tesis, validar aquellas tareas con las cuales aún no se ha experimentado. Es por esto que se ha propuesto el sistema de la Figura 4.2, el cual contiene los elementos necesarios para llevar a cabo los análisis. Así mismo, en la tabla del apéndice B se pueden observar las pruebas a las que será sometido el sistema para verificar si los puntos mencionados anteriormente se cumplen de manera satisfactoria o en caso contrario, identificar posibles puntos de falla.

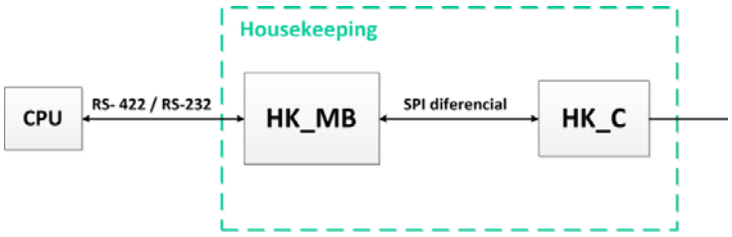


Figura 4.2. Sistema para validar pruebas cuantitativas de la Virtex-5 en el HK.

4.2 Tarjetas que se construyeron para las pruebas y validaciones

Con el propósito de recrear el sistema mínimo de la Figura 4.2, se simularon las tarjetas HK_MB y HK_C, las cuales contienen las interfaces necesarias para la comunicación con CPU y algunos subsistemas del telescopio, de forma que se puedan realizar las pruebas de validación descritas en la tabla del apéndice B.

4.2.1 Simulación de la tarjeta HK_MB

Para simular el funcionamiento de la tarjeta HK_MB, se ha decidido utilizar dos tarjetas, la tarjeta comercial XUPV5-LX110T la cual es una tarjeta de desarrollo que tiene un FPGA Virtex®-5 (Figura 4.3) y la tarjeta de interfaz denominada HK_MB_i (Figura 4.7) la cual se ha diseñado en *Altium Designer* y posee interfaces que le ayudan a la tarjeta de desarrollo a comunicarse con CPU a través de un convertidor RS422 a RS232, ya que se cuenta con un protocolo serial RS232 en la tarjeta comercial por lo que es necesario hacer esta conversión entre estándares, ya que como se mencionó, el subsistema CPU utiliza el protocolo diferencial RS422. El HK_MB_i también posee una interfaz que le permite adquirir datos analógicos del subsistema LVPS.

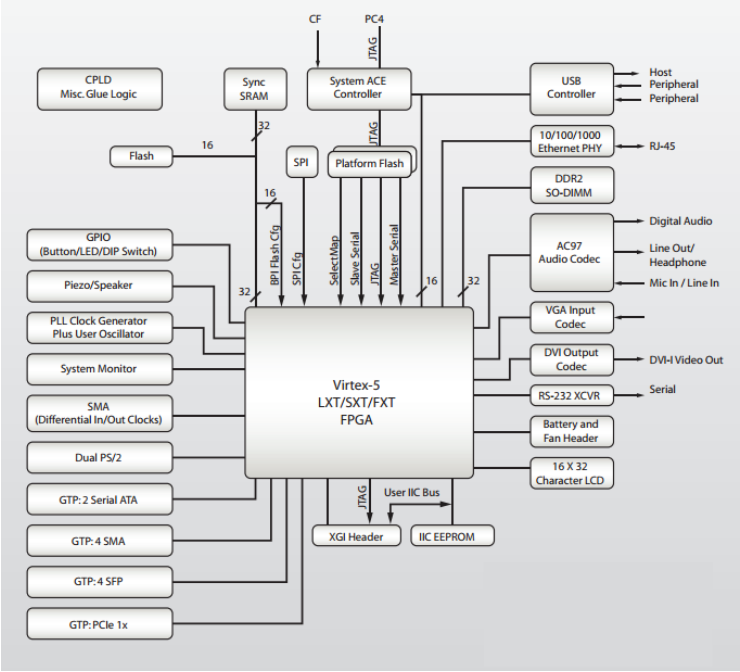


Figura 4.3. Diagrama de bloques de la tarjeta comercial XUPV5-LX110T.

La tarjeta XUPV5-LX110T, es una plataforma de desarrollo que funciona con un FPGA Virtex®-5. Algunos los elementos de esta tarjeta que se utilizan son el puerto serial RS-232, LEDs, *dip switchs*, y terminales de propósito general.

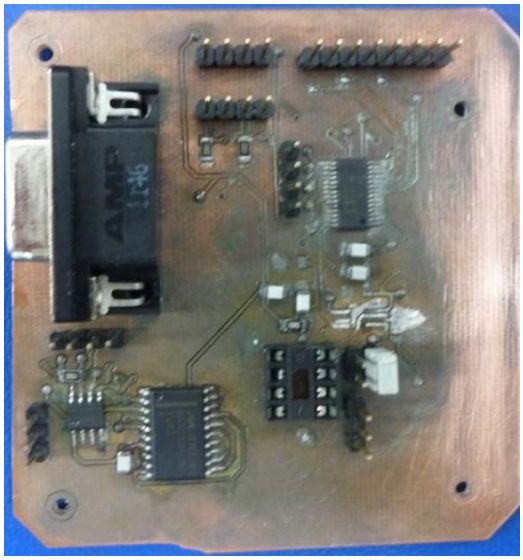


Figura 4.4. Fotografía de la tarjeta HK_MB_i.

En la Figura 4.5 se muestra cómo quedaría la integración de esta tarjeta con los respectivos bancos de prueba para realizar las pruebas de la tabla del apéndice B.

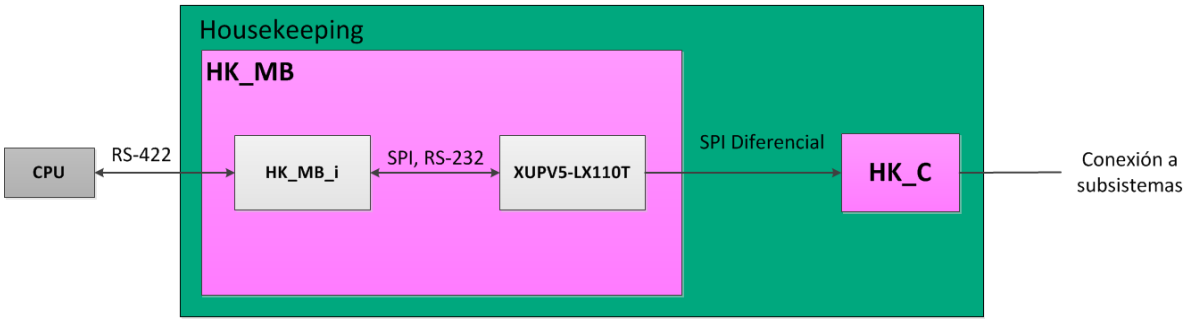


Figura 4.5. Simulación de la tarjeta HK_MB con HK_MB_i y XUPV5-LX110T.

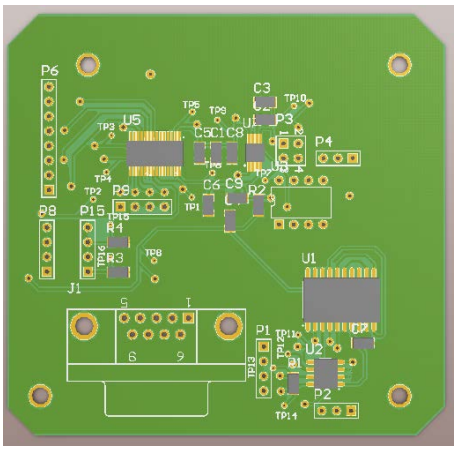


Figura 4.6. Vista 3D de la tarjeta HK_MB_i.

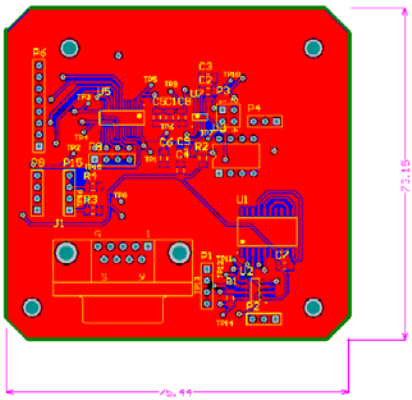


Figura 4.7. Layout de la tarjeta HK_MB_i.

4.2.2 Simulación de la tarjeta HK_C

Como ya se ha mencionado en el apartado 3.5, el HK_C tiene como tarea principal el comunicarse con un grupo de subsistemas y enviar esta información a la tarjeta HK_MB mediante protocolo SPI diferencial. Ya se ha dicho también que por el momento no se ha definido cuál es el microcontrolador que se utilizara, sin embargo, para fines de validación se ha elegido utilizar la plataforma Arduino Mega que contiene un microcontrolador ATmega 2560 de AVR®, en conjunto con una tarjeta de interfaz llamada HK_C_i que permitirá la interacción con el FPGA y con el grupo de subsistemas con los que se debe comunicar la tarjeta HK_C.

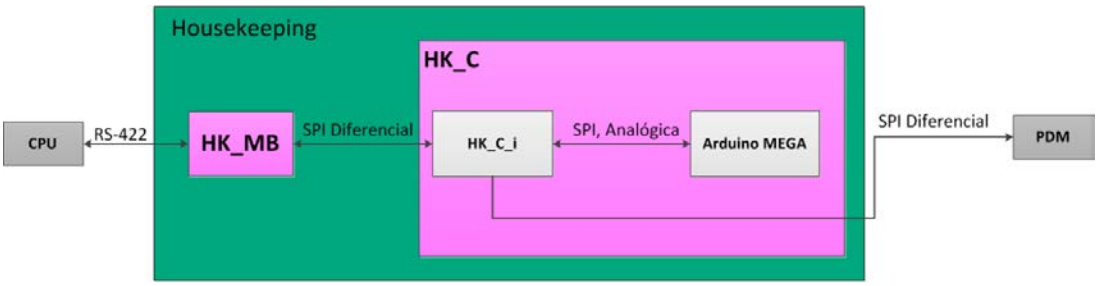


Figura 4.8. Simulación de la tarjeta HK_C con HK_C_i y Arduino MEGA.

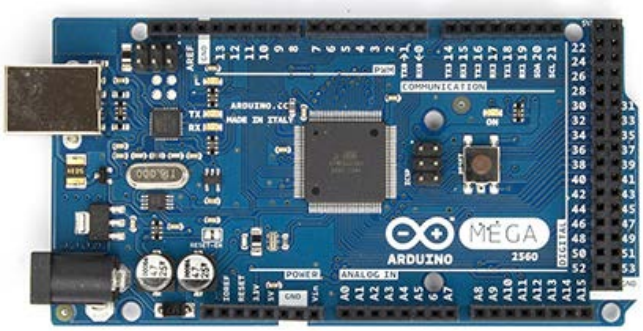


Figura 4.9. Arduino ATMEGA 2560.

La tarjeta HK_C_i por lo tanto, contiene una interfaz diferencial que utiliza transceptores LVDS para convertir las señales tipo *single* del microcontrolador a un estándar diferencial que es el que utilizan los subsistemas PDM y CCB. También tiene una interfaz de lecturas analógicas que utiliza multiplexores para reducir el número de pines de entrada del ADC a solo un pin de lectura.

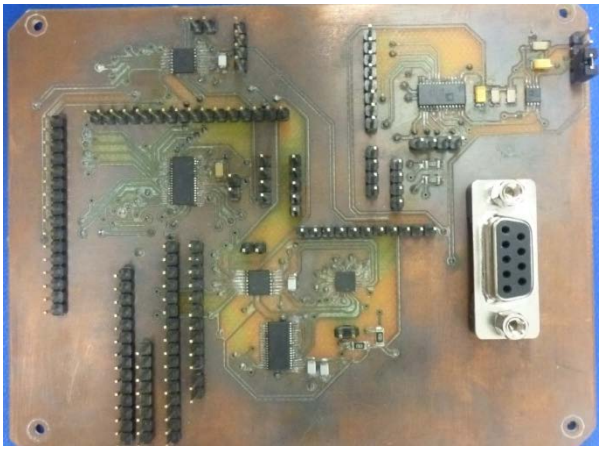


Figura 4.10. Fotografía de la tarjeta HK_C_i.

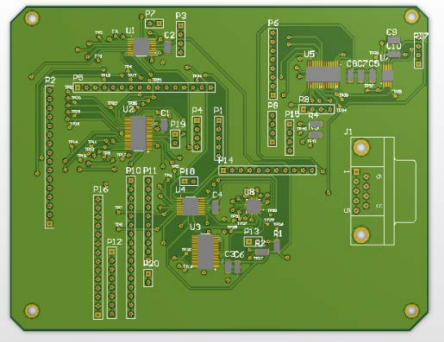


Figura 4.11. Viste 3D de la tarjeta HK_C_i.

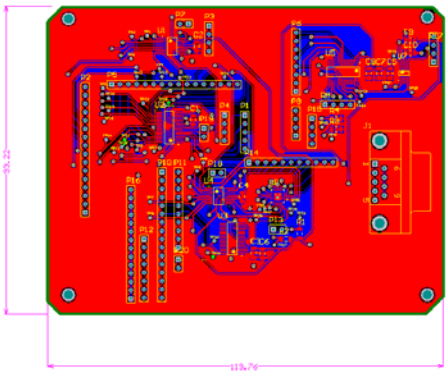


Figura 4.12. Layout de la tarjeta HK_C_i.

4.2.3 Tarjeta de interfaz para comandos de alto nivel (HLC: High Level Command)

Adicionalmente a estas tarjetas, se construyó una tarjeta basada en el modelo descrito en [6], ya que el encendido y apagado que se lleva a cabo a través de los relevadores en los subsistemas LVPS, requiere de un pulso de 12 V, por lo que esta tarjeta funciona como una interfaz entre el comando enviado ya sea por HK_MB o HK_C y el subsistema al que vaya dirigido el comando de encendido o apagado. En un futuro, esta interfaz deberá estar incluida tanto en la tarjeta HK_MB como en la HK_C.

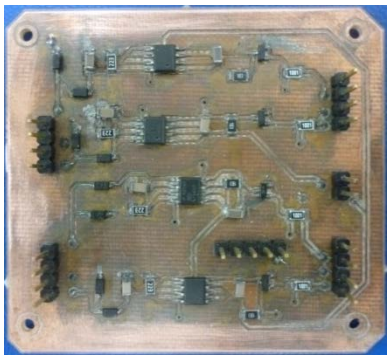


Figura 4.13. Fotografía de la tarjeta HLC.

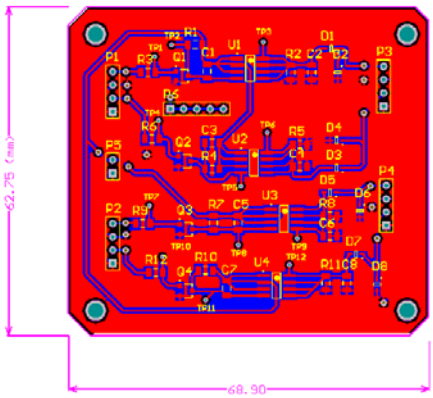


Figura 4.14. Layout de la tarjeta de HLC.

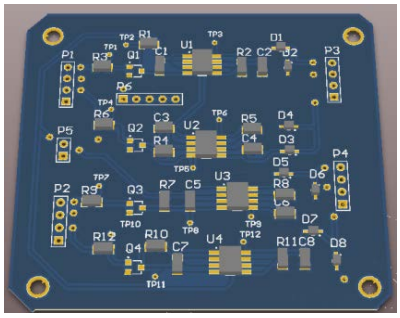


Figura 4.15. Vista 3D de la tarjeta de HLC.

4.3 Resultados de las pruebas de validación del HK con el sistema mínimo

Las siguientes pruebas se realizaron utilizando las tarjetas diseñadas en conjunto con plataformas de desarrollo. Se armó el sistema mínimo de la Figura 4.1 y se hicieron pruebas tanto de funcionalidad de la lógica de monitoreo del HK, así como de la comunicación entre tarjetas a través de la simulación de la condición de distancia entre ellas a la que serán sometidas las tarjetas del HK.

4.3.1 Resultados de las pruebas de funcionalidad del sistema mínimo

La primer prueba que se realizó, fue la de armar el sistema mínimo propuesto en la Figura 4.1, para comprobar que cada una de las distintas interfaces de comunicación con los subsistemas funcionaba de forma correcta. Los resultados se enlistan a continuación:

- La tarjeta HK_MB responde correctamente a las interrupciones de CPU y también envía de forma exitosa registros de alarmas al subsistema CPU.
- La tarjeta HK_MB adquiere y procesa de forma exitosa los datos de voltajes y corrientes de la LVPS correspondiente a CPU y DST.
- La comunicación SPI entre HK_MB y HK_C se realiza de forma exitosa y se tiene una lectura de datos completa. Sin embrago, no siempre hay respuesta a las interrupciones generadas desde HK_MB hacia HK_C, esto se debe a que hace falta una optimización del código descrito para la tarjeta esclava, y también puede deberse a la forma en la que este protocolo ha sido diseñado en este microcontrolador.
- La comunicación con otros esclavos SPI como el GPS desde la tarjeta HK_MB, se realiza de una forma exitosa.
- La lectura de los 7 PDMs, 2 LVPS y un CCB con la tarjeta HK_C, se realiza de una forma óptima, y se observa que si durante el monitoreo de estos subsistemas ocurre una interrupción, esta es atendida de forma correcta y hay un retorno exitoso al ciclo principal de la rutina de monitoreo.

Observaciones:

- El protocolo SPI trabaja a 4 MHz dado que es la máxima frecuencia que soporta la unidad lógica del esclavo (ATMEGA 2560). Sin embargo, esta tarjeta no necesariamente debe contener este microcontrolador que solo se ha utilizado para realizar una simulación. Este microcontrolador no responde al 100% a las interrupciones generadas, es necesario modificar el código para optimizar esta tarea.
- Es necesario agregar esperas del orden de milisegundos para realizar algunas lecturas debido a la forma en la que funciona el microcontrolador ATMEGA 2560.

4.3.2 Resultados de las pruebas de interacción entre HK_MB y HK_C

La comunicación entre estas tarjetas es de mucha importancia para el HK, debido a que en conjunto, las 20 tarjetas HK_C son las encargadas de monitorear los dos subsistemas conformados por la cantidad más grande de elementos del JEM-EUSO: 140 tarjetas PDM y 20 tarjetas CCB, además de monitorear sus respectivas fuentes de alimentación. Por esto se debe validar que existe una correcta transmisión de datos, ya que de lo contrario se estaría perdiendo información sumamente valiosa.

Los objetivos de esta prueba fueron:

- Verificar la transmisión de datos en ambas direcciones entre las tarjetas.
- Verificar el envío de comandos de interrupciones desde HK_MB hacia HK_C, así como la recepción y ejecución de la rutina de interrupción en la tarjeta esclava.
- Verificar que la transmisión de datos no se vea afectada por la longitud del cable que comunica ambas tarjetas.

La propuesta de la arquitectura del HK, como ya se ha analizado, propone la conexión de una tarjeta principal (HK_MB) con 20 tarjetas esclavas (HK_C), pero no se requiere que todas las tarjetas estén en un mismo sitio, sino que se tiene contemplado que las tarjetas HK_C estén distribuidas en la superficie del instrumento JEM-EUSO, que será un círculo de aproximadamente 2m de diámetro.

Por este motivo, la prueba que se ha llevado a cabo, consistió en comunicar a las tarjetas a través de un cable tipo par trenzado de 1.5m de largo, para tratar de recrear en la medida de lo posible, la distancia real a la que se podrían encontrar la tarjetas y realizar la prueba bajo estas condiciones. Por el momento se ha decidido no realizar pruebas detalladas concernientes al comportamiento de las tarjetas HK_MB_i, HK_C_i y la plataforma Arduino MEGA, ya que se prefiere enfocar las pruebas de esta tesis al comportamiento del FPGA como unidad lógica principal del HK.

Las pruebas que se llevaron a cabo fueron:

- Generación de interrupciones durante el ciclo normal de trabajo del HK_C.
- Envío de datos a la tarjeta XUPV5-LX110T.

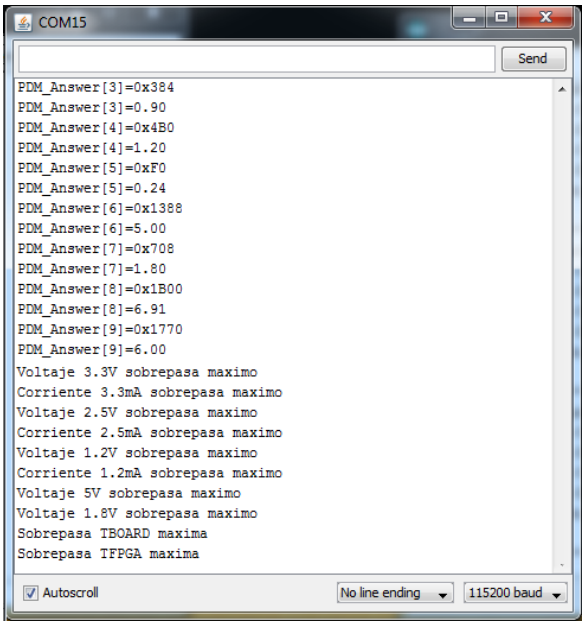


Figura 4.16. Impresión de pantalla de los datos leídos por HK_C y transmitidos a la tarjeta HK_MB.

Como resultado de estas pruebas, se verificó que la comunicación SPI por parte de HK_MB fue exitosa, tanto en el envío como en la recepción de la información proveniente de las tarjetas HK_C, y a pesar de algunos errores suscitados en los datos leídos por parte de HK_C, la tarjeta HK_MB cumplió exitosamente con la tarea de monitoreo.

4.3.3 Resultados de las pruebas de interacción entre HK y CPU

Para realizar esta prueba, se utilizó la configuración del sistema de la Figura 4.17, el cual físicamente se puede observar en la Figura 4.18.

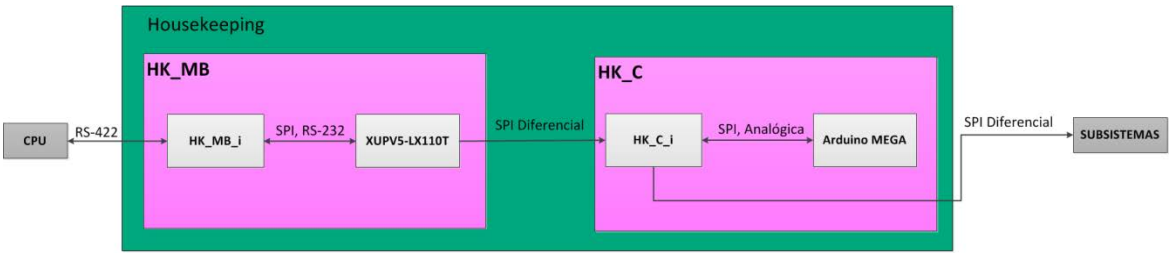


Figura 4.17. Diagrama del sistema para realizar pruebas del HK.

Los objetivos de esta prueba fueron:

- Verificar que HK recibe los comandos de la CPU.
- Verificar que HK ejecuta las tareas que la CPU le indica a través de las interrupciones.
- Verificar que la transmisión de datos no se vea afectada por la longitud del cable que comunica ambas tarjetas.
- Verificar que HK envía vectores de alarmas de alta prioridad a la CPU.

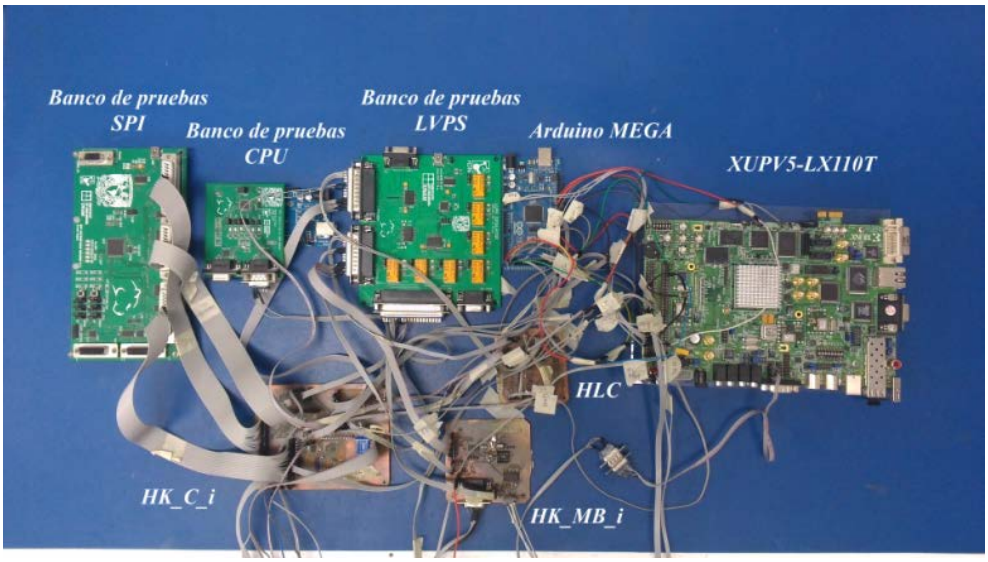


Figura 4.18. Conexión del sistema mínimo bajo prueba.

En este momento no interesa realizar pruebas sobre el diseño de las tarjetas que conforman a este sistema, ya que para esta tesis no se contempla el diseño final de estas tarjetas. Por este motivo, se ha considerado que las tarjetas son ideales, y mientras tanto las pruebas se han enfocado en los elementos externos al HK que permiten la comunicación entre subsistemas, en este caso, en el cable de conexión con la CPU. Este cable, es del tipo par trenzado y mide 1.5 metros, ya que se estima que la máxima longitud de un cable en el JEM-EUSO es de esa longitud.

La metodología para esta prueba fue:

- Se conectó la CPU a la tarjeta HK_MB_i por medio de un cable par trenzado con una longitud de 1.5 m.
- Se simulo la CPU para que ésta estuviese interrumpiendo al HK como se muestra en la Figura 4.19.
- La comunicación SPI entre las tarjetas XUPV5-LX110T y HK_C_i, se realizó a través de un cable par trenzado de 1.5 m, para simular la conexión entre las tarjetas HK_MB y HK_C.

El tiempo total de la prueba fue de 3 horas, durante las cuales CPU envió un total de 2251 interrupciones y de las cuales el HK recibió y ejecutó el 100%.

Con esto se prueba que la longitud del cable no causó ningún problema en la comunicación entre HK y la CPU.

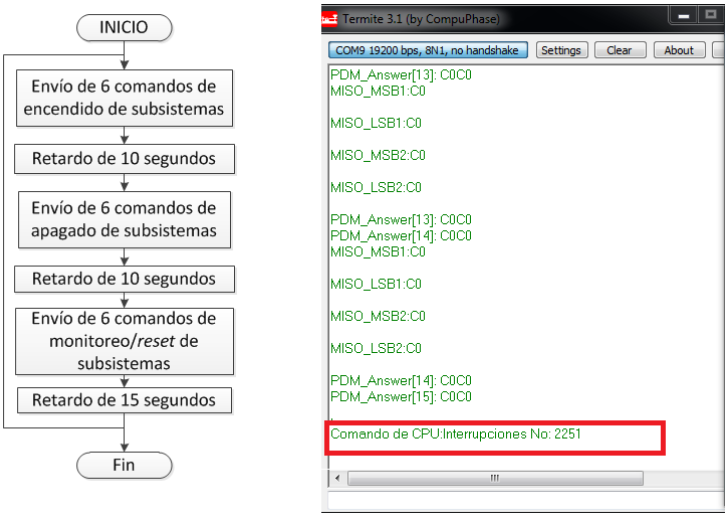


Figura 4.19. Diagrama de flujo del programa de interrupciones e impresión de pantalla de la cuenta de interrupciones captadas por el HK.

4.3.4 Resultados de las pruebas de medición del tiempo de un ciclo de monitoreo del HK

Una de las pruebas más relevantes consistió en la medición del tiempo que el HK tarda en realizar un ciclo completo de monitoreo, ya que no deben pasar más de 3 segundos antes de que el HK vuelva a monitorear el subsistema que monitoreo primero.

Una vez que el HK se ha encendido y que los subsistemas del JEM-EUSO se han inicializado, comienza el monitoreo del HK, el cual consiste en:

- Lectura de 22 esclavos SPI: 20 tarjetas HK_C, CLKB y GPS.
- Lectura de 22 LPVS: 20 LVPS_HK_C, LVPS_CLKB y LVPS_GPS.
- Lectura de 400 sensores de temperatura.

Por el sistema mínimo con el que se realizaron las lecturas, las dos primeras tareas se pueden implementar para medir el tiempo, mientras que la correspondiente a la lectura de los sensores de temperatura se puede calcular y anexar al tiempo de monitoreo y de esta forma obtener el tiempo total que tarda el HK en monitorear a los subsistemas del JEM-EUSO.

Cálculos teóricos del tiempo de monitoreo esperado

La frecuencia con la que trabaja el protocolo SPI es de 4 MHz, ya que la unidad lógica de las tarjetas HK_C es simulada por medio de un microcontrolador ATMEGA 2560 (Figura 4.8) y se tiene la limitación de que en modo esclavo no puede trabajar a una frecuencia más alta.

Se han realizado los cálculos en base a esta forma de operación para estimar el tiempo total de monitoreo.

$$Un\ ciclo\ de\ reloj = \frac{1}{4\ Mhz} = 250\ ns$$

Dado que cada byte es leído en ocho ciclos de reloj:

$$Tiempo\ de\ lectura\ de\ 1\ byte = (250x10^{-9}) \times 8 = 2\ \mu s$$

Hasta el momento, el subsistema PDM proporciona una cantidad de 18 lecturas, cada una formada por:

1 DATO			
Datos parte alta	Datos parte baja	CRCH	CRCL

Por lo que una sola lectura está conformada por 4 bytes, de modo que:

$$Tiempo\ de\ lectura\ de\ un\ dato = (2 \times 10^{-6}) \times 4 = 8\ \mu s$$

Y el tiempo que tarda en realizar la lectura de los datos de 1 PDM es

$$t_{mon\_PDM} = (8 \times 10^{-6}) \times 19 = 152\ \mu s$$

En cuanto al tiempo de lectura de 1 CCB, es el mismo que el de PDM, dado que se tiene la misma cantidad de datos a leer en el mismo formato,

$$t_{mon\_CCB} = 152\ \mu s$$

Cada LVPS hace una lectura de voltaje y corriente (2 datos) del subsistema que le corresponde, y su formato para la lectura también está conformado por 4 bytes, por lo tanto el tiempo de monitoreo de 1 LVPS es:

$$t_{mon\_LVPS} = (8 \times 10^{-6}) \times 2 = 16\ \mu s$$

Dado que cada una de las tarjetas HK_C enviará los datos de 7 PDMs, 1 CCB y 2 LVPS, el tiempo que la tarjeta HK_MB se tardara en adquirir los datos sería de:

$$t_{mon\_HK\_C} = t_{mon\_CCB} + \left(t_{mon\_PDM} \times 7\right) + \left(t_{mon\_LVPS} \times 2\right) \qquad \text{Ec. 1}$$

$$t_{mon\_HK\_C} = (152 \times 10^{-6}) + (152 \times 10^{-6}) \times 7 + (16 \times 10^{-6}) \times 2 = 1.248\ ms$$

El tiempo de monitoreo de CLKB contando envío y recepción de datos es de 10 bytes, por lo que

$$t_{mon\_CLKB} = (2 \times 10^{-6}) \times 10 = 20\ \mu s$$

El tiempo de lectura de GPS es de 8 bytes en total tomando en cuenta los tiempos de envío y recepción de datos, por lo que

$$t_{mon\_GPS} = (2 \times 10^{-6}) \times 8 = 16\ \mu s$$

En cuanto a los sensores de temperatura se ha propuesto, para medir el tiempo de una forma aproximada, un ADC cuya frecuencia de transmisión es de 400 Kb/s y del cual hace una lectura en 45 ciclos de reloj, por lo que el tiempo de lectura de 1 sensor sería de

$$Tiempo\ de\ lectura\ de\ 1\ canal = (2.5 \times 10^{-6}) \times 45 = 112.5\ \mu s$$

Si el ADC tiene 8 canales, entonces

$$Tiempo\ de\ lectura\ de\ un\ ADC = (112.5 \times 10^{-6}) \times 8 = 900\ \mu s$$

Si a cada bus I2C se le conecta un total de 9 convertidores, entonces el tiempo que cada bus tardaría en leer los datos de 72 sensores sería de

$$Tiempo\ de\ lectura\ en\ cada\ bus\ I2C = (900 \times 10^{-6}) \times 9 = 8.1\ ms$$

Por lo tanto, leer 400 sensores, significa leer 400 canales, por lo que el tiempo total de monitoreo de temperaturas sería de

$$t_{mon\_temp} = (112.5 \times 10^{-6}) \times 400 = 45\ ms$$

El tiempo total de monitoreo del HK, se puede calcular de la siguiente forma:

$$t_{mon\_HK} = \left(t_{mon\_HK\_C} \times 20\right) + t_{mon\_CLKB} + t_{mon\_GPS} + t_{mon\_temp} + \left(t_{mon\_LVPS} \times 22\right) \quad \text{Ec. 2}$$

$$t_{mon\_HK} = (1.248 \times 10^{-3} \times 20) + 20 \times 10^{-6} + 16 \times 10^{-6} + 45 \times 10^{-3} + (16 \times 10^{-6} \times 22)$$

$$t_{mon\_HK} = 70.348\ ms$$

Este resultado nos dice que de forma ideal el HK volverá a monitorear la primera tarjeta en tan solo 70.348 ms. Sin embargo, hay que tomar en cuenta otros procesos que se llevan a cabo durante las lecturas como los tiempos de espera de respuesta del microcontrolador, habilitación de esclavos y ejecución de algunas estructuras cíclicas como lo son los "*for*".

Resultados experimentales del tiempo de un ciclo de monitoreo del HK

Las mediciones que se presentan a continuación, se realizaron en el sistema de la Figura 4.17. La primer prueba consistió en la medición del tiempo de lectura que le toma a la tarjeta HK_MB, realizar la lectura completa de los datos de un PDM.

Prueba de tiempo para la lectura de 1 PDM

La prueba contempló las siguientes acciones:

- Envío de los datos de la lectura de 1 PDM por HK_C, a través del protocolo SPI hacia la tarjeta maestra HK_MB.

En esta prueba no se utilizó ningún comando que desplegara cadenas de caracteres o realizara otra acción además de la lectura, para poder captar la lectura más exacta posible. Es decir, que únicamente se está midiendo la transmisión de datos por medio del protocolo SPI.

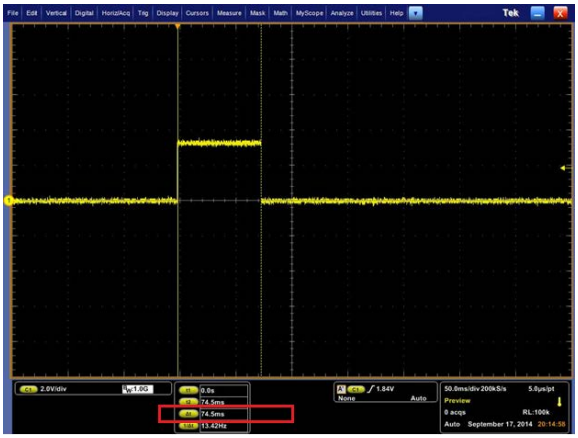


Figura 4.20. Tiempo de monitoreo de 1 PDM.

Como se puede observar en la Figura 4.20, el tiempo que el HK_MB se tarda en realizar la lectura de los datos de 1 PDM es de 74.5 ms.

En comparación con los cálculos teóricos, este tiempo es un orden de magnitud más grande. Esto se debe a que durante las lecturas, la plataforma Arduino requirió de un retardo de 1ms entre el envío de cada byte para tener una transmisión correcta del dato; por lo tanto, la transmisión de 1 dato se hace en 4.008 ms. Es por esto que la diferencia es muy grande, sin embargo si se tomara en cuenta solo la transmisión del byte sin retardos, el tiempo coincide con el calculado.

Prueba de tiempo para la lectura de 1 tarjeta HK_C

Enseguida se implementó una rutina para la lectura de una tarjeta HK_C, la cual envía la información correspondiente de 7 PDMs, 1 CCB y 2 LVPS. Se ha calculado que idealmente, esta transmisión debería de ser de 1.248 ms sin contar el retardo de 1 ms que se ha tenido que añadir.

Sin embrago, se necesita de 1 ms de espera en la plataforma Arduino MEGA para la transmisión de cada byte; el tiempo de lectura para una HK_C que se midió fue de 720 ms.

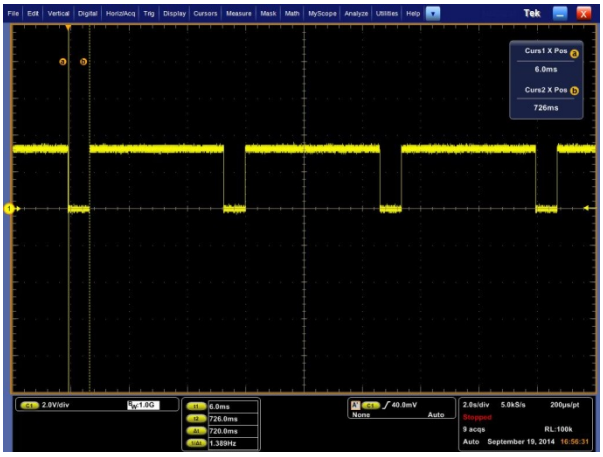


Figura 4.21. Tiempo de monitoreo de una tarjeta HK_C.

Prueba de tiempo para un ciclo de monitoreo del HK

La última prueba consistió en medir el tiempo de un ciclo completo de monitoreo. Esta prueba consistió en la lectura de los datos de las 20 tarjetas HK_C, cada una con el aporte de la información de 7 PDMs, 1 CCB y 2 LVPS; además de la lectura de GPS y CLKB. En cuanto a las lecturas de LVPS, se midió el tiempo equivalente a la lectura de 22 tarjetas LVPS. En estas pruebas, no se ha incluido la medición del tiempo de lectura de los sensores de temperatura, por lo que se tomará el valor que se ha calculado previamente para efectos de obtener el tiempo total de monitoreo.

La metodología para realizar estas lecturas consistió en realizar 20 lecturas de la tarjeta equivalente a HK_C, para tratar de simular la condición de 20 tarjetas esclavas. Las lecturas de CLKB y GPS se realizaron utilizando el banco de pruebas SPI y las lecturas de las 22 LVPS se hicieron leyendo de forma repetida un ADC conectado a una de las tarjetas LVPS del banco de pruebas.

El tiempo total por ciclo que se registró fue de 3.846 s y agregando el tiempo de monitoreo de temperatura se tiene un total de 3.891 s, lo cual quiere decir que el HK tardaría al menos este tiempo en volver a monitorear la primer tarjeta, cuando el tiempo límite establecido es de 3 s, por lo que esta prueba no ha sido totalmente satisfactoria.

Figura 4.22. Tiempo que el HK tarda en realizar un ciclo completo de monitoreo.

Como se ha observado, en todas las pruebas de medición de tiempo, hay un problema común debido a la condición que existe en el envío de datos por parte del microcontrolador de la plataforma Arduino. Por consecuencia, la conclusión de esta prueba experimental es que si se quiere seguir utilizando la plataforma Arduino MEGA, se debe de depurar y optimizar el código para eliminar todos aquellos retardos que están causando un tiempo de lectura tan amplio, ya que como se ha analizado, al realizar una lectura sencilla el tiempo que la HK_MB tarda en adquirir los datos, es muy similar al tiempo calculado, por lo que el problema no se debe a fallas en la comunicación o en las tarjetas, sino que se debe al código implementado en el microcontrolador de la plataforma Arduino MEGA. Otra solución sería implementar la lógica del HK_C con un microcontrolador diferente.

4.3.5 Reporte de recursos utilizados por el FPGA

Una de las cosas que se buscaba medir además del tiempo de monitoreo, es el porcentaje de recursos que utiliza el FPGA Virtex-5 al embeber en el dispositivo la lógica del HK. Para llevar a cabo este análisis, se creó la arquitectura mostrada en la Figura 3.24 y se exportó el diseño a la herramienta PlanAhead™, la cual genera entre otros reportes, uno en el que se muestra el porcentaje de utilización de cada elemento que conforma el diseño implementado.

Como se puede observar en la Tabla 4.1, los recursos que se utilizan para la lógica del HK están por debajo del 50%, lo cual es un buen resultado, ya que esto indica que aún hay recursos disponibles para generar un diseño más complejo, por ejemplo uno donde se tenga tolerancia a fallas.

Elemento	Usados	Disponibles	Porcentaje de utilización
LUTs	6,797	69,120	9%
Slices	3,343	17,280	19%
BRAM	16	148	10%

Tabla 4.1. Recursos utilizados por la arquitectura del HK en Virtex-5.

4.3.6 Reporte de la potencia que consume el FPGA

Con el PlanAhead™, también se puede generar un reporte del consumo de potencia del FPGA a través de su herramienta XPower Analizer, la cual permite simular el consumo de potencia a través de las especificaciones del diseño proporcionadas por los reportes generados en XPS y SDK, así como de las condiciones ambientales. En este caso se realizó el análisis tanto para el sistema completo propuesto en la Figura 3.24, como para el sistema mínimo generado para implementar la lógica del sistema experimental propuesto en la Figura 4.17.

4.3.7 Reporte de la potencia que consume el FPGA del HK

Para generar el reporte del consumo de potencia se propuso una temperatura ambiente de 30°C, un ambiente con un flujo de aire natural y un disipador de alto desempeño para el FPGA.

Environment	
Ambient Temp (C)	30.0
Use custom TJA?	No
Custom TJA (C/W)	NA
Airflow (LFM)	250
Heat Sink	High Profile
Custom TSA (C/W)	NA
Board Selection	Medium (10"x10")
# of Board Layers	12 to 15
Custom TJB (C/W)	NA
Board Temperature (C)	NA

Figura 4.23. Condiciones bajo las cuales se realizó el análisis del consumo de potencia para el diseño del HK.

Device	
Family	Virtex5
Part	xc5vlx110t
Package	ff1136
Temp Grade	Commercial
Process	Typical
Speed Grade	-1

Environment	
Ambient Temp (C)	30.0
Use custom TJA?	No
Custom TJA (C/W)	NA
Airflow (LFM)	250
Heat Sink	High Profile
Custom TSA (C/W)	NA
Board Selection	Medium (10"x10")
# of Board Layers	12 to 15
Custom TJB (C/W)	NA
Board Temperature (C)	NA

On-Chip	Power (W)	Used	Available	Utilization (%)
Clocks	0.156	3	--	--
Logic	0.004	6802	69120	10
Signals	0.006	10771	--	--
BRAMs	0.004	16	148	11
DSPs	0.000	5	64	8
PLLs	0.134	1	6	17
IOs	0.628	98	640	15
Leakage	0.870			
Total	1.802			

Thermal Properties	Effective TJA (C/W)	Max Ambient (C)	Junction Temp (C)
	1.3	82.7	32.3

Supply Summary		Total	Dynamic	Quiescent
Source	Voltage	Current (A)	Current (A)	Current (A)
Vccint	1.000	0.723	0.188	0.535
Vccaux	2.500	0.428	0.034	0.394
Vcco25	2.500	0.004	0.000	0.004

	Total	Dynamic	Quiescent
Supply Power (W)	1.802	0.272	1.530

Figura 4.24. Reporte de potencia obtenido con el XPower Analizer para el diseño del HK.

El resumen de potencia para este sistema obtenido con el XPower Analizer se puede observar en la Figura 4.24, la potencia que consume el FPGA es de 1.802 W, además se tiene otro aporte al consumo de potencia por parte de las fuentes de alimentación que equivale a 1.802 W, por lo que en total, la Virtex-5 tiene un consumo de 3.604 W.

On-Chip Power Summary				
On-Chip	Power (mW)	Used	Available	Utilization (%)
Clocks	155.66	3	---	---
Logic	4.36	6802	69120	10
Signals	6.35	10771	---	---
IOs	627.62	98	640	15
BRAMs	4.10	16	148	11
DSPs	0.00	5	64	8
PLLs	134.22	1	6	17
Static Power	869.82			
Total	1802.13			

Figura 4.25. Consumo de potencia del FPGA del HK con detalle en los elementos que componen el diseño.

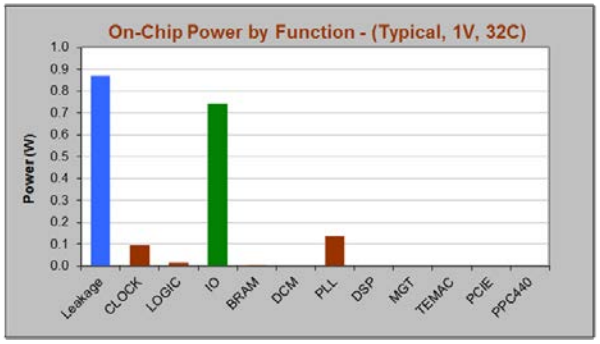


Figura 4.26. Gráfica del consumo de potencia del FPGA del HK con detalle en los elementos que componen el diseño

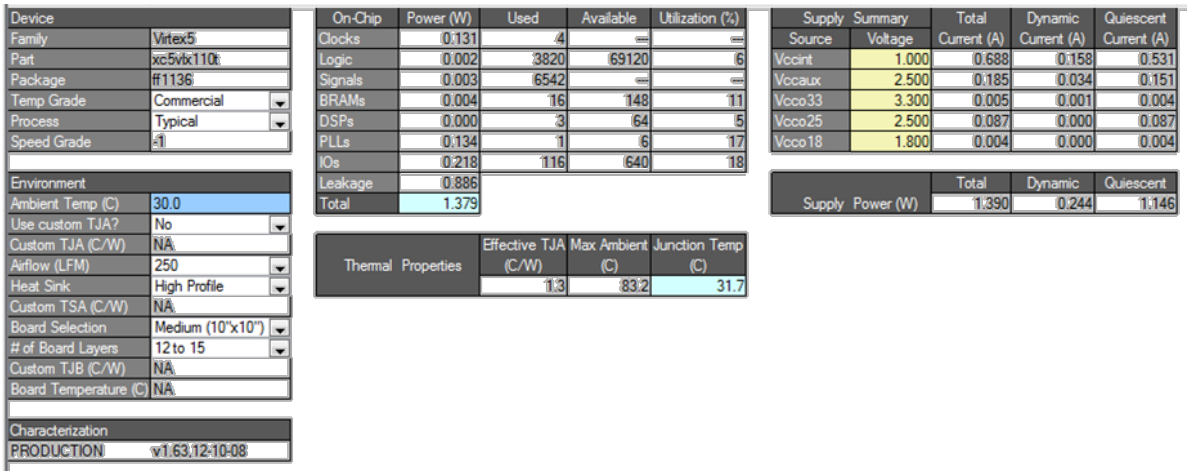
4.3.8 Reporte de la potencia que consume el FPGA del sistema mínimo experimental

El sistema mínimo que se analizó, fue el implementado en el FPGA Virtex-5 para controlar el sistema con el que se llevaron a cabo las pruebas de medición de tiempo y comunicaciones. Para este análisis se propuso una temperatura ambiente de 30°C, un ambiente con un flujo de aire natural y un disipador de alto desempeño para el FPGA.

Environment	
Ambient Temp (C)	30.0
Use custom TJA?	No
Custom TJA (C/W)	NA
Airflow (LFM)	250
Heat Sink	High Profile
Custom TSA (C/W)	NA
Board Selection	Medium (10"x10")
# of Board Layers	12 to 15
Custom TJB (C/W)	NA
Board Temperature (C)	NA

Figura 4.27. Condiciones bajo las cuales se realizó el análisis del consumo de potencia.

En la Figura 4.28, se puede observar el resumen de potencia generado por el XPower Analizer; para este sistema, la potencia que consume el FPGA es de 1.379 W, además se tiene otro aporte al consumo de potencia por parte de las fuentes de alimentación que equivale a 1.390 W, por lo que en total, la Virtex-5 tiene un consumo de 2.796 W.



Device	
Family	Virtex5
Part	xc5vlx110t
Package	ff1136
Temp Grade	Commercial
Process	Typical
Speed Grade	-1

Environment	
Ambient Temp (C)	30.0
Use custom TJA?	No
Custom TJA (C/W)	NA
Airflow (LFM)	250
Heat Sink	High Profile
Custom TSA (C/W)	NA
Board Selection	Medium (10"x10")
# of Board Layers	12 to 15
Custom TJB (C/W)	NA
Board Temperature (C)	NA

Characterization	
PRODUCTION	v1.63,12-10-08

On-Chip	Power (W)	Used	Available	Utilization (%)
Clocks	0.131	4	--	--
Logic	0.002	3820	69120	6
Signals	0.003	6542	--	--
BRAMs	0.004	16	148	11
DSPs	0.000	3	64	5
PLLs	0.134	1	6	17
IOs	0.218	116	640	18
Leakage	0.886			
Total	1.379			

Thermal Properties	Effective TJA (C/W)	Max Ambient (C)	Junction Temp (C)
	1.3	83.2	31.7

Supply Summary		Total	Dynamic	Quiescent
Source	Voltage	Current (A)	Current (A)	Current (A)
Vccint	1.000	0.688	0.158	0.531
Vccaux	2.500	0.185	0.034	0.151
Vcco33	3.300	0.005	0.001	0.004
Vcco25	2.500	0.087	0.000	0.087
Vcco18	1.800	0.004	0.000	0.004

		Total	Dynamic	Quiescent
Supply	Power (W)	1.390	0.244	1.146

Figura 4.28. Reporte de potencia obtenido con el XPower Analizer.

Así mismo, se generó un reporte más detallado para el consumo de potencia cuyos resultados se pueden observar en la Figura 4.29, luego se exportaron los datos a la plataforma XPower Estimator, la cual es una herramienta que antecede al XPower Analizer, para obtener la gráfica de consumo que se observa en la Figura 4.30.

```
--------------------------------------------------------------------------------
|                          On-Chip Power Summary                               |
--------------------------------------------------------------------------------
|        On-Chip        | Power (mW) |  Used  | Available | Utilization (%) |
--------------------------------------------------------------------------------
| Clocks                |     131.14 |      4 |    ---    |       ---       |
| Logic                 |       2.48 |   3820 |     69120 |               6 |
| Signals               |       3.17 |   6542 |    ---    |       ---       |
| IOs                   |     217.74 |    116 |       640 |              18 |
| BRAMs                 |       4.30 |     16 |       148 |              11 |
| DSPs                  |       0.00 |      3 |        64 |               5 |
| PLLs                  |     134.37 |      1 |         6 |              17 |
| Static Power          |     886.08 |        |           |                 |
| Total                 |    1379.27 |        |           |                 |
--------------------------------------------------------------------------------
```

Figura 4.29. Consumo de potencia del FPGA con detalle en los elementos que componen el diseño.

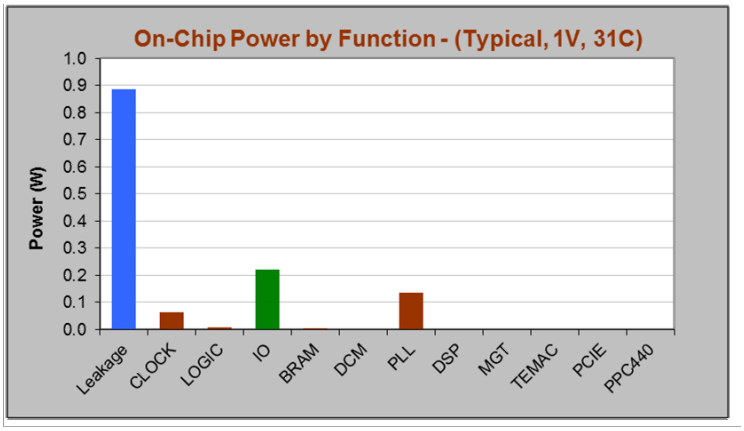


Figura 4.30. Gráfica del consumo de potencia del FPGA con detalle en los elementos que componen el diseño.

Datos experimentales

En el sistema diseñado para las pruebas, no hay una forma de medir la potencia que consume el FPGA, ya que este se encuentra integrado en la plataforma XUPV5-LX110T, por lo que los resultados siguientes no se podrán comparar con los obtenidos en el XPower Analizer. Sin embrago, con estos datos se podrá tener una estimación de la potencia que podría llegar a consumir el sistema final si se implementa la arquitectura propuesta en esta tesis.

La corriente que se midió en la tarjeta XUPV5-LX110T fue de 1.4 A, y el voltaje con el que se alimenta la tarjeta es de 5 V, por lo tanto, la potencia de consumo es de 7 W. Esta potencia es mucho mayor a la que supuestamente debería consumir el FPGA de acuerdo al análisis teórico, pero se debe a que en este caso se mide el consumo de la plataforma de desarrollo y no solo del dispositivo lógico.

Enseguida se midió la corriente de consumo de las tarjetas que conforman al HK_C del sistema mínimo para las pruebas y se obtuvo un consumo total de 1.648 W. Si cada una de las 20 tarjetas que conforman el HK tuviera este consumo, en total se tendría un consumo de 32.96 W y si la potencia de consumo de la tarjeta HK_MB fuera equivalente a la que consume la plataforma de desarrollo, entonces el HK consumiría aproximadamente 40 W, lo cual está dentro de los límites establecidos pero no es del todo deseable ya que no se podría contar con un margen de tolerancia.

Capítulo 5
Conclusiones y recomendaciones

De este trabajo se concluye que la implementación de la lógica del HK en un dispositivo lógico programable y reconfigurable como el FPGA, es la solución óptima para cumplir con los requerimientos del sistema. En el caso específico de Virtex-5, se obtuvo que los recursos que se utilizan son menos del 50%, lo que permitiría que el sistema pudiera escalarse sin ningún problema. Específicamente esto es importante, ya que en esta arquitectura embebida aún no se ha contemplado la implementación de un sistema tolerante a fallas, el cual necesitará de más recursos para implementarse.

En cuanto al comportamiento de Virtex-5 en el análisis de potencia, se obtuvo que esta consumiría alrededor de 4 W-7 W, con lo que quedaría aproximadamente un 80% de recursos de presupuesto energético para utilizarse en las tarjetas del resto del subsistema HK.

Una aportación importante, es que se decidió que la mejor forma de implementar este subsistema es a través de un sistema distribuido conformado por una tarjeta principal y 20 tarjetas esclavas de control local. Entre las ventajas que ofrece una arquitectura como esta, es que se puede implementar un sistema con tolerancia a fallas, lo cual es indispensable para un sistema que en un futuro se planea que opere en el espacio. Así mismo se realizó una aproximación del tamaño de estas tarjetas y el resultado fue que sus medidas se ajustan a los requerimientos de medida para el HK.

Para la implementación de las tarjetas HK_C, se concluyó que de acuerdo a la carga de trabajo a la que serán sometidas y a la cantidad de datos que manejarán, es aconsejable utilizar un microcontrolador.

En esta tesis no se ha propuesto el uso específico de un microcontrolador; sin embargo, debido a los resultados poco satisfactorios de las pruebas que se hicieron utilizando el ATmega 2560 como unidad lógica de estas tarjetas, se recomienda no utilizar este microcontrolador, sino buscar uno que tenga su análogo con calificación espacial y además que posea interfaces suficientes para la comunicación con los subsistemas y el FPGA. También es deseable que el microcontrolador que se elija consuma la menor cantidad de potencia posible.

Uno de los problemas importantes que se encontraron en el funcionamiento del HK, fue la cantidad de tiempo que se requiere para hacer la tarea de monitoreo, ya que sobrepasa al tiempo establecido en los requerimientos. Por lo tanto, se sugiere optimizar el código tanto de la tarjeta HK_MB, como de la tarjeta HK_C, sobre todo de la primera para mejorar el tiempo de los ciclos de lectura. Además, se propone que el diseño incluya un segundo y hasta un tercer procesador que trabajen en paralelo, para poder realizar las lecturas de las tarjetas HK_C de una forma más rápida, lo cual ahorraría tiempo de ejecución de la rutina del HK. Otra alternativa para mejorar el tiempo de monitoreo es el crear un núcleo personalizado que permita realizar lecturas de una forma paralela sin tener que utilizar más de un microprocesador para la ejecución de esta tarea. Otra

solución que se ha encontrado atractiva, es el cambio del protocolo de comunicación entre la tarjeta HK_MB con la tarjeta HK_C, que actualmente es SPI, por uno más veloz para de este modo optimizar también el tiempo dedicado al monitoreo de los subsistemas del JEM-EUSO. Los análisis de potencia que se realizaron, han revelado que el consumo de las tarjetas que se utilizaron para pruebas del HK, es mayor al deseable, por lo que se recomienda que para el diseño de estas tarjetas se ponga atención especial en la potencia de consumo, así como en sus características de ahorro de energía.

Este trabajo de tesis es la base para continuar con el desarrollo del HK bajo una visión que contempla los retos que se enfrentarán en el diseño del subsistema final. Entre las tareas que son parte de un trabajo futuro se encuentra:

- La implementación de un sistema tolerante a fallas, tanto en el sistema distribuido como en la tarjeta HK_MB.
- Diseño y construcción de las tarjetas HK_MB y HK_C.
- Selección de la unidad lógica de la tarjeta HK_C.
- Optimización del código para hacerlo más rápido y tolerante a fallas.
- Implementación de un código no solo de detección sino de corrección de errores.

Pruebas	Objetivos/justificación	Pruebas a realizar	Nivel de importancia	Clasificación de resultados		
				Tarea exitosa	Tarea con fallas	Tarea no cumplida
Comunicación RS232/RS422 (Recepción de comandos)	Validar la comunicación entre CPU y FPGA/El funcionamiento de HK depende de CPU, ya que es un esclavo del mismo	✔	1	Correcta recepción de comandos/Atención a las interrupciones en el momento de su generación.	Falta de atención a una rutina de interrupción por estar dentro de una rutina de interrupción diferente/ perdida de un proceso de monitoreo/Recibir un comando no reconocido/No entrar a la rutina de interrupción/ quedarse atrapado en la interrupción	Falla en el protocolo.
Comunicación SPI con PDM (µc)	Validar lecturas de datos de V, I y T proporcionados por PDM mediante el protocolo SPI/es el sistema de detección principal del JEM-EUSO	✔	1	Se verifica que los datos se leyeron correctamente y que están dentro de los rangos de umbral establecidos o fuera de él pero generando alarmas.	Lectura de datos incorrectos debido al envío de datos erróneos por parte del subsistema/Lecturas de 0x00 o 0xFF	Falla del protocolo SPI.
Comunicación SPI con CCB (µC)	Validar lecturas de datos de V, I y T proporcionados por CCB mediante el protocolo SPI/es el sistema de procesamiento y clasificación de datos de PDM	✔	2	Se verifica que los datos se leyeron correctamente y que están dentro de los rangos de umbral establecidos o fuera de él pero generando alarmas.	Lectura de datos incorrectos debido al envío de datos erróneos por parte del subsistema/Lecturas de 0x00 o 0xFF	Falla del protocolo SPI.
Comunicación SPI con CLKB (FPGA)	Validar lecturas de datos de V, I y T proporcionados por CLKB mediante el protocolo SPI	No se hace	2	Se verifica que los datos se leyeron correctamente y que están dentro de los rangos de umbral establecidos o fuera de él pero generando alarmas.	Lectura de datos incorrectos debido al envío de datos erróneos por parte del subsistema/Lecturas de 0x00 o 0xFF	Falla del protocolo SPI.

Comunicación SPI con GPS (FPGA)	Validar lecturas de datos de V, I y T proporcionados por CLKB mediante el protocolo SPI	✔	2	Se verifica que los datos se leyeron correctamente y que están dentro de los rangos de umbral establecidos o fuera de él pero generando alarmas.	Lectura de datos incorrectos debido al envío de datos erróneos por parte del subsistema/Lecturas de 0x00 o 0xFF	Falla del protocolo SPI.
Lectura analógica de sensores de temperatura	Lectura de sensores de temperatura/monitorear la temperatura en los diferentes subsistemas del telescopio	No se hace	3	-	-	-
Lecturas analógicas de LVPS_HK_C	Lectura de V y I de las fuentes de alimentación de las tarjetas HK_C/verificar la alimentación para el funcionamiento de las tarjetas HK_C	No se hace	1	-	-	-
Encendido del relevador de PDM y verificación de la acción	Verificar que la fuente de alimentación de PDM funciona	✔	1	Recepción del comando de encendido de CPU/Envío del pulso/lectura de CC=5V	-	Lectura de CC=0V/No entra a la interrupción
Encendido del relevador de CCB y verificación de la acción	Verificar que la fuente de alimentación de CCB funciona	No se hace	2	-	-	-
Encendido del relevador de CLKB y verificación de la acción	Verificar que la fuente de alimentación de CLKB funciona	✔	2	Recepción del comando de encendido de CPU/Envío del pulso/lectura de CC=5V	-	Lectura de CC=0V/No entra a la interrupción
Encendido del relevador de GPS y verificación de la acción	Verificar que la fuente de alimentación de GPS funciona	No se hace	2	-	-	-

Encendido del relevador de DST y verificación de la acción	Verificar que la fuente de alimentación de DST funciona	✔	1	Recepción del comando de encendido de un simulador de SIREN/Envío del pulso/lectura de CC=5V	-	Lectura de CC=0V/No entra a la interrupción
Encendido del relevador de CPU y verificación de la acción	Verificar que la fuente de alimentación de CPU funciona	✔	1	Recepción del comando de encendido de CPU/Envío del pulso/lectura de CC=5V	-	Lectura de CC=0V/No entra a la interrupción
Encendido del relevador de EC y verificación de la acción	Verificar que la fuente de alimentación de EC funciona	No se hace	1	Recepción del comando de encendido de CPU/Envío del pulso/lectura de CC=5V	-	Lectura de CC=0V/No entra a la interrupción
Encendido de la tarjeta HK_C	Verificar que la fuente de alimentación de HK_C funciona/alimenta el microcontrolador	No se hace	1	-	-	-
Apagado del relevador de PDM y verificación de la acción	Verificar que la fuente de alimentación de PDM funciona	✔	1	Recepción del comando de apagado de CPU/Envío del pulso/lectura de CC=0V	-	Lectura de CC=5V/No entra a la interrupción
Apagado del relevador de CCB y verificación de la acción	Verificar que la fuente de alimentación de CCB funciona	No se hace	2	-	-	-
Apagado del relevador de CLKB y verificación de la acción	Verificar que la fuente de alimentación de CLKB funciona	✔	2	Recepción del comando de apagado de CPU/Envío del pulso/lectura de CC=0V	-	Lectura de CC=5V/No entra a la interrupción
Apagado del relevador de GPS y verificación.	Verificar que la fuente de alimentación de GPS funciona	No se hace	2	-	-	-

Apagado del relevador de DST y verificación de la acción	Verificar que la fuente de alimentación de DST funciona	✔	1	Recepción del comando de encendido de un simulador de SIREN/Envío del pulso/lectura de CC=5V	-	Lectura de CC=0V/No entra a la interrupción
Apagado del relevador de CPU y verificación de la acción	Verificar que la fuente de alimentación de CPU funciona	✔	1	Recepción del comando de encendido de un simulador de SIREN/Envío del pulso/lectura de CC=5V	-	Lectura de CC=0V/No entra a la interrupción
Apagado del relevador de EC y verificación de la acción	Verificar que la fuente de alimentación de EC funciona	No se hace	1	-	-	-
Apagado o de la tarjeta HK_C	Verificar que la fuente de alimentación de HK_C funciona/alimenta el microcontrolador	No se hace	1	-	-	-
Comunicación SPI con el µC	Validar la comunicación entre FPGA y microcontrolador/El microcontrolador recibirá comandos tanto del FPGA como comandos indirectos de CPU	✔	1	Correcta recepción de comandos/Atención a las interrupciones en el momento de su generación.	Falta de atención a una rutina de interrupción por estar dentro de estas con mayor prioridad/ pérdida del proceso en curso durante la interrupción/No entrar a la rutina de interrupción; quedarse atrapado en la interrupción	Falla del protocolo SPI.
Recepción de ALARM de PDM	Verificación de la recepción de esta alarma de alta prioridad/ envío de alarma de alta prioridad a CPU	✔	1	Entrar a la rutina de interrupción/Registro de la alarma en el vector de alarmas.	Quedarse atrapado en la interrupción/pérdida del proceso en curso durante la interrupción.	No genera la interrupción/No registrar la alarma

Recepción de DONE de PDM	Verificación de la recepción de esta alarma/verificación del registro de esta alarma en el vector de alarmas de baja prioridad	✔	1	Entrar a la rutina de interrupción/Envío del pulso PROG_BIT.	Quedarse atrapado en la interrupción/pérdida del proceso en curso durante la interrupción.	No genera la interrupción/no enviar el pulso de PROG_BIT.
Recepción de ALARM de CCB	Verificación de la recepción de esta alarma de alta prioridad/ envío de alarma de alta prioridad a CPU	No se hace	2	Entrar a la rutina de interrupción/Registro de la alarma en el vector de alarmas.	Quedarse atrapado en la interrupción/pérdida del proceso en curso durante la interrupción.	No genera la interrupción/No registrar la alarma
Recepción de DONE de CCB	Verificación de la recepción de esta alarma/verificación del registro de esta alarma en el vector de alarmas de baja prioridad	No se hace	2	Entrar a la rutina de interrupción/Envío del pulso PROG_BIT.	Quedarse atrapado en la interrupción/pérdida del proceso en curso durante la interrupción.	No genera la interrupción/no enviar el pulso de PROG_BIT.
Recepción de ALARM de CLKB	Verificación de la recepción de esta alarma de alta prioridad/ envío de alarma de alta prioridad a CPU	✔	2	Entrar a la rutina de interrupción/Registro de la alarma en el vector de alarmas.	Quedarse atrapado en la interrupción/pérdida del proceso en curso durante la interrupción.	No genera la interrupción/No registrar la alarma
Recepción de DONE de CLKB	Verificación de la recepción de esta alarma/verificación del registro de esta alarma en el vector de alarmas de baja prioridad	✔	2	Entrar a la rutina de interrupción/Envío del pulso PROG_BIT.	Quedarse atrapado en la interrupción/pérdida del proceso en curso durante la interrupción.	No genera la interrupción/no enviar el pulso de PROG_BIT.
Envío de RESET a PDM	Verificación del envío del pulso de RESET	✔	1	Entrar a la rutina de interrupción/Envío del pulso RESET.	Quedarse atrapado en la interrupción/pérdida del proceso en curso durante la interrupción.	No genera la interrupción/no enviar el pulso de RESET.

Envío de PROG_B a PDM	Verificación del envío del pulso de PROG_B	✔	1	El LED del banco de pruebas que indica esta alarma se apagará	Quedarse atrapado en la interrupción/pérdida del proceso en curso durante la interrupción.	No genera la interrupción/no enviar el pulso de PROG_BIT/No limpiar el vector de alarmas.
Envío de RESET a CCB	Verificación del envío del pulso de RESET	No se hace	2	Entrar a la rutina de interrupción/Envío del pulso RESET.	Quedarse atrapado en la interrupción/pérdida del proceso en curso durante la interrupción.	No genera la interrupción/no enviar el pulso de RESET.
Envío de PROG_B a CCB	Verificación del envío del pulso de PROG_B	No se hace	2	El LED del banco de pruebas que indica esta alarma se apagará	Quedarse atrapado en la interrupción/pérdida del proceso en curso durante la interrupción.	No genera la interrupción/no enviar el pulso de PROG_BIT/No limpiar el vector de alarmas.
Envío de RESET a CLKB	Verificación del envío del pulso de RESET	✔	2	Entrar a la rutina de interrupción/Envío del pulso RESET.	Quedarse atrapado en la interrupción/perdida del proceso en curso durante la interrupción.	No genera la interrupción/no enviar el pulso de RESET.
Envío de PROG_B a CLKB	Verificación del envío del pulso de PROG_B	✔	2	El LED del banco de Pruebas que indica esta Alarma se apagará.	Quedarse atrapado en la Interrupción/pérdida del Proceso en curso Durante la interrupción.	No genera la Interrupción/no Enviar el pulso de PROG_BIT/No limpiar El vector de alarmas
Lecturas analógicas de GPS	Verificación de la correcta lectura de voltajes/verificación de generación de alarmas	No se hace	2	-	-	-
Lecturas analógicas de CLKB	Verificación de la correcta lectura de voltajes/verificación de generación de alarmas	No se hace	2	-	-	-

Lecturas analógicas de CPU	Verificación de la correcta lectura de voltajes/verificación de generación de alarmas	✔	1	Se verifica que los datos se leyeron correctamente y que están dentro de los rangos de umbral establecidos o fuera de él pero generando alarmas.	Mal procesamiento de las lecturas/registro incorrecto de las alarmas en vectores de alarmas.	No hay lecturas/ Falla en el protocolo SPI.
Lecturas analógicas de DST	Verificación de la correcta lectura de voltajes/verificación de generación de alarmas	✔	1	Se verifica que los datos se leyeron correctamente y que están dentro de los rangos de umbral establecidos o fuera de él pero generando alarmas.	Mal procesamiento de las lecturas/registro incorrecto de las alarmas en vectores de alarmas.	No hay lecturas/ Falla en el protocolo SPI.
Lecturas analógicas de EC	Verificación de la correcta lectura de voltajes/verificación de generación de alarmas	✔	1	Se verifica que los datos se leyeron correctamente y que están dentro de los rangos de umbral establecidos o fuera de él pero generando alarmas.	Mal procesamiento de las lecturas/registro incorrecto de las alarmas en vectores de alarmas.	No hay lecturas/ Falla en el protocolo SPI.
Lecturas analógicas de PDM	Verificación de la correcta lectura de voltajes/verificación de generación de alarmas	✔	1	Se verifica que los datos se leyeron correctamente y que están dentro de los rangos de umbral establecidos o fuera de él pero generando alarmas.	Mal procesamiento de las lecturas/registro incorrecto de las alarmas en vectores de alarmas.	No hay lecturas/ Falla en el protocolo SPI.
Hard reset (apagado de DST y CPU)	Verificar la correcta ejecución de la rutina de apagado y encendido de DST y CPU	✔	1	Ejecución de la rutina para apagar/encender DST y CPU/Verificación de los CC=3.3V	-	CC=0
Soft reset (apagado de CPU)	Verificar la correcta ejecución de la rutina de apagado y encendido CPU	✔	1	Ejecución de la rutina para apagar/encender CPU/Verificación de CC=3.3V	-	CC=0

Medición de la potencia consumida por el sistema mínimo del Housekeeping	Cuantificar la potencia que consumirán la tarjetas que componen el sistema mínimo del Housekeeping	✔	2	-	-	-
Simulación del consumo de potencia de la Virtex-5 con el sistema embebido completo	Cuantificar la potencia que consumirá el FPGA Virtex-5	✔	1	Consumo <10 W	-	-

Nota: De todas las tareas a comprobar, se eligieron aquellas que se consideraron más críticas y representativas para evaluar la funcionalidad del sistema, es por esto que algunas de las tareas en la lista no se han realizado. El cuadro en sombra representa el resultado de la prueba.

Pruebas	Objetivos/justificación	Justificación particular	Pruebas a realizar	Nivel de importancia	Clasificación de resultados		
					Tarea exitosa	Tarea con fallas	Tarea no cumplida
Interfaz de comunicación entre HK_MB y HK_C a través del protocolo SPI diferencial	Esta es la segunda tarjeta más importante del subsistema, ya que si no funciona, se pierde la comunicación con PDM y CCB. Por esto se debe validar que existe una correcta transmisión de datos entre estas tarjetas.	Se sabe que la longitud máxima del cable podría llegar a ser de 1.5m.	Construcción de un cable de par trenzado de 1.5m para validar la transmisión de datos bajo esta condición.	1	No hay pérdida de información debido a la longitud del cable.	Se reciben datos, sin embargo los datos recibidos contienen errores.	No se reciben más del 50% de datos sin errores.
		La tarjeta HK_C debe ser capaz de responder a las interrupciones generadas por el HK_MB y después volver al ciclo normal de trabajo sin que sus tareas se vean afectadas.	Generar interrupciones durante el ciclo normal de trabajo del HK_C.	1	Se generan y ejecutan todas las interrupciones.	Se generan y ejecutan más del 50% de las interrupciones.	Se ejecutan menos del 50% de las interrupciones generadas.
		Si la transmisión de datos es más rápida que la lectura, la memoria del buffer podría saturarse, provocando la pérdida de datos.	Prueba de envío de datos al FPGA durante 24 horas para verificar que el buffer de datos nunca se sature. (Verificar FIFO)	1	Siempre hay una lectura de los datos.	-	se pierden datos por saturación del buffer

		El subsistema Housekeeping es un subsistema de velocidad lenta, es decir que el tiempo en el que puede realizar un ciclo está entre 1-5 segundos. Es importante medir el tiempo que tardara en leer a estas tarjetas para ir calculando el tiempo de ejecución de un ciclo completo.	Prueba del tiempo de ejecución de la lectura de 20 tarjetas HK_C	1	Tiempo de lectura menor a 1 segundo	No debe exceder 1.5 segundos	Tiempo de lectura excede los 3 segundos
Comunicación RS422	Es esencial la comunicación con CPU, ya que el Housekeeping recibe de este instrucciones indispensables para el encendido/apagado de todos los subsistemas del JEM-EUSO	Se sabe que la longitud máxima del cable podría llegar a ser de 1.5m.	Construcción de un cable de par trenzado de 1.5m para validar la transmisión de datos bajo esta condición. Esta tarea se realizará durante 24 horas con interrupciones cada 5 minutos.	1	El contador de interrupciones cuenta hasta 280 interrupciones en 24 horas.	Atiende más del 50% de interrupciones.	Atiende menos del 50% de las interrupciones.
		El Housekeeping debe ser capaz de responder a las interrupciones generadas por CPU y después volver al ciclo normal de trabajo sin que sus tareas se vean afectadas.	Generar interrupciones durante el ciclo normal de trabajo del HK.	1	Se generan y ejecutan todas las interrupciones.	Se generan y ejecutan más del 50% de las interrupciones.	Se ejecutan menos del 50% de las interrupciones generadas.

Lectura del ADC (Simula sensores de temperatura)	La tarjeta HK_MB debe encargarse de la lectura de 400 sensores de temperatura. Por el momento no se cuenta con estos, por lo que se hará la lectura simulada de los 400 sensores a partir de un solo sensor de temperatura.	Se quiere ver si el hecho de agregar cables que estén transmitiendo información afecta a la lectura del ADC del sensor.	Realizar las 400 lecturas y ver si esto afecta de forma positiva las lecturas.	2	Las 400 lecturas se realizan de forma correcta y la información transmitida por los otros cables no afecta a las lecturas.	se realizan más del 50% de las lecturas de forma exitosa.	Más del 50% de las lecturas presenta un error.
		El subsistema Housekeeping es un subsistema de velocidad lenta, es decir que el tiempo en el que puede realizar un ciclo está entre 1-5 segundos. Es importante medir el tiempo que tardara en leer los sensores para ir calculando el tiempo de ejecución de un ciclo completo de monitoreo del JEM-EUSO.	Prueba del tiempo de ejecución de la lectura de los 400 sensores.	1	Tiempo de lectura menor a 1 segundo	No debe exceder 1.5 segundos	Tiempo de lectura excede los 3 segundos

Nota: El cuadro en sombra representa el resultado de la prueba.

Referencias

[1] **JEM-EUSO Collaboration.** «EUSO-Balloon: a pathfinder mission for JEM-EUSO», 2011. [En línea]. Disponible: http://euso-balloon.lal.in2p3.fr/IMG/pdf/PurpleBook_2010_v-5_MCM_lightreso.pdf. [Último acceso: 18 Septiembre, 2014].

[2] **A. De la Cruz, C. López, J. Rojas, S. Silvarán, L. Santiago, G. Medina-Tanco. «**Housekeeping UNAM-México**».** México, D.F., 2012.

[3] **Vijay Savani, Nagendra Gajjar** . **«**Development of SEU Monitor System for SEU Detection and Correction in virtex-5 FPGA**»**. INSTITUTE OF TECHNOLOGY, NIRMA UNIVERSITY, AHMEDABAD – 382 481, 08-10 DECEMBER, 2011. 978-1-4577-2168-7/11/$26.00 ©2011 IEEE.

[4] **Dan Fay, Alex Shye, Sayantan Bhattacharya, and Daniel A. Connor**. **«**An Adaptive Fault-Tolerant Memory System for FPGA-based Architectures in the Space Environment**»**. University of Colorado. Second NASA/ESA Conference on Adaptive Hardware and Systems (AHS 2007) 0-7695-2866-X/07 $25.00 © 2007 IEEE.

[5] **Gaceta UNAM**. «Pixqui, Proyecto de Tecnología Espacial,» *Gaceta UNAM,* p. 12, 26 Agosto 2013.

[6] **A. De la Cruz, C. López, J. Rojas, S. Silvarán, L. Santiago, G. Medina-Tanco.** «Diseño, Construcción y Validación del Sistema de Monitoreo y Adquisición de Datos (Housekeeping) para el Foto Detector a Bordo de la Misión EUSO-BALLOON». México, DF. 2013.

[7] **Xilinx, Inc**. «Virtex-5 FPGA Packaging and Pinout Specification (v4.8.1)», Guía de usuario UG195. June 14, 2012.

[8] **Xilinx, Inc.** «Virtex-5 FPGA Data Sheet: DC and Switching Characteristics (v5.3)», Hoja técnica DS202. May 5, 2010.

[9] **Xilinx, Inc**. «XPS General Purpose Input/Output (GPIO) (v1.00a)», Hoja técnica DS569. July 22, 2008.

[10] **Xilinx, Inc**. «Processor Local Bus (PLB) v4.6 (v1.03a)», Hoja técnica DS531. July 28, 2008.

[11] **Alan Burns and Andy Wellings Paperback. «**Real-Time Systems and Programming Languages**».** Ada 2005, Real-Time Java and C/Real-Time POSIX. Addison Wesley Longmain. April, 2009. ISBN: 978-0-321-41745-9

[12] **Gustavo A. Medina-Tanco**. «JEM-EUSO, UNAM », [En línea]. Disponible: http://jem-euso.nucleares.unam.mx. [Último acceso: 17 Septiembre, 2014].

[13] **Gustavo A. Medina-Tanco**. «United Nations Office for Outer Space Affairs», 18 Febrero 2013. [En línea]. Disponible: http://www.oosa.unvienna.org/pdf/pres/stsc2013/tech-35E.pdf. [Último acceso: 17 Septiembre, 2014].

[14] **Grout, Ian. «**Digital Systems Design with FPGAs and CPLDs». Elsevier, 2008. ISBN-13: 978-0-7506-8397-5.

[15] **Sass, Ron y Schmidt, Andrew G.** «Embedded Systems Design with Platform FPGAs Principles and Practices». EUA : Morgan Kaufmann Publishers, 2010.

[16] **Chu, Pong P. «**FPGA Prototyping by VHDL Examples». John Wiley-Interscience, 2008. ISBN: 978-0-470-18532-2.

[17] **Charles H. Roth, Jr.** «Digital Systems Design Using VHDL». EUA: PWS Plushing Company, 1998.

[18] **Zeidman, Bob**. «Designing with FPGAs and CPLDs». EUA : CMP Books, 2002.

[19] **C., Maxfield.** «The Design Warrior's Guide to FPGAs». Elsevier, 2004. ISBN: 0-7506-7604-3.

[20] **Ashenden, Peter J.** «The Designer's Guide to VHDL: Systems on Sylicon». EUA: Morgan Kaufmann, 2001.

[21] **Cofer, R. C. y Harding, Benjamin F.** «Rapid system prototyping with FPGAs: Accelerating the Design Process». Elsevier, 2006. ISBN-13: 978-0-7506-7866-7

[22] **Xilinx, Inc**. «LogiCORE IP Processor Local Bus (PLB) v4.6 (v1.05a)», Hoja técnica, DS531. September 21, 2010. DS531.

[23] **Xlinx, Inc**. «MicroBlaze Processor Reference Guide (v9.0) », Guía de usuario UG081 [en línea]. Disponible: http://www.xilinx.com/support/documentation/sw_manuals/mb_ref_guide.pdf [Último acceso el 10 Septiembre, 2014].

[24] **Xilinx, Inc.** «Embedded System Tools Reference Manual», Guía de usuario UG111 (v14.5). June 23, 2006.

[25] **Tomasi, W.** «Sistemas de Comunicaciones Electrónicas» (Cuarta Edición). Pearson Publications Company. 2003.

[26] **Mano, M. M.** «Digital Design: With an Introduction to the Verilog HDL» (Quinta Edición). Prentice Hall. 2012.

[27] **Smith, M.** «Application Specific Integrated Circuits». Addison-Wesley, 1999. ISBN 0-201-50022-1.

[28] **Skahill, K.,** «VHDL for Programmable Logic». Addison-Wesley, 1996. ISBN 0-201-89573-0.

[29] **Analog Devices.** «SPI Interface», Nota de aplicación Rev. 0 AN-1248. [En línea]. Disponible: http://www.analog.com/static/imported-files/application_notes/AN-1248.pdf. [Último acceso: 15 Septiembre, 2014].

[30] **Texas Instruments.** «AN-214 Transmission Line Drivers and Receivers for TIA/EIA». Reporte de aplicación SNLA137A. Mayo, 2004. Disponible: http://www.ti.com/lit/an/snla137a/snla137a.pdf. [Último acceso: 10 Septiembre, 2014].

[31] **Texas Instruments.** «AN-1031 TIA/EIA-422-B Overview». Reporte de aplicación SNLA044B. Enero, 2000. [En línea]. Disponible: http://www.ti.com/lit/an/snla044b/snla044b.pdf. [Último acceso: 10 Septiembre, 2014].

[32] **George C. Marshall Space Flight Center, NASA**. «EEE Parts Management and Control for MSFC Space Flight Hardware», Document No.: MSFC-STD-3012, December 17, 1999. Pág. 16.

[33] **S. N. S. N. Muhammad A**. «The AVR Microcontroller and Embedded System, Using Assembly and C». New Yersey, USA. Prentice Hall, 2011.

[34] **Xilinx, Inc.** «Xilinx Platform Studio». [En línea] Disponible: http://www.xilinx.com/tools/xps.htm [Último acceso: 9 Septiembre, 2014].

[35] **Xilinx, Inc.** «Xilinx Software Development Kit (SDK)». [En línea]. Disponible: http://www.xilinx.com/tools/sdk.htm [Último Acceso: 9 Septiembre, 2014].

[36] **Xilinx, Inc**. «MicroBlaze Soft Processor Core». [En línea]. Disponible: http://www.xilinx.com/tools/microblaze.htm [Último acceso: 9 Septiembre, 2014].

[37] **Shaoyi Cheng, Vincent Lee, John Wawrzynek**, «EECS150 Components and Design Techniques for Digital Systems: Lab 5B, Xilinx Embedded System Development». University of Caliornia Berkeley: 6 Marzo 2013. [En línea]. Disponible: http://www-inst.eecs.berkeley.edu/~cs150/sp13/lab5/lab5B.pdf. [Último acceso el 10 Septiembre, 2014].

[38] **Xilinx, Inc.** «PicoBlaze 8-Bit Microcontroller for Virtex-E and Spartan-II/IIE Devices», Nota de aplicación XAPP213 (v2.1). Febrero 4, 2003. [En línea]. Disponible: http://www.xilinx.com/support/documentation/application_notes/xapp213.pdf [Último acceso el 10 Septiembre, 2014].

[39] **Xilinx, Inc.** «Embedded Processor Block in Virtex-5 FPGAs Reference Guide», Guía de usuario UG200 (v1.8). Febrero 24, 2010. [En línea]. Disponible: http://www.xilinx.com/support/documentation/user_guides/ug200.pdf [Último acceso: 10 Septiembre, 2014].

[40] **Xilinc, Inc.** «PowerPC 440 Virtex-5». [En línea]. Disponible: http://www.xilinx.com/products/intellectual-property/ppc440_virtex5.htm [Último acceso: 10 Septiembre, 2014].

[41] **Xilinx, Inc.** «Embedded Design with The PowerPC 440 Processor Core» [En línea]. Disponible: http://www.xilinx.com/training/embedded/embedded-design-with-powerpc-440-video.htm [Último acceso: 10 Septiembre, 2010]

[42] **A. S. R. Sass**, «Embedded Systems Design with Platform FPGAs: Principles and Practices». Elsevier, 2010. ISBN 978-0-12-374333-6.

[43] **IBM Microelectronics**. «CoreConnect Bus Architecture», 1999. [En línea]. Disponible: https://www-01.ibm.com/chips/techlib/techlib.nsf/techdocs/852569B20050FF7785256991004DB5D9/$file/crcon_pb.pdf. [Acceso el 14 Septiembre 2014].

[44] **Aguayo E, González I. y Boemo E.** «Tutorial Xilinx MicroBlaze». Escuela Politécnica Superior, Universidad Autónoma de Madrid, España, 2004. [En línea]. Disponible: http://arantxa.ii.uam.es/~ivan/microblaze-jcra04.pdf [Último acceso: 14 Septiembre, 2014].

[45] **A. Burns, A. Wellings**. «Chapter 14: Distributed systems», Abril 2009. [En línea]. Disponible: http://www.cs.york.ac.uk/rts/books/RTSbookFourthEdition/distributedSystems.pdf. [Último acceso: 11 Septiembre 2014].

[46] **U. Meyer-Baese**. «Digital Signal Processing with Field Programmable Gate Arrays». (Segunda edición), Florida: Springer, 2004. [En línea]. Disponible: http://books.google.com.mx/books?id=wzYuOF6HFX0C&pg=PA3&dq=classification+of+VLSI+circuits&hl=es&sa=X&ei=sc4YVNfmCcqrggSghoC4AQ&ved=0CCAQ6AEwATgU#v=onepage&q=classification%20of%20VLSI%20circuits&f=false

[47] **S. M. Spano**, «Achieve Performance Increases and Product Differentiation with OPB Mastering» Embedded magazine, vol. IV, pp. 59-62, 2006.

[53] **ITT Cannon**. **«**Cannon Microminiature Connectors », Catálogo, 2006. [En línea]. Disponible: http://www.farnell.com/datasheets/718323.pdf [Último acceso: 8 Agosto, 2014].

[48] **Xilinx, Inc**. «NASA JPL: Mars Exploration Rovers». [En Línea]. Disponible: http://www.xilinx.com/about/customer-innovation/aerospace-and-defense/mars-exploration-rovers.htm [Último acceso: 23 Octubre, 2014].

[49] **Dmitriy L. Bekker, Thomas A. Werne, Thor O. Wilson, Paula J. Pingree , Kiril Dontchev, Michael Heywood, Rafael Ramos, Brad Freyberg, Fernando Saca, Brian Gilchrist, Alec Gallimore, James Cutler**. «A CubeSat Design to Validate the Virtex-5 FPGA for Spaceborne Image Processing». Jet Propulsion Laboratory-University of Michigan. Aerospace Conference, 2010 IEEE.

[50] JEM-EUSO collaboration. «Web site for the EUSO-Ballon project»,2011-2014. [En Línea]. Disponible: http://euso-balloon.lal.in2p3.fr/. [Último acceso: 20 Octubre 2014].