Universidad de Alcalá
Escuela Politécnica Superior

GRADO EN INGENIERÍA ELECTRÓNICA DE COMUNICACIONES

Trabajo Fin de Grado

Desarrollo de un backend para un receptor radioastronómico de ocultaciones lunares

Autor: Alberto Río Martínez

Ingeniero que avala el proyecto: Pablo García Carreño

Tutor colaborador de la UAH: Raúl Mateos Gil

2023

Resumen

El presente Trabajo Fin de Grado tiene como objetivo fundamental el diseño de un sistema basado en un SoC (System on Chip) programable que permita realizar, en tiempo real, la Transformada Rápida de Fourier sobre señales obtenidas mediante observaciones lunares.

Este trabajo constituye la primera aproximación al uso de FPGAs para la implementación de back-ends radioastronómicos del Observatorio de Yebes, institución que financia y avala el proyecto. Para llevar a cabo esta tarea, se ha optado por emplear la tarjeta de evaluación Zedboard y el conversor analógico digital AD9467, los cuales se pueden conectar entre ellos.

Palabras clave: Zedboard, Backend de recepción radioastronómica, Acceso Directo a Memoria, AD9467, Procesamiento digital de señales con algoritmo FFT.

Abstract

The main goal of the following project is the design of a system based on a programmable SoC (System on Chip) that allows performing, in real time, the Fast Fourier Transform on signals from lunar observations.

This project means the first approach of the “Observatorio de Yebes” on the use of FPGAs for implement radio astronomy back-ends. To carry out this task, the chosen devices are Zedboard evaluation card and AD9467 analog-to-digital converter, which can stablish connections among them.

Main words: Zedboard, Radio Astronomy Reception Back-End, Direct Memory Access, AD9467, Digital signal processing with FFT algorithm.

Resumen extendido

El uso de FPGAs en observaciones astronómicas está muy extendido en la actualidad debido a la gran escalabilidad y calidad que pueden llegar a ofrecer. Sin duda, una de las principales razones de este auge, es el considerable aumento en la fiabilidad de los sistemas basados en estos dispositivos, ya que, las funciones que desempeñan implican un menor número de integrados en la placa de circuitos y, por ende, una reducción en el número de fallos de los mismos [1].

Sin embargo, la razón principal para el uso de estos dispositivos en radioastronomía es la creciente necesidad de procesar grandes cantidades de datos a la máxima velocidad posible. En este marco los SoC basados en FPGA son ideales, ya que ofrecen la escalabilidad y reprogramabilidad de las FPGA ligado a las velocidades de procesamiento del SoC.

El Observatorio de Yebes es una unidad del Instituto Geográfico Nacional dedicada al desarrollo y construcción de instrumentación en el campo de la radioastronomía [2]. Nanocosmos es uno de los numerosos proyectos en los que esta organización se ve envuelta. Su objetivo principal es revelar las condiciones físicas y químicas en la zona de formación de polvo cósmico de las estrellas evolucionadas, haciendo uso de las nuevas técnicas y capacidades de observación. Una de esas técnicas es la observación de ocultaciones lunares, para las cuales son necesarios back-ends de recepción de una resolución mayor a la de los que dispone el Observatorio actualmente.

Las ocultaciones lunares consisten en la alineación o interposición de un cuerpo celeste por la luna. Para realizar la observación de dicho fenómeno, hay que realizar un barrido de datos a muy alta frecuencia y en tiempo real. Uno de los principales problemas que presenta dicha observación es el procesado y tratamiento de grandes volúmenes de datos a muy alta velocidad, por lo que es conveniente el uso de implementaciones basadas en FPGAs, no solo por la velocidad que ofrecen, sino por los múltiples modos de operación que permiten desarrollar.

Para este diseño se pretende utilizar la tarjeta Zedboard, una SoC basado en FPGA de la familia Zynq de Xilinx. Este tipo de dispositivo consta de dos bloques principales. El primero de ellos, denominado Processing System o PS, está compuesto por un procesador ARM9 de doble núcleo y toda una serie de periféricos y controladores de memoria necesarios para su operación. El otro bloque, denominado Programmable Logic o PL, es un bloque donde se modela la lógica configurable. Ambos bloques se conectan mediante enlaces AXI (Advanced eXtensible Interface). Mientras que la lógica programable provee de funcionalidades como lógica de alta velocidad o aritmética, el sistema de procesamiento proporciona soporte para la ejecución de rutinas software y/o sistemas operativos.

El sistema constará de una primera etapa de adquisición y digitalización de los datos, tarea que se llevará a cabo con el conversor analógico digital AD9467, el cual está

conectado a la tarjeta Zedboard por medio de un puerto de alta velocidad de la misma. En la etapa intermedia los datos serán procesados por un bloque que aplicará el algoritmo de la Transformada rápida de Fourier sobre ellos. Por último, se procederá a almacenar los datos en memoria haciendo uso de un controlador de DMA (Direct Memory Access). En el diseño de este enlace de comunicación se buscará la reducción del consumo de recursos.

El Observatorio de Yebes está muy interesado en este tipo de proyectos, hasta el punto de que tiene como objetivo crear un grupo de trabajo dedicado a la generación de back ends con SoCs basados en FPGA. Este proyecto implica una de las primeras aproximaciones de la organización a este tipo de sistemas, y debido a la gran escalabilidad de los mismos podría servir de base a nuevos proyectos de back-ends dedicados a otro tipo de observaciones.

Índice

Índice de figuras

Índice de tablas

Capítulo 1: Introducción

1.1 Objetivos del proyecto

Las dos principales virtudes que presentan los SoC basados en FPGA son las altas velocidades de procesamiento que pueden llegar a ofrecer y la flexibilidad que brinda su implementación, lo que facilita en gran medida la realización de operaciones lógicas a muy alta velocidad. Esta funcionalidad es ideal a la hora de desarrollar algoritmos matemáticos como los implicados en el cálculo de la Transformada de Fourier.

El objetivo fundamental del proyecto es el diseño e implementación en Zedboard de un sistema de adquisición capaz de procesar señales, provenientes de observaciones radioastronómicas, adquiridas con velocidades de muestreo de 50 MHz.

Este trabajo queda englobado dentro del proyecto Nanocosmos en el cual se encuentra incluido el Observatorio Astronómico de Yebes, donde se pretende diseñar y realizar este back-end.

1.2 Organización de la memoria

Para abordar con éxito los objetivos impuestos se va a estructurar el diseño en tres bloques:

1. Bloque de adquisición y calibración de datos con el conversor analógico digital AD9467.
2. Bloque de procesamiento de los datos adquiridos aplicando el algoritmo de la Transformada Rápida de Fourier.
3. Bloque de acceso directo a memoria haciendo uso del controlador DMA de la tarjeta de desarrollo.

Una vez funcionen estos tres bloques de manera independiente, se unirán para crear el sistema final. Para abordar estos bloques de manera efectiva es precisa una base teórica tanto de radioastronomía como de diseño de sistemas de adquisición con SoC basados en FPGA.

A continuación, se expondrán los capítulos conformantes de esta memoria con el fin de esclarecer el contenido de los mismos:

El primer capítulo supone una introducción al documento, y trata de exponer los objetivos del proyecto y el procedimiento empleado para alcanzarlos.

El segundo capítulo supone una introducción teórica a algunos conceptos básicos de radioastronomía y FPGAs que ayudarán a contextualizar el proyecto.

En el tercer capítulo se trata de dar una explicación del hardware empleado, así como de las conectividades que presenta y las herramientas empleadas para desarrollar el proyecto.

El capítulo cuarto hace referencia al desarrollo del proyecto tanto a nivel hardware como a nivel software, así como a las pruebas y resultados obtenidos.

Por último, el documento consta de un apartado donde se exponen las conclusiones tras el desarrollo del proyecto y posibles campos de estudio futuros.

Capítulo 2: Introducción teórica

2.1 Radioastronomía

La radioastronomía es una rama de la astronomía encargada de estudiar los cuerpos celestes por medio de sus emisiones en el dominio radioeléctrico. El nivel de potencia de estas señales es extremadamente débil. Eso hace que su recepción sea muy compleja requiriendo dispositivos diseñados ad hoc para esta tarea, como los radiotelescopios. Generalmente, los radiotelescopios consisten en una gran antena parabólica capaz de captar ondas electromagnéticas.

2.1.1 Contexto histórico

Casi todo lo que sabemos sobre el espacio, es decir, las estrellas y el medio interestelar, se ha obtenido de la radiación electromagnética. Los portadores de información material (meteoritos que impactan la tierra, partículas de rayos cósmicos o muestras de material recolectado) nos proveen una información ínfima a su lado [3].

La radiación electromagnética que proviene del espacio es absorbida, en su gran mayoría, por la atmósfera terrestre. Sin embargo, existen bandas para las cuales la atmósfera es transparente, y son dos de estas bandas las que destacan notablemente sobre las demás por su anchura. La primera de estas bandas es la ventana óptica, que comprende las ondas electromagnéticas pertenecientes al espectro visible, con longitudes de onda desde los 300 a los 1000 nm. La segunda es la ventana radio, que es la que nos ocupa, y comprende longitudes de onda de entre 1 mm y 15 m (300 GHz a 20 MHz). La Figura 1 muestra un diagrama con las ventanas para la radiación electromagnética.

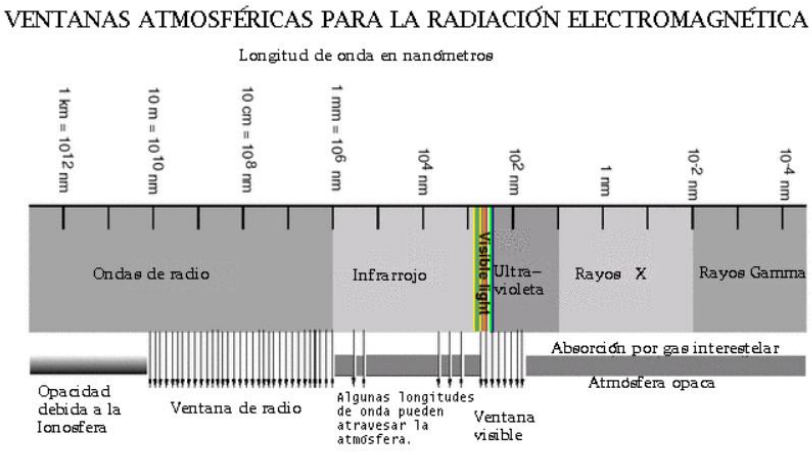


Figura 1: Ventanas atmosféricas para la radiación electromagnética (extraída de https://www.qsl.net/ea1crk/medio/Cap4.htm).

Durante muchos años, la humanidad estuvo restringida a las mediciones realizadas en lo visible. No fue hasta la época de Herschel cuando este rango de longitud de onda se amplió ligeramente hasta longitudes de onda cercanas al infrarrojo, y en 1930, se extendió desde el ultravioleta cercano hasta el infrarrojo cercano: 0,35 μm ≤ λ ≤ 1 μm.

En 1932, Karl Jansky realizó las primeras observaciones radioastronómicas en los laboratorios de Bell Telephone. Allí consiguió captar una señal a 20,5MHz investigando interferencias radioeléctricas. Tras esto, continuó sus observaciones sin mucho impacto hasta que, en 1937, Grote Reber llevó a cabo mediciones de ondas a 160 MHz con un instrumento que desarrolló a partir de las investigaciones de Jansky, y que hoy en día se considera como el primer prototipo de radiotelescopio. Operando en esas frecuencias consiguió captar ondas provenientes del plano de la vía láctea, con las que pudo desarrollar los primeros mapas del cielo en frecuencias de radio, los cuales publicó en 1944. La Segunda Guerra Mundial derivó en una mejora considerable en los receptores para uso militar, pero, una vez concluyó, algunos investigadores dirigieron su atención hacia el "ruido" de radio proveniente de fuentes extraterrestres [3]. La Figura 2 muestra las antenas desarrolladas por Jansky y Reber.

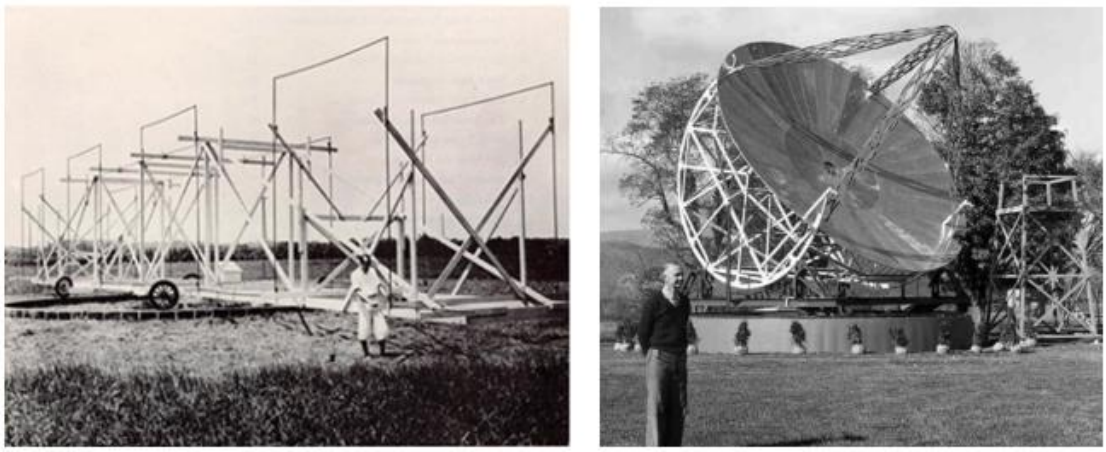


Figura 2: A la izquierda, antena de Karl Jansky (extraído de https://alpoma.net/tecob/?p=5240), a la derecha, antena de Grote Reber (extraído de https://www.astromia.com/fotohistoria/telescopioreber.htm).

Más tarde, las observaciones se vieron ampliadas hasta el infrarrojo intermedio por Pettit y Nicholson, alcanzando longitudes de onda de 1 a 4 micras. Fueron seguidas por las primeras observaciones de estrellas en el infrarrojo lejano, de longitudes de onda de 8 a 14 micras y 4 a 5.5 micras, y fueron desarrolladas por Murray, Wildey, Johnson y Mitchell. Por último, en 1965 Arno Penzias y Robert Wilson captaron accidentalmente la radiación cósmica de fondo trabajando con una antena de bocina de 6 metros, hallazgo por el cual recibieron el premio Nobel en el año 1978.

Este ha sido un pequeño repaso histórico de los inicios de la radioastronomía a nivel global, pero también es interesante exponer algunos de los hallazgos de la radioastronomía en España. El Observatorio de Yebes ha sido una parte fundamental de dichos hallazgos y es una instalación muy importante a nivel internacional.

En la década de 1970, gracias a los Planes de Desarrollo, el Real Observatorio de Madrid (ROM) recibió una inyección de capital que empleó en crear el Observatorio de Yebes. La instrumentación desarrollada para este centro consistía en un astrógrafo doble de 40 cm, un telescopio óptico infrarrojo de 1,5 m de diámetro y un radiotelescopio de 14 m de diámetro para longitudes de onda milimétricas, mostrado en la Figura 3. Este último fue de vital importancia, ya que con él se comenzaron las observaciones de Interferometría de Muy Larga Base (VLBI) en el Observatorio y se llevó a cabo una de las principales aportaciones científicas del momento en el panorama nacional. Esta consistía en el seguimiento de las variaciones de intensidad de la emisión de los máseres circunestelares de óxido de silicio en estrellas evolucionadas.

Figura 3: Radiotelescopio de 14 m del Observatorio Astronómico de Yebes [15].

La creciente mejora de los instrumentos radioastronómicos a finales de los años 80 y principios de los 90 ocasionó una demanda de mejora en las instalaciones del Observatorio de Yebes. En este contexto, para mantener y desarrollar la radioastronomía experimental en España era preciso dejar a un lado el radiotelescopio de 14 metros y desarrollar uno nuevo de mayores prestaciones. Se determinó que la instalación idónea a generar era un radiotelescopio de 40 metros, el cual fue inaugurado el 26 de abril de 2005 y se muestra en la Figura 4. No fue hasta 2007 cuando se llevó a cabo la primera prueba de observación, y en 2008 se realizó la primera observación VLBI. A partir de este punto, el radiotelescopio de 40 metros ha participado en numerosos proyectos de gran interés científico, tanto de carácter nacional como internacional. Algunos de los proyectos en los que se ve inmerso en la actualidad son: RADIONET (BRAND, AETHRA y TOG), ALMA, IRAM (NOEMA), NANOCOSMOS, SKA y JUMPING JIVE.

Figura 4: Radiotelescopio de 40 metros del Observatorio de Yebes [15].

Por último, cabe destacar que el desarrollo histórico ha tendido hacia la obtención de una mayor sensibilidad, una longitud de onda más corta y una mayor resolución angular en las observaciones.

2.1.2 Gestión del espectro en radioastronomía

La gestión del espectro es un conjunto de procedimientos administrativos y técnicos que tienen como fin una correcta asignación del espectro de radiofrecuencia. El espectro es un bien limitado y gestionado por el estado.

En el caso de la radioastronomía, se trata de un servicio de radio pasivo cuyo objetivo principal es la búsqueda y recepción de señales con niveles de potencia muy bajos. Este contexto hace que este tipo de servicio de radio sea especialmente vulnerable a posibles interferencias generadas por otros servicios de radio activos. Dichas interferencias son provocadas, principalmente, por servicios que comparten la misma banda de frecuencia, pero también pueden ser generados a partir de transmisiones que no pertenecen a la misma banda.

El reconocimiento de la radioastronomía como un servicio de radio de importancia pareja a la de otros servicios, como el de las comunicaciones móviles, supuso la creación de una base legal que brinda de protección frente a interferencias perjudiciales para las observaciones. Dicha protección se obtiene a través del establecimiento de zonas de coordinación para los servicios terrestres. En estas zonas de coordinación se garantiza que las transmisiones ajenas a fines radioastronómicos no generen interferencias perjudiciales.

Las bandas dedicadas a la radioastronomía en la región Europa, Asia y África son las siguientes:

- Bandas de 37.5 a 38 MHz y de 38 a 38.25 MHz: Bandas atribuidas globalmente a la radioastronomía con atribuciones secundaria y primaria, respectivamente. Estas bandas son compartidas con servicios fijos y móviles.
- Banda de 406.1 a 410 MHz: Esta banda es una de las más utilizadas en observaciones radioastronómicas continuas. Dispone de atribución primaria compartida con los servicios fijo y móvil.
- Banda de 1400 a 1427 MHz: Muy importante para los estudios de la línea de hidrógeno y la naturaleza de las galaxias. Dispone de atribución primaria compartida con el servicio de exploración de la Tierra por satélite y el servicio de investigación espacial.
- Bandas de 1610.6 a 1613.8 MHz y 1660 a 1670 MHz: Bandas muy utilizadas para realizar observaciones de las líneas espectrales del radical hidrógeno e hidroxilo, las cuales son de vital importancia por su amplia distribución en el universo. Esta distribución permite llevar a cabo estudios sobre la naturaleza y evolución del mismo universo. Dichas bandas comparten categoría primaria con los servicios de móvil por satélite, radionavegación aeronáutica, servicios de investigación espacial y con el servicio de ayudas a la meteorología. Estos servicios con los que comparte banda dificultan mucho las observaciones en este rango de frecuencias.

- Banda de 2690 a 2700 MHz: Banda de vital importancia para la observación astronómica debido a su baja radiación de fondo galáctico y su gran productividad a la hora de realizar estudios de las nubes de hidrógeno ionizado, y de la radiación difusa general de la galaxia. Esta banda dispone de atribución primaria, y está compartida con el servicio de exploración de la Tierra por satélite y el servicio de investigación espacial.

- Bandas de 4800 a 4990 MHz y de 4990 a 5000 MHz: Estas bandas de frecuencia son utilizadas para los estudios de distribuciones de brillo de objetos galácticos tales como, nubes de hidrógeno ionizado y remanentes de supernovas. Las bandas de frecuencia en cuestión disponen de atribuciones secundaria y primaria, respectivamente, y se encuentran compartidas con servicios fijo y móvil.

- Banda 10.68 a 10.7 GHz: Banda con título primario compartida para usos de radioastronomía, exploración de la Tierra por satélite e investigación espacial. Es una de las bandas más utilizadas para observaciones continuas.

- Banda de 14.47 a 14.5 GHz: Esta banda ha sido determinada por los radioastrónomos como una de las más importantes a frecuencias por debajo de 275 GHz. Tanto es así que se solicita a los servicios fijos y móviles que no asignen frecuencias en esta banda.

- Banda de 15.35 a 15.4 GHz: Otra de las bandas predilectas de los radioastrónomos para la realización de observaciones continuas. Esta banda también se encuentra compartida con servicios de exploración de la Tierra por satélite e investigación espacial.

- Banda de 22.21 a 22.5 GHz: Esta banda está definida por los radioastrónomos como “la banda preferida” para la realización observaciones continuas. También dispone de atribución primaria y es compartida con los servicios fijo, móvil, exploración de la Tierra por satélite e investigación espacial.

- Banda de 23.6 a 24 GHz: Está banda es la primaria en servicios de radiocomunicaciones pasivos, es decir, servicios de radioastronomía, exploración de la Tierra por satélite y servicios de investigación espacial.

- Bandas por encima de 40 GHz: Existen bandas de frecuencias superiores a 40 GHz destinadas a la radioastronomía, pero las interferencias generadas por otros servicios activos en estos rangos son mínimas, por lo que no suelen suponer un problema. En cuanto a las observaciones que se llevan a cabo, son de carácter continuo o de líneas espectrales.

Una vez expuestas las bandas frecuenciales del espectro radioeléctrico más relevantes para uso radioastronómico, es menester contextualizar estos rangos frecuenciales para el caso que no ocupa. El observatorio de Yebes dispone de un radiotelescopio de 40 m con el que se pueden llevar a cabo observaciones de antena única y de VLBI, que se llevan a cabo en conjunto con otros radiotelescopios, este concepto será expuesto más adelante con más detalle. Las bandas de frecuencia con las que se realizan las observaciones difieren si son de antena única o de VLBI.

Para el caso de antena única las tres bandas de frecuencia fundamentales son: 18-26 GHz, 32-50 GHz y 72-90 GHz. Las restantes se encuentran en rangos superiores a los 40 GHz, estas bandas son muy selectas ya que no hay demasiados sistemas en la actualidad capaces de operar a tan altas frecuencias.

En cuanto a las observaciones VLBI, el Observatorio de Yebes colabora con redes tales como la red europea de VLBI EVN, la red VLBI milimétrica GMVA, la red geodésica International VLBI Service y otras redes como KaVa y Radioastron.

2.2 El radiotelescopio

El elemento fundamental en toda observación radioastronómica es el radiotelescopio, cuya función principal es captar la energía emitida por una fuente cósmica con una antena parabólica. Dichas señales de energía tienen niveles de potencia muy bajos debido a las grandes distancias a las que se encuentran. El tamaño de la parábola de la antena debe ser grande para captar dichas señales, y depende de la frecuencia de funcionamiento.

En cuanto al funcionamiento, la energía que llega se concentra en el foco, donde se encuentra el receptor. La energía llega a través de una onda plana en la dirección del eje óptico mediante reflexión, generándose un diagrama de difracción centrado en el citado eje óptico. La Figura 5 muestra las partes de un radiotelescopio.

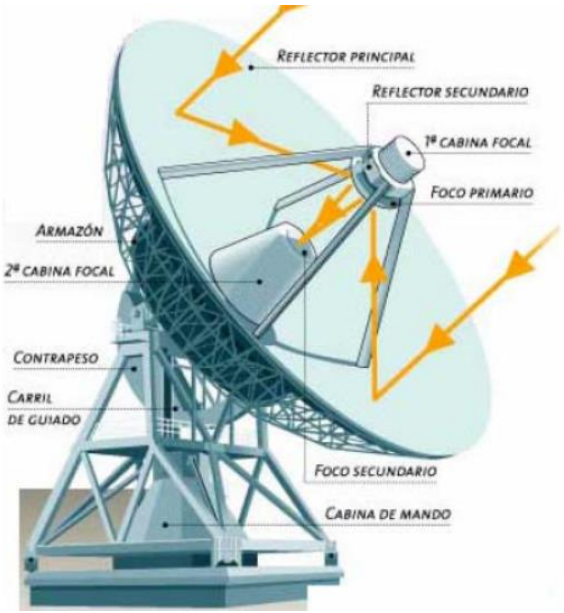


Figura 5: Partes del radiotelescopio (extraída de http://giomepe1214.blogspot.com/2017/09/radiotelescopio.html).

Como se ha indicado anteriormente, para captar señales de manera óptima se deben utilizar grandes antenas o grupos de antenas trabajando en paralelo (array), esto permite observar el espectro radioeléctrico de una parte del cielo. Pero el tamaño de la antena no se ve solo motivado por la capacidad de captación de energía, sino también por la resolución. La resolución del radiotelescopio es la distancia angular que existe entre dos puntos para que puedan ser identificados por separado, y es un parámetro esencial a la hora de diseñar un radiotelescopio.

La antena solo constituye el primer elemento del radiotelescopio, esta va seguida de un receptor y un sistema de procesamiento con el que se lleva a cabo el tratamiento de la señal captada. En la Figura 6 se muestra el esquema básico de un radiotelescopio.

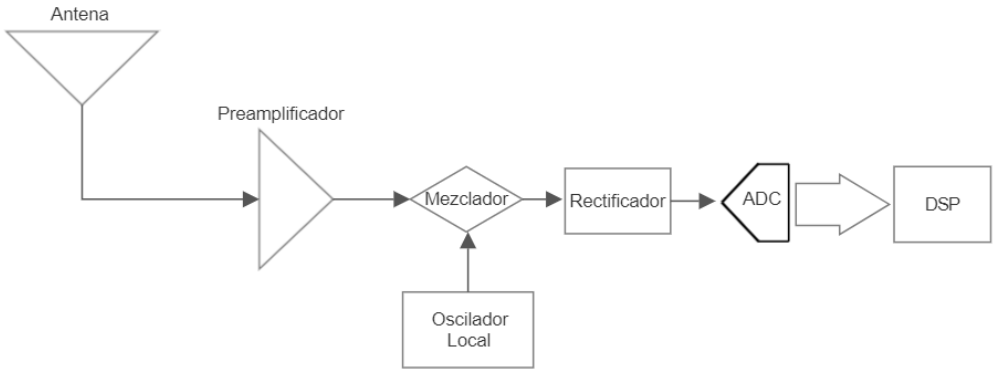


Figura 6: Diagrama de bloques de un radiotelescopio.

2.3 El receptor radioastronómico

Como se ha indicado antes, la energía que es capaz de captar la antena del radiotelescopio se concentra en el foco, donde se encuentra el receptor. Un receptor de radioastronomía es un elemento que se encarga de transformar correctamente la energía captada por la antena. Dichas señales de energía suelen tener niveles de potencia muy bajos, por lo que es necesario amplificarlas adecuadamente en la etapa de recepción. Además, se llevan a cabo procesamientos en las señales tales como conversión a otro rango de frecuencias o filtrado de las mismas que permiten adaptar la señal a las necesidades del backend. Además, si se llevan a cabo técnicas de interferometría se deben añadir señales de calibración y medida de tiempo, con el fin de sincronizar con otros radiotelescopios de la red. Para llevar a cabo todo el tratamiento de señal el receptor dispone de las siguientes partes:

- Criostato: Es el primer elemento que conforma la cadena de recepción de la señal, está compuesto por una bocina y amplificador de bajo ruido que se encuentran a temperaturas inferiores a los 260º bajo cero. La motivación principal de esta etapa del receptor, es que este sea lo más sensible posible, para lo cual sus elementos deben generar muy poco ruido electrónico. Esto se consigue enfriando por vacío criogénico la etapa más crítica del sistema a la temperatura especificada anteriormente.

- Conversores de frecuencia: Su uso es fundamental debido a la dificultad a la hora de procesar señales de muy alta frecuencia. Permiten convertir la señal que proviene del criostato a frecuencias más bajas mezclando la señal de la antena con una proveniente de un oscilador local. Con estas dos señales se genera una señal de frecuencia intermedia, la cual debe ser correctamente filtrada para evitar banda imagen y amplificada para adaptar su potencia a las necesidades del backend. Con esto se consigue reducir la frecuencia de la señal y adaptar su potencia a las necesidades del backend.

- Backend: Es el último elemento de la cadena receptora y su función es procesar las señales provenientes de la etapa anterior. Los backends de radioastronomía generalmente constan de una primera etapa de digitalización de los datos, seguida de un generador de transformada de Fourier. Este es el elemento en el que se va a centrar este TFG, y se ha optado por hacer uso de FPGAs para su desarrollo. El uso de FPGAs cada vez está más extendido en radioastronomía debido a su gran velocidad de procesamiento, baja potencia y flexibilidad de cara a reprogramar diferentes distribuciones hardware.

2.4 VLBI

Las siglas VLBI significan en castellano “Interferometría de muy larga línea de base” [2]. Se trata de una técnica utilizada en radioastronomía que consiste en combinar señales de múltiples radiotelescopios, separados por distancias muy grandes, para crear un telescopio virtual con un tamaño igual a la separación máxima entre los radiotelescopios.

Al combinar las señales de múltiples telescopios, VLBI puede lograr una resolución angular extremadamente alta, lo que permite a los astrónomos estudiar los detalles finos de los objetos celestes con gran precisión. Cada uno de los radiotelescopios recibirá la radiación emitida por el citado objeto en instantes de tiempo distintos, generando un patrón de interferencia denominado franja. Esto permite a la red comportarse como un único instrumento de un tamaño equivalente igual a la distancia entre radiotelescopios [4]. VLBI se utiliza para estudiar una amplia gama de fenómenos astronómicos, incluidos los agujeros negros, los cuásares y la estructura de la galaxia de la Vía Láctea.

Las señales recibidas por cada antena del array están sincronizadas con un reloj atómico de Hidrógeno de altísima precisión, el máser de hidrógeno. A continuación, dichos datos son enviados al correlador por un enlace de fibra óptica. El correlador es un computador cuyo objetivo es captar los datos de todas las antenas del array y multiplicarlos, generando así una imagen en 2-D de la región del cielo observada. La resolución obtenida es proporcional a la frecuencia de observación. Las técnicas de VLBI permiten que la distancia entre telescopios sea mucho más grande que las posibles con la interferometría tradicional. Esta última hacia uso de antenas conectadas con cable coaxial, guía de onda o fibra óptica, debido a que no existe la pérdida de potencia que genera el transporte de la señal por un medio físico.

El array más grande de interferometría existente en el mundo es La red europea de VLBI (EVN) es el array de interferometría más grande actualmente. Dispone de 21 radiotelescopios extendidos por toda Europa que observan en tres periodos por año, dichos periodos se conocen como Sesiones VLBI. La ubicación de dichos radiotelescopios se muestra en la Figura 7.

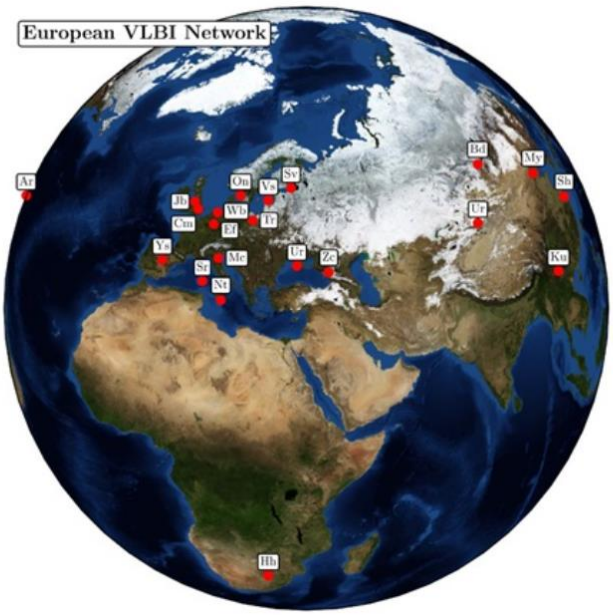


Figura 7: Posición de los radiotelescopios de la red europea de VLBI [15].

2.5 Ocultaciones lunares

Una ocultación es la alineación o interposición de un cuerpo celeste por otro visto desde la Tierra. Dependiendo del tipo de trayectoria o paso entre ambos objetos, estos eventos pueden ser:

- Totales y anulares: si se produce la desaparición o superposición completa.
- Parciales: si únicamente se tapa una parte.
- Rasante: si el paso se produce tangencialmente.

La Figura 8 muestra una representación gráfica de los eventos anteriormente descritos.

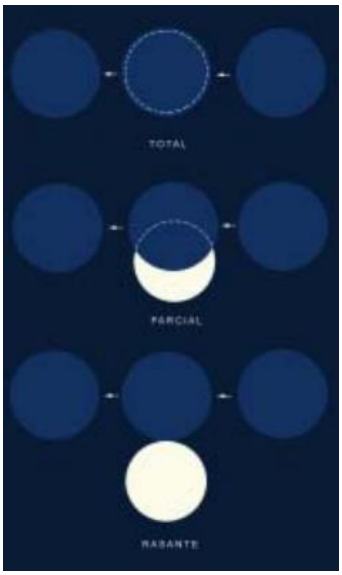


Figura 8: Tipos de ocultaciones según la alineación de los cuerpos entre los que tiene lugar el fenómeno [5].

La diferencia entre los tamaños aparentes (o vistos desde la Tierra) del cuerpo ocultante y del ocultado establecen diferentes tipos de fenómenos:

- Eclipses: cuando las dimensiones aparentes de ambos objetos son similares.
- Tránsitos: cuando el cuerpo ocultante es mucho más pequeño que el ocultado. En tal caso no se produce la desaparición del cuerpo más distante, sino un paso o tránsito del primero proyectado sobre la superficie del mayor.
- Ocultaciones: cuando el cuerpo que causa el fenómeno es mucho mayor aparentemente que el ocultado.

El foco de interés de este proyecto hay que ponerlo en las ocultaciones lunares. Un ejemplo típico son las ocultaciones de estrellas por la Luna, las estrellas son mucho mayores que nuestro satélite, pero sus tamaños aparentes son muy dispares desde nuestra perspectiva y se ven ocultadas.

Las ocultaciones han sido fuente de importantes descubrimientos en la Astronomía, tales como la determinación de la distancia Tierra-Sol, los tamaños y formas de los asteroides, la atmósfera de Plutón o la localización precisa de algunas radiofuentes [5].

2.6 FPGAs

A comienzos de la década de 1980 se hizo notoria la demanda de nuevos dispositivos programables. En un extremo se encontraban los PLD (Dispositivos Lógicos Programables), altamente configurables, pero no capaces de soportar funciones complejas. En el otro extremo se encontraban los ASIC, que sí soportaban funciones complejas pero su diseño era muy costoso y laborioso. Además, los ASIC no son reconfigurables debido a que se congelan en silicio una vez implementados.

En este contexto, y con el fin de paliar esta demanda, Xilinx desarrolló un nuevo tipo de dispositivo programable llamado FPGA (Field-Programmable Gate Array). La arquitectura de las primeras FPGA, aparece representada en la Figura 9, y se basaba en el concepto de un bloque lógico programable (CLB) que comprendía una LUT (tabla de búsqueda) de tres entradas, un flip-flop y un multiplexor [14].

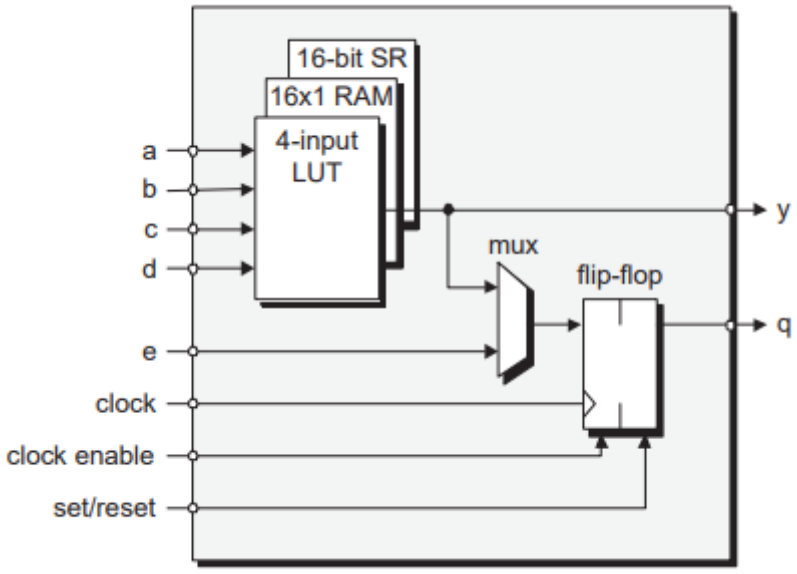


Figura 9: Diagrama de bloques de los primeros CLB [14].

La arquitectura de la parte programable del dispositivo que nos ocupa es de una complejidad mayor a la descrita anteriormente, pero la estructura básica se mantiene. La lógica programable del Zynq SoC consta de bloques lógicos configurables (CLB), cada uno de los cuales contiene dos segmentos. Cada segmento contiene cuatro tablas de búsqueda (LUT), ocho flip-flops (FF) y una matriz de conmutación que lo acompaña. La Figura 10 muestra el diagrama de bloques de un CLB.

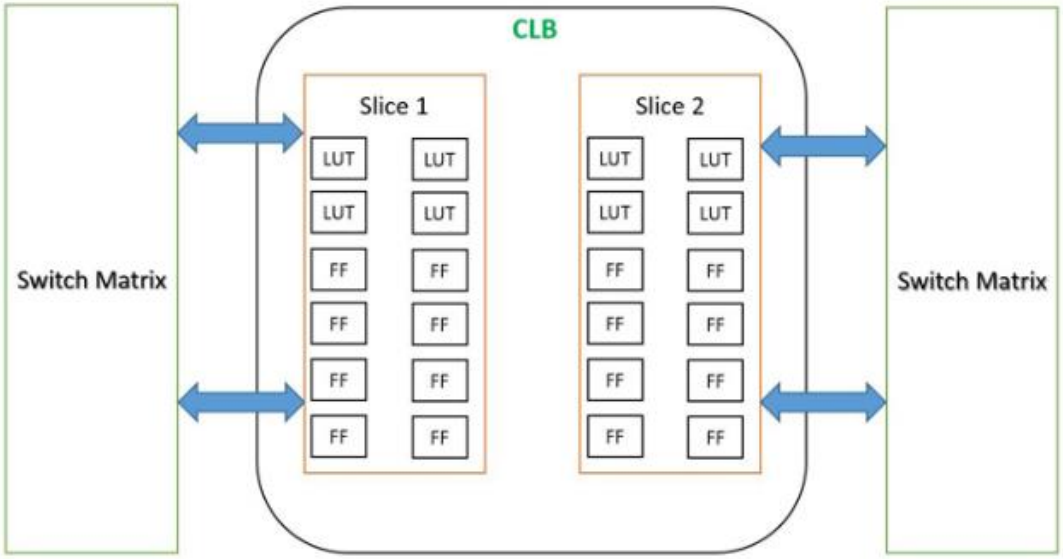


Figura 10: Arquitectura de un CLB (extraída dehttps://www.aldec.com/en/company/blog/144--introduction-to-zynq-architecture).

A continuación, se exponen con más detalle las partes del CLB:

- Slice: Pequeña unidad de lógica dentro de la estructura FPGA que se puede configurar para realizar varias funciones lógicas digitales, tales como implementar circuitos combinacionales y secuenciales.
- Tabla de búsqueda (LUT): Es un circuito digital que se puede configurar para implementar una función booleana de seis entradas. La salida de una LUT está determinada por una tabla de verdad, que es una lista de todas las combinaciones de entrada posibles y sus correspondientes salidas. Una LUT se puede utilizar para implementar lógica combinacional, como puertas AND, OR y NOT, así como funciones booleanas más complejas, como sumadores, comparadores y máquinas de estado.

- Flip-flop (FF): Es un elemento de memoria que puede almacenar un solo bit de información y puede usarse para implementar lógica secuencial. En la arquitectura Zynq se emplea para almacenar la salida de una LUT y ponerla a disposición de otras partes del diseño. También se pueden usar para implementar una lógica secuencial más compleja, como contadores, máquinas de estado y elementos de memoria.
- Matriz de conmutación: Es una red de recursos de enrutamiento programables que permite interconectar señales entre diferentes partes del diseño, como entre segmentos dentro de un CLB, entre los mismos CLB o entre la estructura FPGA y el procesador o las interfaces.

2.7 Uso de FPGAs en radioastronomía

El uso de las FPGAs está muy extendido en aplicaciones de radioastronomía para procesar y analizar las grandes cantidades de datos que suministran los radiotelescopios.

Una de las principales aplicaciones que se les otorga a estos dispositivos es el desarrollo de backends digitales para los radiotelescopios. Un backend digital es un sistema que digitaliza y procesa señales de datos. Las FPGAs resultan interesantes para este tipo de sistemas porque pueden ser programadas para llevar a cabo multitud de tareas de procesamiento de señal en tiempo real. Además, son muy útiles en el desarrollo de algoritmos de procesamiento digital de señal que puedan ser implementados en el hardware.

Las FPGAs se usan también en otros sistemas como conformadores de haz, en los cuales se combinan señales provenientes de múltiples antenas para crear una única con alta resolución. Son ideales para este tipo de sistemas ya que pueden llevar a cabo funciones de correlación en tiempo real.

En resumen, el uso de FPGAs está muy extendido en el campo de la radioastronomía para procesar y analizar grandes cantidades de datos. Pueden ser usados para el desarrollo de backends digitales, conformadores de haz y radiotelescopios de baja frecuencia, debido a su adaptabilidad y a sus capacidades de procesamiento en tiempo real y de implementación de algoritmos de procesamiento digital de señales.

2.8 Observatorio de Yebes

El Observatorio de Yebes constituye actualmente un Centro de Desarrollo Tecnológico para el Instituto Geográfico Nacional (IGN) siendo uno de los diez centros de investigación científica totalmente españoles clasificados como Gran Instalación Científica. Está dedicado al desarrollo y construcción de instrumentación en el campo de la radioastronomía, así como a la realización de observaciones astronómicas tanto de interés astronómico como geodésico o geofísico.

El Observatorio forma parte de infinidad de proyectos con empresas o instituciones internacionales para el desarrollo de receptores de radioastronomía en diferentes bandas. También participa en el desarrollo y fabricación de amplificadores criogénicos ultrasensibles. Dichos amplificadores son usados en los receptores de los radiotelescopios de la institución.

Los receptores desarrollados por el laboratorio del Observatorio son los siguientes:

- Banda S: 2.2 - 2.37 GHz
- Banda CH: 3.22 - 3.39 GHz.
- Banda C: 4.56 - 6.9 GHz
- Banda X: 8.1 - 8.9 GHz
- Banda K: 21 - 24 GHz
- Banda Q: 31.5 - 50 GHz
- Banda W: 72-90.5 GHz y 83-116 GHz.
- Tri-banda: S (2.2-2.7GHz) / X (7.5-9 GHz) / Ka (28-33 GHz) tres bandas simultáneas
- Banda ancha VGOS: 2-14 GHz.

Además, el Observatorio participa en proyectos de carácter nacional e internacional para la creación de nuevos tipos de receptores. En este plano, el Observatorio pretende desarrollar de manera autónoma sus backends de recepción, para lo cual la opción más factible es hacer uso de system on chip basado en FPGA, debido a su gran flexibilidad en cuanto a frecuencias de recepción se refiere.

El radiotelescopio de 40 m es uno de los nodos más importantes de la Red Europea de Interferometría de Muy Larga Base, una de las mayores instalaciones científicas del mundo, y es una estación del Servicio Internacional de VLBI para Geodesia y Astrometría.

Capítulo 3: Descripción del hardware e introducción al desarrollo de un sistema basado en Zynq

En el presente capítulo se expondrán en detalle las características del hardware empleado, así como los procedimientos a llevar a cabo de cara al desarrollo de un diseño basado en el citado hardware.

Para el desarrollo del diseño se ha optado por emplear la tarjeta de desarrollo Zedboard, un SoC basado en FPGA de la familia Zynq de Xilinx. Este tipo de dispositivo consta de dos bloques principales: El Processing System o PS y la Programmable Logic o PL. Ambos bloques se interconectan mediante enlaces AXI (Advanced eXtensible Interface). Mientras que la lógica programable provee de funcionalidades como lógica de alta velocidad u operaciones aritméticas, el sistema de procesamiento proporciona soporte para la ejecución de rutinas software y/o sistemas operativos. La Figura 11 muestra un modelo simplificado de la arquitectura.

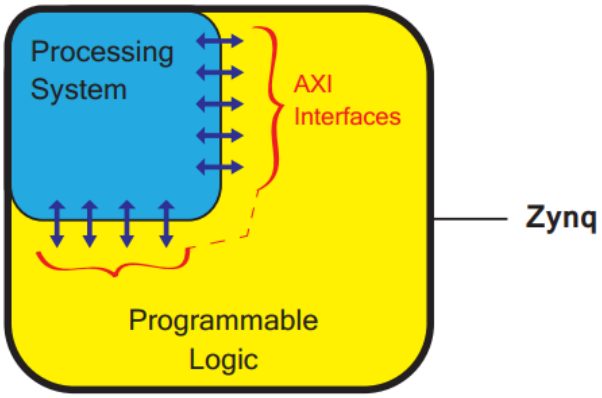


Figura 11: Modelo simplificado de la arquitectura Zynq [7].

Para la digitalización de los datos se hará uso de la tarjeta AD9467-FMC-250EBZ basada en el conversor analógico digital AD9467, que se conecta a la Zedboard por medio del puerto FMC de la misma.

Para llevar a cabo el desarrollo de un sistema basado en Zynq es preciso realizar un diseño dual, es decir, primero se debe abordar el desarrollo hardware y después el desarrollo software.

3.1 Tarjeta de desarrollo Zedboard

ZedBoard es una placa de evaluación y desarrollo basada en Xilinx Zynq-7000 All Programable SoC (AP SoC). Combina un sistema de procesamiento dual Cortex-A9 (PS) con 85.000 celdas de lógica programable (PL) de la serie 7 de Xilinx [6].

La placa proporciona una amplia gama de interfaces y periféricos tales como USB, Ethernet, HDMI e interfaces de tarjeta SD, que se pueden conectar a la FPGA mediante la interconexión programable que esta ofrece. Esto permite el desarrollo de infinidad de aplicaciones y sistemas integrados. Además, ZedBoard está diseñado para ser compatible con una amplia gama de herramientas de desarrollo de software, como Xilinx Vivado y PetaLinux, que pueden ser empleados para crear diseños personalizados para la placa.

3.1.1 Sistema de procesamiento (PS)

ZedBoard incluye un sistema de procesamiento (PS) que combina un procesador ARM Cortex-A9 de doble núcleo con una amplia gama de interfaces periféricas. El sistema de procesamiento está conformado principalmente por la APU (Application Processor Unit), la memoria caché, memoria ROM y RAM, interfaces para memoria externa, controlador DMA y diferentes periféricos e interfaces de entrada/salida.

El procesador ARM Cortex-A9 de doble núcleo de ZedBoard es un procesador de 32 bits que funciona a una frecuencia de reloj de 800 MHz. Es la piedra angular de la APU y se basa en la arquitectura ARMv7-A. Además, la sección PS del dispositivo Zynq incluye una caché de Nivel 1 (L1) y una caché de Nivel 2 (L2). Cada uno de los cores dispone de sus propias cachés independientes de primer nivel (L1) para código y datos, siendo cada una de ellas de 16KB. La caché L2 es de 512 KB, con uso compartido para el almacenamiento de datos e instrucciones. Incluye también una “On-Chip” RAM de 256 KB destinada a almacenamiento de datos. La Figura 12 muestra la arquitectura de la APU.

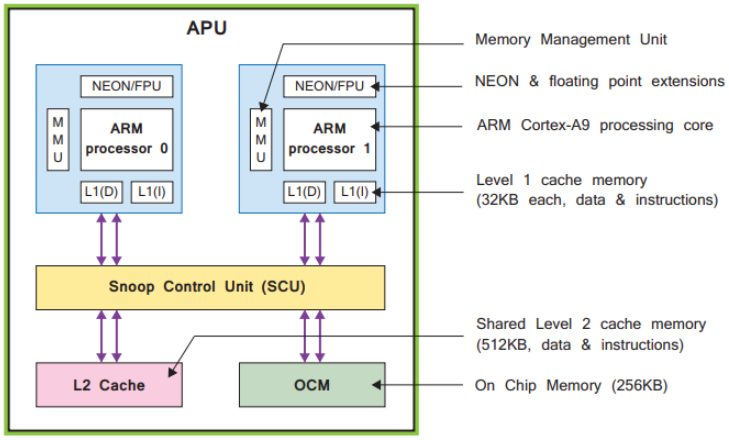


Figura 12: Diagrama de bloques de la APU.

El sistema de procesamiento también incluye una amplia gama de interfaces y periféricos que pueden ser empleados para establecer conexiones con dispositivos externos. Algunas de estas interfaces son USB, Gigabit Ethernet, HDMI, tarjeta SD y UART, entre otras. Además, se pueden conectar a la FPGA mediante la interconexión programable, lo que permite que la FPGA se comunique con los periféricos y realice las tareas de procesamiento necesarias. La citada interconexión se lleva a cabo con una arquitectura de buses denominada AXI, desarrollada por ARM y que será expuesta más adelante en el capítulo.

Además, el sistema de procesamiento incluye un controlador de memoria DDR3, que puede usarse para conectarse a una memoria externa. Esta memoria se puede utilizar para almacenar o acceder a datos para el procesador y la FPGA. Dispone también de un controlador DMA (Direct Memory Access) de ocho canales, encargado de agilizar los accesos a memoria.

3.1.2 Lógica Programable (PL)

La lógica programable en ZedBoard es proporcionada por el FPGA Xilinx Zynq-7000. Más en concreto, hace uso del dispositivo Z-7020, cuya FPGA Artix-7 consta de 85000 celdas lógicas, 53200 LUTs, 106400 biestables, 4,9 Mb de RAM y 220 DSP Slices. Estos se pueden programar para realizar funciones lógicas digitales, tales como puertas lógicas, contadores y máquinas de estado. La interconexión programable se compone de una gran cantidad de puntos de interconexión programables (PIP), que se pueden usar para conectar los bloques CLB y DSP, y así crear circuitos lógicos personalizados.

La lógica programable también incluye un módulo formado por dos ADCs de 12 bits (XADC), con el que se pueden digitalizar tanto entradas externas como parámetros internos de la placa referentes a medidas de temperatura o voltaje.

3.2 Periféricos en Zynq

La tarjeta de desarrollo Zedboard incluye un dispositivo Zynq ZC7Z020, basado en la estructura lógica Artix-7. Éste presenta los siguientes dispositivos:

- Lógica Programable (PL): Es usada para enlazar el dispositivo Zynq con interfaces, customizadas o existentes, y así dotarlo de la funcionalidad que estos aporten.

- Controlador de memoria DDR: Se trata de un controlador de memoria que aparece integrado en dispositivos Xilinx Zynq-7000 AP SoC (All Programmable System on Chip). Éste es el responsable de administrar el flujo de datos entre los núcleos del procesador y la memoria DDR externa, dando soporte a los estándares de memoria de tipo DDR2 y DDR3. La Zedboard tiene una sola ranura de memoria DDR3 y admite una capacidad máxima de memoria de 1GB. Dicha memoria está conectada al AP SoC por medio de una interfaz de 32 bit.

- Gigabit Ethernet: Es un controlador Ethernet de alta velocidad que permite que el dispositivo Zynq establezca comunicaciones con otros dispositivos en la red. Da soporte al estándar IEEE 802.3 para Gigabit Ethernet y puede manejar tasas de transferencia de 10, 100 y 1000 Mbps. También incluye características tales como corrección de errores y control de flujo, y puede ser configurado para operar en modo MII (Media Independent Interface) o GMII (Gigabit Media Independent Interface), que básicamente determinará el número de pines que se va a usar para establecer la conexión con el Ehernet PHY (Physical Layer).

- USB 2.0 (Universal Serial Bus): Se trata de un controlador de interfaz de altas prestaciones que habilita conexiones del dispositivo Zynq con dispositivos USB. El controlador USB 2.0 puede manejar tasas de transferencia de hasta 480 Mbps y da soporte a los modos de operación: USB servidor, dispositivo USB y OTG (On The Go), que es un híbrido de los dos anteriores.

- SD/SDIO (Secure Digital/Secure Digital Input Output): Un controlador de periféricos que habilita al dispositivo Zynq a establecer conexiones con tarjetas SD y SDIO. Permite manejar tasas de transferencia de hasta 48 Mbps.

- General Purpose Input/Output (GPIO): El dispositivo Zynq incluye cuatro bancos de 32 pines cada uno que pueden ser configurados como entrada o salida, y cuyo nivel de voltaje puede ser modificado individualmente. Los bancos 0 y 1 están asociados a la interfaz MIO, mientras que los bancos 2 y 3 están asociados a la interfaz EMIO.

- Universal Asynchronous Receiver-Transmitter (UART): El módulo UART del dispositivo Zynq es un canal de comunicación full-duplex asíncrono que puede operar con tasas de transferencia de hasta 3 Mbps. Incluye un transmisor y un receptor, cada uno de los cuales posee un búfer FIFO (First In First Out). Además, incluye varias características como control de flujo, paridad y longitud de palabra de datos variable.

- Inter-Integrated Circuit (I2C): Este controlador es un módulo que permite establecer comunicaciones entre el procesador y otros dispositivos haciendo uso del protocolo I2C.

- Serial Peripheral Interface (SPI): El controlador SPI del dispositivo Zynq puede ser configurado por software, y posee múltiples líneas SS (Slave Select) para permitir comunicaciones con múltiples dispositivos simultáneamente.

- Direct Memory Access (DMA): El controlador DMA permite que el dispositivo Zynq realice transferencias de alta velocidad entre los periféricos y la memoria sin hacer uso del procesador. Ofrece la posibilidad de operar en modo scatter/gather, en el cual los datos son transferidos a múltiples direcciones de memoria desde un periférico, o viceversa. Además, el controlador DMA puede ser configurado para hacer uso del protocolo AXI, que permite realizar transferencias de datos a alta velocidad conectándose con otros módulos con compatibilidad AXI del diseño.

3.3 Conexiones en Zedboard

Zedboard incluye una gran variedad de interfaces y conectores para enlazar dispositivos externos y periféricos, los cuales aparecen ilustrados en la Figura 13:

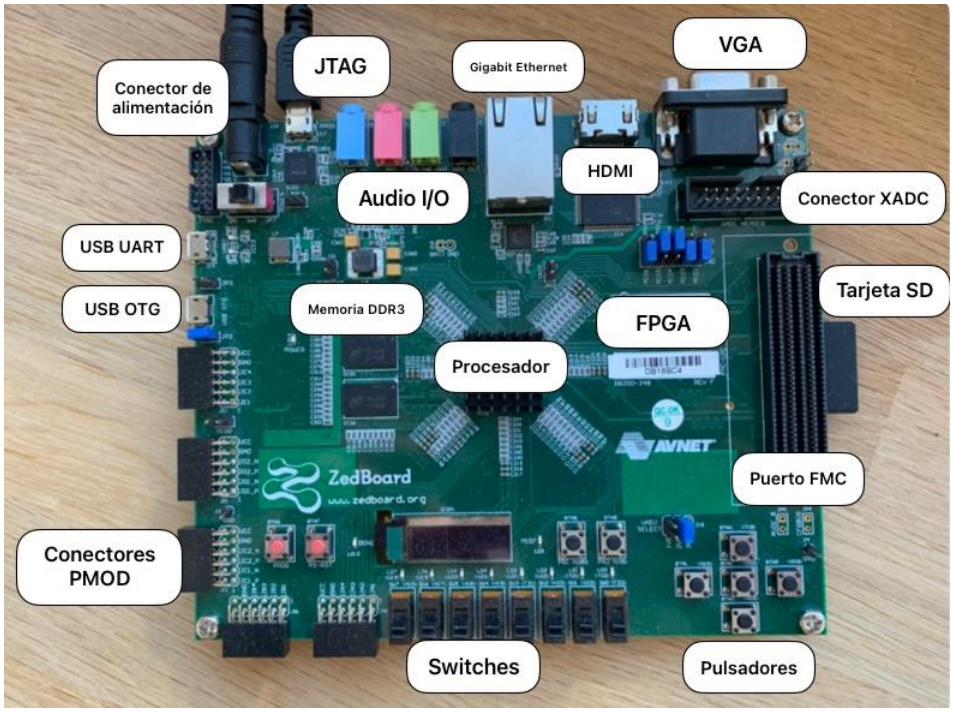


Figura 13: Partes de la Zedboard.

- USB-UART (Universal Asynchronous Receiver/Transmitter): El puente USB-UART convierte los datos enviados a través de USB al protocolo UART, que pueden ser leídos por medio del controlador UART de ZedBoard. Esto permite una fácil comunicación con la placa utilizando un programa de terminal serie estándar en un host.
- USB OTG (On The Go): Es una especificación que permite que un dispositivo USB actúe como host y como dispositivo. Su utilidad es que permite que la Zedboard se comunique con otros dispositivos USB sin necesidad de un ordenador host adicional. Utiliza un conector USB micro-AB que admite los modos de host USB y dispositivo USB. Cuando la ZedBoard está configurada como host puede comunicarse con otros dispositivos USB, como una memoria USB. Mientras que, cuando está configurada como USB, puede conectarse a un ordenador host y ser reconocida como un dispositivo convencional. La interfaz USB OTG en Zedboard está monitorizada por el controlador USB integrado, que forma parte del PS.

- Gigabit Ethernet: Permite que la tarjeta se conecte a una red y se comunique con otros dispositivos a altas velocidades. Está monitorizada por el controlador de acceso a medios (MAC) Ethernet. El MAC es el responsable de controlar el flujo de datos hacia y desde la interfaz Ethernet, así como de realizar la verificación y corrección de errores. La interfaz utiliza un conector RJ-45, que es compatible con cables Ethernet estándar. Además de la conexión física, Zedboard también tiene la capacidad de ejecutar la pila TCP/IP y otros protocolos de red, lo que le permite conectarse a varias redes y también realizar varias funciones de red como DHCP, servidor web, servidor FTP, etc.

- VGA: La interfaz VGA permite que ZedBoard emita datos de vídeo a un monitor o pantalla VGA. Está monitorizado por el controlador VGA integrado, que es responsable de codificar y decodificar los datos de vídeo que se envían y reciben desde la interfaz VGA. Esta utiliza un conector VGA estándar, que es compatible con la mayoría de los cables y pantallas VGA.

- HDMI (High-Definition Multimedia Interface): Es una interfaz digital destinada a la transmisión de vídeo y audio de alta definición. La interfaz HDMI en ZedBoard está controlada por el controlador HDMI integrado. Este es el responsable de codificar y decodificar los datos de vídeo y audio que se envían y reciben desde la interfaz. Además, utiliza un conector HDMI tipo A estándar, que es compatible con la mayoría de los cables y pantallas HDMI.

- Audio: La interfaz de audio permite que ZedBoard capture y transmita señales de audio a dispositivos externos, como micrófonos o auriculares. La interfaz de audio está controlada por el códec de audio incorporado, que es un chip separado en la placa. Este es el responsable de convertir los datos de audio digital en señales de audio analógicas, y viceversa. Por último, utiliza un conector de 3,5 mm para la salida de audio y una entrada de micrófono.

- SD Card: La interfaz de la tarjeta SD permite que ZedBoard lea y escriba datos en una tarjeta SD, que se puede usar para almacenamiento o como medio de arranque. Esta interfaz está monitorizada por el controlador SD incorporado, el cual es responsable de leer y escribir datos en la tarjeta SD, así como de administrar el flujo de datos y realizar la verificación y corrección de errores. Utiliza una ranura para tarjeta SD estándar, que es compatible con la mayoría de las tarjetas SD.

- JTAG (Joint Test Action Group): Es una interfaz estándar de depuración y programación que permite establecer una comunicación entre ZedBoard y un host. JTAG se utiliza para programar la memoria flash integrada, depurar el software que se ejecuta en la placa y acceder a los registros internos de la placa. Utiliza una interfaz de 4 o 5 cables para establecer la conexión, generalmente a través de un adaptador JTAG o un programador USB-JTAG. La interfaz JTAG en ZedBoard está monitorizada por el controlador JTAG integrado.

- PMOD (Peripheral Module): Es una interfaz estándar que permite la conexión de dispositivos periféricos a la placa a través de un conector de 6 o 12 pines. La interfaz PMOD está controlada por la lógica programable integrada y el procesador del SoC.
- Conectores de alimentación: La placa incluye conectores de alimentación que proporcionan la energía necesaria para que funcione la placa. Estos conectores permiten la conexión de fuentes de alimentación externas a la ZedBoard e incluyen un conector de alimentación de DC, que permite una conexión de una fuente externa de 12 V, y un interruptor de alimentación, para encender y apagar el dispositivo.
- Conector XADC: Permite la conexión de señales analógicas externas. El conector XADC en ZedBoard brinda acceso al módulo XADC en el chip, que es un conversor analógico digital de 1 MSPS y 12 bits. El XADC se puede usar para monitorizar varios voltajes integrados, como el voltaje de alimentación y la temperatura del SoC.
- Conector FMC (FPGA Mezzanine Card): Interfaz que permite al usuario conectar una gran variedad de tarjetas secundarias FMC a Zedboard. Estas pueden proporcionar funcionalidades adicionales a la placa. Pueden ser conversores analógico digitales, conversores digital analógicos y otros dispositivos periféricos. El conector FMC cumple con el estándar VITA 57.1, que define las especificaciones mecánicas y eléctricas de las tarjetas FMC. Esto permite utilizar una amplia gama de tarjetas FMC de terceros con la placa.

3.4 Protocolo AXI

El protocolo AXI define un conjunto de señales que se utilizan para la comunicación entre los componentes del diseño de un SoC. Estas señales se pueden agrupar en tres categorías principales:

- Señales de transferencia de datos: Estas señales se utilizan para la transferencia de datos entre los componentes. El protocolo AXI define canales separados para operaciones de lectura y escritura, lo que permite la transferencia simultánea de datos. Las señales de transferencia de datos incluyen:

    - AWADDR: Señal con la dirección para operaciones de escritura.
    - WDATA: Señal de datos para operaciones de escritura.
    - BRESP: Señal de respuesta para operaciones de lectura.
    - ARADDR: Señal que porta la dirección para operaciones de lectura.
    - RDATA: Señal de datos para operaciones de lectura.

- Señales de control: estas señales se utilizan para controlar el flujo de transferencia de datos. Las señales de control incluyen:

    - AWVALID: Indica que hay una dirección de escritura y datos válidos disponibles.
    - AWREADY: Indica que el componente está listo para aceptar una dirección de escritura y datos.
    - WVALID: Indica que hay datos de escritura válidos disponibles.
    - WREADY: Indica que el componente está listo para aceptar datos de escritura.
    - BVALID: Indica que hay disponible una respuesta de lectura válida.
    - BREADY: Indica que el componente está listo para aceptar una respuesta de lectura.
    - ARVALID: Indica que hay disponible una dirección de lectura válida.

- ARREADY: Indica que el componente está listo para aceptar una dirección de lectura.

- Señales de gestión: Estas señales se utilizan para gestionar la comunicación entre componentes. Las señales de gestión incluyen:

  - ACK: Señal de acuse de recibo.
  - INTERRUPT: Señal de interrupción.
  - RETRY: Señal de reintento.

En cuanto a la teoría de operación, el componente maestro inicia una transferencia activando la señal de control adecuada (AWVALID o ARVALID) y proporcionando la información necesaria (dirección y datos) en las líneas de señal correspondientes. El componente esclavo responde activando la señal de control adecuada (AWREADY o ARREADY) para indicar que está listo para aceptar la transferencia. Una vez que se completa la transferencia, el componente esclavo afirma la señal de respuesta adecuada (BRESP o RDATA) para indicar el estado de la transferencia. Luego, el componente maestro acusa recibo de la finalización de la transferencia anulando la señal de control apropiada.

3.4.1 AXI4-Stream

AXI4-Stream es un protocolo diseñado para transportar datos unidireccionalmente de manera arbitraria. El protocolo AXI4-Stream puede utilizar hasta 11 señales, algunas de ellas opcionales. Para este proyecto se utilizarán únicamente 6 señales, una de ellas opcional.

- ACLK: La señal de reloj global. Todas las señales se muestrean en el flanco ascendente de ACLK.
- ARESETn: La señal de reinicio global. ARESETn es activa a nivel bajo.
- TVALID: Procedente del lado maestro. Indica que el maestro está generando una transferencia válida.
- TREADY: Procedente del esclavo. Indica que el esclavo puede aceptar una transferencia en el ciclo actual. Una transferencia solo tiene lugar cuando tanto TVALID como TREADY se afirman.

- TDATA: Procedente del maestro. Es la carga útil principal que se utiliza para proporcionar los datos que pasan a través de la interfaz.
- TLAST: Procedente del maestro. Se utiliza para dividir los datos transferidos en paquetes. Cuando está activo, indica el final de un paquete.

En un AXI4-Stream, los bits contenidos en la señal TDATA se transfieren cada ciclo de reloj. La transferencia se inicia cuando el emisor envía la señal TVALID y el receptor responde enviando la señal TREADY. En este punto, el emisor comenzará a enviar TDATA y TLAST. TLAST señala el último byte del paquete, por lo que el receptor seguirá aceptando el contenido de TDATA hasta que se afirme la señal TLAST [8].

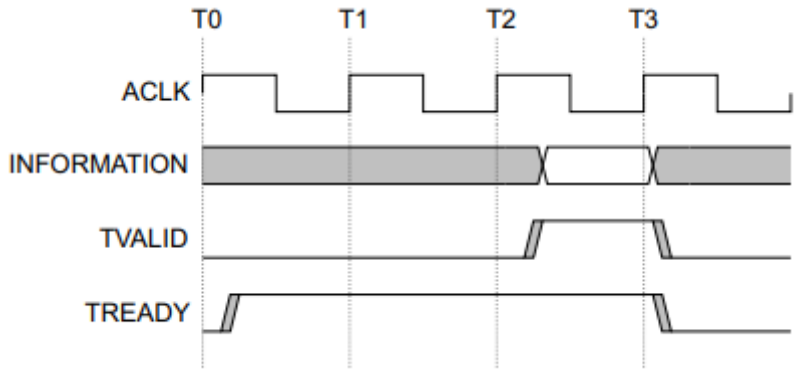


Figura 14: Ejemplo de comunicación AXI-Stream [8].

3.5 AD9467-FMC-250EBZ

AD9467-FMC-250EBZ es una tarjeta de evaluación basada en el componente AD9467 y desarrollada por ANALOG DEVICES. Dicha tarjeta está diseñada para funcionar como un periférico externo de la Zedboard, estableciendo conexión con ella por medio del puerto FMC del que dispone. Además, incorpora un distribuidor de reloj, el AD9517, que permite generar un reloj de frecuencia arbitraria sin necesidad de una fuente externa.

3.5.1 AD9467

El AD9467 es un conversor analógico digital (ADC) de 16 bits. Está optimizado para un alto rendimiento sobre anchos de banda amplios. Funciona con una tasa de conversión de 250 MSPS y está diseñado para receptores inalámbricos, instrumentación y equipos de prueba que requieren un alto rango dinámico.

3.5.1.1 Teoría de operación

La arquitectura del AD9467 está conformada por un ADC segmentado con buffer de entrada que consta de una primera etapa de 3 bits, una segunda de 4 bits, seguida de cuatro etapas de 3 bits y una etapa flash final de 3 bits. Un flash ADC es un tipo de conversor que hace uso de una arquitectura dividida en varios niveles, cada uno de los cuales compara la señal de entrada con unos voltajes de referencia. Para más información se puede consultar la referencia [13], en la que el fabricante de la placa explica esta cuestión más a fondo.

Cada una de las etapas mencionadas proporciona un grado de superposición suficiente como para corregir los posibles errores de flash que puedan suceder en etapas precedentes. Además, el buffer está optimizado para alta linealidad, bajo ruido y baja potencia. Las salidas cuantificadas de cada etapa se combinan en una final, resultando una palabra de 16 bits en la lógica de corrección digital. La Figura 15 muestra un diagrama de bloques con la arquitectura del componente en cuestión.

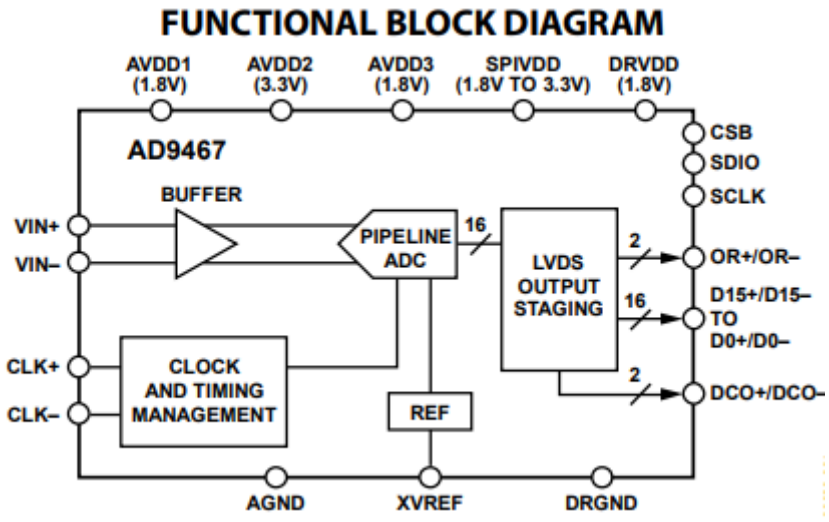


Figura 15: Arquitectura del AD9467 [16].

Los muestreos se llevan a cabo en el flanco ascendente del reloj. Cada etapa del buffer de entrada, excluyendo la última, consta de un flash ADC de baja resolución conectado a un DAC de capacidades conmutadas. La última etapa consiste simplemente en un flash ADC. El bloque de preparación de salida alinea los datos, corrige errores y pasa los datos a los buffers de salida.

El componente hace uso del modo source-synchronous para la transmisión de las señales de datos por el puerto de salida, que consiste en acompañar dichas señales con un reloj de salida sincronizado (Data Clock Output). En los flancos de subida del reloj se transmitirá un dato y en los flancos de bajada el otro. La Figura 16 esclarece la teoría de operación ilustrando un diagrama temporal de las señales del componente.

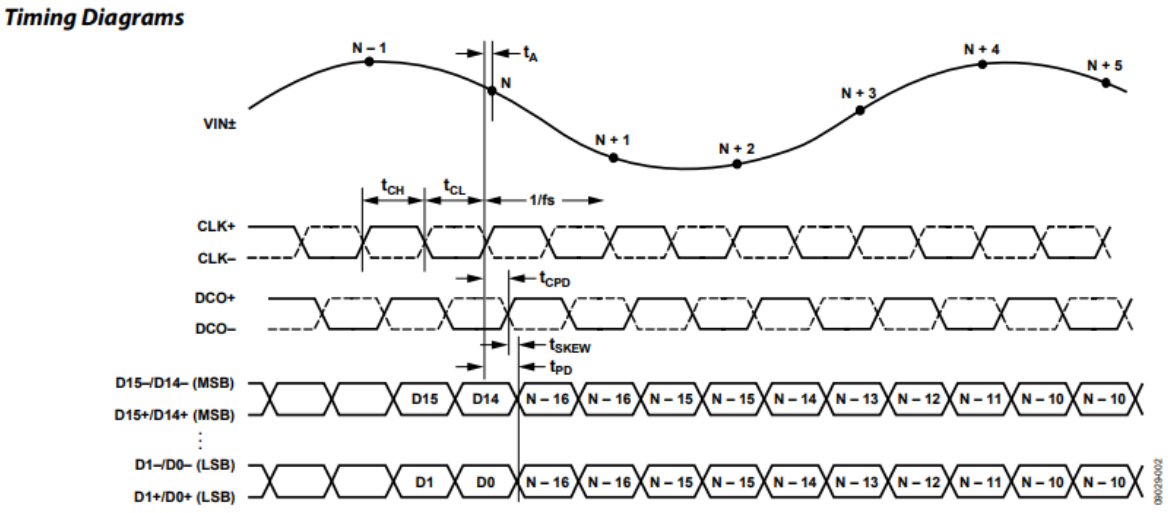


Figura 16: Diagrama temporal de las señales del AD9467 [16].

3.5.1.2 Interfaz de puerto serie (SPI)

La interfaz de puerto serie del AD9467 brinda al usuario la posibilidad de llevar a cabo acciones de configuración sobre el dispositivo, escribiendo y leyendo sobre las direcciones de memoria de un mapa de registros. Dicha memoria se organiza en bytes, que a su vez se pueden dividir en diferentes campos. Se pueden encontrar más especificaciones sobre la funcionalidad de cada byte de los registros en la nota de aplicación AN-877 [19] que proporciona ANALOG DEVICES.

Hay tres pines que definen el funcionamiento de la interfaz SPI: SCLK, SDIO y CSB. El pin SCLK es el reloj de sincronismo de la interfaz y se utiliza para sincronizar la lectura y escribir los datos enviados al ADC. El pin SDIO es un pin bidireccional que permite enviar y leer datos desde los registros internos del mapa de memoria del ADC. El pin CSB es activo a nivel bajo y activa o desactiva los ciclos de lectura y escritura.

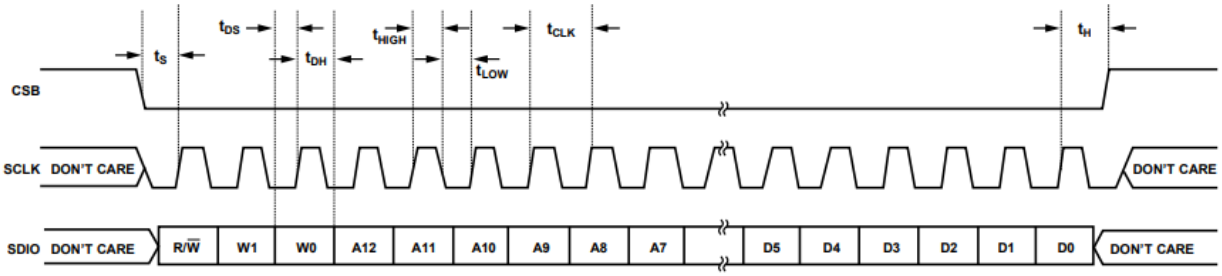


Figura 17: Comunicación SPI [16].

En la Figura 17 se puede observar un ejemplo de transmisión con el protocolo SPI. La estructura del mensaje a enviar en el puerto SDIO consta de un primer bit que determina si el proceso a realizar es de escritura o lectura, seguido de dos bits con los que se especifica el número de bytes a leer/escribir (uno, dos, tres o indefinidos), a continuación se encuentran trece bits con los que se debe indicar la dirección de memoria del banco de registros a la que se quiere acceder, y por último, los datos con un tamaño variable de uno, dos o tres bytes dependiendo de la selección del usuario.

3.5.2 AD9517

El AD9517 es un generador de reloj con un VCO (oscilador controlado por tensión) integrado de 1,6 GHz, aunque también se puede hacer uso de un VCO externo de hasta 2,4 GHz. Dispone de cuatro salidas LVPECL (en dos pares) y cuatro salidas LVDS (en dos pares). Cada salida LVDS se puede reconfigurar como CMOS. Las salidas LVPECL funcionan a 1,6 GHz, las LVDS a 800 MHz y las CMOS a 250 MHz. La Figura 18 muestra un diagrama de bloques con la arquitectura del componente en cuestión.

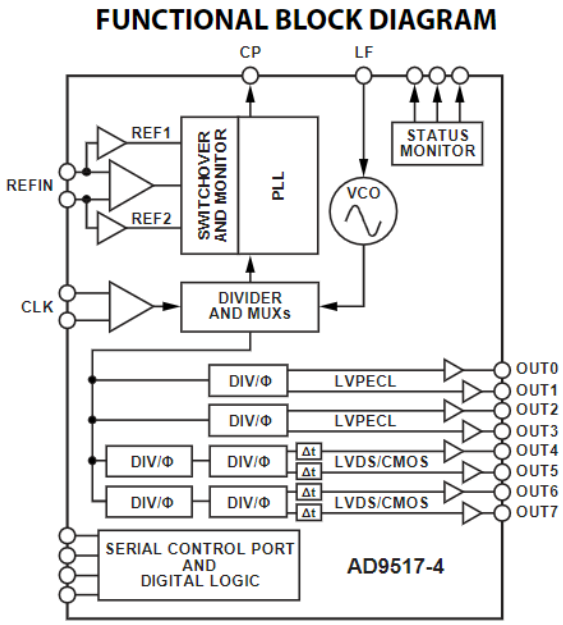


Figura 18: Diagrama de bloques del AD9517 [17].

El modo de funcionamiento del componente es simple, se puede seleccionar como frecuencia base el VCO interno o uno externo. Una vez seleccionado, se indica la división que se desea realizar a la frecuencia base y la salida que se va a emplear. El rango de división para las salidas LVPECL es de 1 a 32. Las salidas LVDS/CMOS permiten un rango de divisiones hasta un máximo de 1024. Además, el integrado incluye una interfaz de puerto serie para realizar configuraciones internas, idéntico al expuesto para el AD9467.

La tarjeta FMC incorpora un AD9517 integrado. La entrada de reloj externa está conectada a un oscilador de 250 MHz, y la salida a utilizar es la tres, que es de tipo LVPECL. El conexionado se puede observar en la Figura 19.

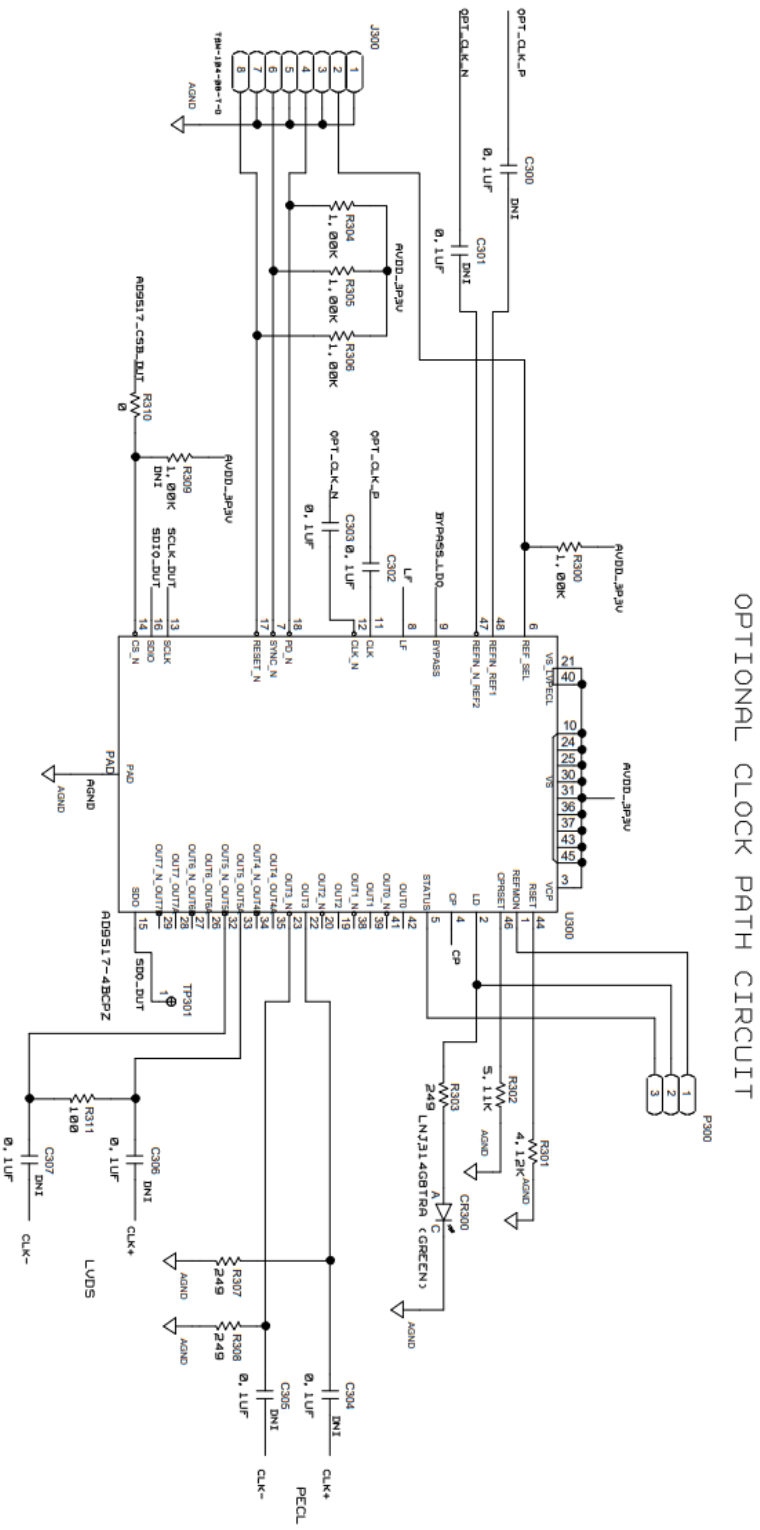


Figura 19: Conexionado del AD9517 en la tarjeta AD9467-FMC-250EBZ [18].

3.6 Desarrollo del diseño hardware con Xilinx Vivado

Para llevar a cabo el diseño hardware se ha utilizado la herramienta de diseño Vivado, en concreto la versión 2017.4. Esta dispone de una interfaz gráfica y asistentes que facilitan mucho su uso, pero puede emplearse vía comandos tcl desde la terminal de la aplicación o desde el sistema operativo.

Si se hace uso de la interfaz gráfica, a medida que se va avanzando en el desarrollo del proyecto se van mostrando en la consola los comandos tcl necesarios para realizar cada acción. Es interesante ir guardando dichos comandos en un documento de extensión .tcl para agilizar el proceso de creación del proyecto en otro terminal y poder guardar el proyecto ocupando muy poca memoria.

Tras crear un nuevo proyecto y especificar la tarjeta de desarrollo a utilizar, es necesario realizar la arquitectura hardware del sistema. Dicha arquitectura se puede desarrollar creando un diagrama de bloques con el asistente que proporciona Vivado. El diagrama está conformado por IP (Intellectual Property) Cores, que son bloques de circuitos digitales prediseñados que se pueden usar para agregar una funcionalidad específica a un diseño de FPGA.

Vivado incluye una amplia librería de IP Cores que abordan muchas de las funcionalidades básicas de la placa. La piedra angular de todo diagrama de bloques es el IP Core referente al sistema de procesamiento de la tarjeta a usar.

En nuestro caso, al tratarse de la Zedboard, el IP Core es el “ZYNQ7 Processing System” y permite establecer la comunicación entre el PS y la PL de la placa por medio de interconexiones AXI, las cuales son claves para el desarrollo óptimo del sistema. Además, el bloque presenta un gran número de parámetros configurables como relojes del sistema, interrupciones, timers y buses AXI, entre otros.

Los bloques a usar en un diagrama no están limitados a los que Xilinx ofrece en sus librerías por defecto, si no que la propia herramienta brinda la posibilidad de crear IP Cores a medida con la opción “Create and package IP”. En este punto, surge un pequeño problema, ya que, para garantizar el correcto funcionamiento del IP a medida, una vez implementado en la arquitectura, es preciso realizar pruebas de simulación. Si hacemos uso del bloque “ZYNQ7 Processing System” estas simulaciones se deben realizar con el hardware volcado en placa, lo cual no es nada práctico, ya que ante cualquier fallo será preciso sintetizar e implementar el hardware de nuevo para modificar el código de nuestro IP. Por ello, la solución más razonable es hacer uso de un modelo BFM (Bus Functional Model) del sistema de procesamiento. Esto es una interfaz software que se comporta de la misma manera que el PS de la placa, permitiendo la simulación del software sin necesidad de volcar el hardware en placa.

Los bloques a usar en nuestra arquitectura deben estar correctamente conectados entre sí, para ello Vivado dispone de un asistente que ayuda a realizar esta tarea y verifica si las conexiones realizadas por el usuario son correctas. Además, para poder llevar a cabo la implementación del hardware, es preciso crear un fichero de restricciones que asigna los puertos del diseño con los pines físicos de la placa. Una vez acabados los pasos anteriores, se procede a realizar la síntesis e implementación del sistema, si se dan estos dos procesos sin fallos se procede a la generación del bitstream.

El bitstream es un archivo que contiene información sobre la configuración lógica de la FPGA, incluyendo qué puertas lógicas están conectadas, qué patrones de señal se utilizan para las entradas y salidas, y cualquier otra información necesaria para configurar el dispositivo para realizar una tarea específica.

3.7 Desarrollo del diseño software con Xilinx SDK

Una vez diseñado el hardware del sistema, se pasa a desarrollar el software, que se lleva a cabo con la herramienta Xilinx SDK (Software Development Kit). Esta provee de un entorno para crear, depurar y desplegar software en dispositivos Xilinx, así como bibliotecas y proyectos. SDK está basada en el modelo de código abierto Eclipse.

Para llevar a cabo el diseño software hay que importar a SDK el hardware generado con Vivado. Una vez exportado, todos los archivos relevantes del hardware se encuentran en Xilinx SDK, por lo que se puede comenzar a desarrollar el software. El primer paso es generar el BSP (Board Support Package), un conjunto de librerías y drivers de bajo nivel que dan soporte al código principal ofreciendo servicios software relacionados con el procesador y periféricos del sistema de procesamiento. Además, es posible añadir bibliotecas que brindan servicios tales como soporte de red TCP/IP.

A continuación, se deberá elaborar una aplicación en lenguaje C o C++. Para poder volcarla en placa es preciso compilar y generar el ejecutable que se cargará en la misma. En primer lugar, se carga en placa el archivo bitstream, cuya finalización está marcada por la activación del LED “Done”. Después se vuelca el software en placa cargando el archivo .elf generado durante la compilación. Este tipo de ejecución software se denomina baremetal, ya que se ejecuta sin sistema operativo.

Capítulo 4: Desarrollo del backend

Para el desarrollo del backend se va a dividir el proyecto en tres fases: la creación de la interfaz para establecer la comunicación con la placa, el desarrollo de la transformada de Fourier y el acceso a memoria por medio del bloque DMA.

4.1 Creación de la interfaz axi_adc_controller para el ADC

Para establecer la conexión entre la Zedboard y la tarjeta AD9467-FMC-250EBZ es preciso crear un bloque IP a medida. Dicho bloque debe constar de dos partes diferenciadas: una interfaz SPI que permita escribir y leer datos para hacer modificaciones en los registros del ADC, y una interfaz de capa física, que se encarga de recibir los datos del ADC por un puerto de entrada y transmitirlos por un puerto de salida.

4.1.1 Interfaz SPI

Su función es dar soporte para llevar a cabo los procesos de lectura y escritura a través de la interfaz del puerto serie del AD9467. Como se ha expuesto antes, las principales señales del protocolo SPI son CSB, SDIO y SCLK, y estas son las que la interfaz tiene que encargarse de generar. A continuación, se exponen la teoría de operación del código VHDL utilizado en el bloque IP a medida, las librerías software utilizadas y las pruebas que verifican el correcto funcionamiento de la interfaz.

4.1.1.1 Teoría de operación

La teoría de operación del código VHDL diseñado para el desarrollo de la interfaz SPI se fundamenta en la existencia de dos flags activos a nivel alto (half_bit_end y last_half_bit) y la señal byte_cnt.

- Half_bit_end: Se activa cuando la cuenta descendente sobre la señal half_bit_time llega su fin. La señal half_bit_time tiene cargado el número de ciclos de reloj del sistema que tienen que darse para que se complete medio ciclo de reloj. Nos referiremos a los medios períodos de SCLK como medios bits, ya que los bits a transmitir/recibir irán sincronizados con la señal de SCLK generada.

- Last_half_bit: Se activa cuando la cuenta descendente sobre la señal half_bit_cnt llega a su fin. El valor máximo de half_bit_count es 15 y decrece cada vez que se detecta un nivel alto en el flag half_bit_end, por lo que cuenta 16 medios bits antes de activar el flag de last_half_bit, o lo que es lo mismo, 8 bits que suponen 1 byte. Por tanto, last_half_bit indica el último bit de un byte.

- Byte_cnt: Se trata de una señal en la que queda registrada el número de bytes transmitidos.

Las recepciones/transmisiones son monitorizadas por una máquina de estados con cuatro estados: idle, cuando el sistema está en reposo; send_cmd, que dilucida si el bit R/W de la trama de información está indicando lectura o escritura; send_data, para enviar datos si se ha seleccionado el modo escritura; y rcv_data para recibir datos si se ha seleccionado el modo lectura.

Para abordar las transmisiones/recepciones con eficacia se ha optado por dejar los bits W0 y W1 siempre a nivel bajo, de tal manera que siempre se van a transmitir dos bytes de información de control y a transmitir/recibir un byte de datos.

En cuanto a las transmisiones, el modo de operación consiste en dividir los datos en tres palabras de un byte cada una, cada una de las cuales se irá cargando en un registro de transmisión de desplazamiento consecutivamente cada 8 bits transmitidos. Una vez la palabra de un byte esté cargada en el registro, se irá desplazando su contenido una posición a la derecha en cada ciclo, tomando el bit más significativo. Por consiguiente, pasados ocho ciclos se habrán tomado y pasado al puerto bidireccional SDIO los ocho bits de información.

Para abordar las recepciones se lleva a cabo el proceso a la inversa, ya que SDIO actúa como entrada y se van cargando los datos individualmente en el bit menos significativo de la señal del registro de transmisión de desplazamiento.

El protocolo se comparte para las comunicaciones con el ADC y el generador de reloj en función del nivel lógico de una entrada de habilitación de la entidad.

4.1.1.2 Librerías software

Para hacer funcionar la interfaz SPI, es preciso diseñar un código software capaz de llevar a cabo la inicialización y configuración de los componentes para la realización de los procesos de escritura/lectura, por ejemplo.

Se han diseñado librerías referentes al AD9467 y al AD9517:

4.1.1.2.1 AD9467

Lleva a cabo la implementación del driver correspondiente al AD9467 (ADC). Se procede a explicar las funciones que contiene esta librería:

- ad9467_setup: Se trata de la función de inicialización del componente, configura el modo test y el modo salida a los valores por defecto. En caso de ejecución con éxito la función devuelve un cero, si no, un valor negativo.

- ad9467_transfer: Inicia una transferencia y espera a que se finalice la operación para acabar. En caso de éxito devuelve un cero, si no, un valor negativo.

- AD9467_spi_write: Escribe datos en un registro para realizar una configuración. Los datos a escribir y la dirección del registro donde van a ser escritos son los argumentos de entrada de la función. En caso de éxito devuelve un cero, si no, un valor negativo.

- AD9467_spi_clear: Limpia los datos previamente escritos en un registro. La máscara de los bits a limpiar y la dirección de su registro correspondiente son los argumentos de entrada de la función. En caso de éxito devuelve un cero, si no, un valor negativo.

- AD9467_spi_read: Función encargada de leer datos en un registro para revisar el estado del mismo. Los datos a leer y la dirección del registro donde van a ser leídos son los argumentos de entrada de la función. En caso de éxito devuelve un cero, si no, un valor negativo.

- set_H_adc_idelay_inc: Se encarga de establecer a nivel alto la señal INC del componente IDELAYE2. Cada vez que está a nivel alto, dicha señal incrementa en 78 ps los retrasos que van a sufrir las señales de datos de salida.

- set_L_adc_idelay_inc: Establece a nivel bajo la señal INC del componente IDELAYE2. Cada vez que está a nivel bajo, dicha señal decrementa en 78 ps los retardos que van a sufrir las señales de datos de salida.

- set_H_adc_idelay_ce: Activa la señal de habilitación del componente IDELAYE2, para que se lleven a cabo los incrementos/decrementos.

- set_H_adc_idelay_rst: Función encargada de activar la señal de reset del componente IDELAYE2.

- rd_adc_calib_sample: Lee el registro de calibración del ADC, para dar soporte durante las calibraciones del componente.

4.1.1.2.2 AD9517

Lleva a cabo la implementación del driver correspondiente al AD517 (generador de reloj). Se procede a explicar las funciones que contiene esta librería a continuación:

- ad9517_setup: Función de inicialización del componente. Resetea el puerto serie y lo configura para instrucciones largas, limpia el bit de soft reset, selecciona el modo referencia del PLL, selecciona la entrada al reloj y resetea con los valores por defecto todas las salidas del componente.

- ad9517_write: Escribe datos en un registro del componente para realizar una configuración. Los datos a escribir y la dirección del registro donde van a ser escritos son los argumentos de entrada de la función. En caso de éxito devuelve un cero, si no un valor negativo.

- ad9517_read: Lee datos en un registro para revisar el estado del mismo. Los datos a leer y la dirección del registro donde van a ser leídos son los argumentos de entrada de la función. En caso de éxito devuelve un cero, si no un valor negativo.

- ad9517_update: Transfiere el contenido de los buffers de los registros dentro de los registros activos.

- dividers_checker: Comprueba si el número, introducido como argumento de entrada, puede ser descompuesto en un producto de dos números, cada uno de los cuales sea más pequeño o igual a 32.

- ad9517_frequency: Función encargada de asignar la frecuencia deseada por el usuario, e introducida como argumento de entrada, en el canal deseado, también disponible como argumento de entrada. La función selecciona el reloj externo como fuente de división y, mediante una serie de operaciones de aproximación, calcula los valores a asignar en las variables divider_low_cycles (número de ciclos de reloj que permanece la señal de salida a nivel bajo) y divider_high_cycles (número de ciclos de reloj que permanece la señal de salida a nivel alto) para que la frecuencia de salida sea la deseada por el usuario, o lo más cercana al valor solicitado posible.

- ad9517_power_mode: Establece el modo de operación, especificado en un argumento de entrada para el canal, especificado también, en un argumento de entrada.

4.1.1.3 Pruebas de funcionamiento

Para comprobar el correcto funcionamiento de la interfaz SPI se ha creado una aplicación software que accede a algunos registros del ADC. Para chequear las lecturas se ha optado por el registro que porta la identificación del dispositivo (dirección 0x00), ya que tiene un valor fijo de 0x50. En el caso de las escrituras, el registro seleccionado es el de offset (dirección 0x10), ya que sus ocho bits son modificables. La prueba de escritura consiste en escribir el valor 0XF0 en el registro de offset, leer el registro para comprobar que la escritura se ha llevado a cabo con éxito, y volver a escribir el valor por defecto del registro (0x00).

Para visualizar los resultados, se ha hecho uso del Integrated Logic Analyzer (ILA). El ILA es una herramienta de depuración que permite al usuario visualizar las señales internas de un diseño. En este caso las principales señales que se han seleccionado son:

- La señal cmd_addr: Indica la dirección a la que se está accediendo.
- La señal cmd_rqt: Muestra cuando el usuario quiere realizar un proceso de escritura/lectura.
- La señal cmd_rnw: Indica si el proceso es de escritura o de lectura.
- La señal tx_data, que porta los datos a leer/escribir.
- La señal tx_shift_reg: Hace referencia al registro de transmisión de desplazamiento.
- La señal sdin_q: Porta los datos que se van a ir cargando en el registro de desplazamiento de transmisión en cada ciclo.

En la Figuras 20 y 21 se puede apreciar el correcto funcionamiento de las comunicaciones SPI, tanto con el componente AD9467, como con el AD9517.

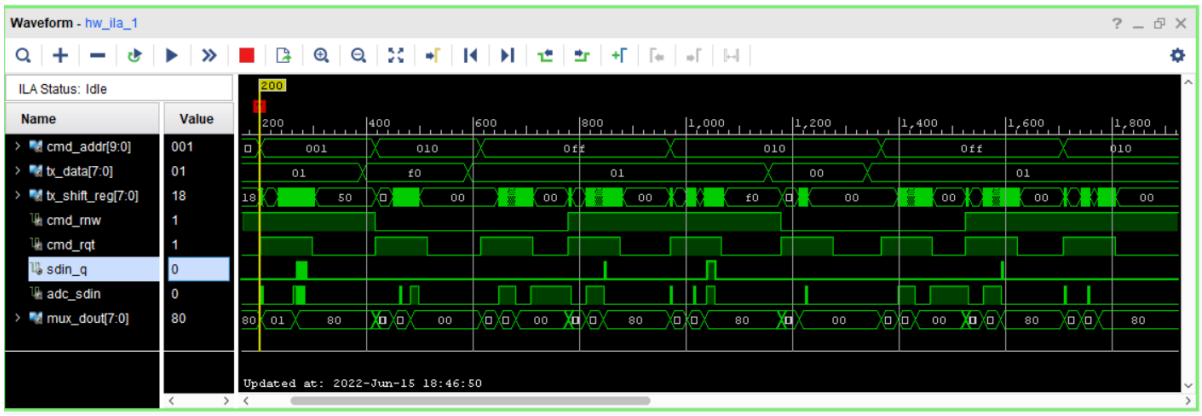


Figura 20: Resultado del test de la comunicación SPI con el componente AD9467.

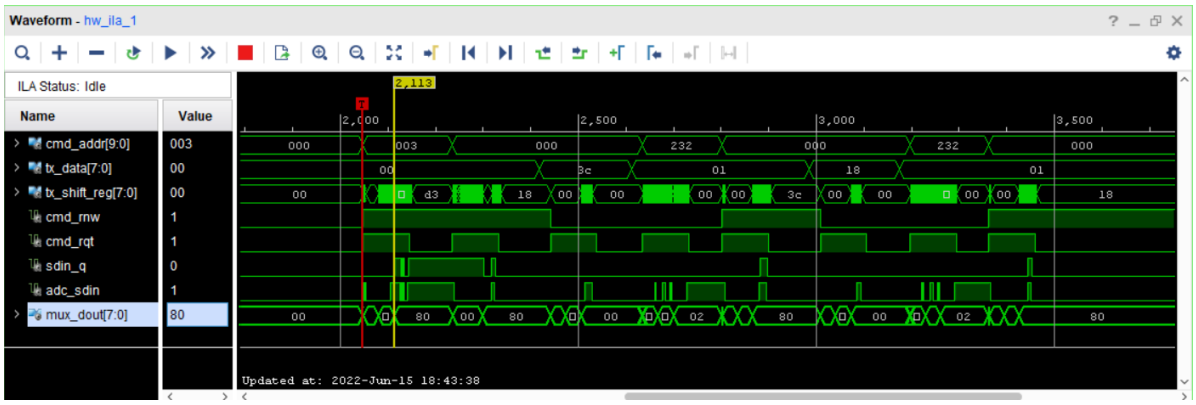


Figura 21: Resultado del test de la comunicación SPI con el componente AD517.

4.1.2 Interfaz física

Además de desarrollar una interfaz para realizar las comunicaciones SPI, es preciso implementar también una interfaz capaz de enlazar con la placa los datos de salida del ADC, los cuales van sincronizados con un reloj. Por tanto, dicha interfaz tendrá como objetivo principal la recepción y el procesamiento de los datos provenientes del ADC, que están en modo diferencial. Además, para garantizar una adquisición exitosa de los datos es preciso realizar una calibración de los mismos.

4.1.2.1 Teoría de operación

El código VHDL utilizado para realizar la interfaz se basa en una arquitectura hardware que puede ser dividida en tres bloques:

1. Bloque de retardo de señales de datos:

Los datos son retardados para llevar a cabo el proceso de calibración de los mismos, el cual se expondrá más adelante. La Figura 22 ilustra la arquitectura del bloque.

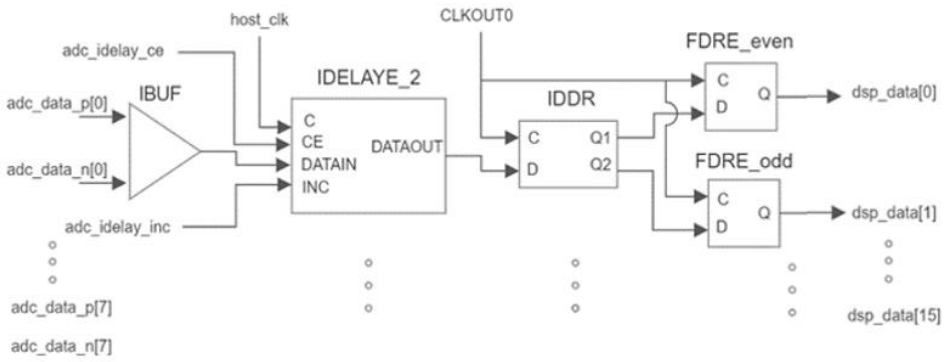
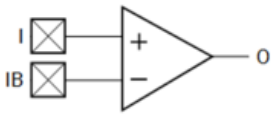


Figura 22: Arquitectura hardware correspondiente al bloque de retardo.

El bloque, básicamente, procesa los datos de entrada para su correcta recepción en el elemento de retardo y, una vez retardados, se sincronizan con el reloj del sistema haciendo uso del componente IDDR. La teoría de operación es la siguiente: los datos en modo diferencial entran a un búfer de entrada diferencial que los pasa a formato no diferencial para poder introducirlos en el bloque IDELAYE2, que es el elemento de retardo. Una vez retardados, los datos saldrán hacia un registro de entrada para datos en formato DDR, para luego ser pasados por un biestable tipo D. A continuación, se expondrán en detalle los componentes usados:

- IBUFDS: Se trata de un buffer de entrada diferencial y salida no diferencial. En la Figura 23 se puede observar un bloque que representa el componente y la tabla de verdad del mismo [9].



I	IB	O
0	0	No Change
0	1	0
1	0	1
1	1	No Change

Figura 23: Representación del componente y tabla de verdad [9].

- IDELAYE2: Es un elemento de retardo que consta de 31 retardos con resolución de retardo calibrada. La resolución de los retardos se varía seleccionando un reloj de referencia IDELAYCTRL, del rango especificado en la hoja de datos de FPGA de la serie 7. Dicho reloj será expuesto más adelante. En la Figura 24 se puede observar el conexionado que se ha aplicado al componente.

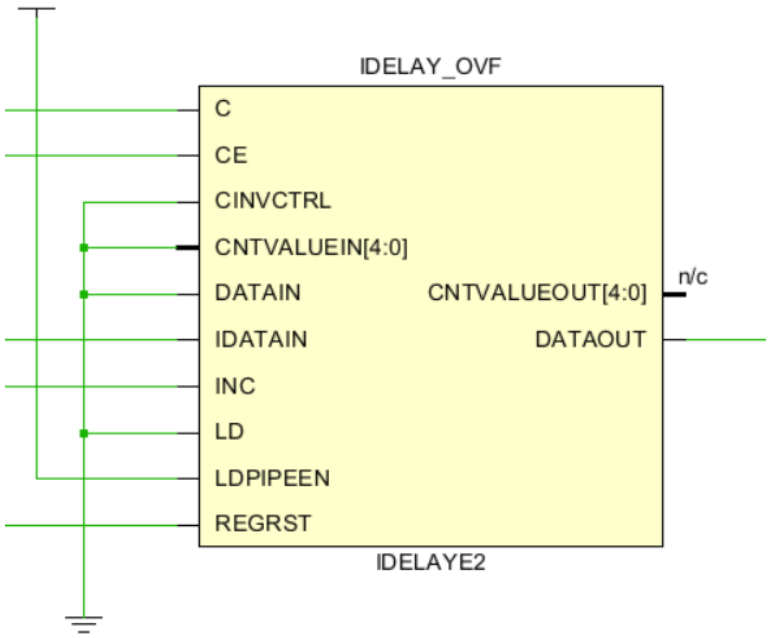


Figura 24: Conexionado del componente IDELAYE2.

A continuación, se procederá a explicar algunos de los puertos más importantes de la entidad [9]:

- C: La entrada de reloj con la que todas las entradas de control están sincronizadas. Se debe conectar un reloj a este puerto cuando IDELAYE2 está configurado en modo "VARIABLE", "VAR_LOAD" o "VAR_LOAD_PIPE". En nuestro caso está configurado en modo VARIABLE.

- CE: Entrada de habilitación activa a nivel alto para la función de incremento/decremento.

- CINVCTRL: Se usa para cambiar dinámicamente la polaridad del pin C. En nuestro caso aparece conectado a masa porque no es preciso su uso.

- CNTVALUEIN: Valor del contador de la lógica de la FPGA para el valor de entrada de los incrementos recargables dinámicamente.

- DATAIN: La entrada de datos es conducida por la lógica FPGA, que proporciona una línea de retardo lógica accesible. Los datos regresan a la lógica FPGA a través del puerto DATAOUT con un retraso establecido por IDELAY_VALUE.

- IDATAIN: La entrada IDATAIN está controlada por su E/S asociada. Los datos pueden enviarse a un bloque de registro de entrada o ISERDESE2, directamente a la lógica FPGA, o a ambos a través del puerto DATAOUT con un retraso establecido por IDELAY_VALUE.

- INC: Selecciona si los retardos incrementarán o disminuirán. Los retardos incrementarán cuando esté a nivel alto, y disminuirán cuando esté a nivel bajo.

- LD: Solo describiré su comportamiento en modo VARIABLE, ya que es el que empleamos. En dicho modo carga el valor establecido por el atributo IDELAY_VALUE.

- LDPIPEEN: Cuando está a nivel alto, carga el registro de segmentación con el valor de CNTVALUEIN. En nuestro caso aparece conectado a nivel alto, por lo que estará cargando en el registro de canalización el valor de CNTVALUEIN.

- REGRST: Cuando es alto, restablece el registro de canalización a cero. Solo se usa en el modo "VAR_LOAD_PIPE".

- DATAOUT: Saca los datos retrasados provenientes de IDATAIN o DATAIN.

- CNTVALUEOUT: Los pines CNTVALUEOUT se utilizan para informar del valor de conmutación dinámica del elemento de retardo. Solo está disponible cuando IDELAYE2 está en modo "VAR_LOAD" o "VAR_LOAD_PIPE", por lo que la salida queda al aire sin usar.

- IDDR: Este elemento es un registro de entrada diseñado para recibir señales externas de tipo DDR (Double Data Rate). El modo de captura a utilizar es el SAME_EDGE, en el que los datos se capturan en el flanco de subida y bajada del mismo ciclo. La Figura 25 ilustra los registros de entrada DDR y las señales asociadas en el modo same edge, así como un diagrama temporal que ilustra el funcionamiento de dichas señales.

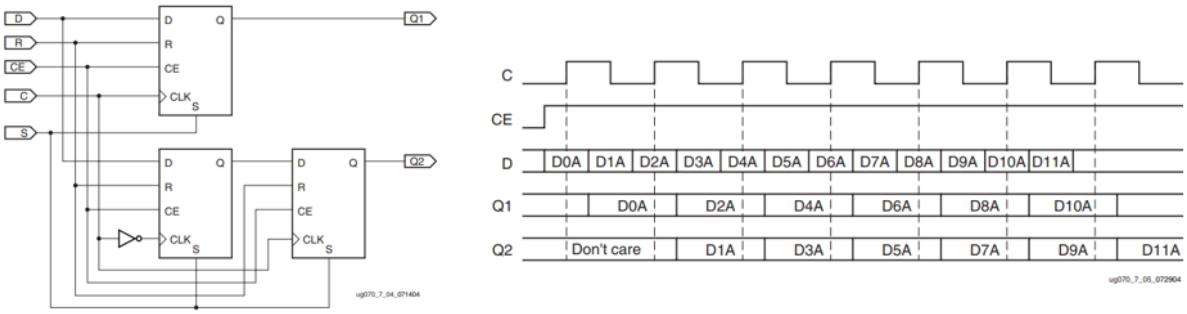


Figura 25: A la izquierda, registros de entrada DDR y señales asociadas en el modo same Edge. A la derecha, diagrama temporal que ilustra el funcionamiento de dichas señales. [20].

El objetivo es capturar y registrar las señales de datos entrantes con un reloj de doble velocidad de datos (host_clk), y sincronizarlos con el reloj del sistema (CLKOUT0). Para ello, Q1 deberá cambiar en los flancos de subida del reloj del sistema, y Q2 en los flancos de bajada. La configuración del componente para conseguir este objetivo consiste en: asociar los datos en cuestión a la entrada D, conectar CE a nivel alto (para cargar nuevos datos en el flip-flop), R y S a masa (estos pines modifican el nivel lógico de las salidas a bajo y alto, respectivamente), y C al reloj del ADC [9]. La Figura 26 muestra una representación gráfica del bloque IDDR.

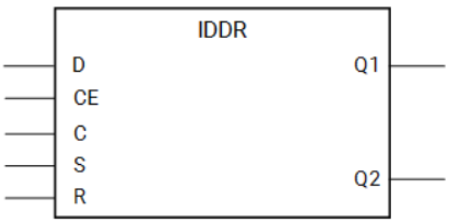
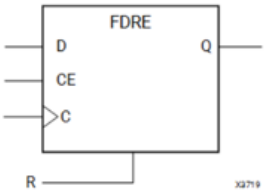


Figura 26: Bloque representativo del componente IDDR [9].

- FDRE: Este elemento es un simple flip-flop tipo D con señal de clock enable (CE) y de reset síncrono. Cuando CE está a nivel alto y la señal de reset no está activada, los datos en la entrada D se transfieren a la salida Q en los flancos de reloj (C). Este es, precisamente, el conexionado escogido para el componente. En la Figura 27 se puede observar una representación del componente en cuestión, así como la tabla de verdad del mismo [9].



Inputs				Outputs
R	CE	D	C	Q
1	X	X	↑	0
0	0	X	X	No Change
0	1	D	↑	D

Figura 27: Representación del componente y tabla de verdad [9].

2. Bloque de procesamiento de señal de reloj

En la Figura 28 se puede observar un diagrama de bloques que ilustra la arquitectura hardware usada para recibir la señal de reloj proveniente del ADC.

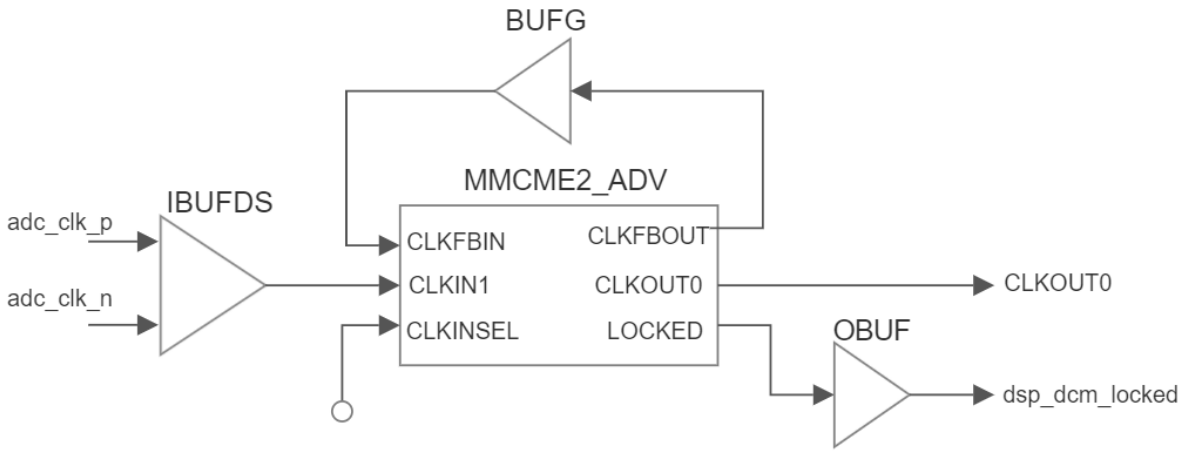


Figura 28: Diagrama ilustrativo del bloque de procesamiento de la señal de reloj.

La teoría de operación consiste en lo siguiente:

Los datos en modo diferencial entran a un búfer de entrada diferencial que los pasa a formato no diferencial para poder introducirlos en un gestor de reloj de señal mixta (MMCME2). Dicho gestor de reloj está diseñado para abordar la síntesis de frecuencia, la corrección de la red de reloj y la reducción de la fluctuación de fase. Cada una de las salidas de reloj puede tener una división, un cambio de fase y un ciclo de trabajo individuales. En nuestro caso, solo usaremos la salida CLKOUT0. A continuación, se expondrán en detalle los puertos usados del componente [9]:

- CLKFBIN: Pin de realimentación de reloj al MMCM.
- CLKIN1: Entrada de reloj primaria.
- CLKINSEL: Si está a nivel alto, como es el caso, selecciona CLKIN1 como entrada de reloj.
- CLKFBOUT: Salida de realimentación de reloj.
- CLKOUT0: Salida de reloj.

- LOCKED: Indica cuándo el MMCM ha logrado la alineación de fase dentro de una ventana predefinida y la coincidencia de frecuencia dentro de un rango de PPM predefinido. Se anulará LOCKED si el reloj de entrada se detiene o se viola la alineación de fase.

Además, cabe destacar algunos de los genéricos más importantes de la entidad:

- CLKFBOUT_MULT_F: Establece la cantidad por la que se van a multiplicar todas las salidas de reloj CLKOUT. Este número, en combinación con los valores de CLKOUT#_DIVIDE y DIVCLK_DIVIDE, determinará la frecuencia de salida.

- CLKIN1_PERIOD: Precisa el período de entrada en ns para la entrada CLKIN1.

- CLKOUT0_DIVIDE_F: Especifica la cantidad por la que se va a dividir la salida de reloj CLKOUT0. Este número, en combinación con los valores CLKFBOUT_MULT_F y DIVCLK_DIVIDE, determinará la frecuencia de salida.

- CLKOUT0_DUTY_CYCLE: Establece el ciclo de trabajo de la salida de reloj CLKOUT asociada en porcentaje (es decir, 0.50 generará un ciclo de trabajo del 50%).

- CLKOUT0_PHASE: Precisa el desplazamiento de fase en grados de la salida de realimentación del reloj. Cambiar el reloj de retroalimentación da como resultado un cambio de fase negativo de todos los relojes de salida.

- DIVCLK_DIVIDE: Especifica la relación de división para todos los relojes de salida con respecto al reloj de entrada.

3. Bloque de control del componente IDELAY

En la Figura 29 se puede observar un diagrama de bloques que ilustra la arquitectura hardware usada para para controlar el componente IDELAY.

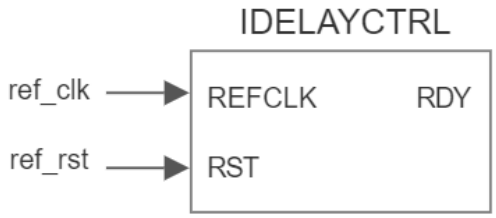


Figura 29: Diagrama ilustrativo del bloque de control del componente IDELAY.

El módulo IDELAYCTRL dispone de una entrada de reloj de referencia que permite que los circuitos internos obtengan una polarización de voltaje, independiente de las variaciones de PVT (proceso, voltaje y temperatura), para definir valores de tap-delay precisos para el componente IDELAYE2 [9].

REFCLK es la referencia de tiempo de IDELAYCTRL para calibrar el módulo IDELAYE2. Su frecuencia debe ser de 200 MHz para garantizar el valor de tap-delay especificado en la hoja de características.

En cuanto a la señal de RST, se trata de una simple señal de reset asíncrona activa a nivel alto.

La señal RDY indica cuando el componente IDELAYE2 está calibrado. Ya que no se precisa de su funcionalidad, se ha optado por dejar sin conectar.

4.1.2.2 Calibración

Para una correcta recepción en la FPGA de los datos digitalizados es preciso realizar un proceso de calibración. Esto se debe a que los flancos de la señal de reloj sincronizada con los datos del ADC coinciden con los bordes de los mismos. En dichos bordes aparecen unas zonas de transición en las que es conveniente evitar tomar muestras, ya que, si la señal de entrada cambia justo antes del flanco de reloj, el circuito puede volverse metaestable, lo que significa que puede entrar en un estado impredecible durante un período de tiempo indeterminado.

Por ello, el instante de muestreo deberá llevarse a cabo en el centro de la anchura efectiva de la trama de datos, que es el punto más alejado de las dos zonas de transición del mismo, y al que llamaremos centro de temporización ideal. La Figura 30 muestra un ejemplo de un proceso de calibración análogo al que nos ocupa. En este caso se lleva a cabo el centrado de la señal estroboscópica de datos (DQS) en un sistema DRAM para garantizar que esté alineado con los datos que se transfieren en el bus de datos (DQ).

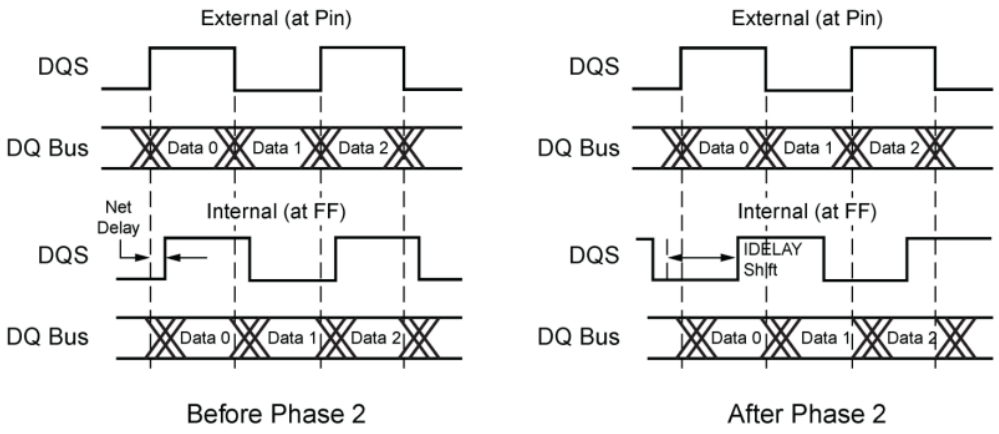


Figura 30: Proceso de calibración del bus de datos (extraído de https://support.xilinx.com/s/article/43667?language=en_US).

El proceso de calibración se fundamenta en aplicar retardos a las señales de datos, con esto se consigue que los flancos de captura de la señal de reloj caigan sobre el centro del dato y no sobre los bordes, asegurando así de que los datos capturados son correctos.

Para retardar las señales de datos entra en escena el componente IDELAYE2, expuesto anteriormente. Cualquier entrada (y algunas salidas) de la Zedboard puede ser retrasada individualmente hasta en 32 incrementos de 78 ps o 52 ps haciendo uso del componente IDELAY. El número de retardos se puede configurar en el componente y se pueden incrementar/decrementar durante el funcionamiento haciendo uso de los puertos INC y CE.

Para comprobar que la calibración se llevó a cabo de manera satisfactoria, hay que tener en cuenta que el ADC transmite los datos en formato DDR utilizando una aproximación source synchronous. En este formato DDR, cada línea del bus de datos transmite un par de bits consecutivos: en el semiciclo positivo el bit 2k+1 y en el semiciclo negativo el bit 2k.

Para calibrar, necesitamos un patrón que genere sobre cada una de las líneas del bus de datos una señal con la misma forma de onda que la señal de reloj síncrono. El ADC implementa algunos patrones de testeo, de entre los cuales se ha escogido el patrón Midscale Short, que reproduce continuamente la palabra mostrada en la Figura 31.

Output Test Mode Bit Sequence	Pattern Name	Digital Output Word 1	Digital Output Word 2	Subject to Data Format Select
0000	Off (default)	N/A[1]	N/A[1]	N/A[1]
0001	Midscale short	1000 0000 0000 0000	Same	Yes
0010	+Full-scale short	1111 1111 1111 1111	Same	Yes
0011	−Full-scale short	0000 0000 0000 0000	Same	Yes
0100	Checkerboard	1010 1010 1010 1010	0101 0101 0101 0101	No
0101	PN sequence long[2]	N/A[1]	N/A[1]	Yes
0110	PN sequence short[2]	N/A[1]	N/A[1]	Yes
0111	One-/zero-word toggle	1111 1111 1111 1111	0000 0000 0000 0000	No

Figura 31: Patrones de testeo del AD9467 [16].

El procedimiento llevado a cabo para realizar la calibración consiste en aplicar incrementos a la línea de datos, al mismo tiempo que se chequean los resultados de la captura de los mismos. Como en el modo midscale short los dos primeros bits de la palabra son constantes, se tomarán estos dos bits para comprobar que las recepciones son correctas. Si los datos capturados por la entidad coinciden con los datos aplicados a su entrada, será un indicio de que nos encontramos dentro de la anchura efectiva del dato. Sin embargo, si el valor de los datos cambia, indicará que el flanco de captura ha pasado al siguiente dato o a la zona de transición, o lo que es lo mismo, se ha alcanzado el final de la anchura efectiva de la trama de datos. Por tanto, su longitud temporal será igual al número de incrementos aplicados. Para encontrar el centro de dato, simplemente habrá que dividir entre dos la cifra de incrementos aplicada.

En la Figura 32 se muestra la línea de datos de la interfaz del ADC a través del analizador lógico ILA aplicando el modo Checkerboard, una vez los datos han sido calibrados.

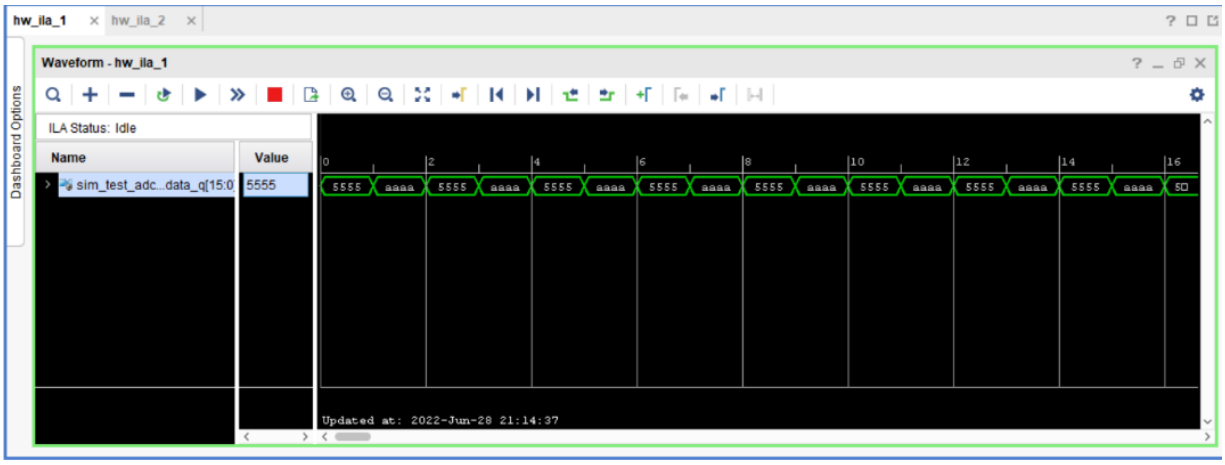


Figura 32: Línea de datos del ADC en modo checkerboard calibrados.

4.1.3 Banco de registros

La entidad incluye un banco de registros para realizar configuraciones desde una aplicación software, haciendo uso de una interfaz AXI Lite. Los registros disponibles son:

- CTRL_REG: Se trata del registro de control de la entidad y dispone de las funcionalidades de habilitación del core (bit 0), habilitación de interrupciones (bit 1), reseteo del software (bit 3), reseteo del bloque de señal mixta (MMCME2) (bit 4) y reseteo de la señal de streaming (bit 5).

- VERSION_REG: Comprueba la versión de la entidad a utilizar, contiene cuatro campos: REVISION (se utiliza para almacenar el número de la versión del diseño), MINOR VERSION (se utiliza para almacenar el número de versión menor del diseño), MAJOR VERSION (se utiliza para almacenar el número de versión mayor del diseño) y MAGIC NUMBER (se utiliza para almacenar un número único y codificado que identifica el diseño y verifica que sea compatible con el sistema que se está utilizando).

- INT_ENABLE_REG: Habilita las interrupciones en la entidad.

- INT_FLAG_REG: Habilita el flag de interrupción.

- SPI_CMD_REG: Este registro genera los dos primeros bytes del mensaje del protocolo SPI, que son los que portan la información necesaria para llevar a cabo los procesos de lectura o escritura.

- SPI_DATA_REG: Registro que almacena los datos recibidos a la entidad a través del protocolo SPI.

- ADC_IO_CALIB_REG: Accediendo al bit 4 de este registro se pueden llevar a cabo incrementos en elemento de retardos IDELAYE2.

- CALIB_SAMPLE_REG: Toma las muestras de datos provenientes del ADC, que son utilizadas para realizar el proceso de calibración.

- FRAME_REG: Registro encargado de generar el parámetro referente a la longitud de la trama.

- DECIMATE_REG: Genera el factor de diezmado. El factor de diezmado es un número que se elige como una potencia de 2 y determina cuántas muestras de entrada se agrupan y procesan en paralelo durante cada etapa

4.1.4 Librerías para el software

La librería adc_fmc_v1_0 contiene los ficheros que conforman el driver del componente en cuestión. El fichero adc_fmc.c incluye las funciones para realizar operaciones en el componente:

- adc_fmc_LookupConfig: Función de configuración inicial que establece el valor de los parámetros DEVICE_ID, BASE_ADDRESS y ADC_WIDTH.

- adc_fmc_get_version: Toma la versión de revisión del ADC.

- adc_fmc_get_enable: Comprueba si el core está habilitado.

- adc_fmc_set_enable: Habilita o deshabilita el core en función del nivel lógico del entero que se pasa como argumento de entrada.

- adc_fmc_get_sw_reset: Comprueba si se está llevando a cabo un reseteo del software.

- adc_fmc_set_sw_reset: Habilita o deshabilita la función de reseteo de software dependiendo del nivel lógico del entero que se pasa como argumento de entrada.

- adc_fmc_set_gie: Habilita o deshabilita la función de Global Interrupt Enable (gie) dependiendo del nivel lógico del entero que se pasa como argumento de entrada.

- adc_fmc_get_int_enable: Comprueba el estado de la función de habilitación de interrupción.

- adc_fmc_set_int_enable: Aplica la máscara de interrupciones al registro de habilitación de interrupciones.

- adc_fmc_get_int_flags: Comprueba el estado de los flags de interrupción.

- adc_fmc_clr_int_flags: Limpia los flags de interrupción.

Todas las funciones anteriormente descritas hacen uso de unas funciones de acceso a registros, las cuales se exponen a continuación:

- _set_reg: Establece un valor en un registro.

- _get_reg: Toma el valor de un registro.

- _set_reg_bits: Establece un valor en los bits seleccionados de un registro.

- _clr_reg_bits: Limpia los bits de un registro seleccionados con una máscara.

Por último, la librería también incluye las siguientes funciones de inicialización:

- adc_fmc_init_cfg: Función que inicializa la configuración cargando en el driver del core los valores establecidos en la función “adc_fmc_LookupConfig”.

- adc_fmc_init_hw: Función encargada de inicializar el hardware. Lleva esto a cabo reseteando el bloque, inhabilitando todas las interrupciones y limpiando flags de interrupción.

- adc_fmc_initialize: Función de inicialización del core. Su funcionamiento se fundamenta en la llamada de las funciones “adc_fmc_LookupConfig”, “adc_fmc_init_cfg” y “adc_fmc_init_hw”.

- adc_fmc_set_reg: Establece un valor en un registro del ADC.

- adc_fmc_get_reg: Toma el valor de un registro del ADC.

- adc_fmc_dcm_reset: Función encargada de habilitar/deshabilitar el reseteo del DCM (Digital Clock Manager) del componente.

- adc_fmc_dcm_locked: Comprueba si el DCM se encuentra bloqueado.

- adc_fmc_get_decimation: Toma el valor del registro de diezmado del core.

- adc_fmc_set_decimation: Establece un valor en el registro de diezmado del core.

- adc_fmc_get_frame_length: Toma el valor del registro de longitud de trama del core.

- adc_fmc_set_frame_length: Establece el valor de longitud de trama deseado por el usuario en el registro de longitud de trama del componente.

- adc_fmc_stream_rst: Función que habilita/deshabilita el reseteo de la transmisión en streaming de la interfaz.

- adc_fmc_transfer: Función que inicia una transferencia del ADC y espera a que esta termine.

- adc_calibrate: Función de calibración del ADC. Para ello, se va a seleccionar el modo de test Midscale Short (accediendo al registro TEST_IO_REG_IDX), como se expuso anteriormente. A continuación, se inicia una transmisión de datos, se resetea el componente IDELAYE_2 y se accede a un bucle que, en cada iteración, toma los dos bits más significativos de la trama de datos actual y los compara con los de la trama anterior. Si el valor de las tramas no es el mismo, se aplicará un incremento (accediendo a ADC_CALIB_REG_IDX) y se procesará otra iteración del bucle hasta que el valor coincida y sea igual a “10”. Esto indica que se ha superado la zona de skew y se ha alcanzado el primer borde de la curva de bañera. Para localizar el segundo borde se aplicarán incrementos en la trama de datos hasta que el valor de las tramas no coincida y sea distinto de “10”. Una vez obtenidos los dos bordes se calculará el centro de temporización ideal calculando su punto medio.

4.2 Creación de la interfaz “sys_FFT” para la generación de la Transformada de Fourier de los datos

La transformada rápida de Fourier (FFT) es un algoritmo empleado para transformar una señal en el dominio del tiempo en una señal en el dominio de la frecuencia. El bloque IP FFT de Xilinx Vivado proporciona una implementación del algoritmo FFT que se adapta a las capacidades específicas de las FPGA de Xilinx. Por ello, se ha optado por emplear dicho bloque IP para llevar a cabo la Transformada de Fourier de los datos introducidos a la Zedboard.

El algoritmo FFT que presenta el bloque se basa en un enfoque de divide y vencerás, donde los datos de entrada se dividen recursivamente en subconjuntos, hasta que son lo suficientemente pequeños como para ser resueltos de manera directa. La operación de mariposa es la piedra angular del algoritmo FFT y se utiliza para combinar los resultados de los subconjuntos.

El bloque dispone de diferentes métodos de operación. En este caso se ha optado por el método entrada/salida de transmisión segmentada (Pipelined Streaming I/O), ya que permite que se procese un flujo continuo de datos en paralelo y en tiempo real. Esto es así porque los datos de entrada se procesan en pequeños bloques, y la salida de cada bloque se usa como entrada para el siguiente bloque.

En la subdivisión de los datos de entrada, cada uno de los bloques se procesa a través de una serie de etapas de segmentación. Cada etapa de segmentación realiza una operación específica en los datos, como una operación mariposa. Estas operaciones implican sumar y restar números complejos y multiplicarlos por "factores de giro" complejos. Además, se realizan en paralelo, lo que permite un procesamiento de alto rendimiento y baja latencia.

Los argumentos de entrada del bloque FFT son los datos de salida de la interfaz del ADC, pero el conexionado de ambos bloques no es directo, por lo que hay que realizar una etapa previa de acondicionamiento de las señales. Se ha optado por crear un bloque IP con nombre “sys_FFT” que acondicione las señales de entrada para realizar un correcto conexionado con el bloque IP “Fast Fourier Transform”. Para su explicación, se va a dividir la arquitectura del bloque en cinco partes:

1. Sincronización de señales de información de transmisión.

En este bloque se lleva a cabo la sincronización de las señales host_n_frames, host_dec_factor y host_pkg_len, que portan la información referente al número de tramas, el factor de diezmado y la longitud de paquete, respectivamente. Estos parámetros deben ser definidos por el usuario en la aplicación software, por lo que accederán a la interfaz a través del canal S_AXI, cuyo reloj de referencia es el S_AXI_ACLK. La función del bloque es sincronizar las citadas señales con la frecuencia de reloj de los datos capturados por el ADC (dsp_clk), con el fin de poder usarlas para obtener señales de monitorización que indiquen parámetros como última trama o recepción en curso (señal running). En la Figura 33 se muestra un diagrama de bloques que ilustra la arquitectura a usar.

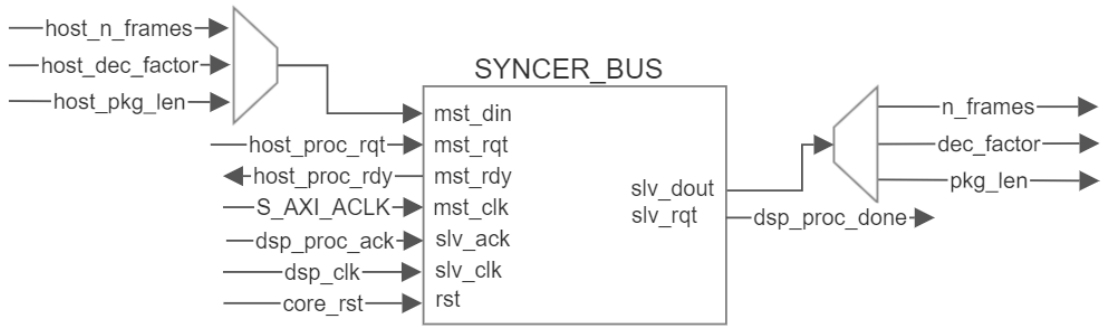


Figura 33: Diagrama de bloques de la etapa de sincronización de las señales de información de transmisión.

2. Prefifo y conexionado a bloque IP FFT.

El bloque consta de una FIFO en la que se introducen la señal de los datos provenientes del ADC y las señales de monitorización obtenidas con las señales sincronizadas en el anterior bloque. La función de esta FIFO es almacenar y suministrar correctamente los datos provenientes del ADC hacia el bloque FFT.

La estructura de los datos a introducir en la FIFO es la siguiente: los bits 0 al 15 están destinados a la palabra de datos que extrae el ADC (adc_data), el bit 16 a la señal de último bit de la muestra (sample_end & point_end), el bit 17 a la señal de última trama (iframe_end) y el bit 18 a la señal de final de paquete (oframe_end). La señal de validación de los datos de entrada de la FIFO se activará cuando lo esté también la señal S_AXIS_TVALID de la entidad (que está conectada a la señal M_AXIS_TVALID de la interfaz del ADC), la señal de recepción en curso y la señal de última muestra.

Los datos de salida de la FIFO se demultiplexan para obtener de nuevo las señales de último bit de la muestra (fft_din_last), última trama (fft_din_last_frame) y fin de paquete (fft_din_packet_end), junto con la señal de datos de 16 bits. Esta última constituirá la parte real de los datos a introducir en el bloque FFT, la parte imaginaria se implementará concatenando una señal de 16 bits de valor 0 a la señal de datos reales, resultando así una señal de datos de 32 bits.

Para acabar el conexionado con el bloque FFT, se asociarán las señales de último bit de la muestra y de validación de los datos de salida de la prefifo, a las señales tlast y tvalid del puerto esclavo del enlace AXI Streaming, respectivamente. El resultado, tras aplicar el algoritmo a los datos de entrada, será una palabra de 64 bits que saldrá por la señal tdata del puerto maestro del enlace AXI Streaming, acompañada de las respectivas señales de tlast, tready y tvalid. En cuanto a la señal de reloj que utiliza el bloque, se trata de la señal de reloj del ADC (dsp_clk). En la Figura 34 se muestra un diagrama de bloques que ilustra la arquitectura a usar.

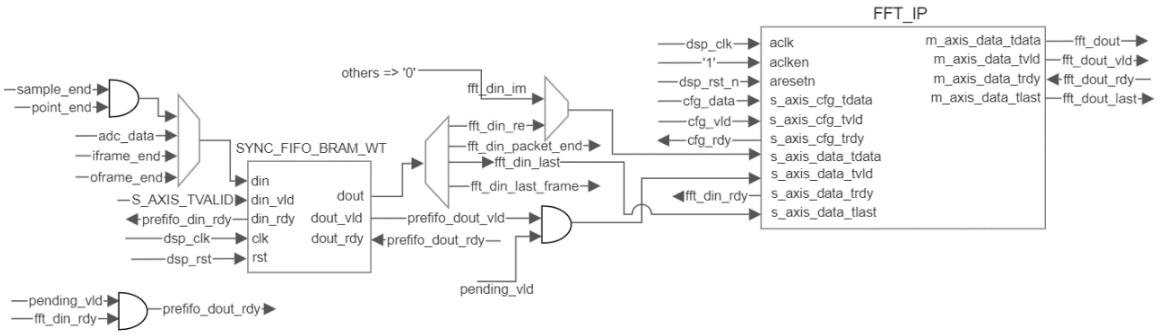


Figura 34: Diagrama de bloques de la etapa de prefifo y bloque FFT.

3. FIFO para la obtención de las señales de última trama y final de paquete.

Este bloque tiene como principal objetivo obtener unas señales de última trama y final de paquete sincronizadas con los datos de salida del bloque IP FFT. Para ello, la señal de reloj deberá ser la asociada los datos de salida del ADC. En cuanto a las dos señales de salida de la prefifo, portadoras de la información en cuestión, se introducen como argumento de entrada para el puerto de datos de una FIFO.

La señal de validación de dicho puerto de entrada se activará cuando comiencen las transmisiones de datos desde la prefifo al bloque FFT, mientras que la señal de validación del puerto de salida lo hará cuando haya acabado la transmisión de un paquete en el puerto de salida del bloque FFT. En la Figura 35 se muestra un diagrama de bloques que ilustra la arquitectura a usar.

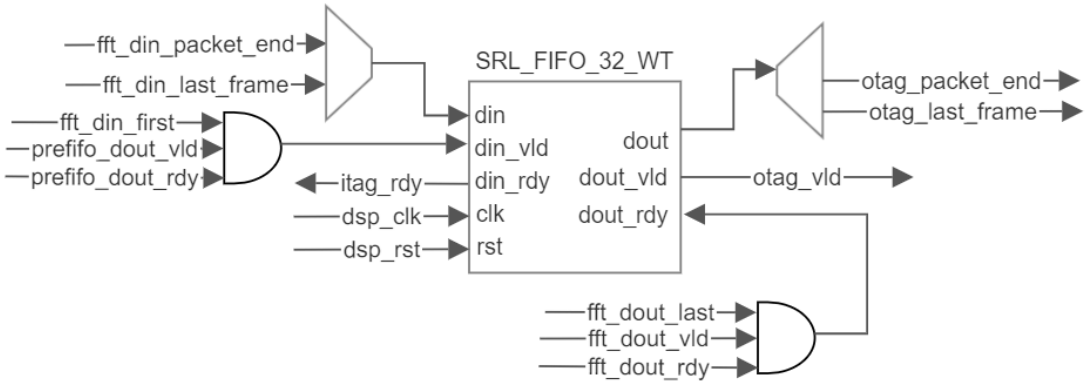


Figura 35: Diagrama de bloques de la etapa de FIFO para la obtención de las señales de última trama y final de paquete.

4. Postfifo

El bloque está compuesto por una FIFO de salida a la que llamaremos postfifo. La funcionalidad de la postfifo es almacenar y suministrar correctamente los datos de salida del bloque FFT, que se dirigen hacia el controlador DMA. Por tanto, su señal de reloj deberá ser la del ADC.

La estructura de los datos a introducir en el puerto din de la postfifo es la siguiente: los bits 0 al 63 están destinados a la trama de datos que extrae el bloque FFT (fft_dout), el bit 64 a la señal de último bit de la muestra (fft_dout_last), el bit 65 a la señal de última trama (otag_last_frame) y el bit 18 a la señal de final de paquete (otag_packet_end). Las señales de validación y listo de los datos de entrada de la postfifo están asociadas a las señales fft_dout_vld y fft_dout_rdy del bloque FFT, respectivamente.

En la parte referente al puerto de salida de la postfifo se obtienen, demultiplexando la señal dout, las señales de datos y último bit, que se conectarán al controlador DMA, y las señales de última trama y final de paquete. En cuanto a la señal de validación y listo, están conectadas a sus señales equivalentes del controlador DMA a través del enlace AXIS. En la Figura 36 se muestra un diagrama de bloques que ilustra la arquitectura a usar.

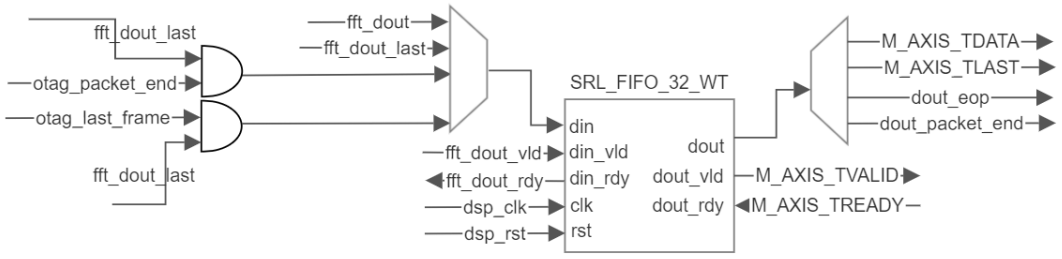


Figura 36: Diagrama de bloques de la etapa de postfifo.

5. Sincronizador para la obtención de interrupción al finalizar trama.

Este último bloque se va a encargar de generar una interrupción al finalizar el procesado de una trama de datos. Para ello se va a hacer uso de un sincronizador, con el que se recibirá una señal que indique la última trama sincronizada con el reloj del ADC. El sincronizador se encargará de extraer esta señal sincronizada con el reloj S_AXI_ACLK, perteneciente al enlace AXI de la interfaz con el que se accede al banco de registros. De esta manera, se podrá incluir la señal de interrupción en el banco de registros y podrá ser monitorizada desde una aplicación software.

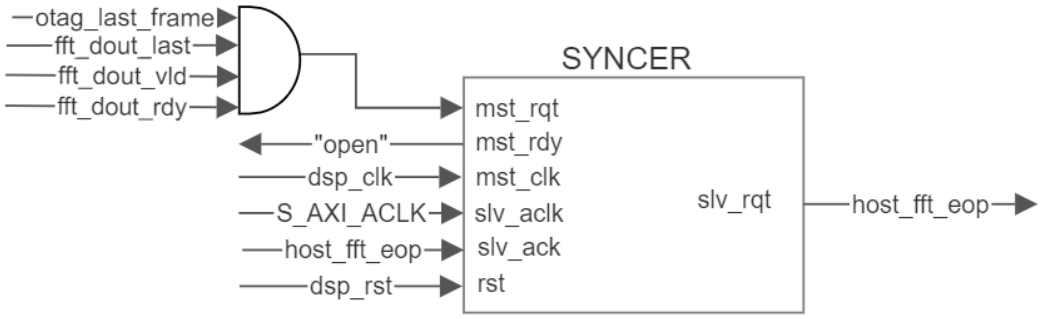


Figura 37: Diagrama de bloques de la etapa de sincronización para la obtención de interrupción.

4.2.1 Banco de registros:

La entidad incluye un banco de registros para realizar configuraciones desde una aplicación software, haciendo uso de los enlaces AXI Lite. Los registros disponibles son:

- CTRL_REG: Se trata del registro de control de la entidad y dispone de las funcionalidades de habilitación del core (bit 0), habilitación de interrupciones (bit 1), reseteo del software (bit 2), y modo usuario (bit 4).

- VERSION_REG: Comprueba la versión de la entidad a utilizar, contiene cuatro campos: REVISION (se utiliza para almacenar el número de la versión del diseño), MINOR VERSION (se utiliza para almacenar el número de versión menor del diseño), MAJOR VERSION (se utiliza para almacenar el número de versión mayor del diseño) y MAGIC NUMBER (se utiliza para almacenar un número único y codificado que identifica el diseño y verifica que sea compatible con el sistema que se está utilizando).

- INT_ENABLE_REG: Habilita las interrupciones en la entidad.

- INT_FLAG_REG: Habilita el flag de interrupción.

- PROC_CMD_REG: Este registro, a partir de las especificaciones del usuario en la aplicación software, genera las señales referentes al número de tramas a procesar, el factor de diezmado y a la disponibilidad del procesador para realizar operaciones.

- PKT_LEN_REG: Registro que genera la señal referente a la longitud del paquete de datos, a partir de las especificaciones del usuario en la aplicación software.

4.2.2 Librerías para el software.

La librería astro_fft_v1_0 contiene los ficheros que conforman el driver del componente en cuestión. El fichero astro_fft.c porta las funciones para realizar operaciones en el core:

- fft_LookupConfig: Función de configuración inicial que establece el valor de los parámetros DEVICE_ID, FFT_BASE_ADDRESS y FFT_SIZE.

- fft_get_version: Toma la versión de revisión del core a medida.

- fft_get_enable: Comprueba si el core está habilitado.

- fft_set_enable: Habilita o deshabilita el core en función del nivel lógico del entero que se pasa como argumento de entrada.

- fft _get_sw_reset: Comprueba si se está llevando a cabo un reseteo del software.

- fft_set_sw_reset: Habilita o deshabilita la función de reseteo de software dependiendo del nivel lógico del entero que se pasa como argumento de entrada.

- fft_set_gie: Habilita o deshabilita la función de Global Interrupt Enable (gie) dependiendo del nivel lógico del entero que se pasa como argumento de entrada.

- fft_get_int_enable: Comprueba el estado de la función de habilitación de interrupción.

- fft_set_int_enable: Aplica la máscara de interrupciones al registro de habilitación de interrupciones.

- fft_get_int_flags: Comprueba el estado de los flags de interrupción.

- fft_clr_int_flags: Limpia los flags de interrupción.

Todas las funciones anteriormente descritas hacen uso de unas funciones de acceso a registros, las cuales se exponen a continuación:

- _set_reg: Establece un valor en un registro.

- _get_reg: Toma el valor de un registro.

- _set_reg_bits: Establece un valor en los bits seleccionados de un registro.

- _clr_reg_bits: Limpia los bits de un registro seleccionados con una máscara.

Por último, la librería también incluye las siguientes funciones de inicialización:

- fft_init_cfg: Función que inicializa la configuración cargando en el driver del core los valores establecidos en la función “fft_LookupConfig”.

- fft_init_hw: Función encargada de inicializar el hardware. Lleva esto a cabo reseteando el bloque, inhabilitando todas las interrupciones y limpiando flags de interrupción.

- fft_initialize: Función de inicialización del core. Su funcionamiento se fundamenta en la llamada de las funciones “fft_LookupConfig”, “fft_init_cfg” y “fft_init_hw”.

- fft_get_n_points: Toma el valor de la variable n_points que define el usuario.

- fft_get_run_mode: Comprueba si el core se encuentra en modo run.

- fft_set_run_mode: Función encargada de habilitar/deshabilitar el modo run del core.

- fft_get_packet_len: Toma el valor del registro de longitud de paquete del core.

- fft_set_packet_len: Establece un valor en el registro de longitud de paquete del core.

- fft_get_n_frames: Toma el valor de la variable referente al número de tramas n_frames, que define el usuario.

- fft_get_decimation: Toma el valor de la variable referente al factor de diezmado que define el usuario.

4.3 Bloque DMA (Direct Memory Access)

Direct Memory Access (DMA) es una técnica que permite que un dispositivo realice escrituras o lecturas de datos hacia o desde la memoria del sistema, sin necesidad de hacer uso de la CPU. Por lo tanto, permite liberar en gran medida la carga de trabajo en la misma, llegando incluso a realizar otras operaciones mientras las escrituras y lecturas de datos están en curso. Además, las escrituras y lecturas que realiza un controlador DMA tienen tasas de transferencia mucho más altas que las que la CPU podría ofrecer.

Los canales DMA son utilizados para establecer comunicaciones entre diferentes direcciones de memoria, diferentes periféricos, o entre un periférico y memoria [10]. Para transferir los datos procesados con el algoritmo FFT hacia el bloque de memoria DDR3 se va a incorporar en la PL de la Zedboard el controlador DMA "AXI Direct Memory Access", incluido en las librerías de Xilinx. Una vez acabada la transmisión, se activará una interrupción con la que se monitorizarán futuras recepciones.

Un controlador DMA puede funcionar en diferentes modos. Los principales son el modo registro directo, el modo scatter/gather y el modo cíclico.

En el modo registro directo es preciso configurar el controlador DMA cada vez que se quiere llevar a cabo una transmisión. Cuando dicha transmisión acaba, se lanza una interrupción, entonces si se desea realizar otra transferencia se debe reprogramar el controlador DMA.

La problemática relacionada con la programación del componente que surgía en el modo registro directo puede ser paliada con el modo scatter/gather. Este modo tiene especial relevancia cuando la serie requerida de transmisiones es conocida, constante y se desea que sucedan de manera automática. Permite realizar todas las transmisiones, sin necesidad de reprogramar el controlador en cada una de ellas, creando unas estructuras llamadas descriptores de buffer.

Estos descriptores de buffer (BD) son estructuras de datos que contienen un puntero que apunta hacia la dirección inicial de una región de memoria. Almacenan información referente a las transmisiones que se van a llevar a cabo, como la cantidad de datos a ser leídos/escritos en dicha dirección de memoria. Una ventaja adicional de estas estructuras es la posibilidad de realizar transferencias hacia direcciones de memoria no contiguas, lo cual optimiza el uso de la memoria.

En cuanto al modo cíclico, en este modo el controlador DMA obtiene y procesa los mismos BDs cíclicamente sin interrupción. El controlador DMA continúa recuperando y procesando hasta que se detiene o reinicia.

Pero no todo son ventajas, el uso de los descriptores de buffer está limitado por el uso que hacen de la memoria DDR, en la que almacenan la información de las transferencias y de la propia región que ocupa el descriptor. Cuando se completa la transferencia de un descriptor de buffer, el controlador DMA pasa automáticamente a la siguiente

transferencia, sin dejar espacio entre transmisiones. Es más, una entrada común en un BD es un puntero a otro BD que debe procesarse inmediatamente después de que concluya la transferencia de datos hacia o desde la dirección señalada [11].

En cuanto a la organización de los descriptores de buffer, cada BD dispone de dos palabras de 32 bit: una referente a la dirección del buffer de recepción, y otra a la configuración y estado.

Los BDs incluyen unos bits de control de estado: los bits de inicio de trama/transmisión (SOF) y los bits de final de trama/transmisión (EOF), que indican el primer y último paquete a transmitir, respectivamente.

En cuanto a la máquina de estado que sigue un BD, dispone de cuatro estados: Free, Pre-Process, Hardware y Post-Process, los cuales se expondrán a continuación.

Para llevar a cabo la inicialización del componente se va a hacer uso de las funciones pertenecientes al driver XaxiDma, incluido en el BSP (Board Support Package) de la aplicación software. El primer paso es obtener una estructura de configuración (dirección base, ID de interrupción, etc) para el dispositivo DMA en cuestión, haciendo uso de la función "XaxiDma_LookupConfig", e inicializar dicho dispositivo con los citados ajustes de configuración (establecer la dirección base del dispositivo, configurar ajustes de interrupción, etc), haciendo uso de la función "XaxiDma_CfgInitialize".

A continuación, se creará un anillo de descriptores de buffer para el dispositivo DMA. Un anillo de BDs es una estructura de datos que contiene la información necesaria referente a los buffers para que la transferencia se lleve a cabo con éxito. Para su creación se hace uso de la función "XaxiDma_BdRingCreate", a la que hay que pasar como argumento un puntero que apunte al descriptor de anillo (obtenido con la función "XaxiDma_GetRxRing"), que contiene información como la dirección base de memoria, el número total de BDs y el tamaño de cada BD. Una vez definidas estas cuestiones se crea el anillo, se inicializa con los parámetros especificados y lo establece en el estado "Free", o inactivo. En este estado los descriptores todavía no están programados para realizar transferencias DMA.

Para que el software pueda tomar control de los descriptores y, por ende, puedan ser programados para realizar transferencias, se debe pasar al estado "Pre-Process", llamando a la función "XaxiDma_BdRingAlloc". Dicha función se utiliza para asignar un cierto número de BD del anillo de descriptores para que los utilice el motor DMA. Estos BD se pueden configurar con la dirección de buffer, la longitud o los campos de control, entre otros parámetros.

Seguidamente, se pueden configurar aspectos tales como: especificar las direcciones a las que deben apuntar los descriptores (llamando a la función "XaxiDma_BdSetBufAddr"), establecer la longitud del buffer en un BD (con la función "XaxiDma_BdSetLength"), establecer campos de control (incluyen opciones como habilitar/deshabilitar generación de los flags de interrupción EOF y SOF, coherencia de caché o manejo de errores) en un BD (llamando a la función "XaxiDma_BdSetCtrl") y

establecer el campo ID (campo definido por el usuario que se puede emplear para asociar una BD específica con una transferencia de datos o un búfer específicos) en un BD (con la función “XaxiDma_BdSetId”). En nuestro caso se desactivarán todas las opciones de control, ya que no son necesarias, y se asociará el BD con el buffer de recepción de datos.

Posteriormente, es necesario vaciar el contenido de las cachés del BD y el buffer de recepción (este último no es estrictamente necesario si la coherencia de caché está activada). Esto se hace con el fin de que se actualice el contenido de la memoria principal, ya que habrá que llevar a cabo los pasos anteriores de configuración tantas veces como descriptores haya en el anillo.

El siguiente paso es enviar los descriptores al dispositivo DMA para su procesamiento, lo que se lleva a cabo llamando a la función “XaxiDma_BdRingToHw()”. Al llamar a esta función los descriptores entran en el estado “Hardware”. Por lo general, se llama después de que la aplicación haya preparado uno o más BD para la transferencia, pero en nuestro caso ha sido llamado en cada iteración del bucle para poner en cola a cada descriptor individualmente. Se ha optado por este método para aprovechar al máximo la memoria disponible, ya que el espacio de memoria alineada a asignar a cada descriptor individualmente es menor que si se asigna de manera colectiva a todos los descriptores.

Antes de comenzar la transferencia de datos hay que configurar las interrupciones del sistema para notificar el final de las rutinas de interrupción, y conectar la fuente de interrupción con su handler asociado. Estas dos tareas se llevarán a cabo con funciones del driver del “XScuGic”, un controlador de software proporcionado por Xilinx para su IP "ScuGic" (SCU General Interrupt Controller), un controlador de interrupción programable que está integrado en Zynq. La primera tarea se llevará a cabo llamando a la función “XScuGic_CPUWriteReg”, con la cual se escribirá el ID de interrupción del controlador DMA en el registro EOI (End Of Interrupt). La segunda llamando a la función “XScuGic_Connect”, con la que se conectará el ID de la interrupción del controlador DMA con el handler asociado.

Para iniciar la trasferencia de datos y habilitar el motor DMA se hace uso de la función “XaxiDma_BdRingStart”. También es interesante llamar a la función “XaxiDma_BdRingSetCoalesce”, que establece el número de transferencias que debe realizar el dispositivo (normalmente es igual al número de BDs del anillo, para que se genere la interrupción al acabar de procesar el mismo) antes de que se genere una interrupción. Por último, llamando a la función “XaxiDma_BdRingIntEnable” se habilitan las interrupciones en el dispositivo.

Una vez se ha procesado el anillo de descriptores, los BDs permanecen en el hardware hasta que el usuario recupere el control llamando a la función “XaxiDma_BdRingFree”. Al hacerlo, se pasa al estado “Post-Process”, en el cual el usuario vuelve a tener acceso a los BD y puede determinar si las transacciones fueron exitosas o no. La Figura 38 muestra la máquina de estados de un BD.

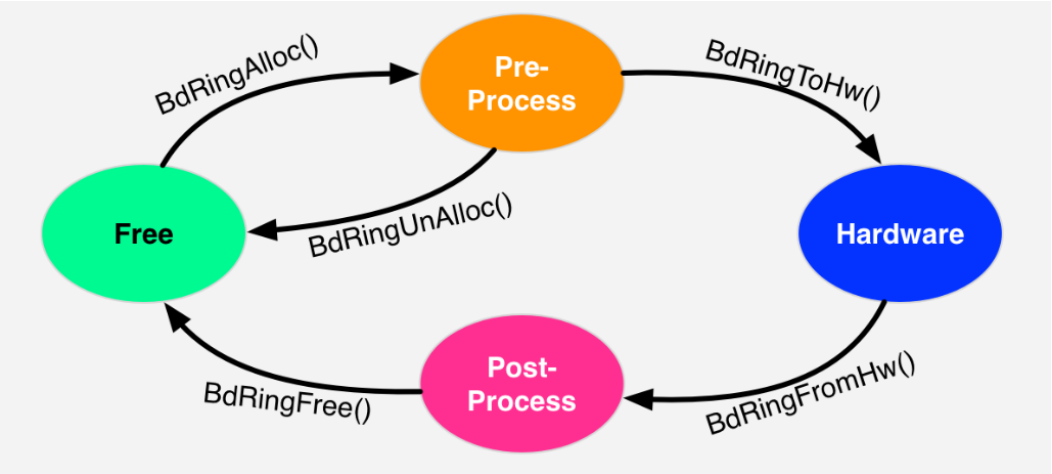


Figura 38: Diagrama representativo de la máquina de estados de un BD [11]

En el caso de nuestra aplicación, el objetivo es transferir a memoria los datos provenientes de un ADC, después de ser procesados con el algoritmo FFT. Para ello se debe reconfigurar el anillo de descriptores cada vez que finalice la transmisión de un paquete de datos. Para reconfigurarlo, primero hay que liberar los BDs del anillo estableciendo estos en el estado Post-Process. Una vez liberados, se lleva a cabo el proceso de configuración expuesto anteriormente. Esta reconfiguración se debe llevar a cabo una vez finalizada una transmisión, por lo que se lleva a cabo dentro del handler de la interrupción de final de transmisión del controlador DMA.

Para obtener una transferencia continua e indefinida de datos hacia memoria es preciso que el flujo de datos hacia el controlador DMA sea continuo también. Para conseguir esto se hace uso de una unidad de almacenamiento que vaya suministrando los datos de forma continua, como por ejemplo una FIFO. El controlador DMA permite mover los datos en grupos de 8 bytes cada ciclo, que pueden verse como palabras de 64 bits.

4.4 Desarrollo hardware

Para llevar a cabo el desarrollo hardware se va a hacer uso de la herramienta Vivado Design Suite en su versión 2017.4. El procedimiento de trabajo llevado a cabo con la citada herramienta consiste en unir bloques IP (ya sean proporcionados por las librerías de Xilinx o customizados), por medio de asistentes de cableado que proporciona la propia herramienta, con el fin de conseguir la funcionalidad deseada.

ZYNQ7 PROCESSING SYSTEM

La piedra angular de todo diagrama de bloques es el bloque IP Zynq7 Processing System. Este bloque representa el sistema central de procesamiento y el resto de bloques del sistema deberán de estar ligados al mismo. La Figura 39 muestra una representación gráfica del bloque IP referente al Zynq7 Processing System.

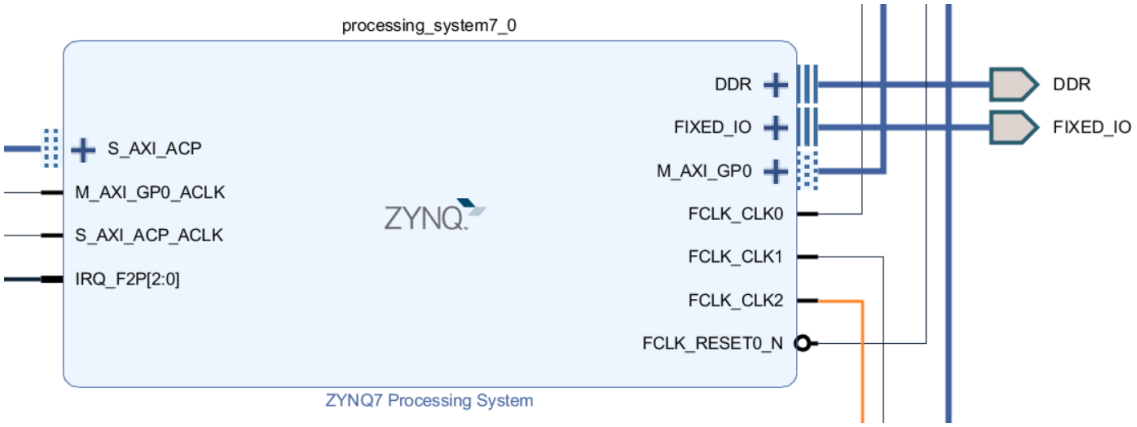


Figura 39: Bloque Zynq7 Processing System.

Al hacer doble clic sobre el componente se accede a un asistente que permite configurar el bloque para hacer uso de la funcionalidad deseada por el usuario. En la Figura 40 se muestra la interfaz gráfica del asistente.

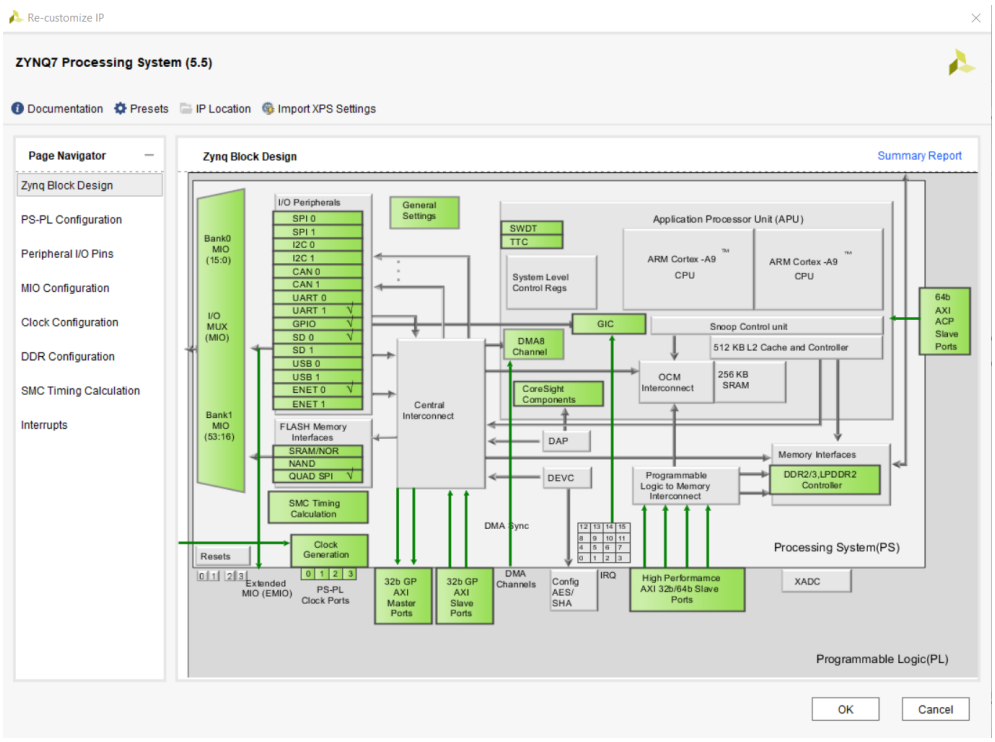


Figura 40: Asistente gráfico para configuración del bloque IP.

Las funcionalidades que debe abordar el procesador de nuestro sistema deben comprender:

- Comunicación DMA: Para poder gestionar las transmisiones y recepciones en memoria del controlador DMA, es preciso habilitar el interfaz AXI esclavo ACP (AXI Coherency Port). El citado puerto permite que el sistema de procesamiento (PS) y la lógica programable (PL) compartan memoria y se comuniquen entre sí de manera eficiente al mantener la coherencia de la memoria caché. Esto permite que el PS y la PL accedan a la misma ubicación de memoria sin necesidad de una transferencia de datos explícita entre ellos, lo que puede mejorar el rendimiento y reducir la latencia del sistema.

- Generación de frecuencias de reloj internas: Es preciso generar con el procesador dos frecuencias de reloj. Una de ellas hace referencia a la frecuencia de reloj interna de 100MHz, con la cual funcionarán las conexiones AXI Lite del sistema. La otra, es la señal de reloj de referencia de 200MHz necesaria en el controlador del componente IDELAY, expuesto anteriormente.

- Interfaces de memoria DDR2/3: Serán usados para almacenar la información captada por el ADC.

- Soporte para canales S_AXI: Para dar soporte a los interfaces esclavos AXI de los componentes del sistema es preciso habilitar la interfaz maestra GP0 (General Purpose I/O), que hace referencia a los puertos de propósito general.

- Gestión de las interrupciones del sistema: Las interrupciones generadas por el sistema son tres: la que indica el final de procesamiento en la interfaz del ADC, la que indica el final de procesamiento en el bloque FFT y la que indica que se ha completado el proceso de escritura en memoria por parte del controlador DMA (s2mm_introut). Para poder monitorizarlas se deben concatenar (haciendo uso del bloque IP “concat”) y conectar al sistema de procesamiento a través del puerto de interrupción IR_F2P. Las señales IRQ_F2P están conectadas al PS GIC (Generic Interrupt Controller), que es el responsable del manejo de interrupciones.

AXI INTERCONNECT

El bloque AXI Interconnect tiene la función de interconectar puertos esclavos con puertos maestros del sistema que siguen el protocolo AXI. A la vez que interconecta los bloques, sincroniza sus frecuencias de reloj respectivas.

En nuestro sistema, los bloques AXI interconnect son usados para enlazar los puertos ACP y GP0 con sus respectivos puertos maestros y esclavos, respectivamente. En la Figura 41 se puede observar la representación del bloque.

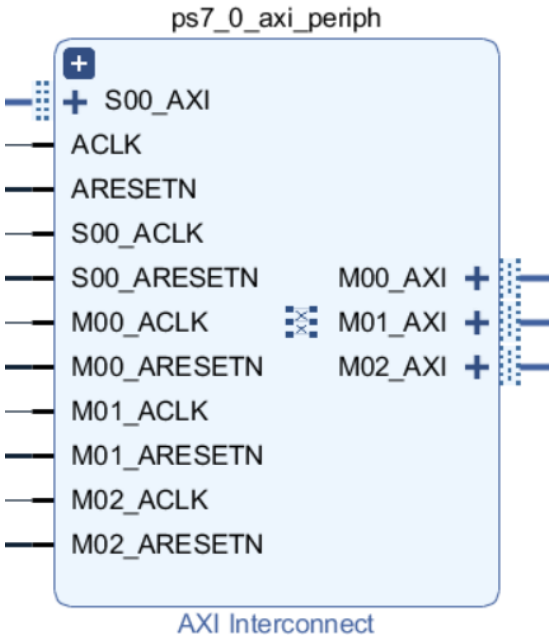


Figura 41: Bloque IP AXI Interconnect.

PROCESSOR SYSTEM RESET

El bloque Processor System Reset genera resets de forma síncrona para los elementos del sistema, tales como el procesador, los bloques de interconexión y demás periféricos. La Figura 42 muestra el bloque de generación de reset usado en el sistema.

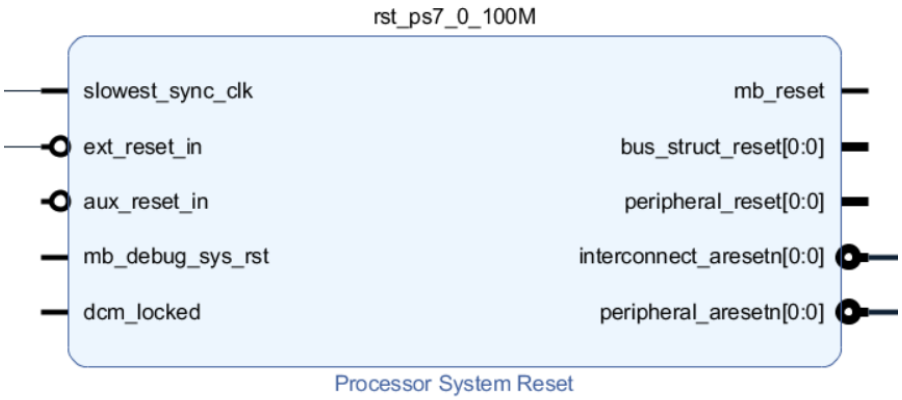


Figura 42: Bloque IP Processor System Reset.

Los puertos más importantes del bloque son [12]:

- Slowest_sync_clk: Reloj de sincronismo del sistema.

- Ext_reset_in: Entrada de señal de reset externa. Es activo a nivel bajo o alto en función del estado de su genérico asociado “Ext Reset Active Polarity”.

- Aux_reset_in: Entrada de señal de reset auxiliar. Es activo a nivel bajo o alto en función del estado de su genérico asociado “Auxiliary Reset Active Polarity”.

- Interconnect_aresetn: Señal de reset activa a nivel bajo dedicada a dar soporte a los bloques de interconexión del sistema.

- Peripheral_aresetn: Señal de reset activa a nivel bajo dedicada a dar soporte a los periféricos cuyo reloj coincide con el reloj de sincronismo del sistema.

AXI ADC CONTROLLER

El bloque AXI ADC Controller es un IP a medida encargado de establecer la interfaz de unión entre el ADC y la Zedboard, unidos por medio del puerto FMC de esta última. Su teoría de operación fue descrita en apartados anteriores, por lo que en este nos centraremos en los puertos del bloque y en su funcionalidad en el conjunto del sistema. La Figura 43 muestra una representación gráfica del bloque IP referente al AXI ADC Controller.

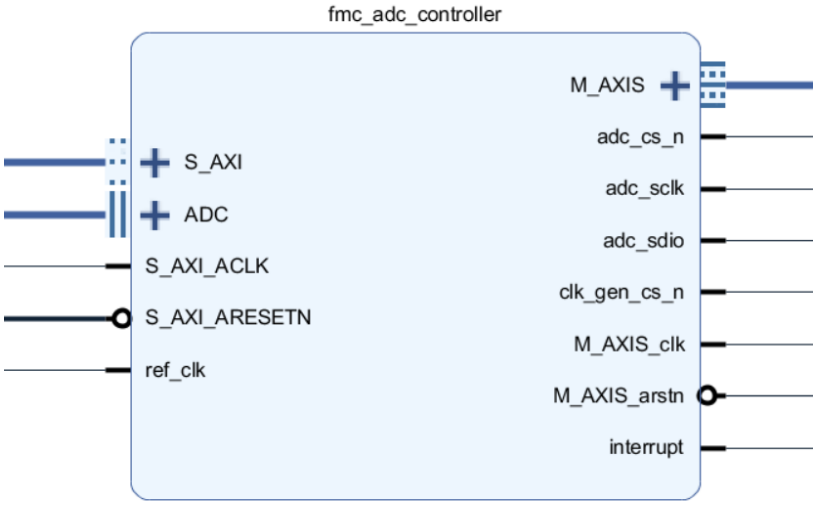


Figura 43: Bloque IP AXI ADC Controller.

Los puertos más relevantes del bloque son:

- S_AXI: Un canal AXI esclavo que da soporte para la modificación de registros y otras labores de comunicación con la entidad. Sus señales de reloj y reset asociadas son las conectadas a los puertos S_AXI_ACLK y S_AXI_ARESETN, respectivamente.

- ADC: Interfaz para la conexión de los puertos físicos del ADC, entre los que se incluyen las señales diferenciales de datos, reloj y overflow. Dichos puertos se conectan a las entradas de la interfaz física del bloque, descrita anteriormente.

- Ref_clk: Entrada de reloj dedicada a conectar la señal de 200MHz que precisa el controlador del componente IDELAY para garantizar el valor de tap_delay.

- M_AXIS: Canal maestro de AXI-Stream con la función de extraer los datos introducidos por el puerto “ADC”. Dichos datos de salida están sincronizados con

las señales de reloj y reset M_AXIS_CLK y M_AXIS_RST. El canal maestro dispone de señal de salida de datos, señal de tvalid, tlast y tuser.

- adc_cs_n: Puerto de salida portador de la señal CSB de habilitación de los ciclos de lectura/escritura del componente AD9467, que está conectado al puerto SPI de entrada correspondiente del citado componente.

- adc_sclk: Puerto de salida portador de la señal de reloj SCLK del componente AD9467, que está conectado al puerto de entrada correspondiente del citado componente.

- Adc_sdio: Puerto bidireccional asociado a la señal de datos SDIO del componente AD9467, ya sea para procesos de lectura o escritura. Dicho puerto está conectado al puerto correspondiente del citado componente.

- Clk_gen_cs_n: Puerto de salida portador de la señal que define si se va a establecer comunicación con el AD9467 o el AD9517. Este irá conectado al puerto correspondiente en la placa.

- Interrupt: Puerto de salida portador de la señal de interrupción de la interfaz del ADC.

Para que el sistema asocie correctamente los puertos de la entidad relativos a la interfaz física (ADC) y a la interfaz SPI (adc_cs_n, adc_sclk, adc_sdio y clk_gen_cs_n), es preciso enlazarlos a los puertos correspondientes de la placa FMC por medio de un fichero de restricciones.

SYS_FFT

El bloque sys_FFT es un IP customizado encargado de aplicar el algoritmo FFT sobre los datos muestreados por el ADC. Su teoría de operación fue descrita en apartados anteriores, por lo que en este nos centraremos en los puertos del bloque y en su funcionalidad en el conjunto del sistema. En la Figura 44 se puede observar la representación del bloque.

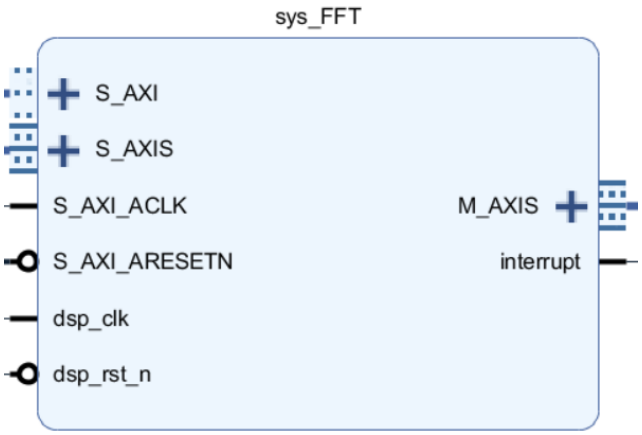


Figura 44: Bloque IP sys_FFT.

Los puertos más relevantes del bloque son:

- S_AXI: Un canal AXI esclavo que da soporte para la modificación de registros y otras labores de comunicación con la entidad. Sus señales de reloj y reset asociadas son las conectadas a los puertos S_AXI_ACLK y S_AXI_ARESETN, respectivamente.

- S_AXIS: Canal AXI Streaming de entrada para la conexión con el canal M_AXIS de la interfaz del ADC.

- dsp_clk: Entrada dedicada a la señal de reloj que va sincronizada con los datos de salida del ADC (DCO).

- Dsp_rst_n: Entrada dedicada al reset asociado al reloj del ADC.

- M_AXIS: Puerto AXI-Stream maestro, utilizado para extraer los datos hacia el DMA. Dichos datos de salida están sincronizados con las señales de reloj y reset M_AXIS_CLK y M_AXIS_RST. Dispone las señales m_axis_tdata, m_axis_tready, m_axis_tlast y m_axis_tvalid.

- Interrupt: Puerto de salida portador de la señal de interrupción del core.

AXI GPIO

El bloque AXI_GPIO está disponible en las librerías de Xilinx y proporciona una interfaz entrada/salida de propósito general. El bloque está diseñado para interactuar con el canal S_AXI, que está conectado al maestro de propósito general del sistema de procesamiento. En la Figura 45 se puede observar la representación del bloque.

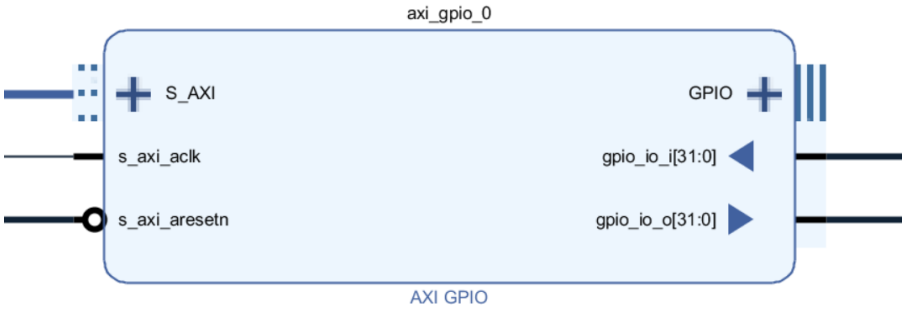


Figura 45: Bloque IP AXI Gpio.

AXI_DMA

Para abordar el acceso directo a memoria se va a usar el bloque IP AXI DMA, disponible en las librerías de Xilinx. Dicho bloque proporciona un acceso directo a memoria de gran ancho de banda entre interfaces de memoria mapeada AXI4 y AXI4-Stream [10]. Se accede a los registros de estado e inicialización del bloque a través de un interfaz esclavo AXI4-Lite.

El flujo de datos principal entre la memoria del sistema y el IP se da por medio del canal de lectura AXI4 MM2S (memory-mapped to stream), y el canal de escritura AXI4 S2MM (stream to memory-mapped). Los canales MM2S y S2MM funcionan de forma independiente y pueden realizar transferencias al mismo tiempo. En nuestro caso, solo se hará uso del canal S2MM, ya que solo se precisa de realizar escrituras en memoria de los datos del ADC procesados con el algoritmo FFT en el bloque sys_FFT. Además, el bloque AXI DMA provee de soporte de realineamiento de datos a nivel de byte, que permite que la memoria lea y escriba a partir de cualquier ubicación de bytes desplazada.

El canal de datos en streaming es el AXI4-Stream Slave (puerto S_AXIS_S2MM). El canal S2MM dispone de una interfaz AXI-Stream de control (puerto M_AXI_S2MM) para recibir datos de la aplicación del usuario desde la IP de destino. A este canal de control maestro hay que añadir uno adicional para el modo Scatter-Gather, el M_AXI_SG, que será el empleado y se explicará a continuación.

El puerto M_AXI_SG (Scatter Gather) se utiliza para transferir datos mediante la técnica DMA de dispersión y recopilación. Dicha técnica permite que el controlador DMA transfiera datos desde múltiples ubicaciones de memoria en el PS a múltiples ubicaciones de memoria en la PL, o viceversa. Esta técnica es útil en escenarios donde los datos no son contiguos en la memoria y permite que el controlador DMA transfiera datos en fragmentos más pequeños, lo que reduce las posibilidades de errores y aumenta el rendimiento.

Adicionalmente, la interrupción generada por el controlador DMA (s2mm_introut) está conectada al sistema de procesamiento a través del puerto IRQ_F2P. En la Figura 46 se puede observar la representación del bloque.

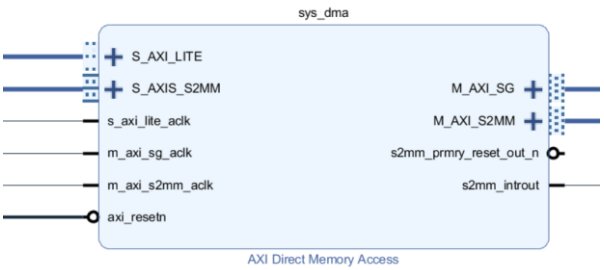


Figura 46: Bloque IP AXI Direct Memory Access.

DISEÑO GLOBAL

Una vez descritas todas las partes del sistema, la arquitectura del sistema será la siguiente:

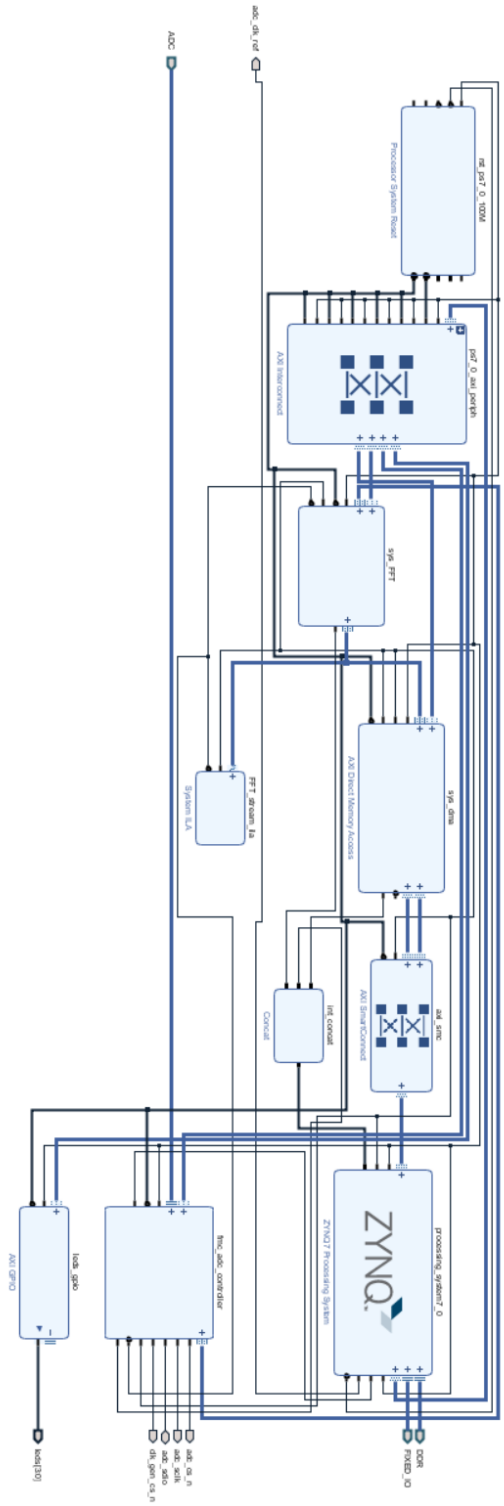


Figura 47: Diagrama de bloques del sistema.

El diagrama de bloques de la Figura 47 ha sido generado con el integrador de IPs de Vivado. Una vez definido el conexionado es preciso mapear los puertos de las interfaces de la arquitectura. Vivado lleva a cabo esta operación de manera automática, aunque cabe destacar que, en diseños que utilizan controladores de acceso a memoria, puede resultar necesario excluir ciertos segmentos de direcciones del acceso de DMA a la memoria.

La principal razón por la que se lleva a cabo la exclusión es evitar que el controlador DMA acceda a la memoria que ya está asignada a otras partes del sistema. Así se obtiene una mejora del sistema en cuanto a la seguridad, ya que puede evitar el acceso no autorizado a regiones de memoria privadas; estabilidad, ya que evita accesos a memoria que podrían provocar que el sistema se bloquee o se vuelva inestable; y rendimiento, ya que evita accesos a memoria que podrían ralentizar el sistema.

En la Figura 48 se pueden observar las direcciones de memoria asignadas a los puertos de la arquitectura.

Cell	Slave Interface	Base Name	Offset Address	Range	High Address
axi_dma_0					
Data_SG (32 address bits : 4G)					
proces...	S_AXI_ACP	ACP_DDR...	0x0000_0000	512M	0x1FFF_FFFF
proces...	S_AXI_ACP	ACP_QSPI...	0xFC00_0000	16M	0xFCFF_FFFF
Excluded Address Segments (2)					
proc...	S_AXI_ACP	ACP_IOP	0xE000_0000	4M	0xE03F_FFFF
proc...	S_AXI_ACP	ACP_M_A...	0x4000_0000	1G	0x7FFF_FFFF
Data_S2MM (32 address bits : 4G)					
proces...	S_AXI_ACP	ACP_DDR...	0x0000_0000	512M	0x1FFF_FFFF
proces...	S_AXI_ACP	ACP_QSPI...	0xFC00_0000	16M	0xFCFF_FFFF
Excluded Address Segments (2)					
proc...	S_AXI_ACP	ACP_IOP	0xE000_0000	4M	0xE03F_FFFF
proc...	S_AXI_ACP	ACP_M_A...	0x4000_0000	1G	0x7FFF_FFFF
processing_system7_0					
Data (32 address bits : 0x40000000 [1G])					
axi_dm...	S_AXI_LITE	Reg	0x4040_0000	64K	0x4040_FFFF
axi_ast...	S_AXI	S_AXI_RE...	0x43C0_0000	64K	0x43C0_FFFF
axi_gpi...	S_AXI	Reg	0x4120_0000	64K	0x4120_FFFF
axi_ad...	S_AXI	S_AXI_RE...	0x43C1_0000	64K	0x43C1_FFFF

Figura 48: Asistente para asignación de bancos de memoria de Vivado.

A continuación, se lleva a cabo la síntesis. En Vivado, la síntesis es el proceso en el cual un diseño hardware, modelado con un lenguaje de alto nivel como VHDL, pasa a ser un diseño que puede ser implementado en un dispositivo destino.

Durante el proceso de síntesis, Vivado utiliza la descripción RTL (Register Transfer Level) del diseño y las restricciones proporcionadas por el usuario para asignar el diseño al dispositivo de destino. El resultado del proceso es una lista de conexiones que describe el diseño en términos de puertas lógicas y otros componentes disponibles del dispositivo.

El paso intermedio entre la síntesis y la implementación es la elaboración, durante la cual se toma la descripción RTL (Register Transfer Level) del diseño, junto con las restricciones y los núcleos IP para crear una lista de conexiones que se utiliza para el posterior análisis e implementación.

A continuación, se llevará a cabo el proceso de implementación, cuya tarea principal es la de establecer la ubicación de los elementos de la lógica creada por el usuario en el espacio físico de la FPGA. Primero se lleva a cabo una pre-ubicación durante la cual se verifica que las conexiones sean ruteables, las restricciones físicas sean válidas y que no se lleve a cabo una sobreutilización de recursos. Acto seguido, se ubican los interfaces I/O y relojes, después las macros y demás bloques básicos y, por último, los flip-flops y LUTs. En la Figura 49, se puede observar una representación del diseño implementado en la placa.

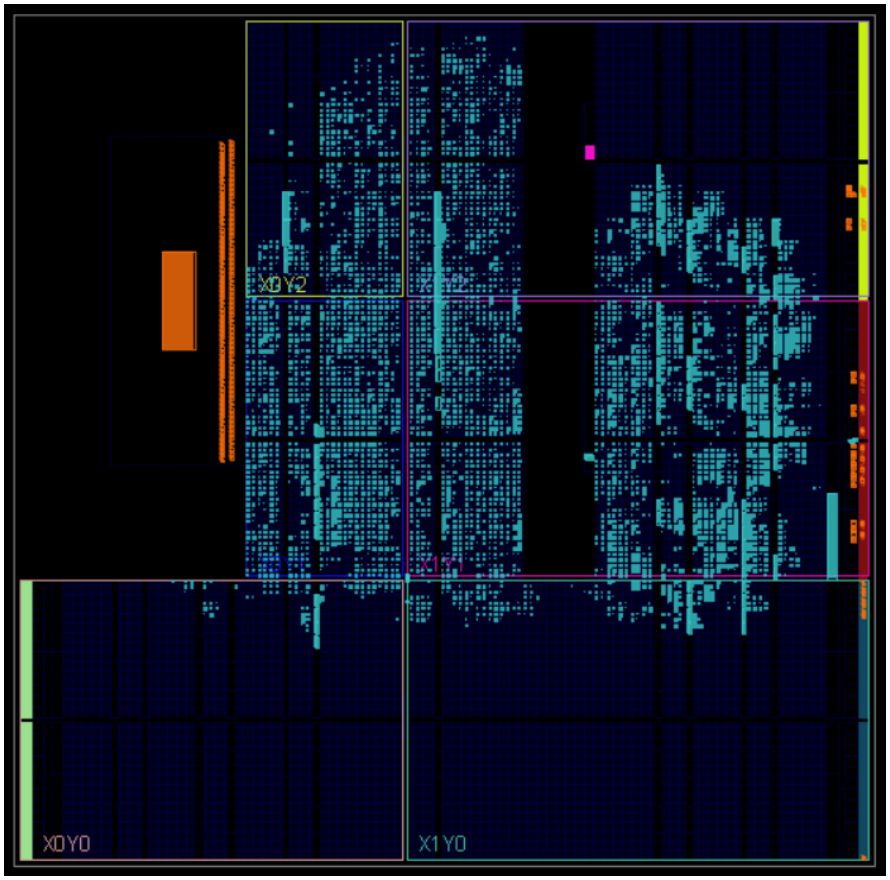


Figura 49: Representación del modelo implementado del sistema.

Una vez se ha completado el proceso de colocación y enrutamiento de la implementación, se lleva a cabo el análisis de restricciones de tiempo. Durante este proceso, Vivado analiza el diseño para garantizar que cumpla con los requisitos de tiempo especificados.

Por último, se lleva a cabo una optimización de consumo. Una de sus tareas principales es la de las habilitaciones de reloj (clock gating), con la que se alcanzan reducciones del consumo dinámico de hasta un 30%. En la Figura 50, se puede observar un gráfico con la optimización de consumo.

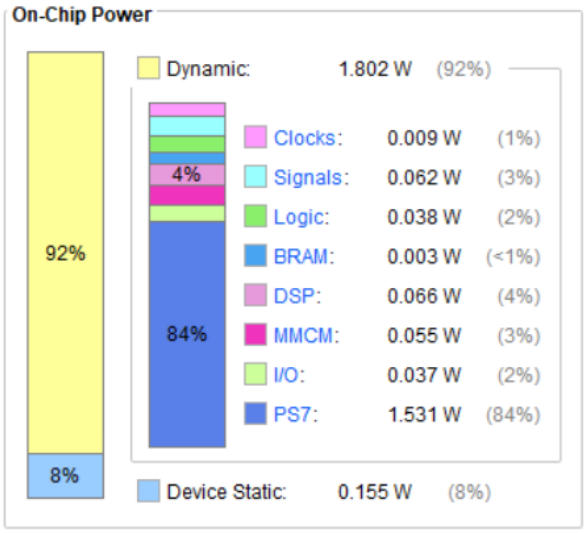


Figura 50: Gráfico de la optimización de consumo del sistema.

El último paso antes de exportar el hardware a la aplicación software, es la generación del archivo bitstream. La función Generar Bitstream crea el archivo tomando el diseño implementado de Vivado junto con las restricciones y configuraciones específicas. El bitstream contiene los datos de configuración para todos los recursos lógicos programables del diseño, tales como LUTs, flip-flops... Una vez que se genera el archivo, se puede cargar en el hardware de destino, configurando los PLD para implementar el diseño.

4.5 Desarrollo Software

Para llevar a cabo el desarrollo de la aplicación software se va a hacer uso del entorno de desarrollo Xilinx SDK (Software Development Kit). A continuación, se procede a exponer, en orden de ejecución, el código empleado para inicializar el sistema:

1. Init_platform: Función de que realiza la configuración inicial de los cores del sistema. Contiene las funciones:
    a) Setup_fmc_adc: Función de inicialización de la interfaz del ADC. Tras llevar a cabo el proceso de configuración e inicialización de la entidad (función adc_fmc_initialize), se procede a realizar la calibración de los datos provenientes del ADC (función adc_calibrate). Para ello, se va a seleccionar el modo de test Midscale Short (accediendo al registro TEST_IO_REG_IDX), como se expuso anteriormente. A continuación, se inicia una transmisión de datos, se resetea el componente IDELAYE_2 y se accede a un bucle que, en cada iteración, toma los dos bits más significativos de la trama de datos actual y los compara con los de la trama anterior. Si el valor de las tramas no es el mismo, se aplicará un incremento (accediendo a ADC_CALIB_REG_IDX) y se procesará otra iteración del bucle hasta que el valor coincida y sea igual a "10". Esto indica que se ha superado la zona de skew y se ha alcanzado el primer borde de la curva de bañera. Para localizar el segundo borde se aplicarán incrementos en la trama de datos hasta que el valor de las tramas no coincida y sea distinto de "10". Una vez obtenidos los dos bordes se calculará el centro de temporización ideal calculando su punto medio.

    b) Setup_fft: Función de inicialización del bloque FFT. En primer lugar, se llevan a cabo las funciones de configuración e inicialización del hardware de la interfaz. A continuación, se resetea el core y se definen los parámetros de n_points (número de puntos de la señal de entrada que se transformarán, debe ser una potencia de 2, y cuanto mayor sea el valor, más precisa será la FFT, aunque también dará como resultado un tiempo de cálculo más largo) y packet_len (número de muestras en un solo cálculo de FFT, a mayor sea la longitud del paquete, mayor será la resolución de frecuencia y menor será la resolución temporal).

2. Init_interrupts: Función que inicializa las interrupciones del sistema. Se ejecuta después de la configuración inicial de los componentes y dispone de las siguientes funciones:
    a) fft_conf_interrupts: Función que inicializa las interrupciones del bloque FFT. En primer lugar, se limpia el bit EOI (End Of Interrupt) del GIC (Generic Interrupt Controller), lo cual permite que este último pase a la siguiente interrupción y la maneje, asegurando que el sistema pueda continuar operando eficientemente y respondiendo a nuevas interrupciones de manera oportuna. A continuación, se asocia la interrupción en cuestión con el handler "fft_handler", el cual limpia los flags de interrupción. El último paso es habilitar la interrupción por final de paquete usando la máscara FFT_EOP_INT_MASK.
    b) adc_fmc_conf_interrupts: Función que inicializa las interrupciones de la interfaz del ADC. En primer lugar, se limpia el bit EOI, lo cual va seguido de la conexión de la interrupción de la interfaz con el handler "adc_fmc_handler", que limpia los flags de interrupción. El último paso es habilitar la interrupción de la interfaz.
    c) xdma_conf_interrupts: Función que inicializa las interrupciones del controlador DMA. En primer lugar, se limpia el bit EOI, lo cual va seguido de la conexión de la interrupción de la interfaz con el handler "axidma_rcv_handler", que libera y reconfigura los descriptores de búfer del anillo para llevar a cabo una nueva escritura en memoria.

3. Fft_execute: Función de ejecución del bloque FFT. Genera el comando de procesamiento con los datos referentes a número de tramas (número de tramas utilizadas en un cálculo de FFT, donde cada trama representa un segmento de la señal de entrada que se analiza por separado) y factor de diezmado (especifica el factor por el cual se reduce la frecuencia de muestreo de una señal).

4.6 Consumo de recursos en el sistema

En este apartado se dispone a exponer el consumo de recursos de la Zedboard que efectúa el sistema diseñado, así como el consumo de recursos de cada elemento con respecto al propio sistema. Los recursos a exponer serán: LUT, Flip-Flop, Slice, Block RAM y DSP Slice.

En primer lugar, la Tabla 1 muestra los datos sobre consumo de recursos del diseño frente a la totalidad de recursos disponibles en Zedboard, así como el tanto por ciento de consumo de cada recurso por el sistema.

Recurso	Consumo del sistema	Recursos disponibles en Zedboard	% de consumo
LUT	10792	53200	20,3
Flip-Flop	16669	106400	15,7
Slice	4648	13300	34,9
Block RAM	27	140	19,3
DSP	38	220	17,3
IOB	30	200	15

Tabla 1: Tabla de datos de consumo del sistema frente a la totalidad de recursos disponibles en Zedboard.

Para mostrar de una manera más esclarecedora los datos de consumo de recursos se ha elaborado una gráfica que se muestra en la Figura 52.

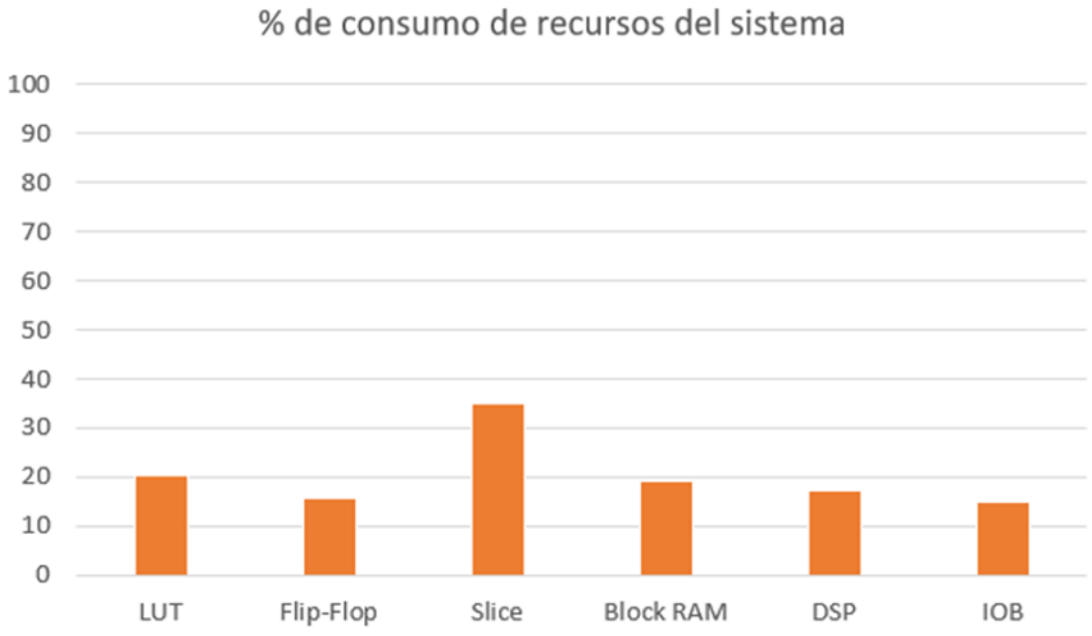


Figura 51: Gráfica del porcentaje de consumo de recursos del sistema.

Se puede apreciar que el consumo de recursos del sistema no es muy elevado y no supera en ningún caso el 50%. A continuación, se procederá a exponer el consumo de recursos del sistema por parte del diseño RTL frente al del debug hub. El debug hub es un centro de depuración integrado en las FPGA de Xilinx, que proporciona funcionalidades como depuración JTAG y analizador lógico. La Tabla 2 muestra la comparación de consumo de recursos del sistema, por las partes del mismo mencionadas anteriormente.

Recurso	Consumo del diseño RTL	Consumo del debug hub	Recursos disponibles en el sistema	% de consumo del diseño RTL	% de consumo de dbg_hub
LUT	10335	457	10792	95,8	4,2
Flip-Flop	15947	722	16669	95,7	4,3
Slice	4403	245	4648	94,7	5,3
Block RAM	27	0	27	100	0
DSP	38	0	38	100	0
IOB	30	0	30	100	0

Tabla 2: Tabla de consumo de recursos del sistema por el diseño RTL frente al del debug hub.

Al igual que antes, se ha elaborado una gráfica para comparar de manera visual los resultados de consumo que se muestra en la Figura 52.

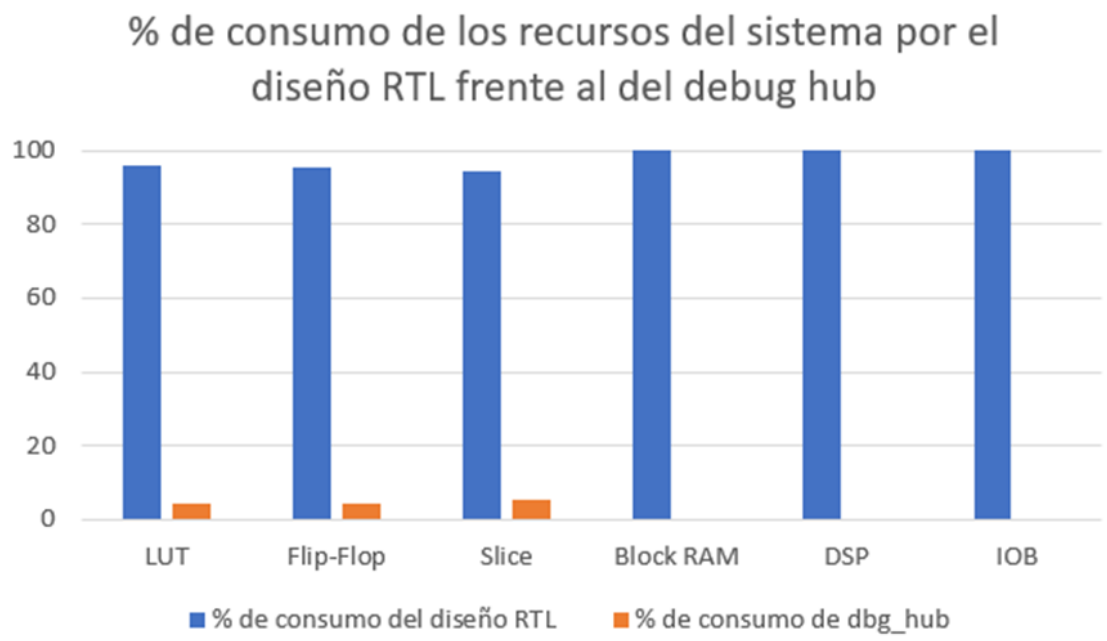


Figura 52: Gráfica de procentaje de consumo de recursos del sistema por el diseño RTL frente al del debug hub.

Se puede apreciar que el diseño RTL consume la mayor parte de los recursos del sistema. Por ello, se va a proceder a analizar el consumo de recursos de cada uno de los elementos del diseño para visualizar como está repartido. En las Tablas 3, 4, 5, 6, 7, 8, 9 y 10 se muestra el consumo de recursos por parte de la interfaz del ADC, el bloque generador de FFT, el controlador de DMA, el bloque GPIO, el bloque "Smart Connect", el bloque interconexión estándar, el generador de reset y el ILA, respectivamente.

Recurso	Consumo de axi_adc_controller	Recursos disponibles en el diseño RTL	% consumo de axi_adc_controller
LUT	136	10335	1,3
Flip-Flop	312	15947	2
Slice	98	4403	2,2
Block RAM	0	27	0
DSP	0	38	0

Tabla 3: Tabla de consumo de recursos del sistema por axi_adc_controller.

Recurso	Consumo de sys_FFT	Recursos disponibles en el diseño RTL	% consumo de sys_FFT
LUT	4127	10335	39,9
Flip-Flop	7215	15947	45,2
Slice	1646	4403	37,4
Block RAM	21	27	77,8
DSP	38	38	100

Tabla 4: Tabla de consumo de recursos del sistema por sys_FFT.

Recurso	Consumo de axi_dma	Recursos disponibles en el diseño RTL	% consumo de axi_dma
LUT	1436	10335	13,9
Flip-Flop	2811	15947	17,6
Slice	791	4403	18
Block RAM	1,5	27	5,6
DSP	0	38	0

Tabla 5: Tabla de consumo de recursos del sistema por axi_dma.

Recurso	Consumo de axi_gpio	Recursos disponibles en el diseño RTL	% consumo de axi_gpio
LUT	41	10335	0,4
Flip-Flop	68	15947	0,4
Slice	21	4403	0,5
Block RAM	0	27	0
DSP	0	38	0

Tabla 6: Tabla de consumo de recursos del sistema por axi_gpio.

Recurso	Consumo de axi_smc	Recursos disponibles en el diseño RTL	% consumo de axi_smc
LUT	2872	10335	27,8
Flip-Flop	2855	15947	17,9
Slice	1073	4403	24,4
Block RAM	0	27	0
DSP	0	38	0

Tabla 7: Tabla de consumo de recursos del sistema por axi_smc.

Recurso	Consumo de axi interconnect	Recursos disponibles en el diseño RTL	% consumo de axi interconnect
LUT	579	10335	5,6
Flip-Flop	707	15947	4,4
Slice	264	4403	6
Block RAM	0	27	0
DSP	0	38	0

Tabla 8: Tabla de consumo de recursos del sistema por axi interconnect.

Recurso	Consumo de processor system reset	Recursos disponibles en el diseño RTL	% consumo de processor system reset
LUT	18	10335	0,2
Flip-Flop	37	15947	0,2
Slice	9	4403	0,2
Block RAM	0	27	0
DSP	0	38	0

Tabla 9: Tabla de consumo de recursos del sistema por processor system reset.

Recurso	Consumo de ILA	Recursos disponibles en el diseño RTL	% consumo de ILA
LUT	1126	10335	10,9
Flip-Flop	1942	15947	12,2
Slice	650	4403	14,8
Block RAM	4,5	27	16,7
DSP	0	38	0

Tabla 10: Tabla de consumo de recursos del sistema por el ILA.

Por último, la Figura 53 muestra una gráfica que compara el consumo de recursos de cada uno de los componentes del diseño. Se puede observar que el bloque generador de FFT es el que más recursos consume. Esto se debe a que la implementación hardware de algoritmos de generación FFT precisan de una gran cantidad de interconexiones para llevar a cabo las operaciones lógicas pertinentes, como las operaciones mariposa.

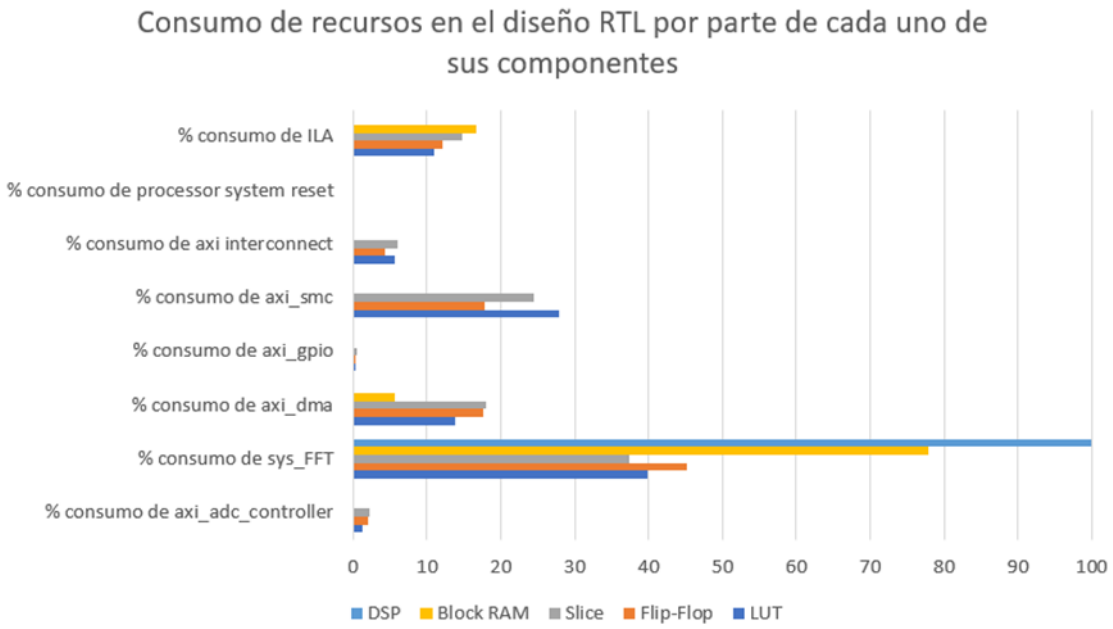


Figura 53: Gráfica de consumo de recursos en el diseño RTL por parte de cada uno de sus componentes.

4.7 Resultados

Con la finalidad de comprobar el correcto funcionamiento del sistema, se ha introducido una señal analógica de frecuencia constante e igual a 1,5 MHz. Dicha señal es mostrada en la Figura 54. Los datos de la transformada de Fourier se han extraído del búfer de recepción para almacenarlos en un archivo .bin. Este procedimiento se ha llevado a cabo haciendo uso del comando TCL mrd (memory read), el cual lee datos con tamaño definido y desde la dirección de memoria especificada por el usuario, y los escribe en formato binario en el archivo .bin indicado.

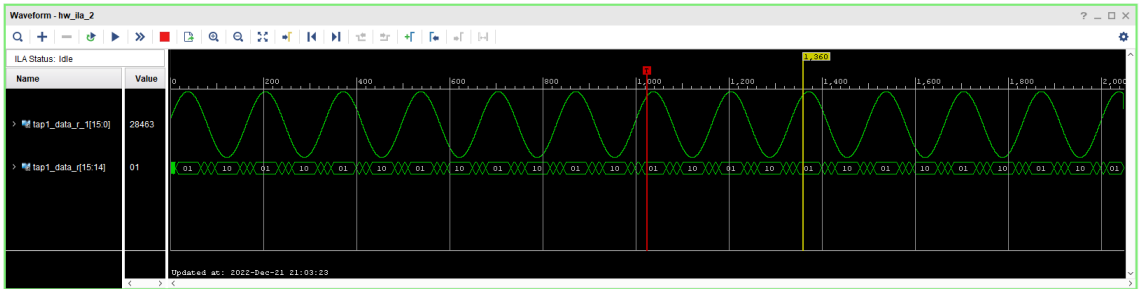


Figura 54: Señal aplicada para el test de frecuencia constante.

Una vez almacenados en un archivo binario, se accede a los datos desde MATLAB y se procesan. El procesamiento consiste en subdividir el array de datos en dos arrays, uno con los números reales y otro con los complejos. Con estos arrays, se aplicará la transformada de Fourier de los datos y se representará como el espectro de amplitud en función de la frecuencia. La Figura 55 muestra el resultado de la gráfica obtenida en MATLAB.

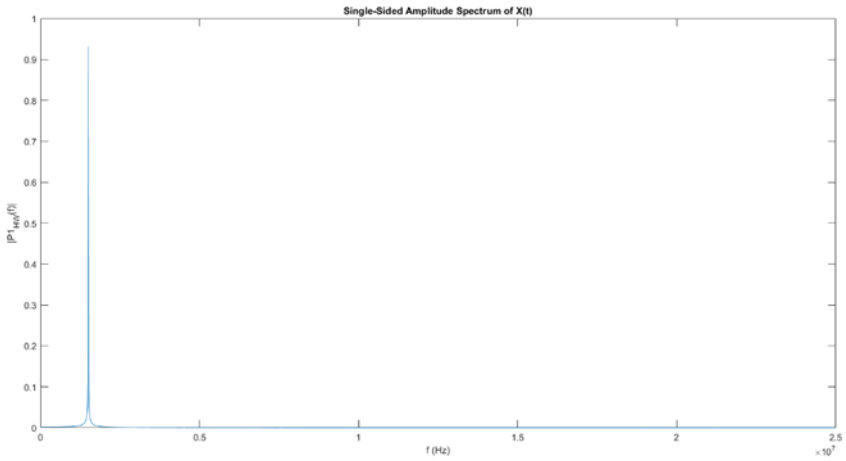


Figura 55: Representación de la FFT de los datos realizada en Matlab.

Conclusiones y líneas futuras de investigación

Conclusiones

Este proyecto ha supuesto un estudio a fondo de la arquitectura y la teoría de operación de los SoC basados en FPGA, así como del desarrollo hardware y software de aplicaciones en este tipo de dispositivos.

Uno de los principales problemas que puede surgir a la hora de crear sistemas cuyo funcionamiento depende de componentes conectados a la placa (como es el caso que nos envuelve con la tarjeta FMC-AD9467-250EBZ) es crear una interfaz capaz de enlazar el citado componente con la tarjeta de evaluación. Para comprobar el correcto funcionamiento de esta interfaz es muy recomendable realizar un modelo BFM, con el que se pueden llevar a cabo simulaciones que no pongan en riesgo el hardware de la tarjeta y que ayude a identificar problemas de una manera más dinámica.

En el proceso de captura de datos de los sistemas de adquisición de altas prestaciones surge otro problema, ya que los datos deben experimentar un proceso de calibración para una adquisición sin errores. La técnica llevada a cabo para realizar esta calibración ha supuesto otro importante punto de aprendizaje, la cual se fundamenta en el uso de un elemento retardador.

Llevar a cabo el desarrollo de la Transformada de Fourier en la PL de la FPGA puede consumir una cantidad muy elevada de recursos. Además, para una entrada de 16 bits proporciona una salida de 64 bits, lo que implica un aumento más que considerable de los datos en cada muestra procesada.

El almacenamiento de un conjunto de datos procesados en memoria puede llegar a ser un procedimiento bastante tedioso, sobre todo en sistemas de adquisición en tiempo real, ya que la memoria es limitada y el flujo de datos es abundante y constante. El problema surge porque a la hora de almacenar datos en memoria se precisa de grandes espacios de memoria alineada si no se enfoca la cuestión de la manera adecuada. La técnica de almacenamiento de datos en memoria basada en el uso de anillos de descriptores, cuyas direcciones de BDs y búfers de recepción se van actualizando en cada ciclo permite hacer uso de espacios de memoria alineada pequeños, aprovechando al máximo la memoria disponible.

En conclusión, en su diseño los sistemas de adquisición de señales de altas prestaciones presentan una serie de técnicas que hay que conocer para abordarlos con éxito. De no ser así, las demandas de adquisición en tiempo real no podrán ser solventadas.

Líneas futuras de investigación

El trabajo propuesto es un backend de recepción radioastronómica destinado a observaciones de ocultaciones lunares, pero el diseño presenta una gran escalabilidad debido a que su arquitectura es común en todos los backends destinados a estos fines. Por lo tanto, variando algunas características del diseño, este puede amoldarse a otras frecuencias de muestreo y ser empleado en otro tipo de observaciones radioastronómicas.

Una importante mejora sería la implementación en el diseño de un enlace Ethernet, que conectado al puerto Gigabit Ethernet de la placa sea capaz de transmitir los datos procesados a un host. En dicho host se podrían visualizar los datos en tiempo real.

En cuanto a la tarjeta de evaluación escogida, Zedboard es una placa de desarrollo basada en el Zynq-700 AP SoC. Esta placa tiene prestaciones limitadas, pero Zynq provee de una gran variedad de productos de mayores prestaciones como UltraScale+, el cual integra la última tecnología Xilinx UltraScale+ FPGA con procesadores ARM Cortex-A53. Migrando el proyecto a una tarjeta de mayores prestaciones, las prestaciones del sistema también incrementarían, ya que las velocidades de procesamiento serían mayores debido a la mejora del PS, y la mejora de la PL permitiría implementar algoritmos de Transformada Rápida de Fourier más complejos.

Bibliografía

[1] “Desarrollo de las herramientas CASPER para diseño de instrumentación radioastronómica de alto rendimiento.” – Universidad Nacional de La Plata. Informe Final 2020. Autores: Alan Szeinfeld, Ariel Saidman. Directores: Guillermo Gancio, Leandro García.

[2] “Receptores en las estaciones de VLBI mundial. Estado actual y nueva generación de receptores VLBI en el CAY” - Informe Técnico IT-OAN 2005-10. J.M.Serna Puente, J.A.López Fernández. Fuente: https://icts-yebes.oan.es/report

[3] T. L Wilson, K. Rohlfs, S. Hüttemeister. “Tools of Radioastronomy”. ISBN: 978-3-540-85121-9, Fifth Edition, Ed. Springer, 2009.

[4] IGN Astronomía. [En línea]. Instituto Geográfico Nacional interferometría de muy larga bases Disponible en: http://astronomia.ign.es/interferometria-de-muy-larga-base [Consulta: 7 de Septiembre de 2022].

[5] UNIDAD DIDÁCTICA OCULTACIONES – Instituto de Astrofísica de Canarias. Juan Carlos Casado, Miquel Serra-Ricart, Luis Cuesta, 2004, ISBN: 84-688-6966-X. Edita: Gabinete de Dirección del IAC.

[6] ZedBoard (Zynq™Evaluation and Development) Hardware User’s Guide - Version 2.2. 27 January 2014.

[7] The Zynq Book. Embedded Processing with the ARM Cortex-A9 on the Xilinx Zynq7000 All Programmable SoC – 1st edition. Louise H. Crockett, Ross A. Elliot, Martin A. Enderwitz, Robert W. Stewart, Strathclyde Academic Media, ISBN-13: 978-0992978709, julio 2014.

[8] AMBA AXI-Stream Protocol Specification, ARM, Issue B, 9 April 2021.

[9] Vivado Design Suite 7 Series FPGA and Zynq-7000 SoC Libraries Guide (UG953).

[10] AXI DMA v7.1 LogiCORE IP Product Guide (PG021).

[11] Understanding the Gigabit Ethernet Controller's DMA on ZYNQ Devices. [En línea] https://igorfreire.com.br/2016/12/28/understanding-gigabit-ethernet-controllers-dma-zynq-devices/#Buffers_and_Buffer_Descriptors_in_the_GEM [Consulta: 12 de Noviembre de 2022].

[12] Processor System Reset Module v5.0 LogiCORE IP Product Guide (PG164).

[13] Understanding Flash ADCs. [En línea] Disponible en: https://www.analog.com/en/technical-articles/understanding-flash-adcs.html [Consulta: 3 de Diciembre de 2022].

[14] "FPGAs World Class Designs" - Clive "Max" Maxfield, ISBN: 978-1-85617-621-7, Newnes, 2009.

[15] Observatorio de Yebes. [En línea] https://astronomia.ign.es/icts-yebes/acercade [Consulta: 12 de Enero de 2022].

[16] AD9467 16-Bit, 200 MSPS/250 MSPS Analog-to-Digital Converter. [En línea]. Disponible en: https://www.analog.com/media/en/technical-documentation/data-sheets/AD9467.pdf [Consulta: 22 de Enero de 2022].

[17] AD9517 12-Output Clock Generator with Integrated 1.6 GHz VCO. [En línea]. Disponible en: https://pdf1.alldatasheet.com/datasheet-pdf/view/516304/AD/AD9517-4.html [Consulta: 22 de Enero de 2022].

[18] FMC Schematics [En línea]. Disponible en: https://wiki.analog.com/_media/resources/fpga/xilinx/fmc/9467fmc01c_sch.pdf [Consulta: 22 de Enero de 2022].

[19] AN-877 APPLICATION NOTE – Interfacing to High Speed ADCs via SPI [En línea]. Disponible en: https://www.analog.com/media/en/technical-documentation/application-notes/AN-877.pdf [Consulta: 1 de Febrero de 2022].

[20] Xilinx UG070 Virtex-4 FPGA User Guide, User Guide, v2.6, December 1, 2008.

ESCUELA POLITECNICA
SUPERIOR