

Trabajo Fin de Grado

Ingeniería Electrónica, Robótica y Mecatrónica

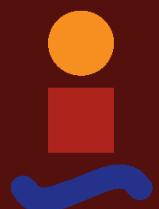
Autor: Miguel Ángel Lara Guarino

Tutores: Fernando Muñoz Chavero

José María Hinojo Montero

Dpto. de Ingeniería Electrónica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2024



Proyecto Fin de Carrera
Ingeniería Electrónica, Robótica y Mecatrónica

Autor:

Miguel Ángel Lara Guarino

Tutores:

Fernando Muñoz Chavero
Catedrático de Universidad

José María Hinojo Montero
Investigador Post-doctoral

Dpto. de ingeniería Electrónica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2024

Trabajo Fin de Grado: Diseño e implementación de un sistema de adquisición de datos para un sensor óptico.

Autor: Miguel Ángel Lara Guarino

Tutores: Fernando Muñoz Chavero

José María Hinojo Montero

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2024

El secretario del Tribunal

Agradecimientos

Me gustaría comenzar expresando mi más sincero agradecimiento a Enrique López y Fernando Muñoz, quienes despertaron en mí una pasión por las FPGAs durante mi tercer año de carrera. En especial, quisiera agradecer a Fernando por brindarme la oportunidad de adentrarme en el mundo de la electrónica profesionalmente y hacer posibles estos seis meses de prácticas. Agradezco también a José María Hinojo por el tiempo dedicado durante mi estancia de prácticas; le estoy profundamente agradecido por todo lo que he aprendido de él, tanto a nivel laboral como personal.

También quiero expresar mi gratitud a los compañeros con quienes he tenido la fortuna de compartir el día a día: Mario Palmero, José Ignacio, Paula González, Katherine Alexandra Guerrero, Mario Cruz y Jorge Jiménez. Son personas extraordinarias de las que he aprendido mucho y siempre llevaré en mi memoria. En particular, quiero destacar a Jorge Jiménez y Mario Cruz, de quienes he aprendido muchísimo en el ámbito de la electrónica. También me gustaría resaltar a José Ignacio, con quien compartí gran parte de mi tiempo como becario y de quien me llevo una entrañable amistad que recordaré siempre con cariño. Todos han sido como una familia para mí.

A nivel académico, me gustaría agradecer a todos mis compañeros que me han acompañado diariamente en las clases y en las prácticas de laboratorio. En especial, quiero destacar a mi amigo Luis Páez, por toda la ayuda que siempre me ha brindado y con quien he tenido la suerte de enfrentar la carrera.

Finalmente, en el ámbito personal, me gustaría dar las gracias a mi familia por todo el apoyo incondicional que me habéis dado a lo largo de mi carrera. Tampoco quiero olvidar a mi pareja Alba Vega, quien ha sido un pilar fundamental en mi vida, ayudándome a enfrentar y superar las situaciones más difíciles.

Gracias a todos por la ayuda que he recibido, porque es gracias a ello que este trabajo ha sido posible y del que me llevo un valioso conocimiento tanto laboral como personal.

Miguel Ángel Lara Guarino

Sevilla, 2024

Resume

Dada la complejidad del manejo de los datos en tiempo real presentada por los sensores ópticos, en este trabajo se desarrolla el diseño e implementación de un sistema de adquisición completo que permite configurar, extraer y almacenar en un fichero la información generada. Dicho sistema será desarrollado mediante lenguaje de descripción Hardware. Además, se incorpora una aplicación para permitir el control y tratamiento de los datos por parte del usuario de manera eficiente. Esto permite realizar la caracterización del sensor y asegurar que cumpla con los requisitos del entorno. Finalmente, es importante mencionar que este Trabajo Fin de Grado se ha llevado a cabo dentro del contexto de un proyecto de investigación junto a la compañía Alter Technology TÜV Nord. El objetivo de dicho proyecto es el desarrollo de una plataforma modular para la caracterización de sensores de imagen.

Abstract

Given the complexity of handling real-time data presented by optical sensors, this work develops the design and implementation of a complete acquisition system that allows for the configuration, extraction, and storage of the information generated in a file. This system will be developed using Hardware Description Language. Additionally, an application is incorporated to allow efficient control and processing of the data by the user. This enables the characterization of the sensor and ensures it meets the environmental requirements. Finally, it is important to mention that this Final Degree Project has been carried out within the context of a research project in collaboration with the company Alter Technology TÜV Nord. The objective of this project is the development of a modular platform for the characterization of image sensors.

Índice

Agradecimientos	7
Resumé	9
Abstract	11
Índice	13
Índice de Tablas	15
Índice de Figuras	17
Notación	19
1 Introducción	1
1.1 Objetivos	2
1.2 Estructura del trabajo	2
2 Descripción de la plataforma	3
2.1 Kit de Desarrollo ZC702	3
2.1.1 Zynq 7000	4
2.2 Sensor	5
2.2.1 Funcionamiento teórico	5
2.2.2 Configuración	8
2.2.3 Operación	9
2.3 ADC	11
2.3.1 Configuración	12
2.3.2 Operación	15
2.4 Tarjeta de adaptación	16
3 Arquitectura Hardware	19
3.1 DMA	20
3.2 FIFO data stream	22
3.3 IP PICO	23
3.3.1 Registros	25
3.3.2 Descripción de la entidad	25
3.4 IP DAQ_ADC3244	27
3.4.1 Registros	29
3.4.2 Descripción de la entidad	30
3.4.3 Sub-bloques. AXI Master	33
3.4.4 Sub-bloques. SPI	33
4 Aplicación software	35
4.1 PetaLinux	35
4.2 Módulo del kernel para la DMA	37
4.3 Aplicación en C	37
4.3.1 Estructura de la aplicación	38
5 Validación de la Plataforma de Adquisición Implementada	47
5.1 Prueba 1: Comprobación de la descripción VHDL desarrollada por medio de un testbench	47

5.2	Prueba 2: Lectura/Escritura de los registros del ADC3244	52
5.3	Prueba 3: Lectura de tramas personalizadas del ADC3244	53
5.4	Prueba 4: Uso de la DMA para la lectura de un contador programado en el ADC3244	54
5.5	Prueba 5: Aplicación de la configuración al PICO1024	55
5.6	Prueba 6: Señal sinusoidal a la entrada del ADC3244 para verificar la etapa de acondicionamiento	
	56	
6	Conclusiones	57
6.1	Líneas futuras de trabajo	57
Bibliografía		11

ÍNDICE DE TABLAS

Tabla 2.1: Características del kit de desarrollo ZC702 [3].	4
Tabla 2.2: Características eléctricas principales del PICO1024 Gen2 [2].	5
Tabla 2.3: PICO1024. Características y descripción de sus terminales.	6
Tabla 2.4: Trama de configuración para el PICO1024 [2].	9
Tabla 2.5: Características del ADC3244.	11
Tabla 2.6: Características de los pines del ADC3244 [4].	12
Tabla 2.7: ADC3244. Registros internos.	14
Tabla 2.8: ADC3244. Descripción de parámetros configurables.	15
Tabla 3.1: Mapa de memoria.	19
Tabla 3.2: DMA. Registro de control.	21
Tabla 3.3: Descripción de los puertos del PICO IP.	24
Tabla 3.4: Descripción de los registros del IP PICO.	25
Tabla 3.5: DAQ_ADC3244. Descripción de los puertos.	29
Tabla 3.6: DAQ_ADC3244. Descripción de los registros.	29
Tabla 4.1: Petalinux. Comandos para configurar.	35
Tabla 4.2: Petalinux. Configuración global.	35
Tabla 4.3: Petalinux. Configuración del kernel.	36
Tabla 4.4: Petalinux. Configuración del rootfs.	36
Tabla 5.1: Descripción del banco de pruebas.	47

ÍNDICE DE FIGURAS

Figura 1.1: Esquema básico de un sistema de adquisición.	1
Figura 2.1: Diagrama de interconexión del sistema de prueba implementado.	3
Figura 2.2: Imagen del kit de desarrollo ZC702 [3].	3
Figura 2.3: Imagen del PICO1024 [2].	5
Figura 2.4: PICO1024. Diagrama que representa la estructura del sensor [2].	7
Figura 2.5: PICO1024. Estructura esquemática de un píxel [2].	7
Figura 2.6: PICO1024. Diagrama interno del ROIC [2].	8
Figura 2.7: PICO1024. Cronograma para modo de compatibilidad UL05251 [2].	8
Figura 2.8: PICO1024. Cronograma para configuración nominal [2].	9
Figura 2.10: PICO1024. Periodo completo de la señal INT, Ecuación 2.1.	10
Figura 2.11: PICO1024. Modo de operación [2].	10
Figura 2.12: PICO1024. Señal de muestreo [2].	10
Figura 2.13: PICO1024. Secuencia de encendido, configuración y apagado del sensor [2].	11
Figura 2.14: ADC3244. Vista Frontal [4].	12
Figura 2.15: ADC3244. Escritura en los registros de configuración [4].	13
Figura 2.16: ADC3244. Lectura en los registros de configuración [4].	13
Figura 2.18: ADC3244. Modo de operación 'One-Wire', Ecuación 2.2. [4].	16
Figura 2.19: ADC3244. Modo de operación 'Two-Wire', Ecuación 2.3. [4].	16
Figura 2.20: Imagen de la placa de adaptación suministrada.	17
Figura 3.1: Esquema General del lenguaje HDL desarrollado.	20
Figura 3.2: DMA. Diagrama interno [5].	20
Figura 3.3: DMA. Configuración de la IP.	22
Figura 3.4: FIFO. Configuración de la IP.	22
Figura 3.5: Señales Entrada/Salida del PICO IP.	23
Figura 3.6: PICO IP. Diagrama de bloques.	23
Figura 3.7: PICO1024. Configuración de la IP.	25
Figura 3.7: PICO_top. Diagrama de bloques.	26
Figura 3.8: PICO_controller. Diagrama de estados.	26
Figura 3.9: DAQ_ADC3244 IP. Puertos de entrada/salida.	27
Figura 3.10: DAQ_ADC3244 IP. Diagrama de bloques.	27
Figura 3.17: ADC3244. Configuración de la IP.	29
Figura 3.11: ADC3244_ACQ. Diagrama de bloques.	30
Figura 3.12: ADC3244_ACQ. Registros par e impar en el proceso de adquisición (Two-Wire).	31

Figura 3.13: ADC3244_ACQ. Registros par e impar en el proceso de adquisición (One-Wire).	31
Figura 3.14: ADC3244_ACQ. Comportamiento de los registros en la adquisición ($F_{clk_i} = '1'$).	32
Figura 3.15: ADC3244_ACQ. Comportamiento de los registros en la adquisición ($F_{clk_i} = '0'$).	32
Figura 3.16: Protocolo de comunicación; DAQ - Axi Stream.	33
Figura 3.17: Protocolo de comunicación; DAQ - AXI Stream. Ejemplo.	33
Figura 3.18: SPI. Diagrama de bloques.	34
Figura 4.1: Modulo Kernel. Esquema conceptual de su utilidad.	37
Figura 4.2: Aplicación. Esquema de ficheros.	37
Figura 4.3: Aplicación. Diagrama de flujo.	39
Figura 4.4: Aplicación. Menú.	39
Figura 4.5: Aplicación. Diagrama de flujo de la tarea READ ADC CONF.	40
Figura 4.6: Aplicación. TEST ADC.	40
Figura 4.7: Aplicación. Diagrama de flujo del TEST ADC.	42
Figura 4.8: Aplicación. TEST PICO1024 SENSOR.	43
Figura 4.9: Aplicación. Cambiar la configuración del PICO1024.	43
Figura 4.10: Aplicación. Diagrama de flujo de la configuración.	43
Figura 4.11: Aplicación. Diagrama de flujo de la prueba PICO1024.	44
Figura 5.1: Diagrama del Testbench.	48
Figura 5.2: Prueba 1. Resultados del ADC3244 IP para Two-Wire.	48
Figura 5.3: Prueba 1. Resultados del ADC3244 IP para One-Wire.	49
Figura 5.4: Prueba 1. Resultados del protocolo de configuración del PICO IP.	49
Figura 5.5: Prueba 1. Resultados del modo de operación del PICO IP.	50
Figura 5.6: Prueba 1. Resultados del stop del PICO IP.	50
Figura 5.7: Prueba 1. Comunicación DMA – FIFO.	51
Figura 5.8: Prueba 1. Comunicación DMA – FIFO (Ampliación).	51
Figura 5.9: Prueba 1. Resultado del envío de paquetes en la DMA IP.	51
Figura 5.10: Prueba 1. Resultados del envío de paquetes en la DMA IP (Un paquete).	51
Figura 5.11: Prueba 1. Resultados del envío de paquetes en la DMA IP (Ampliación).	52
Figura 5.12: Prueba 2. Resultado de la escritura/lectura de registros internos del ADC3244.	52
Figura 5.13: Prueba 3. Resultados del ADC3244 con un patrón constante.	53
Figura 5.14: Prueba 3. Resultados del ADC3244 con un patrón constante (Aplicación).	53
Figura 5.15: Prueba 3. Resultados del ADC3244 con el patrón de contador.	54
Figura 5.16: Prueba 3. Resultados del ADC3244 con el patrón senoidal.	54
Figura 5.17: Prueba 4. Resultados de la DMA con un contador.	55
Figura 5.18: Prueba 4. Resultados de la DMA con un contador (Matlab).	55
Figura 5.19: Prueba 6. Resultado de una señal sinusoidal.	56

Notación

FPGA	Field Programmable Gate Array
PL	Programmable Logic
PS	Processing System
XGA	
SoC	System-on-Chip
ADC	Analog-Digital Converter
FMC	FPGA Mezzanine Card
HDL	Hardware Description Language
PCB	Printed Circuit Board
SPI	Serial Periferal Interface
ROIC	Read Out Integrated Circuit
MC	Master Clock
SC	Sample Clock
FCLK	Frame Clock
DCLK	Data Clock
IP	Intellectual Property
GP	General Purpose (Axi Interface)
HP	High Performance (Axi Interface)
MSPS	Mega Sample Per Second

1 INTRODUCCIÓN

Los sensores de adquisición de imagen y video juegan un papel crucial en el sector espacial, con aplicaciones que abarcan desde la observación remota hasta la seguridad y defensa, [1]. Algunos ejemplos de estas aplicaciones incluyen:

- **Observación remota:** capturar imágenes de alta resolución de la superficie terrestre para monitorear cambios ambientales, evaluar desastres naturales y estudiar la vegetación.
- **Monitorización continua:** proporcionar datos actualizados para aplicaciones como agricultura, seguimiento de incendios forestales y detección de deforestación.
- **Exploración planetaria:** ofrecer imágenes detalladas de la superficie de Marte, la Luna y otros cuerpos celestes, ayudando a los científicos a comprender su geología y atmósfera.
- **Navegación y posicionamiento:** permitir el correcto posicionamiento de los satélites en su órbita esperada.
- **Seguridad y defensa:** detectar movimientos militares o realizar seguimiento de embarcaciones y monitoreo fronteras.

Para lograr satisfacer las necesidades de las aplicaciones descritas anteriormente, los sensores empleados deben presentar una gran resolución (elevado número de píxeles), una alta precisión en la adquisición de la señal y una alta tasa de refresco, que permita capturar eventos cuya duración temporal sea reducida. Esto provoca que, no sólo el propio sensor debe presentar unas altas prestaciones, si no que el resto de los elementos que componen la cadena de adquisición requiere unas prestaciones elevadas.

Esta cadena de adquisición incluye una etapa de acondicionamiento de señal, el conversor analógico-digital, el cual juega un papel vital transformando las señales analógicas del sensor en datos digitales que pueden ser almacenados y, posteriormente, procesados; y el hardware específico para su manejo, configuración y lectura. Este hardware debe asegurar que todas las partes del sistema de adquisición de datos trabajen juntas de manera sincronizada y eficiente para evitar pérdidas de datos o la aparición de errores en la captura de imágenes que inutilicen la información capturada. Además, debe ser capaz de operar en las condiciones extremas del espacio, donde factores como la radiación, la temperatura y la presión pueden afectar significativamente el funcionamiento del equipo. En la **Figura 1.1**, se puede apreciar un esquema básico donde se contemplan las partes mencionadas.

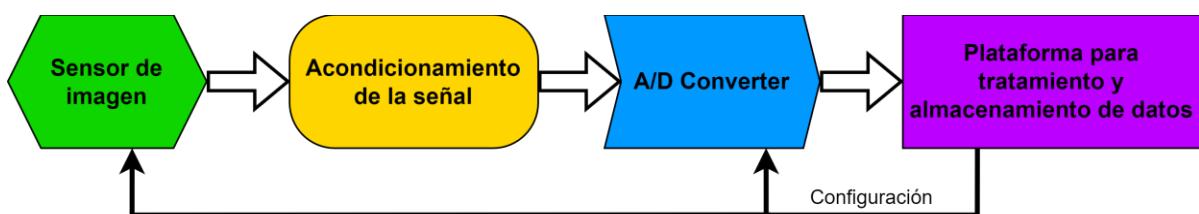


Figura 1.1: Esquema básico de un sistema de adquisición.

Cabe destacar que estos sistemas requieren como paso previo a su integración en el sistema final del diseño de una plataforma de pruebas que permita operar y evaluar el sensor bajo prueba. El diseño de este sistema supone un gran desafío tecnológico que abarca diferentes áreas del diseño electrónico. Por un lado, integrar adecuadamente el sensor con un sistema de control de alta velocidad (normalmente implementado en una FPGA) implica, no sólo manejar eficientemente la interfaz y protocolos de transmisión de datos; sino también, optimizar el procesamiento de datos en tiempo real. Esto es crucial para ejecutar algoritmos de procesamiento

de imagen complejos que aseguren la calidad y funcionalidad de las imágenes obtenidas. Por otro lado, es esencial disponer de un sistema fiable, repetitivo y altamente configurable a la hora de probar los sensores en entornos que recreen las condiciones extremas de operación como son las grandes variaciones de temperatura en cortos períodos de tiempo (estrés térmico) o la radiación, típicas del entorno espacial. Por ello, es necesario un diseño robusto tanto en hardware como en software que permita evaluar la durabilidad y rendimiento del sensor en condiciones reales.

Dada la complejidad a la hora de evaluar toda la cadena de adquisición que requiere la integración de un sensor de imagen, se propone en este Trabajo Fin de Grado la realización de una plataforma modular para el sensor PICO1024. Es importante resaltar el carácter modular de la solución ya que los sensores existentes en el mercado presentan múltiples interfaces. Por tanto, para evitar tener que diseñar un sistema para cada sensor, se va a proponer una arquitectura que independice el control de la interfaz física del sensor de la adquisición y procesamiento de la información.

Por último, cabe indicar que el presente Trabajo Fin de Grado se ha realizado en el marco de un proyecto de investigación con la compañía Alter Technology TÜV Nord.

1.1 Objetivos

Para lograr un sistema de adquisición de datos con la suficiente adaptabilidad para un gran abanico de sensores, se plantearon los siguientes objetivos en este Trabajo Fin de Grado:

- Diseño e implementación de una plataforma modular que permita la rápida y fácil adaptación a nuevos sensores de imagen.
- Desarrollo e implementación de los módulos necesarios, mediante un lenguaje de descripción hardware (HDL), para la configuración y control del sensor de imagen PICO1024 y para la gestión del convertidor analógico-digital.
- Desarrollo y despliegue de un sistema operativo que permita la ejecución de una aplicación de alto nivel de control.
- Desarrollo de una aplicación de control que sea capaz de adquirir y procesar los datos generados por el sensor de imagen. Además, esta herramienta deberá permitir la verificación tanto de la correcta configuración del sensor como la integridad de los datos adquiridos.

1.2 Estructura del trabajo

La tarea desarrollada en el presente Trabajo Fin de Grado consiste tanto en el diseño hardware de la etapa de adquisición, como del software necesario para la configuración de los dispositivos y almacenamiento de los datos. Por ello, la estructura seguida consta de los siguientes puntos:

- **Sección 2. Descripción de la plataforma:** proporciona una explicación detallada de los distintos componentes hardware involucrados en el sistema junto con el procedimiento necesario para la configuración y puesta en marcha de cada uno de ellos.
- **Sección 3. Arquitectura Hardware:** describe el proceso de diseño de los módulos HDL implementados para la configuración y gestión de los componentes detallados en la sección 2.
- **Sección 4. Aplicación software:** se presenta la aplicación desarrollada tanto para la gestión y control del sistema por parte del usuario, como para el posterior tratamiento de los datos adquiridos.
- **Sección 5. Validación de la Plataforma de Adquisición Implementada:** se expone el banco de pruebas realizado y los resultados obtenidos para cada uno de las pruebas realizadas.
- **Sección 6. Conclusiones:** recoge las conclusiones y futuras líneas de investigación del presente trabajo.

2 DESCRIPCIÓN DE LA PLATAFORMA

La plataforma de caracterización que se va a desarrollar para el sensor PICO1024, provisto por el fabricante Lynred [2], se compone del propio sensor óptico, el convertidor Texas Instruments ADC3244, una placa de adaptación para acondicionar la señal entre el sensor óptico y el ADC, desarrollado por la empresa Alter Technology TÜV Nord, y el kit de desarrollo ZC702 de la compañía Xilinx [3].

El sistema quedará como el representado en el diagrama de la **Figura 2.1** en el que dispondremos del kit de desarrollo conectado a la tarjeta de adaptación mediante el conector FMC LPC (FPGA Mezzanine Card Low Pin Count) que presenta la tarjeta ZC702. En dicha tarjeta de adaptación, se incluye el ADC3244, etapa de acondicionamiento y espacio adaptado para la interfaz del sensor PICO1024.

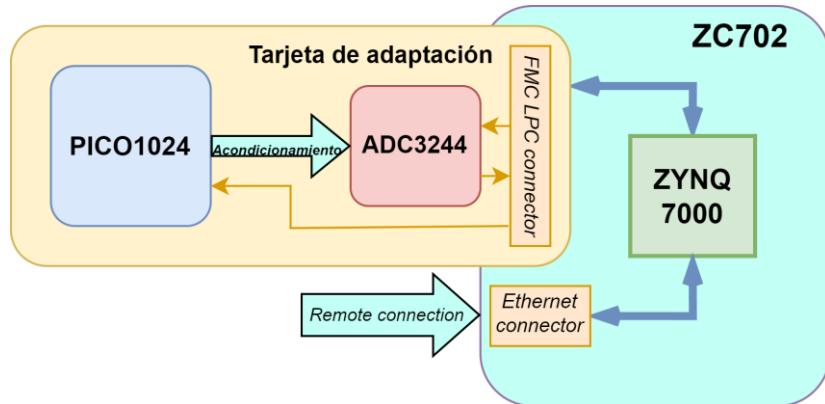


Figura 2.1: Diagrama de interconexión del sistema de prueba implementado.

2.1 Kit de Desarrollo ZC702

Se ha optado por utilizar la tarjeta de pruebas ZC702 fabricada por AMD Xilinx, mostrada en la **Figura 2.2**, [3]. Dicha plataforma permite evaluar y desarrollar aplicaciones empotradas gracias a la inclusión de una FPGA SoC en el chip Zynq-7000. Esta se caracteriza por integrar un procesador ARM Cortex-A9 junto a la lógica programable de la FPGA. Entre sus numerosas características, se puede destacar las descritas en la **Tabla 2.1**.



Figura 2.2: Imagen del kit de desarrollo ZC702 [3].

Características	Descripción
Conectores FPGA Mezzanine Card (FMC)	Facilitan la conexión de alta velocidad con otras tarjetas
Memoria	Incluye 1 GB de memoria DDR3
Conectividad Serial	Conectividad serial a través de USB OTG, UART e IIC
Comunicación	Ofrece conectividad Ethernet 10/100/1000 Mbps
Salida de Video	Implementa aplicaciones de video a través de HDMI
FPGA Z7020-CLG484-1	• Lógica de celdas: 85 K
	• Bloques de RAM: 4.9 Mb
	• Segmentos DSP: 220
	• Pines E/S Máximos: 200

Tabla 2.1: Características del kit de desarrollo ZC702 [3].

2.1.1 Zynq 7000

Un SoC FPGA (System-on-Chip Field Programmable Gate Array) es una tecnología avanzada que combina un procesador y una FPGA en un único dispositivo, permitiendo tanto la flexibilidad de programación de la FPGA como la potencia de procesamiento de un procesador estándar. En este caso particular, utilizamos el modelo ZC702, el cual está compuesto principalmente por una FPGA Z7020-CLG484-1 y un procesador ARM Cortex-A9. Este tipo de configuración resulta ser especialmente útil en aplicaciones que requieren un alto grado de personalización y desempeño computacional.

El lado de la lógica programable, conocido como PL (Programmable Logic), desempeña las funciones críticas dentro de la plataforma de pruebas desarrollada. En este contexto, el PL se encarga de configurar y controlar el sensor y el convertidor analógico-digital (ADC), lo cual es vital para la correcta captura de los datos. Además, el PL gestiona la lógica asociada a la adquisición de datos y su almacenamiento, garantizando que la información recogida por el ADC sea almacenada de forma segura para su posterior análisis y/o transmisión.

Por otro lado, el sistema de procesamiento, o PS (Processing System), ofrece funcionalidades complementarias, pero igualmente críticas. Este componente gestiona el procesamiento de la información recolectada, lo que implica filtrar, analizar y preparar los datos para su uso final o para decisiones en tiempo real. Además, el PS tiene la responsabilidad de transmitir esta información a equipos remotos, facilitando así la comunicación una aplicación de alto nivel. Simultáneamente, el PS supervisa el funcionamiento del PL, asegurando que todo el sistema opere de manera coherente y eficiente. Además, actúa como un intermediario entre el usuario y el hardware integrado en el SoC FPGA.

Esta integración entre el PL y el PS en un sólo chip proporciona una plataforma poderosa y versátil, capaz de adaptarse a una amplia variedad de necesidades tecnológicas y de aplicación. Por tanto, el SoC FPGA lo hace un componente hardware ideal para la aplicación del proyecto llevado a cabo.

2.2 Sensor

El sensor utilizado es el PICO1024 Gen2 del fabricante Lynred. Es un sensor de imagen infrarroja diseñado para entornos exigentes y que ofrece una resolución XGA (Extended Graphics Array) y alto contraste para aplicaciones de detección a larga distancia [2]. Podemos observar las características más destacables en la **Tabla 2.2**. Adicionalmente, se dispone de una imagen física del sensor junto a una descripción de sus terminales en la **Figura 2.3**.

Parámetro	Valor
Sensibilidad Térmica	< 50 mK (f / 1, 300K, 30Hz, FPA 25°C)
Rango de Temperaturas de funcionamiento	[-40°C ; +85°C]
Tasa de fotogramas máxima	120 Hz
Tiempo de constante térmico	< 12 ms
Consumo de energía	< 220 mW
Calificación	Standard MIL810 – MIL883

Tabla 2.2: Características eléctricas principales del PICO1024 Gen2 [2].

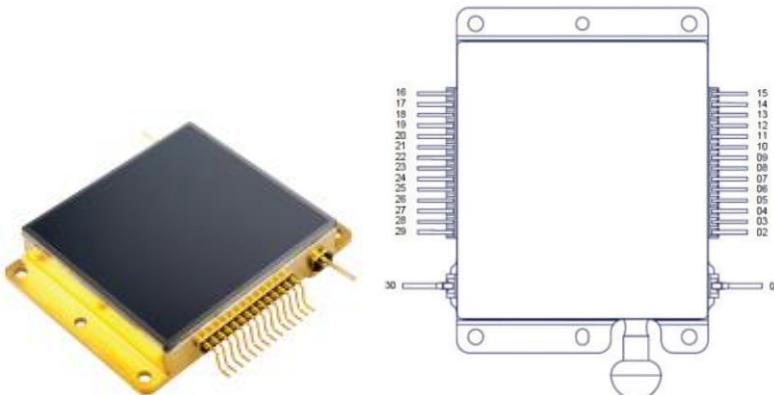


Figura 2.3: Imagen del PICO1024 [2].

A continuación, se expone en la **Tabla 2.3** los pines que componen el PICO1024 junto con el nombre de cada uno de ellos, tipo de señal (analógica, digital o Tierra) y la función que desempeña.

2.2.1 Funcionamiento teórico

2.2.1.1 Principio del sensor infrarrojo

En términos generales, el sensor detecta el incremento de temperatura causado por la radiación infrarroja a través de una membrana térmicamente aislante. Para ello, la membrana, compuesta de un material capaz de absorber la radiación infrarroja, está en contacto directo con un elemento sensor de temperatura. Este último registra el aumento de temperatura provocado por la radiación infrarroja entrante y lo transforma en una señal eléctrica.

El tipo de detección más habitual es mediante un bolímetro resistivo, cuya resistencia varía con los cambios de temperatura. Considerando una matriz bidimensional de estos elementos sensibles (denominados píxeles), el circuito integrado de lectura (ROIC, por sus siglas en inglés) está generalmente diseñado para medir la resistencia de cada bolímetro y compilar los datos en un flujo único de información para aplicaciones de video.

Además, debido a la relación directa entre el aislamiento térmico y la sensibilidad del dispositivo, los sensores IR de alta eficacia se suelen operar en vacío dentro de un empaque especial que incluye una ventana infrarroja, [2].

PIN	SEÑAL	TIPO	FUNCIÓN
1	GETTER	—	—
2	NC	—	—
3	NC	—	—
4	NC	—	—
5	Reservado	—	—
6	NC	—	—
7	Reservado	—	—
8	NC	—	—
9	VSSL	Tierra	Tierra digital
10	VDDL	Entrada analógica	Alimentación digital (3.6 V)
11	NC	—	—
12	SERDAT	Entrada digital	Serial Bus Control
13	RESET	Entrada digital	Sincronización de cuadros
14	MC	Entrada digital	Sincronización de píxeles
15	INT	Entrada digital	Tiempo de Integración
16	OUT1	Salida analógica	Salida de video 1
17	VTEMP	Salida analógica	Temperatura del plano focal
18	OUT3	Salida analógica	Salida de video 3
19	NC	—	—
20	GFID	Entrada analógica	Puerta de transistor del microbolómetro
21	VDDA	Entrada analógica	Alimentación analógica (3.6 V)
22	VSK	Entrada analógica	Comparador del microbolómetro
23	VDET	Entrada analógica	Activa el microbolómetro
24	VSSA	Entrada analógica	Tierra analógica
25	VBUS	Entrada analógica	Referencia CTIA
26	GSK	Entrada analógica	Compensador del microbolómetro
27	OUT2	Salida analógica	Salida de video 3
28	GOC	Entrada analógica	Nivel de continua en el microbolómetro
29	OUT4	Salida analógica	Salida de video 4
30	GETTER	—	—

Tabla 2.3: PICO1024. Características y descripción de sus terminales.

La **Figura 2.4** muestra un diagrama que representa la colocación de los diferentes elementos que componen el sensor empleado.

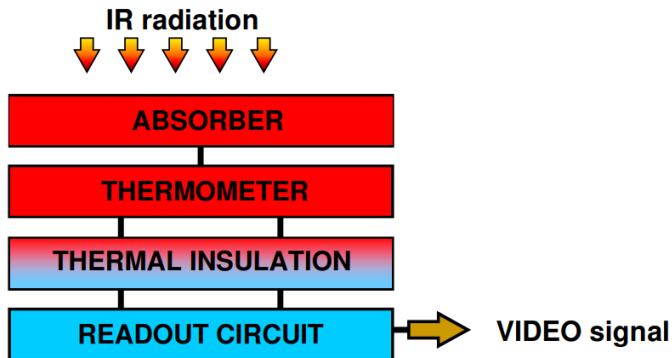


Figura 2.4: PICO1024. Diagrama que representa la estructura del sensor [2].

2.2.1.2 Estructura de un píxel

Cada píxel está eléctricamente conectado a una entrada del Circuito Integrado de Lectura (ROIC, por sus siglas en inglés) que está diseñado específicamente para leer cada termómetro y multiplexar las señales producidas por cada píxel hacia una salida de video (Véase la **Figura 2.5**). El ROIC es un Circuito Integrado de Aplicación Específica (ASIC, por sus siglas en inglés), desarrollado utilizando tecnología CMOS especialmente para esta función, [2].

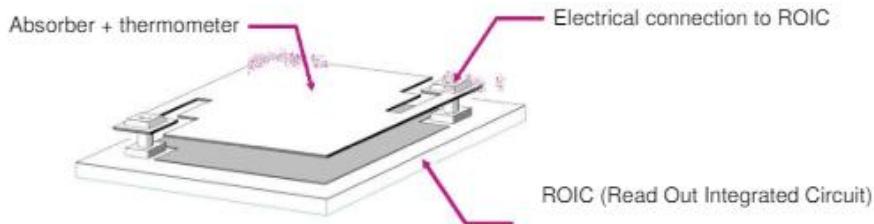


Figura 2.5: PICO1024. Estructura esquemática de un píxel [2].

2.2.1.3 Circuito Integrado de Lectura (ROIC)

La función principal del ROIC es transformar la radiación infrarroja en señales electrónicas, basadas en la emisión térmica de los objetos en la escena. El ROIC detecta los cambios en la resistencia de cada píxel causados por la radiación infrarroja y multiplexa las señales eléctricas resultantes hacia dos o cuatro salidas de video. Estas salidas de video deben estar sincronizadas con las señales de reloj proporcionadas por un sistema de control externo, lo que es esencial para la correcta reconstrucción de la imagen en etapas posteriores.

La estructura del ROIC, expuesta en la **Figura 2.6**, se compone de una parte analógica y otra digital. La principal función de los bloques analógicos es extraer una señal que produce cada bolómetro resistivo y generar una salida de video analógica para cada píxel. Por otro lado, los bloques digitales se encargan de la sincronización de cada uno de los píxeles y su multiplexación en las salidas de video. Además, se encarga de establecer la comunicación externa para la configuración de los diferentes aspectos del sensor.

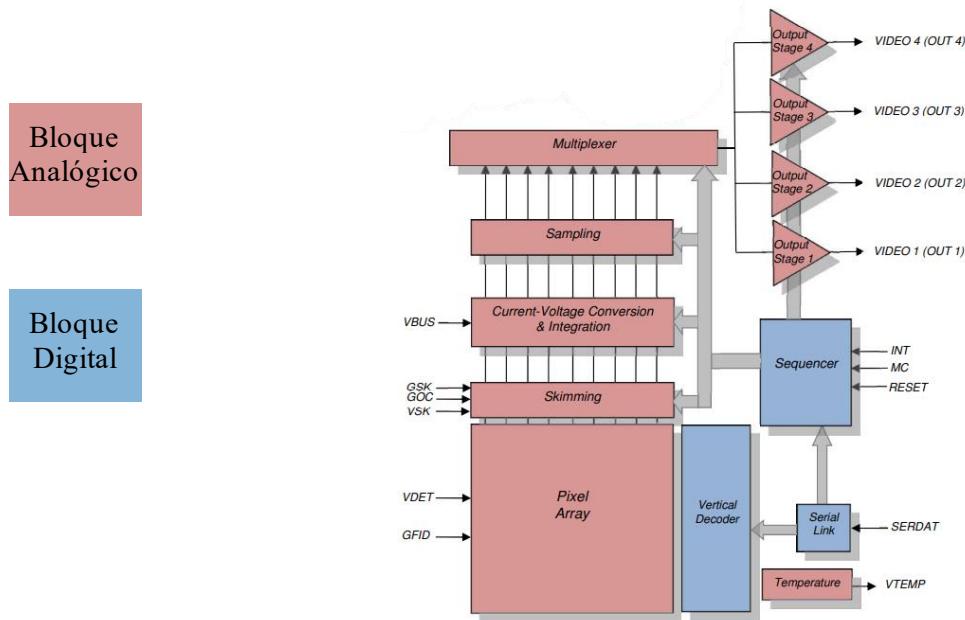


Figura 2.6: PICO1024. Diagrama interno del ROIC [2].

2.2.2 Configuración

El sensor PICO1024 tiene dos modos funcionamiento: modo nominal PICO1024 y modo de compatibilidad UL05251. La diferencia principal entre ambos modos es que el modo nominal incluye nuevas funcionalidades como el ‘MODE_4OUTPUTS’ que permite habilitar dos salidas más para aumentar el procesamiento en paralelo. La diferencia sustancial es que el número de bits que deban transmitirse para configurar el dispositivo sea mayor.

A la hora de establecer una nueva configuración del sensor, la comunicación requiere comenzar con el primer bit, llamado condición de START, a nivel alto. Seguidamente, se debe enviar de forma consecutiva. El flanco de subida del reloj maestro (MC) se utiliza para enviar el nuevo bit a transmitir; mientras que los flancos de bajada son usados por el sensor para capturar la información transmitida. El orden de estos bits se describe en la **Tabla 2.4** y la longitud de esta trama dependerá de si queremos el modo de compatibilidad nominal o el UL05251, tal y como se muestra en las imágenes **Figura 2.7** y **Figura 2.8**.

Para finalizar y aplicar dicha configuración, es importante aplicar un pulso de RESET en el periodo posterior al último bit de configuración enviado.

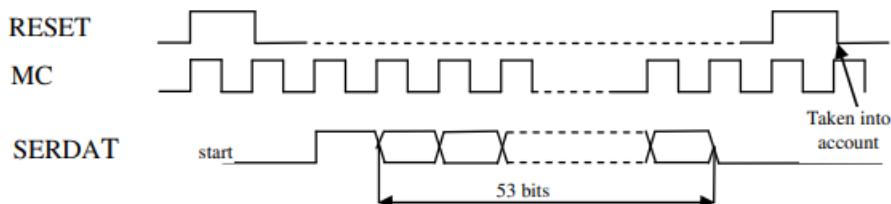


Figura 2.7: PICO1024. Cronograma para modo de compatibilidad UL05251 [2].

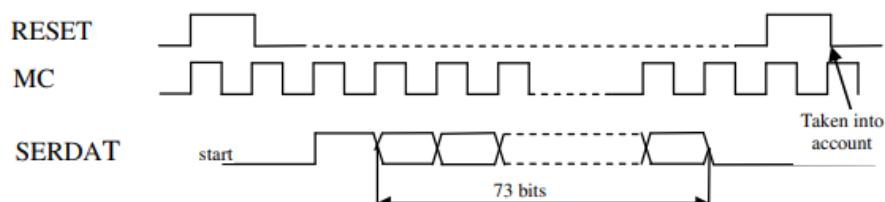


Figura 2.8: PICO1024. Cronograma para configuración nominal [2].

Tal y como hemos indicado anteriormente, la **Tabla 2.4** detalla la posición, nombre, tamaño y valor por defecto de cada uno de los bits que componen la trama de configuración tanto para el modo de compatibilidad UL05251 como el modo nominal PICO1024.

POSICIÓN	MODO DE COMPATIBILIDAD UL05251	MODO PICO1024	TAMAÑO	VALOR POR DEFECTO
0 → 0	START	START	1	1
1 → 1	CALOC	CALOC	1	0
2 → 5	SELQ <2:0>, VALK	SELQ <2:0>, VALK	4	1000
6 → 8	Reservado	Reservado	3	000
9 → 9	VALOC	VALOC	1	0
10 → 11	GAIN	GAIN	2	10
12 → 12	UPCOL	UPCOL	1	1
13 → 13	UPROW	UPROW	1	1
14 → 14	SIZEA	SIZEA	1	1
15 → 15	SIZEB	SIZEB	1	1
16 → 25	YFIRST	YFIRST	10	0000000000
26 → 35	YLAST	YLAST	10	1011111111
36 → 44	XFIRST	XFIRST	9	000000000
45 → 53	XLAST	XLAST	9	111111111
54 → 54		MODE_PICO1024	1	0
55 → 55		Reservado	1	0
56 → 56		Reservado	1	0
57 → 57		Reservado	1	0
58 → 58		Reservado	1	0
59 → 62		Reservado	4	0000
63 → 63		Reservado	1	0
64 → 64		MODE_4OUTPUTS	1	0
65 → 65		Reservado	1	0
66 → 66		Reservado	1	0
67 → 67		Reservado	1	0
68 → 70		Reservado	3	000
71 → 71		Reservado	1	0
72 → 72		Reservado	1	0
73 → 73		Reservado	1	0

Tabla 2.4: Trama de configuración para el PICO1024 [2].

2.2.3 Operación

Primero, se debe cargar la configuración al censor, tal y como se ha detallado en el punto **2.2.2**. Justo después de dar la señal de reset tras el envío de la configuración a través de la línea SERDAT, procederemos con la etapa de operación que está definida por el MC, para la sincronización de los píxeles, y el uso de la señal INT, para marcar el final de cada fila de píxeles de la imagen. Por lo tanto, la duración de un periodo completo de la señal INT dependerá del número de columnas y el modo de funcionamiento que hayamos configurado previamente (MODE_4OUTPUTS), tal y como se muestra en la **Figura 2.9**. Además, se muestran las ecuaciones necesarias para el cálculo de los períodos en la **Ecuación 2.1**.

$$INT_{low} = 17 \text{ (Periodos del MC)}$$

$$INT_{high} = \text{Número de Columnas} \div \text{Número de Salidas} \text{ (Periodos del MC)}$$

$$\text{Número de Columnas} \in [160,1024]$$

$$\text{Número de Salidas} \in \{2,4\}$$

Ecuación 2.1. Estimación de la duración de la señal INT.

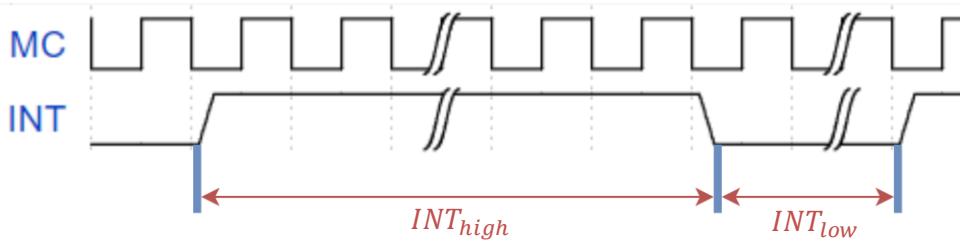


Figura 2.9: PICO1024. Período completo de la señal INT, Ecuación 2.1.

Por cada imagen deberán existir tantos períodos completos del INT como filas disponga el modo configurado, tal y como se observa en la **Figura 2.10**. Esta representa el envío de una imagen completa, compuesta por la lectura de 1024 columnas y 768 filas, en el modo de 4 salidas.

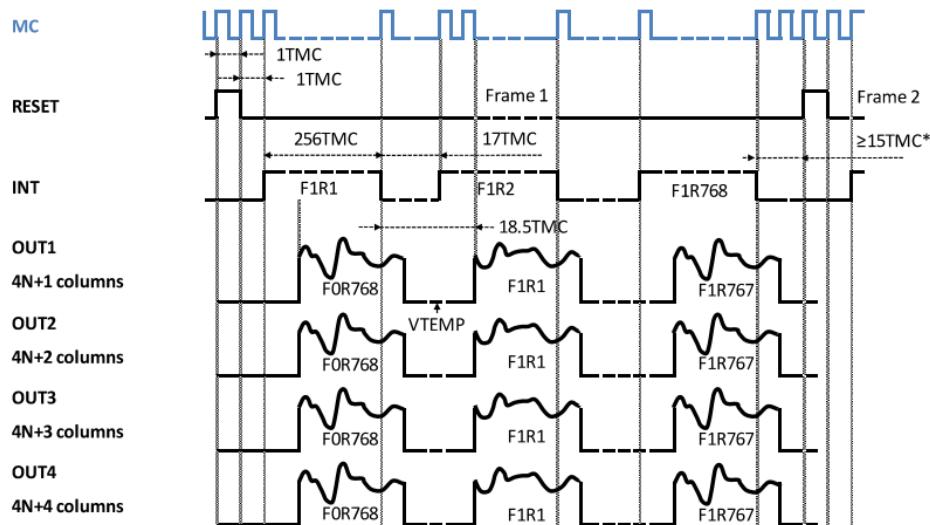


Figura 2.10: PICO1024. Modo de operación [2].

Finalmente, para sincronizarse con la muestra correcta de la señal analógica de cada uno de los píxeles, se debe crear una señal de reloj de igual periodo que el MC, pero con un cierto desfase, tal y como se aprecia en la **Figura 2.11**. Esta señal la llamaremos Sample Clock (SC) y permite establecer el instante de tiempo en el que el convertidor tomará la muestra de la salida de video correspondiente.

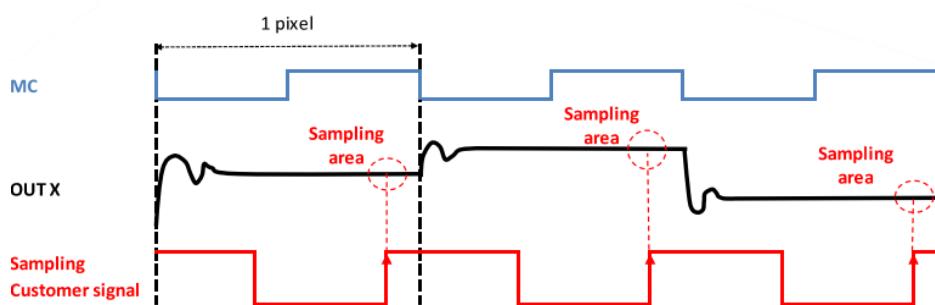


Figura 2.11: PICO1024. Señal de muestreo [2].

Es importante destacar que, tanto para el encendido como el apagado del sensor, debemos seguir un orden específico a la hora de activar las tensiones de alimentación y transmitir la configuración. Este proceso se representa en la **Figura 2.12**.

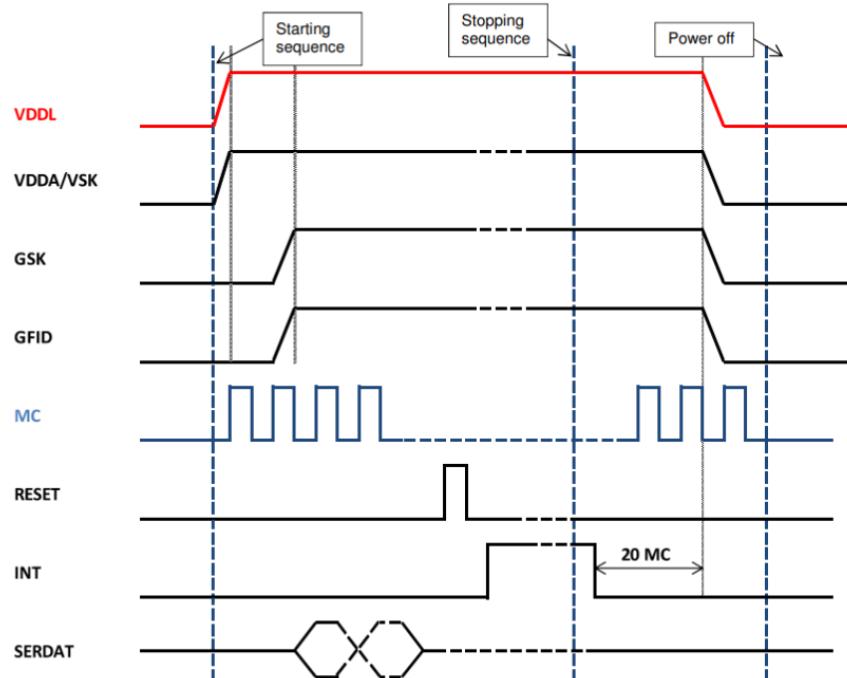


Figura 2.12: PICO1024. Secuencia de encendido, configuración y apagado del sensor [2].

2.3 ADC

Se ha utilizado el convertidor ADC3244 del fabricante Texas Instruments. Este circuito integrado incorpora dos canales de conversión de hasta 14 bits de resolución y con un rango de velocidades que comprende desde los 25 MSPS (puede reducirse hasta los 15 MSPS con una serie de configuraciones específicas) hasta los 125 MSPS. [4]. Algunas de las características más importantes del dispositivo figuran en la **Tabla 2.5**. Además, se puede observar una imagen del chip en la **Figura 2.13**.

Parámetro	Valor
Número de canales	2
Resolución	14 bits
SNR	72.4 dBFS
Suministro de tensión	1.8 V
Consumo de energía	< 116 mW/ Canal
Aislamiento entre canales	105 dB
Encapsulado	VQFN-48 (7 mm × 7 mm)

Tabla 2.5: Características del ADC3244.

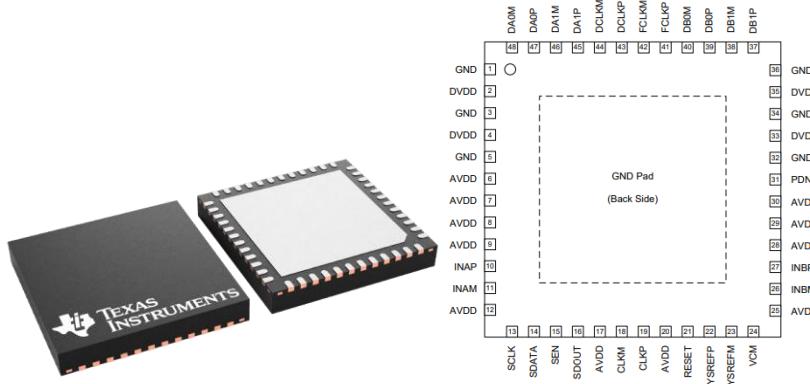


Figura 2.13: ADC3244. Vista Frontal [4].

A continuación, la **Tabla 2.6** recoge los pines que componen el ADC3244 junto con su nombre, dominio de señal al que pertenece (analógica, digital o tierra) y la función que realiza.

PIN	SEÑAL	TIPO	FUNCIÓN
1, 3, 5, 32, 34, 36	GND	Tierra	—
2, 4, 33, 35	DVDD	Entrada digital	Alimentación digital (1.8 V)
6-9, 12, 17, 20, 25, 28-30	AVDD	Entrada analógica	Alimentación analógica (1.8 V)
24	VCM	Salida analógica	Tensión del modo común
13	SCLK	Entrada digital	Entrada del reloj (Interfaz serie)
14	SDATA	Entrada digital	Entrada de datos (Interfaz serie)
15	SEN	Entrada digital	Habilitación (Interfaz serie)
16	SDOUT	Salida digital	Salida (Interfaz Serie)
18, 19	CLKP CLKM	Entrada digital	Señal diferencial del reloj de muestro del ADC
21	RESET	Entrada digital	Reinicia los registros del ADC
22, 23	SYSREFP SYSREFM	Entrada digital	Señal diferencial externa de referencia
10, 11, 26, 27	INxP INxM	Entrada analógica	Señal diferencial del canal ' x ' \in A, B
31	PDN	Entrada digital	Apagado de control
37 – 40, 45 - 48	DxyP DxyM	Salida digital	Señal diferencial del canal ' x ' \in A,B y puerto ' y ' \in 0,1
41, 42	FCLKP FCLKM	Salida digital	Señal diferencial del reloj de sincronización de tramas
43, 44	DCLKP DCLKM	Salida digital	Señal diferencial del reloj de sincronización de datos

Tabla 2.6: Características de los pines del ADC3244 [4].

2.3.1 Configuración

El ADC utiliza una interfaz SPI (Serial Peripheral Interface) para la configuración de sus registros. Este protocolo se compone de cuatro señales:

- **SCLK:** Es el reloj de sincronización para la interfaz serie.
- **SEN:** Es una señal activa a nivel bajo que marca el inicio y el fin de una comunicación.

- **SDATA:** Es la entrada de datos serie del ADC y está compuesta por 24 bits:

- **Bit 0:** operación a ejecutar:
 - **Escritura:** 0
 - **Lectura:** 1

○

BIT	0	1	2-15	16-23
Valor	Escribir = '0' Leer = '1'	constante	Dirección del registro	Datos para escribir en el registro

- **SDOUT:** Es la salida de datos serie del ADC y se compone de 2 bytes.

Se muestra, a continuación, un ejemplo de escritura en la **Figura 2.14** y un ejemplo de lectura en la **Figura 2.15**.

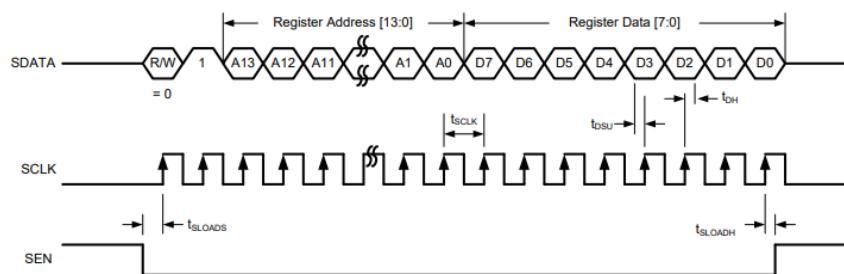


Figura 2.14: ADC3244. Escritura en los registros de configuración [4].

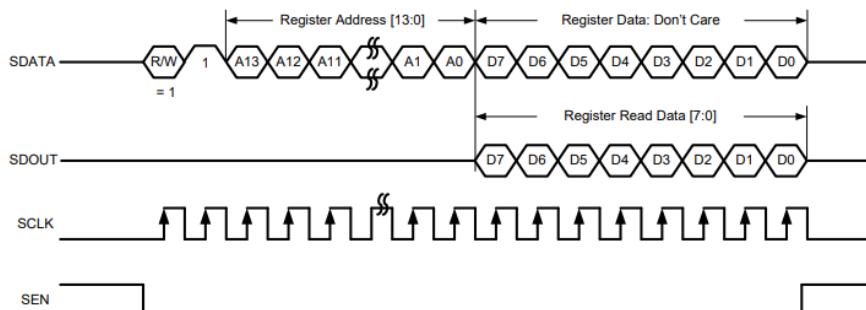


Figura 2.15: ADC3244. Lectura en los registros de configuración [4].

Los registros internos del ADC están expuestos en la Tabla 2.7, donde se detalla su dirección y la asignación de cada uno de los bits a su parámetro configurable correspondiente. Además, se explican algunos de estos parámetros en la Tabla 2.8 por su relevancia en el proyecto. El resto de los parámetros se encuentran explicados con mayor detalle en el datasheet del dispositivo [4].

Dirección del registro [13:0](Hex)	Datos del registro							
	D7	D6	D5	D4	D3	D2	D1	D0
01	0	0	DIS DITH CHA		DIS DITH CHB		0	0
03	0	0	0	0	0	0	0	ODD EVEN
04	0	0	0	0	0	0	0	FLIP WIRE
05	0	0	0	0	0	0	0	1W-2W
06	0	0	0	0	0	0	TEST PATTERN EN	RESET
07	0	0	0	0	0	0	0	OVR ON LSB
09	0	0	0	0	0	0	ALIGN TEST PATTERN	DATA FORMAT
0A	0	0	0	0	CHA TEST PATTERN			
0B	CHB TEST PATTERN				0	0	0	0
0E	CUSTOM PATTERN [13:6]							
0F	CUSTOM PATTERN [5:0]						0	0
13	0	0	0	0	0	0	LOW SPEED ENABLE	
15	0	CHA PDN	CHB PDN	0	STANBY	GLOBAL PDN	0	CONFIG PDN PIN
25	LVDS SWING							
27	CLK DIV		0	0	0	0	0	0
41D	0	0	0	0	0	0	HIGH IF MODE0	0
422	0	0	0	0	0	0	DIS CHOP CHA	0
434	0	0	DIS DITH CHA	0	DIS DITH CHA	0	0	0
439	0	0	0	0	SP1 CHA	0	0	0
51D	0	0	0	0	0	0	HIGH IF MODE1	0
522	0	0	0	0	0	0	DIS CHOP CHB	0
534	0	0	DIS DITH CHB	0	DIS DITH CHB	0	0	0
539	0	0	0	0	SP1 CHB	0	0	0
608	HIGH IF MODE[3:2]			0	0	0	0	0
70A	DIS CLK FLIT	0	0	0	0	0	0	PDN SYSREF

Tabla 2.7: ADC3244. Registros internos.

Parámetro	Dirección [13:0](Hex)	Posición	Descripción
1W-2W	05	D0	Permite elegir entre OneWire ('1') o TwoWire ('0')
RESET	06	D0	Reinicia los registros del ADC a '0'
TEST PATTERN EN	06	D1	Habilita el patrón de pruebas a las salidas del ADC
DATA FORMAT	09	D0	Permite seleccionar las salidas en complemento A2 ('0') o con establecer un offset ('1')
CHA TEST PATTERN	0A	D3 → D0	Permite seleccionar el patrón en el canal A
CHA TEST PATTERN	0B	D7 → D4	Permite seleccionar el patrón en el canal B
CUSTOM PATTERN	0E	D7 → D0	8 bits más significativos del patrón de prueba
	0F	D7 → D2	6 bits más significativos del patrón de prueba
LOW SPEED ENABLE	13	D1 → D0	('10') en TwoWire y ('10') en OneWire para trabajar con velocidades entre 20-25 MHz

Tabla 2.8: ADC3244. Descripción de parámetros configurables.

2.3.2 Operación

A la hora de transmitir el resultado de la conversión, el ADC3244 utiliza las señales de reloj diferenciales DCLK, FCLK, CLK; y las interfaces de datos serie diferenciales DA0, DA1, DB0 y DB1. Dichas señales de reloj se utilizan para la sincronización de los datos durante la comunicación. De este modo, el FCLK sincroniza el inicio y fin de una trama mientras que el DCLK indica cuando hay un dato disponible para muestrear mediante el flanco de subida y de bajada.

Este método de transmisión de las tramas dispone de dos modos de operación distintos: 'Two-Wire' y 'One-Wire'. La principal diferencia es el uso de una única interfaz serie en cada canal para la transmisión de los datos ('One-Wire', **Figura 2.16**) frente al uso de dos ('Two-Wire', **Figura 2.17**). Como se puede observar, para ambos casos los períodos de reloj serán diferentes tal y como se exponen en las ecuaciones **Ecuación 2.2** y **Ecuación 2.3**.

$$T_{FCLK} = T_{CLK}$$

$$T_{DCLK} = T_{FCLK} \div 14$$

Ecuación 2.2.

$$T_{FCLK} = 2 \times T_{CLK}$$

$$T_{DCLK} = T_{FCLK} \div 7$$

Ecuación 2.3.

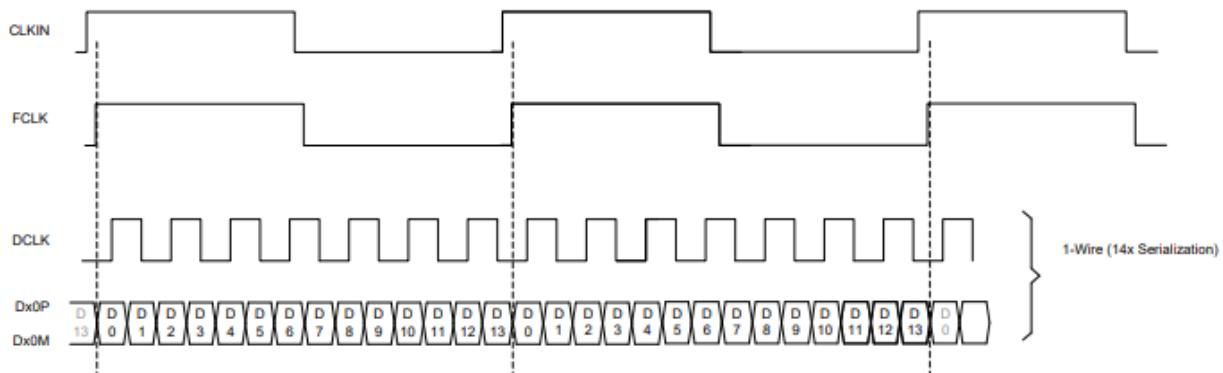


Figura 2.16: ADC3244. Modo de operación 'One-Wire', Ecuación 2.2. [4].

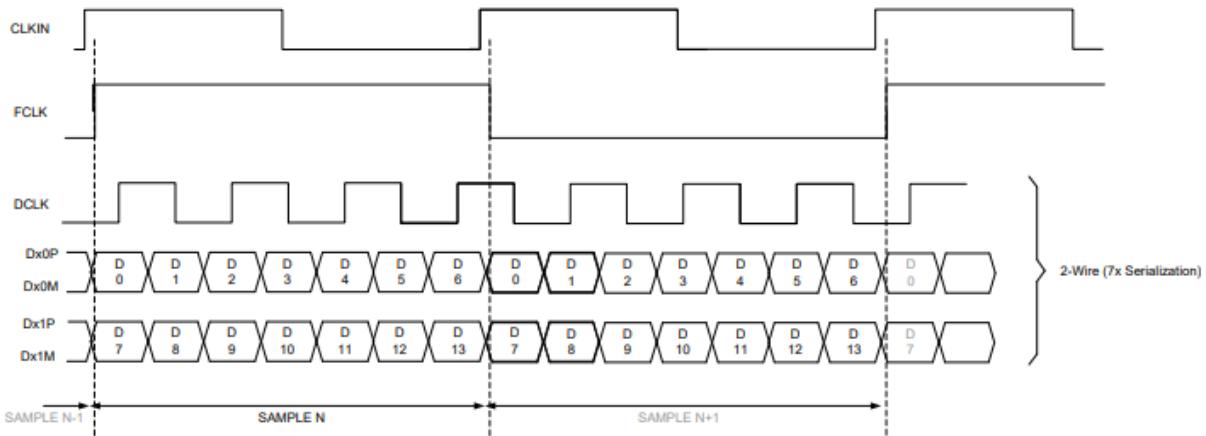


Figura 2.17: ADC3244. Modo de operación 'Two-Wire', Ecuación 2.3. [4].

2.4 Tarjeta de adaptación

Tarjeta suministrada por Alter Technology TÜV Nord que adapta la interfaz del sensor al ADC y a la FPGA junto con la generación de las tensiones de alimentación necesarias.

Por un lado, el sensor transmite los datos por dos salidas single-ended, mientras que nuestro ADC trabaja con señales diferenciales. Por lo tanto, la adaptación está compuesta por una etapa de conversión single-ended a diferencial basada en los circuitos integrados AD8139 y el AD8066.

Por último, la conexión a la FPGA se realiza a través de un conector FMC, **Figura 2.18**.

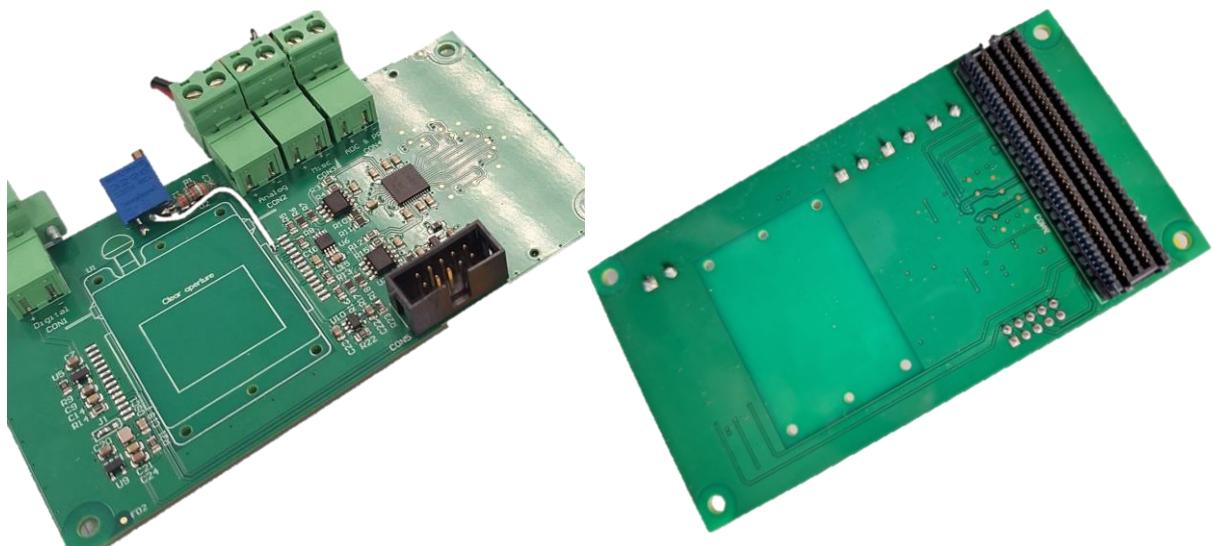


Figura 2.18: Imagen de la placa de adaptación suministrada.

3 ARQUITECTURA HARDWARE

El software utilizado para el desarrollo del lenguaje HDL (Hardware Description Language) en este proyecto es Vivado 2022.2, una potente herramienta de diseño ofrecida por la compañía Xilinx. Este entorno de desarrollo es ampliamente reconocido por su capacidad para facilitar el diseño y simulación de circuitos digitales complejos y está especialmente adaptado para trabajar con los dispositivos FPGA y SoC de Xilinx, [3].

La arquitectura hardware del proyecto se estructura alrededor de varios componentes claves: el PL (Programmable Logic), el PS (Processing System) y las interfaces AXI para la comunicación entre el PS y el PL, específicamente las interfaces GP (General Purpose) y HP (High Performance). Estas interfaces son fundamentales para gestionar la transferencia eficiente de datos entre el procesador y la lógica programable, permitiendo que ambos componentes del SoC FPGA interactúen de manera efectiva.

En este apartado, nos centraremos en la composición del PL, que desempeña un papel crucial en el procesamiento y manejo de los datos dentro del dispositivo. Una de las piezas centrales de esta configuración es la IP de DMA (Direct Memory Access), proporcionada por Xilinx. La DMA es esencial para el manejo eficiente de datos, ya que permite la transferencia de grandes volúmenes de información entre la memoria y el PL sin la intervención constante del procesador central, liberando recursos del sistema para otras tareas y aumentando el rendimiento general del sistema.

Además, se integra una FIFO (First In, First Out) que es necesaria para retener paquetes de datos antes de ser procesados por la DMA. Esta FIFO actúa como un buffer temporal que ayuda a gestionar flujos de datos de entrada y salida, asegurando que la DMA pueda operar de manera continua y eficiente sin sufrir interrupciones por variaciones en la velocidad o disponibilidad de los datos entrantes.

Por último, el PL también incluye módulos desarrollados específicamente en HDL para manejar la funcionalidad de dispositivos externos como el ADC3244 y el PICO1024. Se pude observar el esquema general del proyecto en la *Error! No se encuentra el origen de la referencia.*, donde se marca en naranja el bloque representativo del PS (Processing System), y las direcciones de memoria reservadas para los registros de configuración de cada uno de los componentes en la Tabla 3.1.

Módulo	Tamaño	Dirección
PS	1 G	0x0000_0000 → 0x3FFF_FFFF
PICO1024	4 K	0x4000_0000 → 0x4000_1000
ADC3244	4 K	0x4000_1000 → 0x4000_1FFF
DMA	64 K	0x4040_0000 → 0x4040_FFFF

Tabla 3.1: Mapa de memoria.

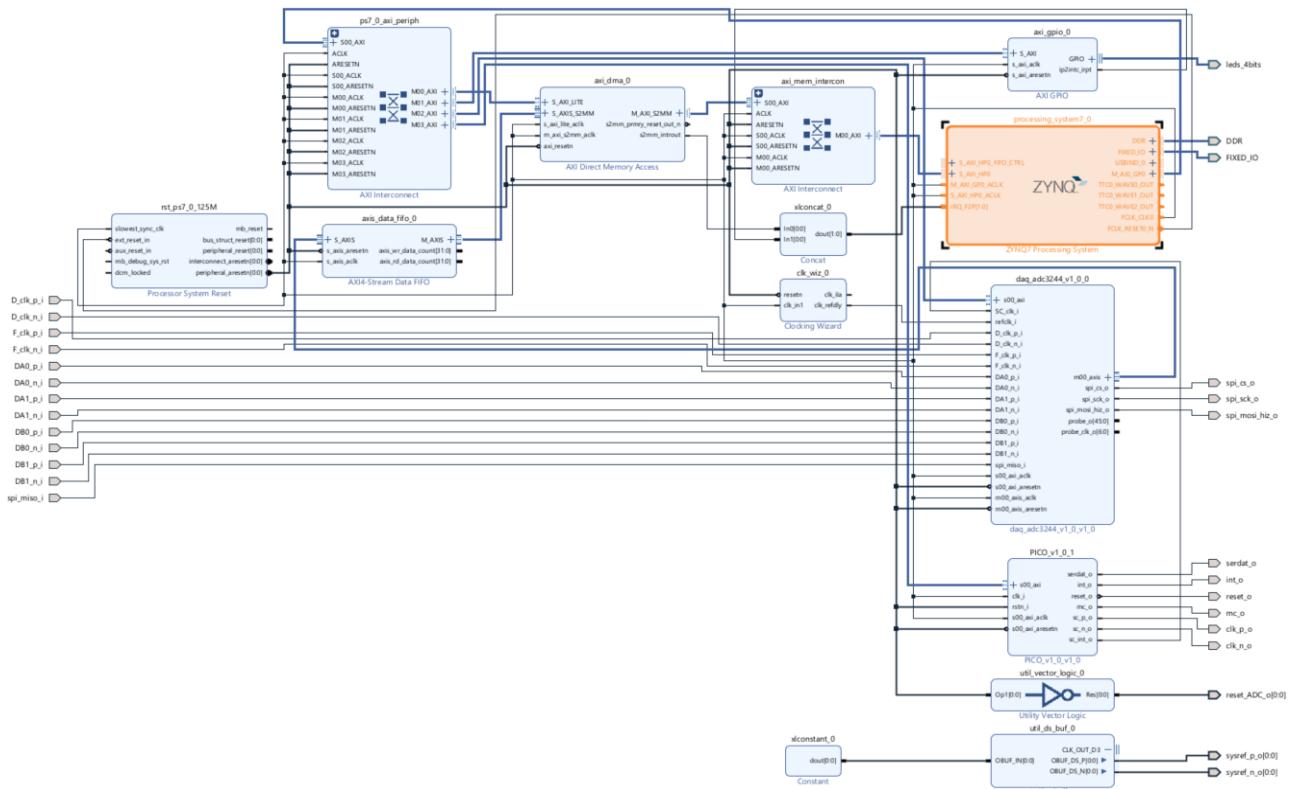


Figura 3.1:Esquema General del lenguaje HDL desarrollado.

3.1 DMA

La DMA o *Direct Memory Access* facilita una conexión de alta velocidad para mover datos directamente entre las interfaces de memoria AXI4 y AXI4-Stream. Los registros para iniciar, supervisar y gestionar estas operaciones se acceden mediante una interfaz esclava AXI4-Lite. En la **Figura 3.2** siguiente se muestra una imagen de la composición interna del diagrama de bloques.

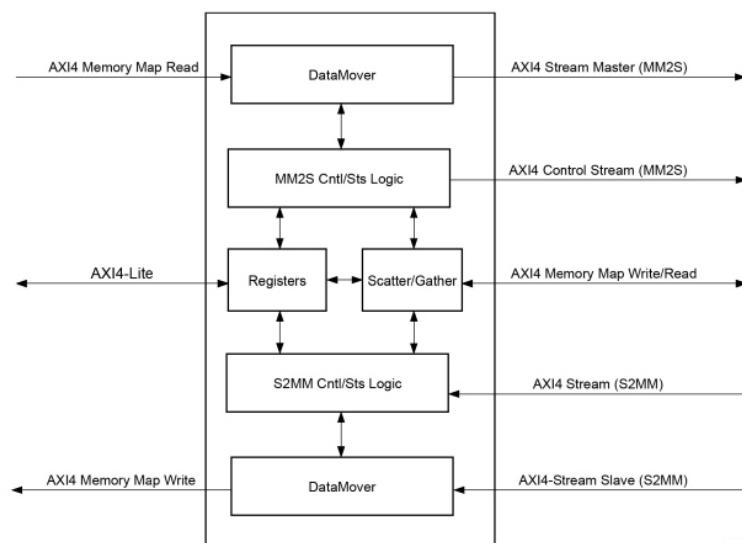


Figura 3.2: DMA. Diagramma interno [5]

La transferencia de datos de alta velocidad entre la memoria del sistema y el destino se realiza a través de una interfaz maestra AXI4 Stream tipo MM2S (de memoria a flujo) hacia una interfaz esclava AXI4 Stream. Para escribir en la memoria del sistema, se conecta la interfaz esclava AXI4 Stream tipo S2MM (de flujo a memoria) a una interfaz maestra AXI4 Stream. La DMA permite hasta 16 canales de datos simultáneos en ambas direcciones (MM2S y S2MM) en configuración de dispersión/recolección.

Los canales MM2S y S2MM funcionan de manera independiente. El canal MM2S admite un flujo de control AXI para recibir datos de la aplicación del usuario al módulo de destino. En cuanto al canal S2MM, se proporciona un flujo AXI de datos para su escritura desde el módulo desarrollado a la memoria del sistema.

En la aplicación, sólo se utiliza el canal S2MM para almacenar paquetes de 32 palabras de 4 bytes y transmitir dichos paquetes a una dirección de memoria especificada en los registros de control. Dichas palabras de datos son leídas desde una FIFO (First In First Out), la cual almacenará los datos recibidos por el IP **DAQ_ADC3244** de manera secuencial. El objetivo final es poder guardar en memoria los datos recibidos sin ninguna pérdida entre tramas.

Para la configuración y puesta en marcha de la DMA, se debe realizar primero un reset de los registros de control mediante una escritura (de valor hexadecimal 0x4) en la dirección del canal S2MM mapeado en memoria (en nuestro caso 0x4040000) con un offset de 0x30. Después, se debe especificar la nueva dirección de destino escribiendo dicho valor en 0x40400048 y la cantidad de bytes que se tienen que transmitir en 0x40400058. Finalmente, para inicializar la DMA se debe escribir en la dirección de control 0x40400030 el valor 0x5001. En la **Tabla 3.2**, se observa la descripción del control aplicado.

BIT	NOMBRE	VALOR INICIAL	VALOR ASIGNADO	DESCRIPCIÓN
0	RS	0	1	Control de Arranque/Parada para iniciar y parar el canal S2MM de la DMA.
1	Reservado	1	0	—
2	Reset	0	0	Realiza un reinicio del núcleo AXI DMA. Finaliza en el momento en que este bit vuelve a ‘0’.
3	Keyhole	0	0	Inicia el canal S2MM escribiendo en el modo de dirección no incremental.
4	Cyclic BD Enable	0	0	Permite utilizar la DMA en modo de Descriptores de Búfer Cíclicos (BD) sin intervención del usuario. Se ignora el bit ‘Completado’.
5 ⊕ 11	Reservado	0	0	—
12	IOC_IrqEn	0	1	Indica que debe generar una interrupción cuando finalice la transferencia.
13	Dly_IrqEn	0	0	Indica que debe generar una interrupción cuando transcurra el tiempo indicado en IRQDelay
14	Err_IrqEn	0	1	Indica que debe generar una interrupción en caso de un error
15	Reservado	0	0	—
16 ⊕ 23	IRQThreshold	01h	00h	Se realiza una cuenta regresiva sobre el valor asignado por cada interrupción generada y transmite una interrupción a la salida de la DMA cuando la cuenta finalice.
24 ⊕ 31	IRQDelay	00h	00h	Permite establecer un temporizador para generar una interrupción. Este temporizador se inicia al finalizar un paquete y se reinicia con la llegada de otro. La cuenta es el numero asignado × 125 × Periodo del reloj

Tabla 3.2: DMA. Registro de control.

El módulo utilizado para el control y monitorización del estado de la DMA es proporcionado por la compañía Xilinx. Toda la información relevante a su configuración viene detallada en su página oficial [5]. Además, se adjunta la **Figura 3.3** donde se muestra la configuración aplicada a la IP en Vivado.

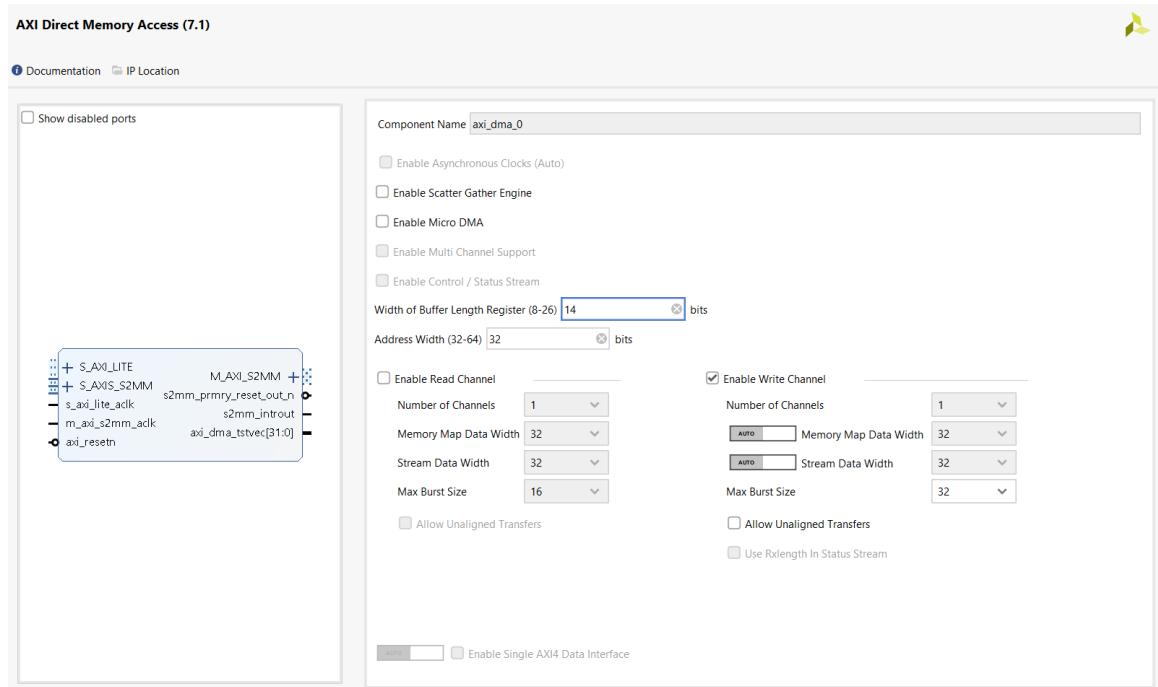


Figura 3.3: DMA. Configuración de la IP.

Es importante resaltar que el ancho de la longitud del buffer define el bus de direcciones admisibles para su escritura en la memoria del sistema. Por tanto, este debe ajustarse al tamaño máximo del bloque de datos que se va a transmitir.

3.2 FIFO data stream

Se encuentra ubicado como intermediario entre el DAQ_ADC3244 y la DMA. Su objetivo es minimizar fluctuaciones de los datos y buffer de la DMA, así como compensar el tiempo que requiere el sistema en reasignar una nueva ubicación a la DMA para el envío de los próximos datos. El tamaño de la FIFO es de 1024 palabras de 32 bits.

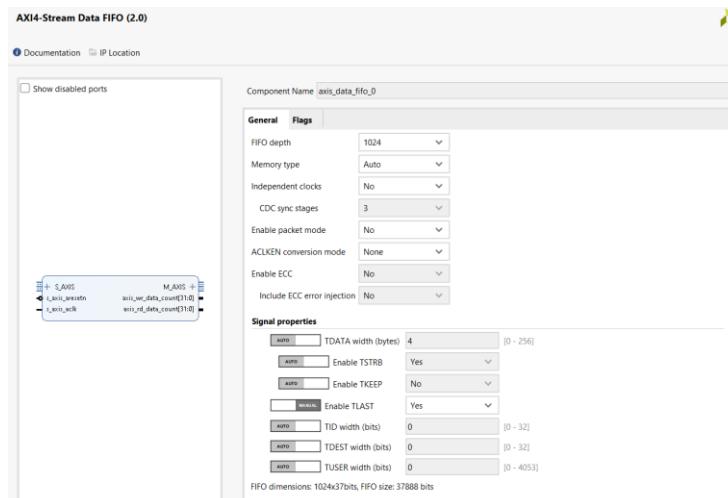


Figura 3.4: FIFO. Configuración de la IP.

3.3 IP PICO

Esta IP tiene dos objetivos principales. Por un lado, debe configurar el sensor de acuerdo con el protocolo especificado en su datasheet y con los valores proporcionados por la aplicación de usuario. Por otro lado, debe generar un reloj que establezca la frecuencia de envío de los datos (en este caso, píxeles) del PICO al ADC3244, denominado MC (Master Clock). Simultáneamente, debe generar un segundo reloj con el mismo período, pero con un cierto desfase para marcar el momento de muestreo del ADC3244, conocido como SC (Sampling Clock). Véase el diagrama de dichos en relojes en la **Figura 2.11**.

A continuación, la **Figura 3.5** muestra un diagrama de bloques de la entidad, donde se pueden apreciar los puertos de entrada y salida descritos en la **Tabla 3.3**. Este diagrama se compone principalmente de módulo que implementa una interfaz AXILite Slave para la configuración de los registros y un segundo bloque, denominado PICO, donde se define el comportamiento de la IP (Véase la **Figura 3.6**).



Figura 3.5: Señales Entrada/Salida del PICO IP.

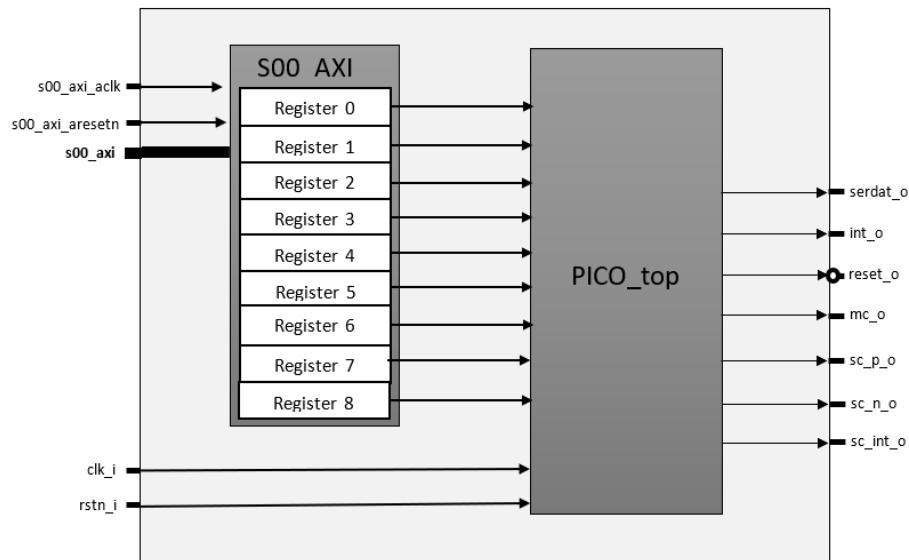


Figura 3.6: PICO IP. Diagrama de bloques.

En la **Tabla 3.3**, se describen cada uno de los puertos presentes en la IP dedicada al PICO1024.

	Puerto	Input/Output	Descripción
AXI_SLAVE_00	s00_axi_awaddr [4:0]	INPUT	Interfaz AXI-Lite dedicada a la escritura/lectura de los registros. <i>Para más información consulta la documentación de Xilinx.</i>
	s00_axi_awprot [2:0]	INPUT	
	s00_axi_awvalid	INPUT	
	s00_axi_awready	OUTPUT	
	s00_axi_wdata [31:0]	INPUT	
	s00_axi_wstrb [3:0]	INPUT	
	s00_axi_wvalid	INPUT	
	s00_axi_wready	OUTPUT	
	s00_axi_bresp [1:0]	OUTPUT	
	s00_axi_bvalid	OUTPUT	
	s00_axi_bready	INPUT	
	s00_axi_araddr [4:0]	INPUT	
	s00_axi_arprot [2:0]	INPUT	
	s00_axi_arvalid	INPUT	
	s00_axi_arready	OUTPUT	
	s00_axi_rdata [31:0]	OUTPUT	
	s00_axi_rresp [1:0]	OUTPUT	
	s00_axi_rvalid	OUTPUT	
	s00_axi_rready	INPUT	
PICO_TOP	s00_axi_aclk	INPUT	Reloj interno Reset Asíncrono. Transmite los datos de configuración del sensor PICO 1024 según el protocolo detallado en 2.2.3 Operación
	s00_axi_aresetn	INPUT	
	clk_i	INPUT	
	rstn_i	INPUT	
	serdat_o	OUTPUT	
	int_o	OUTPUT	
	reset_o	OUTPUT	
	mc_o	OUTPUT	Sample Clock.
	sc_p_o (diff pair)	OUTPUT	
	sc_n_o (diff pair)	OUTPUT	
	sc_int_o (single-ended)	OUTPUT	

Tabla 3.3: Descripción de los puertos del PICO IP.

Por último, se muestra la **Figura 3.7** de la configuración del módulo en Vivado.

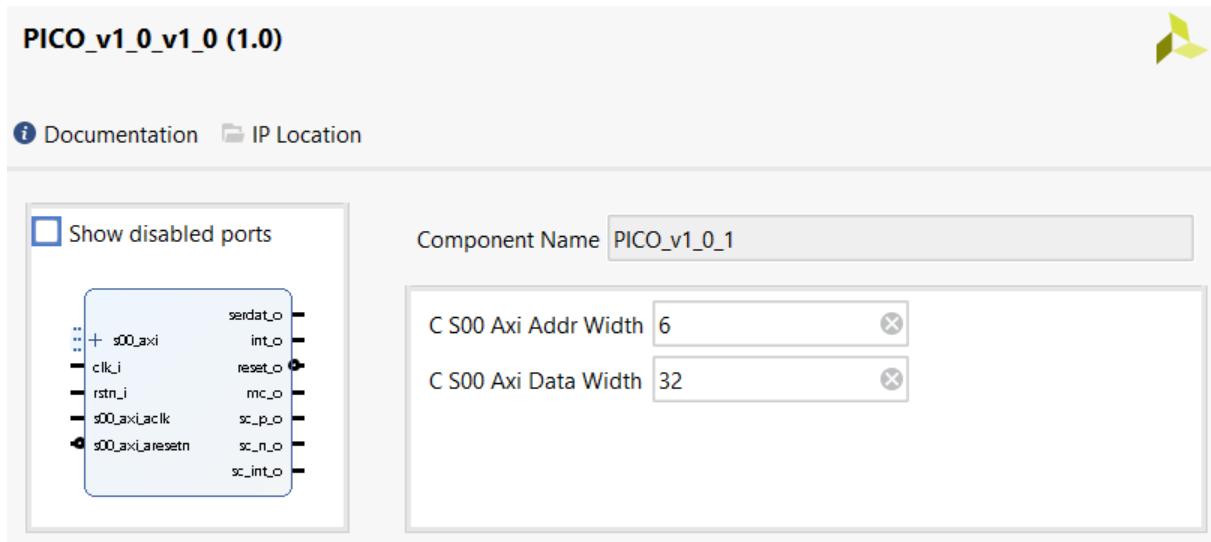


Figura 3.7: PICO1024. Configuración de la IP.

3.3.1 Registros

Los registros que incorpora el módulo permiten tanto configurar los parámetros del PICO1024, como monitorizar el estado de la adquisición de los fotogramas procedentes del sensor. La zona de memoria asignada para los registros es la 0x40001000 con un tamaño de 32 palabras (Véase la **Tabla 3.4** con la descripción de cada uno de ellos).

Registro [31:0]	Offset	Descripción
Slv_reg0	0x00	Registro de control para indicar tanto el inicio de la configuración y posterior arranque del sensor PICO1024, como el stop en caso de querer apagar el sensor, realizar una pausa o reconfigurarlo.
Slv_reg1	0x04	Permite monitorizar el estado del sensor según una codificación en One Hot.
Slv_reg2	0x08	
Slv_reg3	0x0C	Se almacenan una serie de parámetros de configuración que deben ser transmitidos al Sensor Óptico para su correcto funcionamiento.
Slv_reg4	0x10	
Slv_reg5	0x14	Número de periodos del MC que debe tener la señal “int_o” en ‘HIGH level’ (Número de Columnas/2 del fotograma)
Slv_reg6	0x18	Número de periodos del “clk_i” para generar un periodo completo del MC.
Slv_reg7	0x1C	Número de periodos del “clk_i” que debe tener el desfase entre el MC y el SC
Slv_reg8	0x20	Número de filas del fotograma

Tabla 3.4: Descripción de los registros del IP PICO.

3.3.2 Descripción de la entidad

Como se ha visto previamente en la **Figura 3.6**, el bloque se compone de la interfaz AXI Slave, cuya descripción hardware viene aportada y resuelta por Xilinx, y el submódulo PICO. Este último se encarga de establecer el

protocolo de comunicación indicado en la subsección 2.2.3 para la configuración y puesta en marcha del sensor. Para ello, se utilizan dos entidades, el PICO_controller (encargado de la supervisión de la configuración y sincronización con el PICO) y el PICO_phys_layer (encargado de generar los relojes y cumplir los tiempos de los protocolos de configuración y sincronización con el PICO), tal y como se observa en la **Figura 3.8**.

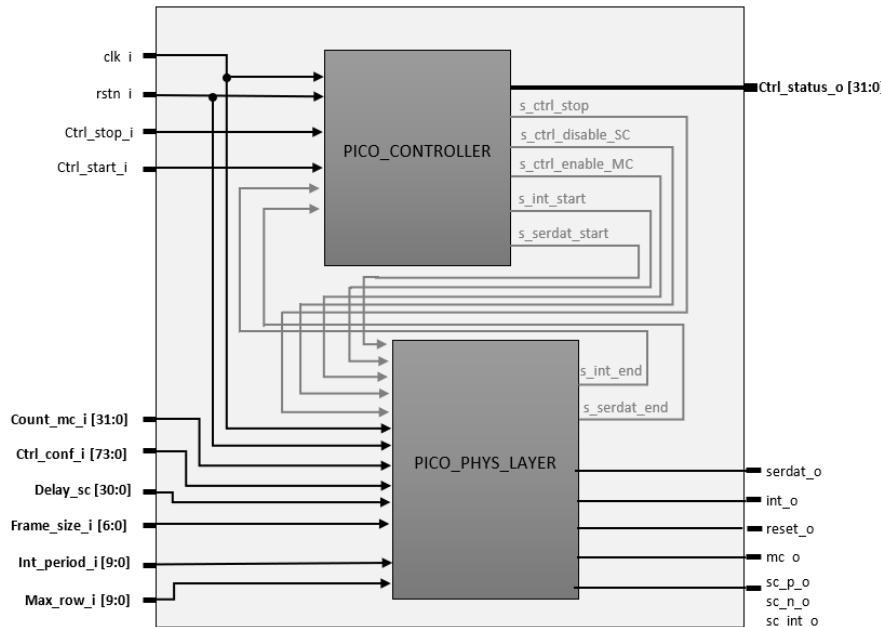


Figura 3.8: PICO_top. Diagrama de bloques.

3.3.2.1 PICO_controller

Este sub-bloque es responsable de supervisar el inicio de la configuración del sensor y el inicio del tiempo de integración. Además, antes de apagar el sensor, el controlador monitoriza las señales de salida para garantizar un apagado adecuado del sensor cuando recibe la señal de detención.

Se muestra, a continuación, el diagrama de estados desarrollado:

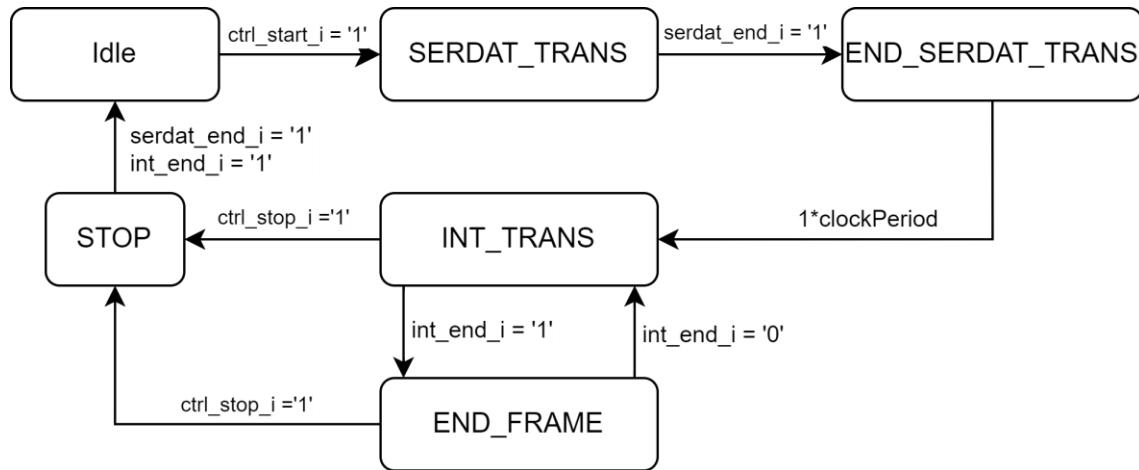


Figura 3.9: PICO_controller. Diagrama de estados.

3.3.2.2 PICO_phys_layer

Este sub-bloque es responsable de generar el reloj principal, el reloj de muestreo, la salida de datos en el canal serdat_o y la salida en el canal int_o basándose en las señales del controlador PICO, respetando los intervalos de tiempo especificados entre cada señal y la señal de reinicio, tal como se detalla en 2.2.3.

3.4 IP DAQ_ADC3244

El propósito de esta IP es configurar los registros del ADC3244 e interpretar los relojes de 'Frame clock' (F_clk_i) y 'Data clock' (D_clk_i), junto con los datos (Dx0 y Dx1) de cada canal, para la lectura de las tramas de 14 bits del ADC. Además, debe ser capaz de enviar estas tramas de datos a una FIFO asegurando que no se pierda ninguna de los dos canales (canales A y B).

A continuación, la **Figura 3.10** muestra un diagrama de bloques de la entidad, donde se pueden apreciar los puertos de entrada y salida descritos en la **Tabla 3.5**. Este diagrama se compone principalmente del AXI Slave para la configuración de los registros, el AXIStream y la capa de adquisición donde se define el comportamiento de la IP (Véase la **Figura 3.11**).

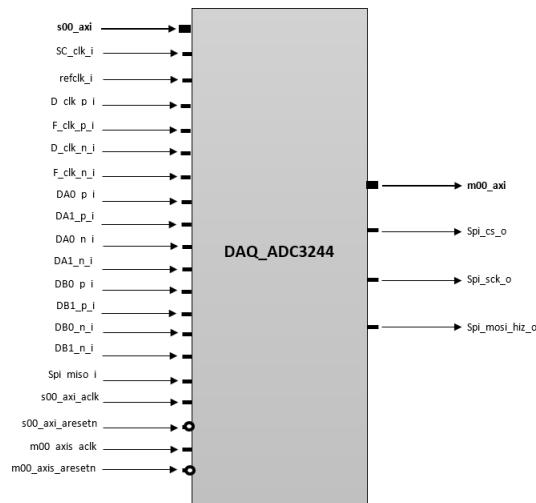


Figura 3.10: DAQ_ADC3244 IP. Puertos de entrada/salida.

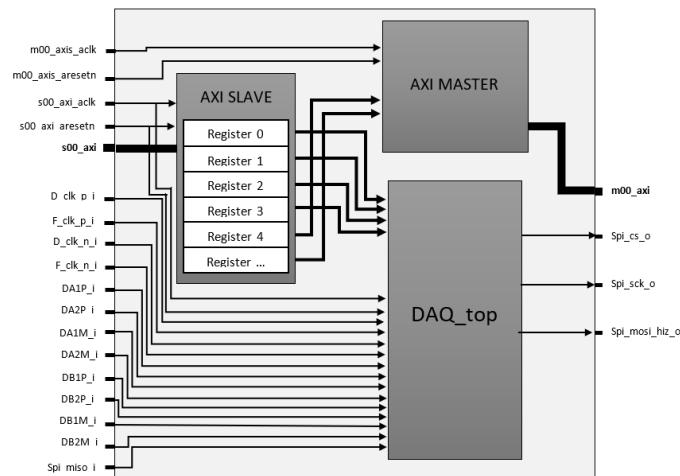


Figura 3.11: DAQ_ADC3244 IP. Diagrama de bloques.

En la **Tabla 3.5** se describen cada uno de los puertos presentes en la IP dedicada al ADC3244.

	Puerto	Input/Output	Descripción
AXI_SLAVE	s00_axi_awaddr [4:0]	INPUT	Interfaz AXI-Lite dedicada a la escritura/lectura de los registros. <i>Para más información consulta la documentación de Xilinx.</i>
	s00_axi_awprot [2:0]	INPUT	
	s00_axi_awvalid	INPUT	
	s00_axi_awready	OUTPUT	
	s00_axi_wdata [31:0]	INPUT	
	s00_axi_wstrb [3:0]	INPUT	
	s00_axi_wvalid	INPUT	
	s00_axi_wready	OUTPUT	
	s00_axi_bresp [1:0]	OUTPUT	
	s00_axi_bvalid	OUTPUT	
	s00_axi_bready	INPUT	
	s00_axi_araddr [4:0]	INPUT	
	s00_axi_arprot [2:0]	INPUT	
	s00_axi_arvalid	INPUT	
	s00_axi_arready	OUTPUT	
	s00_axi_rdata [31:0]	OUTPUT	
	s00_axi_rrresp [1:0]	OUTPUT	
	s00_axi_rvalid	OUTPUT	
	s00_axi_rready	INPUT	
AXI_MASTER	s00_axi_aclk	INPUT	Interfaz dedicada a la escritura de la FIFO.
	s00_axi_aresetn	INPUT	
	m00_axis_tdata [31:0]	OUTPUT	
	m00_axis_tstrb [3:0]	OUTPUT	
	m00_axis_tlast	OUTPUT	
DAQ_TOP	m00_axis_tvalid	OUTPUT	Señal diferencial de reloj del dato Señal diferencial de reloj de la trama Señal diferencial del canal 'x' ∈ A,B y puerto 'y' ∈ 0,1 MISO data from ADC3244. Chip Select from SPI_top entity to ADC3244. Clock Signal from SPI_top entity to ADC3244. MOSI data from SPI_top entity to ADC3244. Reference clock of 200 MHz for the IODELAY buffer Sampling clock
	m00_axis_tready	INPUT	
	D_clk_p_i, D_clk_n_i	INPUT	
	F_clk_p_i, F_clk_n_i	INPUT	
	Dxy_p_i, Dxy_n_i	INPUT	
	Spi_miso_i	INPUT	
	Spi_cs_o	OUTPUT	
	Spi_sck_o	OUTPUT	

Tabla 3.5: DAQ_ADC3244. Descripción de los puertos.

Por último, se muestra la **Tabla 2.7** de la configuración del módulo en Vivado.

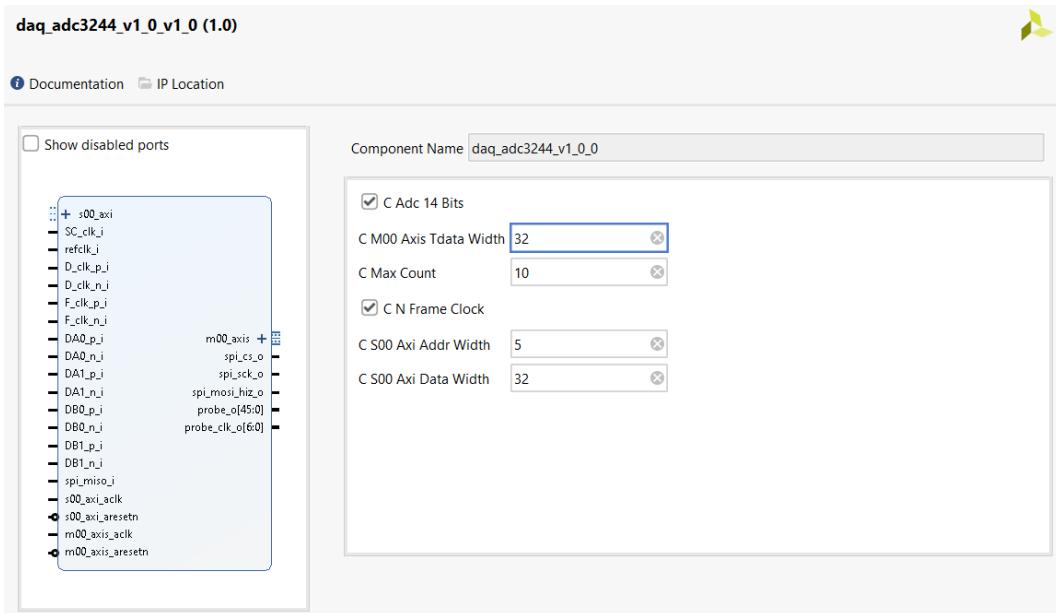


Figura 3.12: ADC3244. Configuración de la IP.

3.4.1 Registros

Los registros permiten tanto configurar los parámetros del ADC3244 como configurar algunos aspectos del método de adquisición de las tramas de datos. La zona de memoria asignada para los registros es la 0x40000000 con longitud de 32 k (Véase la TABLA con la descripción de cada uno de ellos)

Registros [31:0]	Offset	Descripción
Slv_reg0	0x00	Registro de control que regula una serie de parámetros del proceso de adquisición y configuración. Por ejemplo, el channel_mode o el periodo del SCLK del SPI
Slv_reg1	0x04	Permite monitorizar el estado de la adquisición y si hay tramas SPI encoladas en alguna de lasdos FIFOs
Slv_reg2	0x08	Trama SPI para cargar en la FIFO TX
Slv_reg3	0x0C	Trama SPI recibida de la FIFO RX. Después de leerla, se cargará en este registro la siguiente trama en la cola de la FIFO.
Slv_reg4	0x10	Dato de 14 bits más reciente leído del canal A del ADC3244
Slv_reg5	0x14	Dato de 14 bits más reciente leído del canal B del ADC3244
Slv_reg6	0x18	Máximo número de tramas de datos que se pueden enviar a la DMA por cada solicitud de paquetes

Tabla 3.6: DAQ_ADC3244. Descripción de los registros.

3.4.2 Descripción de la entidad

Como se ha visto previamente en la **Figura 3.11**, el bloque se compone del AXI Slave, cuya descripción hardware viene aportada y resuelta por Xilinx, el AXI Master y el ‘DAQ_top.vhd’, el cual está dividido internamente por dos bloques que trataremos como independientes, ‘SPI_top.vhd’ y ‘ADC3244_ACQ.vhd’.

3.4.2.1 Sub-bloques. ADC3244_ACQ

El propósito de esta parte de la IP es interpretar el protocolo de comunicación explicado en **2.3.2 Operación** para la lectura de diversas tramas de datos.

Este sub-bloque se divide principalmente en dos tipos de entidades. Por un lado, está el “en_controller”, cuya función es activar el proceso de adquisición de tramas cuando detecta que el reloj ‘Frame clock’ es correcto. Por otro lado, se encuentran las entidades dedicadas a la lectura de tramas de 14 bits para ambos canales. Además, utiliza algunos elementos aportados por Xilinx como son el IBUFDS para la conversión de diferencial a single-ended y el IODELAY para solventar problemas de retardo en la señal del ‘Data Clock’. Esta última es fundamental para asegurar la sincronización con el ‘Frame Clock’. Véase la **Figura 3.13** donde se observa el diagrama de bloques interno de este sub-bloque.

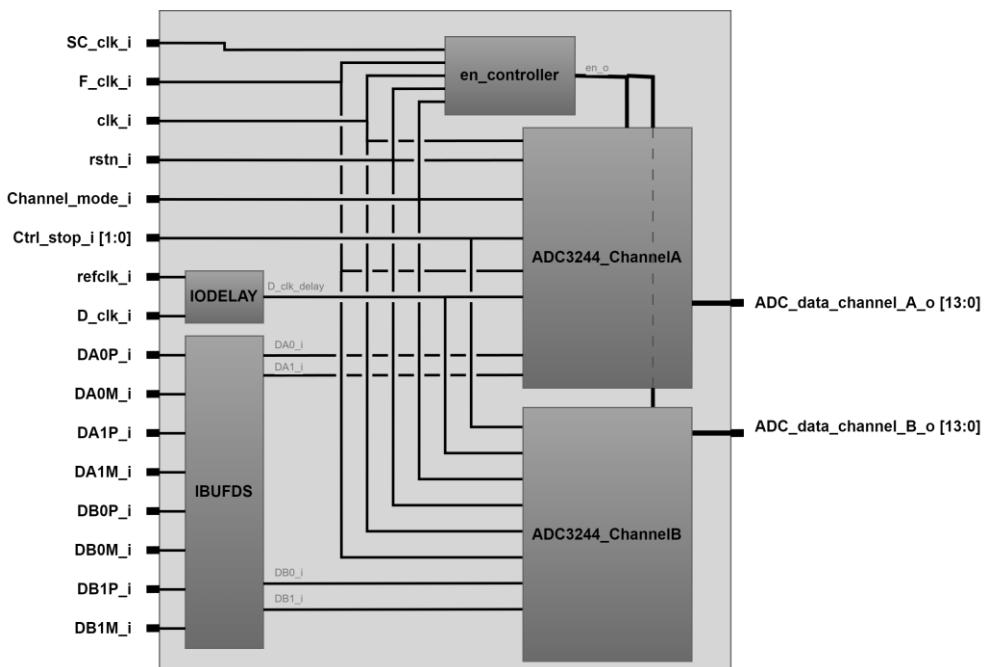


Figura 3.13: ADC3244_ACQ. Diagrama de bloques.

Como se mencionó en la sección **2.3.2**, nuestro diseño incluye dos modos de operación identificados por la señal “channel_mode”: el modo One-Wire y el modo Two-Wire. Por ello, para almacenar los datos recibidos en un registro, los datos del puerto DX0 (refiriéndose de forma genérica a los puertos DB0 y DA0) se entrarán al principio del registro de desplazamiento mientras que los datos del DX1 entrarán en mitad. De este modo, la entrada del puerto Dx0 se mantendrá independientemente del modo y el Dx1 se deshabilitará para el modo One-Wire. Además, si observamos la **Figura 2.16** y la **Figura 2.17**, se muestra que la entrada de datos se sincroniza con dos eventos diferentes: el flanco de subida y el flanco de bajada del ‘Data Clock’. Este enfoque introduce un desafío en la sincronización del registro que almacena los datos de diferentes tramas. Inicialmente, se optó por implementar dos registros de desplazamiento, denominados ‘even_reg’ y ‘odd_reg’, para capturar los datos en

posiciones pares e impares, activados por el flanco de subida y de bajada, respectivamente. Posteriormente, se realiza una reorganización de los bits para obtener la trama completa de 14 bits. La **Figura 3.14** y la **Figura 3.15** ilustran cómo se capturan los primeros 6 bits en los modos Two-Wire y One-Wire.

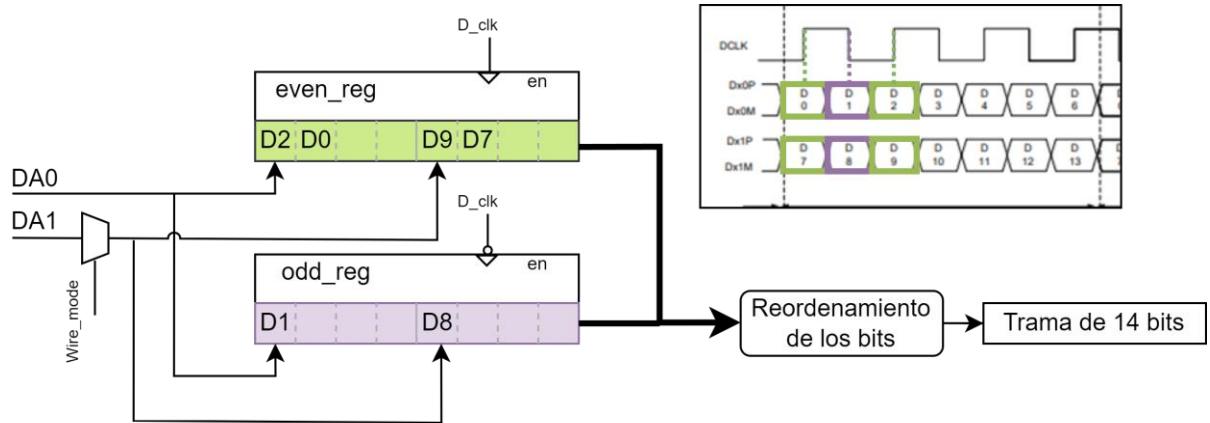


Figura 3.14: ADC3244_ACQ. Registros par e impar en el proceso de adquisición (Two-Wire).

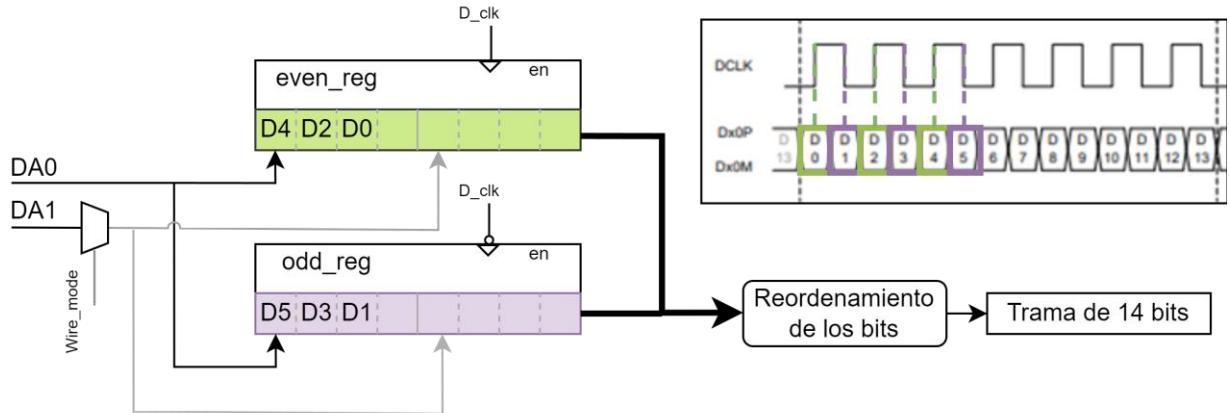


Figura 3.15: ADC3244_ACQ. Registros par e impar en el proceso de adquisición (One-Wire).

El segundo desafío que plantea este sistema es la velocidad de adquisición. En el escenario más desafiante, con un modo One-Wire a una frecuencia de MC de 25 MHz, el 'Data Clock' alcanza los 175 MHz. Esto deja aproximadamente 1.43 ns para procesar la trama recibida en el registro de 14 bits. Dado que nuestro reloj interno opera a 125 MHz (extensible hasta 500 MHz), este tiempo no es suficiente para procesar la trama y prepararse para la siguiente. Por lo tanto, se propone duplicar los registros mencionados: durante el periodo de nivel alto del 'Frame Clock', los registros 'even_reg_Fhigh' y 'odd_reg_Fhigh' almacenarán los datos; y durante el periodo de nivel bajo, lo harán 'even_reg_Flow' y 'odd_reg_Flow'. Es importante destacar que para el modo One-Wire, se duplica el reloj internamente para mantener una operación simétrica en ambos modos. Gracias a este sistema, mientras un par de registros se centran en la captura de la nueva trama, su gemelo reordena los bits de la captura anterior y lo envía a la capa superior para su procesado. La **Figura 3.16** y la **Figura 3.17** ilustran el funcionamiento de los registros durante los periodos alto y bajo del 'Frame Clock', respectivamente, para los primeros 6 bits recibidos de la trama actual.

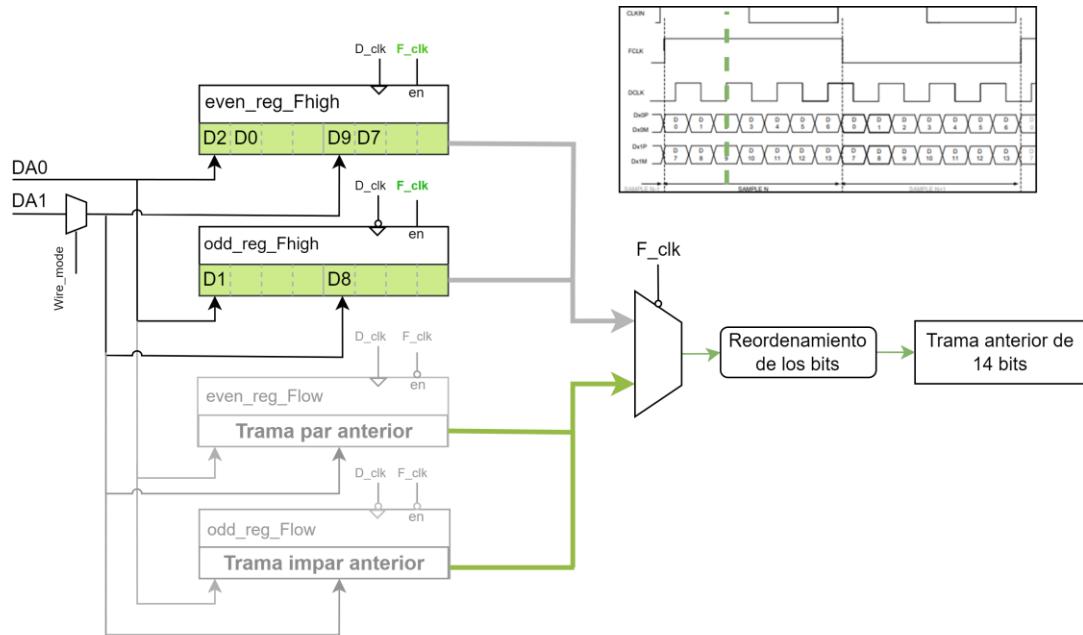


Figura 3.16: ADC3244_ACQ. Comportamiento de los registros en la adquisición ($F_{clk_i} = '1'$).

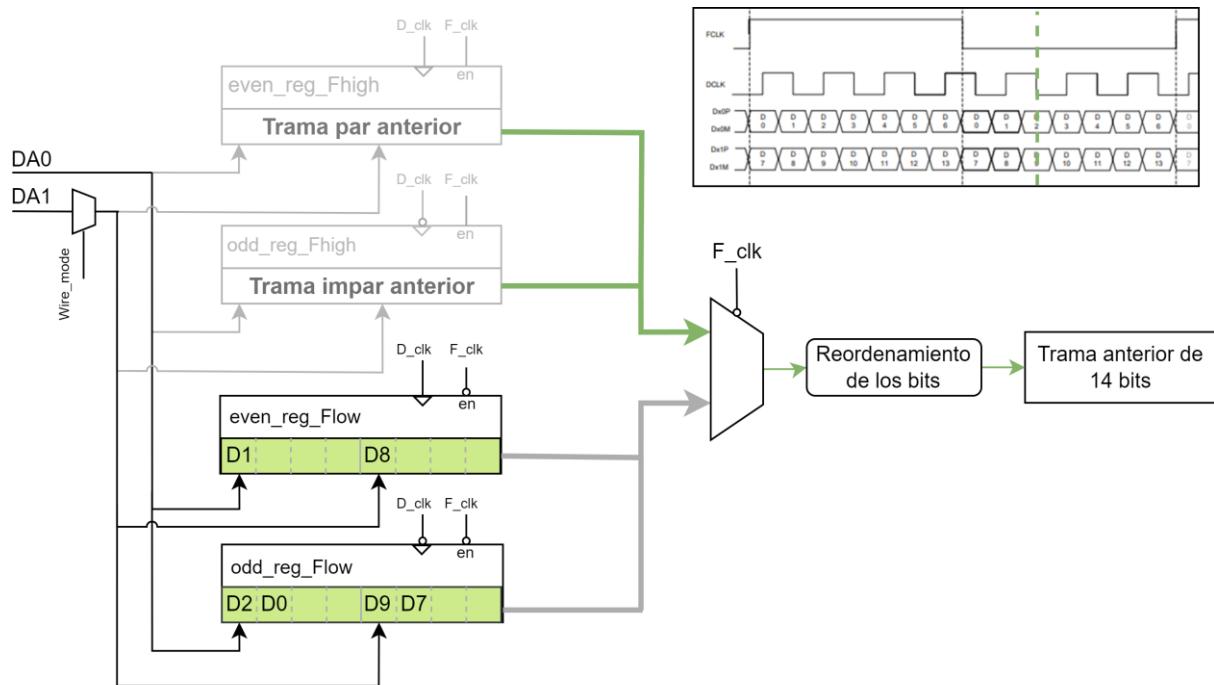


Figura 3.17: ADC3244_ACQ. Comportamiento de los registros en la adquisición ($F_{clk_i} = '0'$).

Cabe destacar que, en el caso específico del modo Two-Wire durante la señal ‘Frame Clock’ a nivel bajo, el comportamiento de los registros par e impar está invertido tal y como se observa en la **Figura 3.17**. Esto ocurre porque la nueva trama comienza con el flanco de bajada seguido del flanco de subida. Por ello, se ha tenido en cuenta que este caso requiere un reordenamiento de los bits distinto.

3.4.3 Sub-bloques. AXI Master

Para retener las tramas de datos leídas para su procesamiento futuro sin perder detalles, se requiere el uso de una FIFO como contenedor, junto con un mecanismo para la escritura. La entidad AXI Master se encarga de recibir las tramas de ambos canales y luego las envía de manera secuencial a la FIFO, acompañadas de una etiqueta identificativa para cada canal.

Por ejemplo, para el caso de un dato del canal A, 0x13B3, se almacenará en la FIFO como 0x800013B3. En caso contrario, si es del canal B será 0x400013B3

Para la comunicación entre la capa inferior de adquisición y el AXI Master, se utiliza el protocolo de comunicación detallado en la **Figura 3.18**.

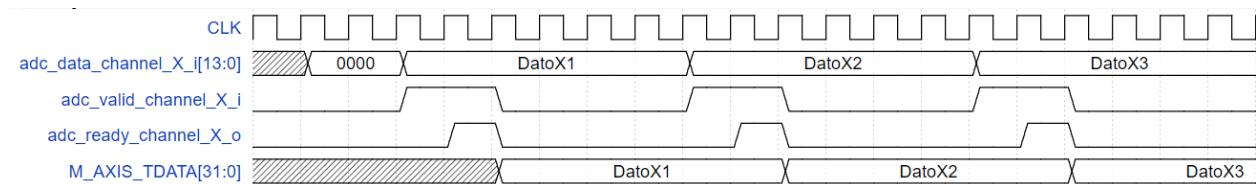


Figura 3.18: Protocolo de comunicación; DAQ - Axi Stream.

Cuando se completa el registro de un dato en el sistema de adquisición de datos (DAQ), este es recibido en la señal '*adc_data_channel_X_i[13:0]*', junto con la señal '*adc_valid_channel_X_i*', indicando la disponibilidad del dato (siendo X el canal A o B).

Una vez que el dato ha sido registrado por el AXI Master, se activa la señal '*adc_ready_channel_X_o*' con un valor de '1', comunicando al DAQ que el dato ha sido recibido y que puede dejar de transmitir la señal '*valid*'. Seguidamente, el dato previamente registrado es transferido a la FIFO a través del bus de datos '*M_AXIS_TDATA[31:0]*'. La FIFO es responsable de almacenar datos para la DMA (Acceso Directo a Memoria).

Debido a que la llegada de datos ocurre simultáneamente para los canales A y B, se priorizan los datos del canal A sobre los del canal B. Véase el ejemplo de la **Figura 3.19**.

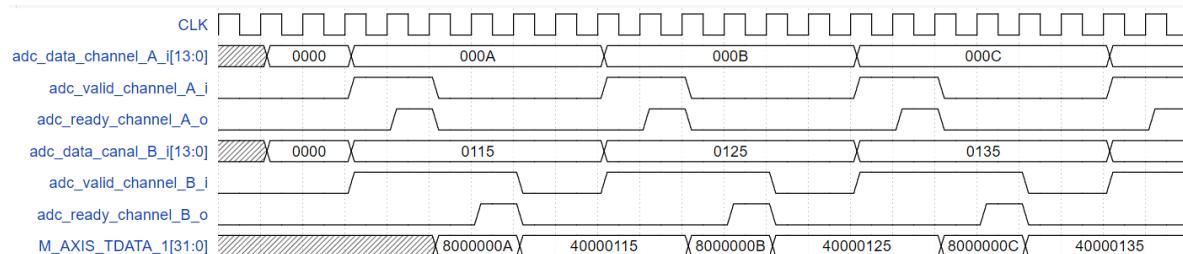


Figura 3.19: Protocolo de comunicación; DAQ - AXI Stream. Ejemplo.

3.4.4 Sub-bloques. SPI

El ADC3244 emplea el protocolo de comunicación "Serial Peripheral Interface" (SPI) para configurar sus registros. Este protocolo implica cuatro señales: MISO (Master Input, Slave Output), MOSI (Master Output, Slave Input), CS (Chip Select), y SCLK (Reloj de Muestreo).

Por consiguiente, nuestra IP específica para el ADC incluye un módulo dedicado para replicar este protocolo, acompañado por un par de FIFOs: una para transmisión (FIFO TX) y otra para recepción (FIFO RX), tal y como se detalla en la FIGURA. Estas FIFOs facilitan la acumulación de tramas a enviar y de tramas recibidas, a la

espera de ser procesadas por la capa de aplicación.

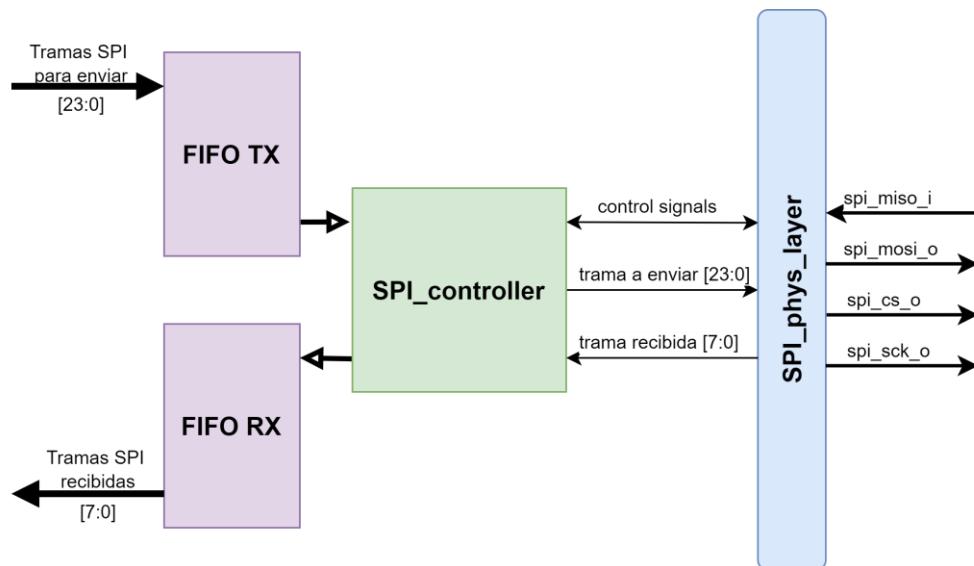


Figura 3.20: SPI. Diagrama de bloques.

4 APPLICACIÓN SOFTWARE

Es imprescindible desarrollar una plataforma intuitiva y fácil de usar para el usuario final, que simplifique tanto el tratamiento de los datos como la supervisión del sistema. Por ello, es fundamental contar con recursos de conectividad robustos, como Ethernet, que posibilitan una comunicación remota y eficiente por parte del usuario, así como una interfaz intuitiva. Esta capacidad de acceso telemático es esencial para garantizar la operatividad y la monitorización continua del sistema.

En este contexto, la elección del sistema operativo es un aspecto crítico. Poreste motivo, la elección de Petalinux se fundamenta en su versatilidad, ofreciendo una amplia gama de herramientas y funcionalidades que facilitan la gestión eficiente de recursos y la comunicación entre procesos. Además, Petalinux proporciona una plataforma sólida y estable que se adapta a las necesidades específicas de nuestro sistema.

4.1 PetaLinux

PetaLinux es una herramienta especializada desarrollada por Xilinx que desempeña un papel crucial en la creación de sistemas embebidos basados en FPGAs. Esta plataforma ofrece una integración fluida entre el software y el hardware al adaptar un entorno Linux a la arquitectura de la serie Zynq 7000. La flexibilidad de PetaLinux permite aprovechar al máximo las capacidades de las FPGAs, ofreciendo un sistema operativo robusto y adaptable para una amplia gama de aplicaciones. [6]

Durante el proceso de configuración y montaje de PetaLinux, se ha empleado el software proporcionado por Xilinx en un entorno Linux, concretamente en la distribución Ubuntu 22.04.

Además, para implementar la aplicación deseada, se necesitan realizar ciertos ajustes específicos, tanto en la generación de la imagen como en el árbol de dispositivos (device-tree). Para empezar, se trabaja con una plantilla aportada por PetaLinux para la familia de FPGAs con la que se va a trabajar. En nuestro caso, se ha seleccionado la familia Zynq, ya que el kit de desarrollo empleado presenta una FPGA de este tipo.

A continuación, para la correcta configuración de la imagen, es necesario modificar tres apartados: la configuración global, la configuración del rootfs y la configuración del kernel. La **Tabla 4.1** recoge los nombres de los comandos empleados para configurar cada una de estas secciones. A su vez, en las Tabla 4.2, Tabla 4.3 y Tabla 4.4 se recogen las principales opciones a configurar junto con una breve descripción de su significado.

Nombre de la configuración	Comando	Configuración
Global	<i>Petalinux-config</i>	Tabla 4.2
Rootfs	<i>Petalinux-config -c rootfs</i>	Tabla 4.4
Kernel	<i>Petalinux-config -c kernel</i>	Tabla 4.3

Tabla 4.1: Petalinux. Comandos para configurar.

Nombre	Selección	Descripción
ROOTFS EXT4	Si	Indica el sistema de archivos que utiliza la tarjeta SD en la que crearemos la imagen de petalinux.
SDROOT	/dev/mmcblk0p2	Directorio raíz donde se ubica la partición en la que queremos cargar la imagen
HOSTNAME	ptlnx_pico	El nombre de usuario escogido
YOCO BUILDTOOLS EXTENDED	Si	Es necesario para el uso de algunas herramientas de versiones superiores.

Tabla 4.2: Petalinux. Configuración global.

Nombre	Selección	Descripción
DMA	DMADEVICES DMA Engine support DMABUF	Habilita el uso del dispositivo DMA y su controlador integrado en el kit de desarrollo.
UIO DRIVER	PDRV GENIRQ DMEM GENIRQ	Habilita el controlador para el manejo de interrupciones

Tabla 4.3: Petalinux. Configuración del kernel.

Nombre	Selección	Descripción
FILESYSTEM PACKAGE	admin	sudo Necesario para el acceso a comandos como administrador
	base	busybox i2c-tools netbase shell → bash tar tzdata util-linux
	console → utils	bzip2 file gawk git grep gzip screen sed unzip vim zip Aplicaciones, bibliotecas y módulos esenciales para el correcto funcionamiento del sistema, la comunicación con el usuario y la gestión en tiempo real.
	devel	autoconf automake binutils diffstat make gmp gnu-config
	misc	ger gcc-runtime gdb glib-2.0 packagegroup-core powertop tcf-agent
PACKAGEGROUP-PETALINUX	Sí lmsensors networking-debug networking-stack utils	Herramientas adicionales específicas de petalinux
MODULES	dma_module	Habilitar módulo desarrollado y detallado en 4.2

Tabla 4.4: Petalinux. Configuración del rootfs.

Para el caso del device-tree, se debe añadir el controlador UIO al kernel y configurarlo para que se active automáticamente en cada arranque del sistema. Este controlador, conocido como UIO (User-space I/O), es esencial ya que permite al kernel gestionar las interrupciones generadas por el hardware, proporcionando una interfaz eficiente y directa para el manejo de eventos críticos en el sistema.

4.2 Módulo del kernel para la DMA

Debido a que las interrupciones detectadas por el controlador UIO solo pueden ser manejadas en el espacio del kernel, surge la necesidad de implementar un módulo adicional que supervise estas interrupciones y las convierta en señales que puedan procesarse en el espacio de usuario. Este módulo desempeña un papel crucial al actuar como intermediario entre la capa hardware, que genera las interrupciones, y la capa de aplicación, que necesita responder a ellas.

En este contexto, se requiere un componente que esté especialmente diseñado para leer la interrupción con el identificador 54, que está asociado a la interrupción generada por la DMA. Una vez detectada, el módulo debe estar configurado para enviar una señal específica, denominada SIGUSR1, al proceso denominado "pico1024", que constituye el proceso de nuestra aplicación. Véase la **Figura 4.1** relativa a esta explicación.

Este proceso de intermediación asegura una comunicación fluida y eficiente entre el hardware y el software de la aplicación, permitiendo una respuesta rápida y precisa a las interrupciones generadas por el sistema.

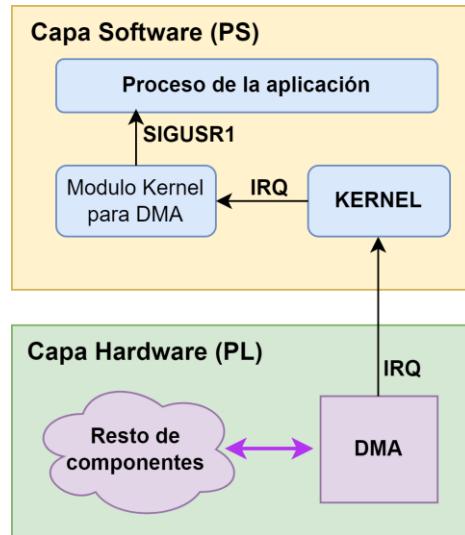


Figura 4.1: Modulo Kernel. Esquema conceptual de su utilidad.

4.3 Aplicación en C

El principal objetivo es disponer de una aplicación sencilla para el usuario y que, a su vez, nos permita comprobar los diferentes aspectos de la adquisición de datos para, así, asegurar su correcto funcionamiento. La aplicación incorpora un menú básico que nos permite seleccionar entre varios tipos de pruebas que podemos realizar, así como un arranque completo del sistema (PICO1024 y ADC3244) para capturar y almacenar los datos (píxeles) generados por el sensor de imagen.

La aplicación se compone jerárquicamente del siguiente conjunto de archivos, **Figura 4.2**.

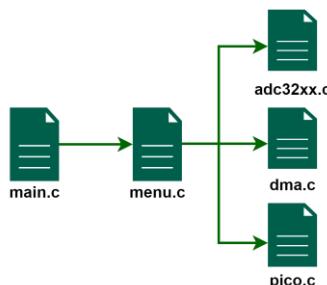


Figura 4.2: Aplicación. Esquema de ficheros.

- **Main.c:** La función ‘main’ inicializa el sistema de menús y gestiona la navegación a través de ellos. Además, se encarga de limpiar la terminal y muestrear continuamente el menú actual hasta que el usuario realiza una selección. Si el menú actual no tiene submenús, ejecuta la tarea asociada con él y regresa al menú principal.
- **Menu.c:** Se encarga de mostrar el texto de cada menú y submenú asociado. Además, ejecuta las tareas asociadas a cada opción haciendo uso de las funciones declaradas en los archivos de cabecera “Adc32xx.h”, “DMA.h” y “Pico.h”. Cada tarea se compone de 5 partes: petición de parámetros específicos de la prueba al usuario, mapeado en memoria de los componentes a utilizar, escritura de los registros de cada componente, inicio de la prueba y apagado de los componentes.
- **Adc32xx.c:** Este archivo contiene las funciones para controlar la familia de dispositivos ADC32XX. Incluye funciones para leer y escribir los registros, inicializar el dispositivo y leer los datos obtenidos por el convertidor
- **DMA.c:** Este archivo contiene la implementación de un controlador DMA. Proporciona las funciones necesarias para la configuración y arranque de la DMA. Además, tras completarse una zona de memoria, se encarga de reubicar la DMA al mismo tiempo que almacena dichos datos en un fichero para su posterior procesamiento.
- **Pico.c:** Este archivo contiene las funciones para controlar el dispositivo PICO1024. Incluye funciones para leer y escribir los registros, inicializar el dispositivo y cambiar su configuración interna.

Cabe iniciar que cada fichero de código fuente contiene una descripción exhaustiva de las funciones y parámetros desarrollados, siguiendo las directrices de escritura y documentación del código recogidas en el proyecto. Esta organización facilita que el lector pueda consultar los archivos individuales, donde encontrará tanto descripciones generales del código como explicaciones detalladas de cada función. Adicionalmente, para facilitar la lectura y su accesibilidad, se ha generado documentación complementaria utilizando Doxygen. Este software es una herramienta avanzada de documentación para lenguajes de programación. Esta documentación es especialmente útil para obtener una perspectiva global y coherente del proyecto, permitiendo una navegación eficaz y un entendimiento profundo de la estructura y funcionalidad del código.

4.3.1 Estructura de la aplicación

La aplicación sigue el comportamiento descrito en el diagrama de flujo expuesto en la **Figura 4.3**. En los siguientes puntos se describirá cada uno de los submenús junto con el diagrama de flujo de la entidad “TAREA” del diagrama.

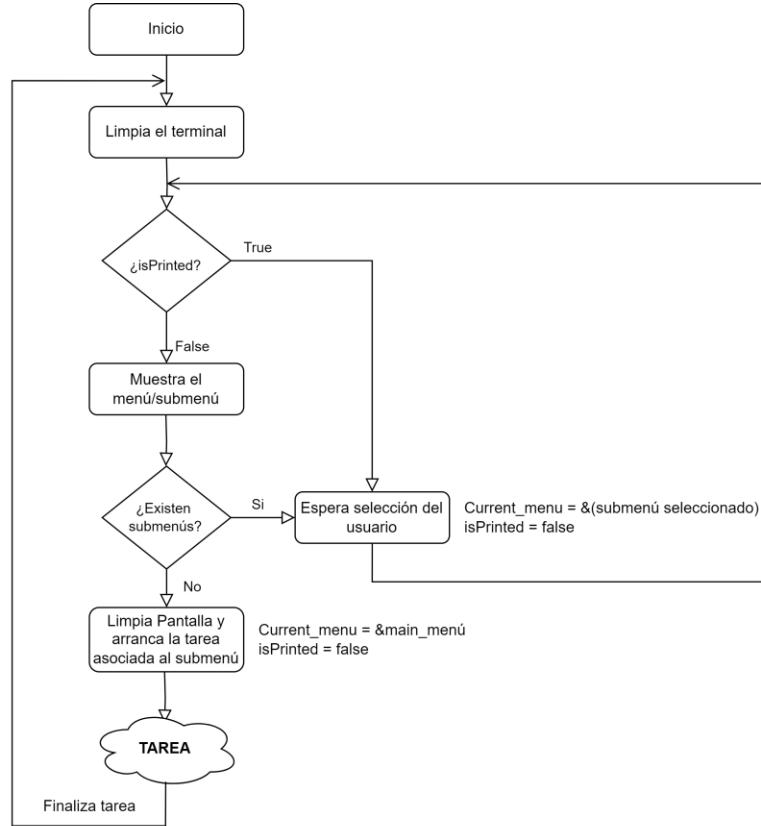


Figura 4.3: Aplicación. Diagrama de flujo.

Inicialmente, para el arranque de la aplicación es necesario que sea mediante el administrador, ya que se requieren permisos de acceso a memoria. Una vez iniciado se desplegará el menú expuesto en la **Figura 4.4**

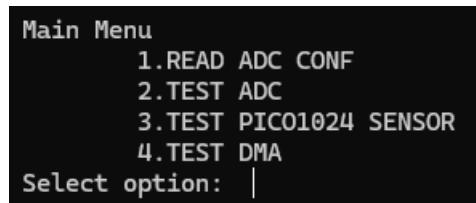


Figura 4.4: Aplicación. Menú.

4.3.1.1 READ ADC CONF

Esta prueba realizará una escritura en todos los registros del ADC3244 y, posteriormente, una lectura. Con esto se pretende comprobar que la comunicación SPI entre el ADC3244 y la ZC702 es correcta. La configuración enviada son unos valores default declarados en una matriz del fichero “main.c”. El diagrama de flujo relativa a esta tarea se expone en la **Figura 4.5**.

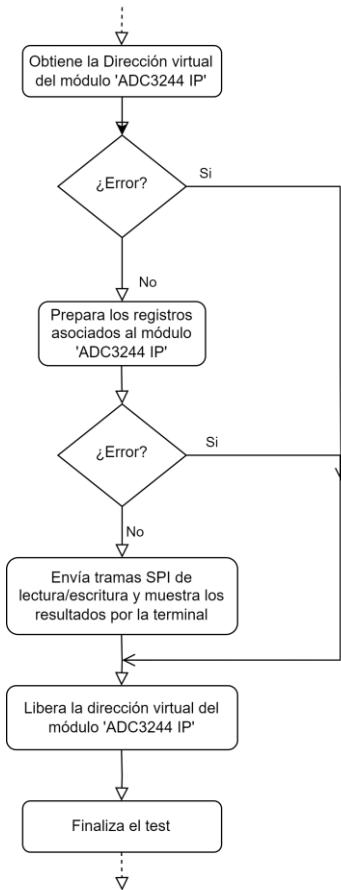


Figura 4.5: Aplicación. Diagrama de flujo de la tarea READ ADC CONF.

4.3.1.2 TEST ADC

Cuando seleccionamos esta prueba, se nos abrirá el submenú expuesto en la **Figura 4.6**.

```

Main Menu
 1.READ ADC CONF
 2.TEST ADC
 3.TEST PICO1024 SENSOR
 4.TEST DMA
Select option: 2
 2.TEST ADC
    2.1.Toggle pattern: 0x2AAA - 0x1555
    2.2.Custom pattern
    2.3.Digital ramp: 0x0000 -> 0x3FFF
    2.4.Sine-wave
Select option: |
  
```

Figura 4.6: Aplicación. TEST ADC.

El objetivo de esta prueba es comprobar el comportamiento del ADC3244 y la correcta adquisición de los datos por parte de nuestra Plataforma.

- **2.1 Toggle Pattern:** Permite verificar si estamos recibiendo la trama 0x2AAA y 0x1555 que comutan constantemente entre uno y otro.

- **2.2 Custom Pattern:** Permite solicitar al ADC3244 que mande una trama de 14 bits específica.
- **2.3 Digital Ramp:** Para esta prueba es recomendable asegurar que la DMA está funcionando correctamente (véase la sección **4.3.1.4 TEST DMA**). El ADC comenzará a transmitir los valores de un contador de 14 bits.
- **2.4 Sinewave:** El ADC envía los valores de una señal sinusoidal.

Cuando se elige uno de los modos disponibles, se solicitará ciertos parámetros para la configuración. Estos incluyen la selección del canal que se va a utilizar (canal A, B o ambos), el método de comunicación a emplear (1 wire o 2 wire), la cantidad de tramas o paquetes DMA que se desean capturar y, en el caso del modo 'Custom Pattern', la trama específica que se desea recibir.

Una vez que se han seleccionado los parámetros requeridos, la aplicación procederá a mapear la zona de memoria correspondiente al ADC para realizar la escritura de las tramas SPI necesarias. Estas tramas se utilizan para configurar los registros internos del ADC3244. Si la configuración se lleva a cabo sin inconvenientes y todo es correcto, se procede a realizar la escritura en el registro de control para inicializar el ADC. Posteriormente, se leen los datos recibidos a través de los registros asociados a los canales A y B.

El diagrama de flujo relativa a esta tarea se expone en la **Figura 4.7**. Cabe destacar que, en función del submenú seleccionado, se realizará una lectura de los registros del módulo del ADC3244 IP o se realiza un volcado de la información escrita por la DMA en la memoria un fichero de texto. Dicho fichero será generado en la ubicación de los archivos de la aplicación

Por último, cabe destacar que en cualquiera de las pruebas se debe utilizar el módulo 'PICO1024 IP' ya que es el encargado de generar la señal de reloj SC detallada en el punto **2.2.3**. Este reloj es necesario para el ADC3244 ya que marca los relojes 'Frame Clock' y 'Data Clock' para la adquisición de las tramas, como se especifica en el punto **2.3.2**.

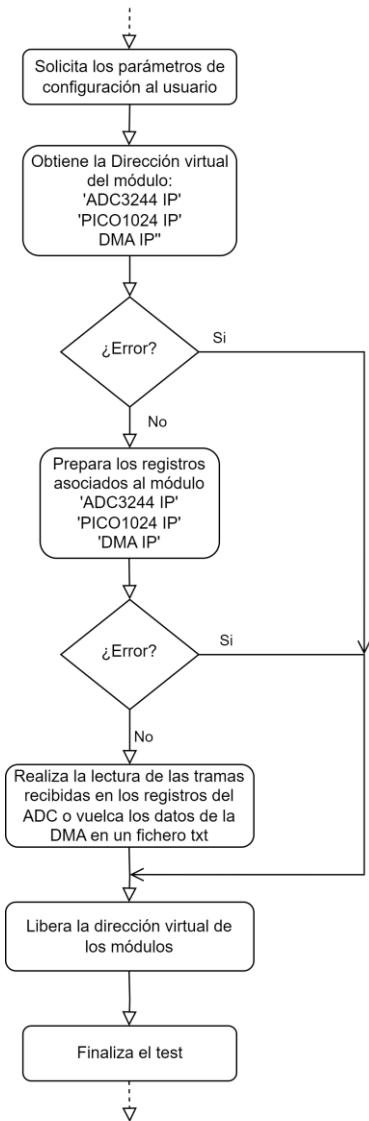


Figura 4.7: Aplicación. Diagrama de flujo de TEST ADC.

4.3.1.3 TEST PICO1024 SENSOR

Cuando se selecciona este menú, se desplegará el submenú mostrado en la **Figura 4.8**. Con la primera opción puede seleccionar de forma sencilla los valores de configuración que enviaremos al PICO1024. Este cuenta con dos modos de configuración según si queremos el modo de compatibilidad UL05251 o el modo nominal PICO1024. Este cambio se realiza tan solo indicando el valor ‘1’ o ‘0’ en la función “PICO_MODE”. Esta diferenciación entre un modo u otro es necesaria ya que el protocolo de comunicación cambia ligeramente. Como puede verse en el dataheet, para el modo UL05251 tenemos una trama de datos a enviar más corta que en el caso del modo PICO1024, véase **Figura 2.7** y **Figura 2.8**.

Una vez sea definida la configuración, seleccionando la segunda opción del submenú se iniciará una prueba sobre el sensor. De este modo, se creará un fichero de texto donde se almacenarán los datos capturados por la DMA de todos los píxeles de cada imagen. Posteriormente, habrá que comprobar los datos del fichero con ayuda de algún software que nos permita visualizarlo (Ejemplo: Matlab).

```

Main Menu
1.READ ADC CONF
2.TEST ADC
3.TEST PICO1024 SENSOR
4.TEST DMA
Select option: 3
3.TEST PICO1024 SENSOR
    3.1.Change PICO1024 configuration
    3.2.Start PICO1024 test
    3.3.Run MC
Select option: |

```

Figura 4.8: Aplicación. TEST PICO1024 SENSOR.

TEST PICO1024: CHANGE CONFIGURATION					TEST PICO1024: CHANGE CONFIGURATION				
Actual Configuration: UL05251 Compatibility mode					Actual Configuration: PICO1024 mode				
ID	Functions	MaxSize	DefaultValue	ActualValue	ID	Functions	MaxSize	DefaultValue	ActualValue
01	CALOC	01	0x000	0x000	01	CALOC	01	0x000	0x000
02	SELQ	03	0x000	0x000	02	SELQ	04	0x000	0x000
03	VALQ	01	0x001	0x001	03	VALQC	01	0x000	0x000
04	VALOC	01	0x000	0x000	04	GAIN	02	0x002	0x002
05	GAIN	02	0x002	0x002	05	UPCOL	01	0x001	0x001
06	UPCOL	01	0x001	0x001	06	UPROW	01	0x001	0x001
07	UPROW	01	0x001	0x001	07	SIZEA	01	0x001	0x001
08	SIZEA	01	0x001	0x001	08	SIZEB	01	0x001	0x001
09	SIZEB	01	0x001	0x001	09	YFIRST	10	0x000	0x000
10	YFIRST	10	0x000	0x000	10	YLAST	10	0x2FF	0x2FF
11	YLAST	10	0x2FF	0x2FF	11	XFIRST	09	0x000	0x000
12	XFIRST	09	0x000	0x000	12	XLAST	09	0x1FF	0x1FF
13	XLAST	09	0x1FF	0x1FF	13	PICO_MODE	01	0x000	0x001
14	PICO_MODE	01	0x000	0x000	14	MODE_4OUTPUTS	01	0x000	0x000
16	Default				16	Default			
17	Exit				17	Exit			
ID conf:					ID conf:				

Figura 4.9: Aplicación. Cambiar la configuración del PICO1024.

El diagrama de flujo relativa a la tarea de configuración se expone en la **Figura 4.10**.

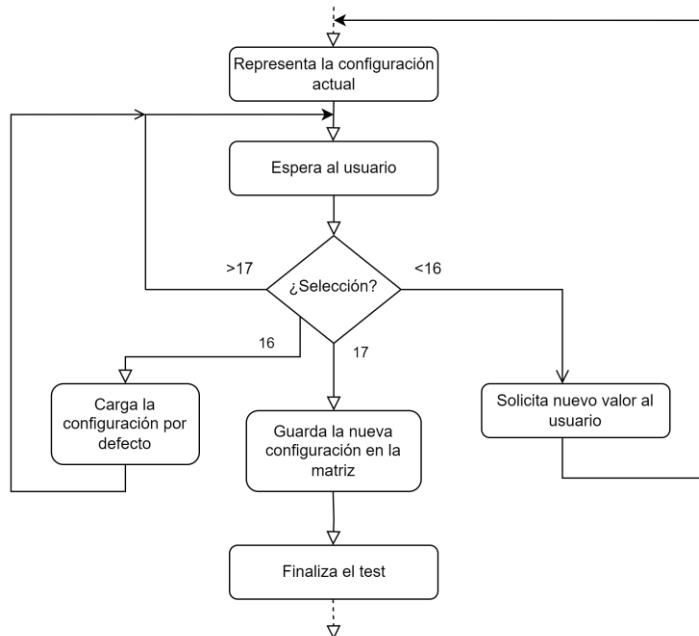


Figura 4.10: Aplicación. Diagrama de flujo de la configuración.

El diagrama de flujo relativa a la tarea de la prueba del PICO1024, se expone en la Figura 4.11.

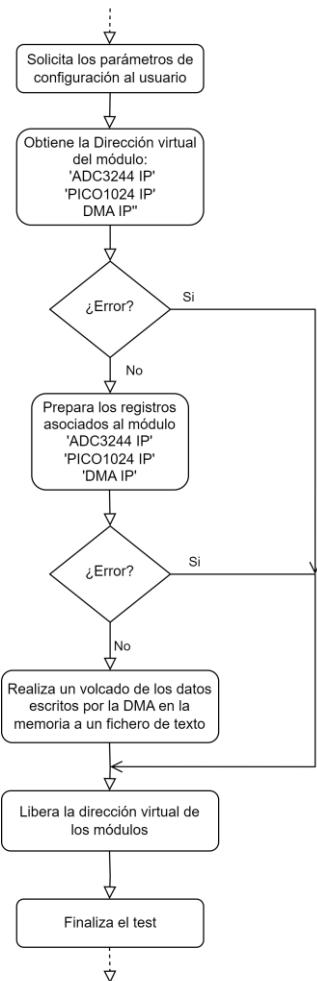


Figura 4.11: Aplicación. Diagrama de flujo de la prueba PICO1024.

4.3.1.4 TEST DMA

El objetivo de la prueba es realizar una comprobación del funcionamiento de la DMA con el ADC transmitiendo tramas constantes. Al principio se nos pedirán algunos parámetros ya comentados anteriormente como el canal a utilizar (canal A, B o ambos), el método de comunicación a emplear (1 wire o 2 wire), la cantidad de tramas o paquetes DMA que se desean capturar y la trama específica que se desea recibir. Cabe destacar que la cantidad de datos que se recibirá será $N_{paquetes} * TamañoDMA * N_{repeticiones}$.

TamañoDMA = 32 (Este valor está definido en el código. Su modificación requiere recomilar el programa)

$N_{repeticiones}$ = 10 (Valor definido en el programa. Puede aumentarse)

Por ejemplo, para un numero de paquetes de 16, tendremos 5120 tramas de datos.

El diagrama de flujo relativa a esta tarea es el igual a la **Figura 4.11** con la diferencia de que se ajustan los registros internos del ADC para transmitir una trama constante.

5 VALIDACIÓN DE LA PLATAFORMA DE ADQUISICIÓN IMPLEMENTADA

A lo largo del proyecto se han realizado una serie de pruebas que nos han permitido verificar la correcta funcionalidad de cada componente y su integración completa en el sistema. Esta validación de la plataforma se ha llevado a cabo mediante la evaluación del banco de pruebas descrito en la Tabla 1. Cada una de estas pruebas está orientada a validar el correcto funcionamiento de un módulo del sistema. Nótese que en aquellas pruebas que hacen uso del hardware, se ha empleado la aplicación de usuario descrita en el párrafo 4.3.1, validando su funcionamiento.

Número	Descripción
1	Comprobación de la descripción VHDL desarrollada por medio de un testbench
2	Lectura/Escritura de los registros del ADC
3	Lectura de tramas personalizadas del ADC3244
4	Uso de la DMA para la lectura de un contador programado en el ADC3244
5	Aplicación de la configuración al PICO1024
6	Señal sinusoidal a la entrada del ADC3244 para determinar el ruido del hardware de adaptación.
7	Lectura de los píxeles del PICO1024 por medio del ADC3244 (Sistema Completo)

Tabla 5.1: Descripción del banco de pruebas.

Por último, es importante resaltar que en las pruebas realizadas donde se ha empleado el hardware descrito, se ha introducido el módulo ILA (Integrated Logic Analyzer) de Xilinx. Esta es una herramienta de depuración dentro de la suite de herramientas Vivado que nos permite observar y analizar el comportamiento interno en tiempo real. Gracias a él, hemos podido seleccionar y visualizar señales específicas, capturar datos, establecer disparadores y facilitar el análisis para identificar y solucionar problemas de lógica en el diseño.

Para el contexto de las medidas ha sido una herramienta muy útil que nos ha permitido determinar retrasos y/o verificar el correcto comportamiento del sistema. Por ello, en alguna de las pruebas se incorporarán capturas de datos por medio de dicha herramienta.

5.1 Prueba 1: Comprobación de la descripción VHDL desarrollada por medio de un testbench

Realizar un testbench en Vivado es esencial para asegurar la correcta funcionalidad del diseño antes de su implementación física. Este proceso permite verificar y depurar mediante simulaciones que aplican diversos estímulos, facilitando la detección de errores.

Para ello se ha diseñado un testbench con la capacidad de simular el comportamiento de las señales del ADC3244 (véase el punto 3.4 donde se detallan las señales de entrada de la IP adc3244). Además, para replicar el comportamiento de la escritura y lectura de registros por parte de la CPU, se ha utilizado la IP desarrollada por Xilinx, ‘Traffic Generator’. El diagrama completo de esta prueba queda como el expuesto en la **Figura 5.1**.

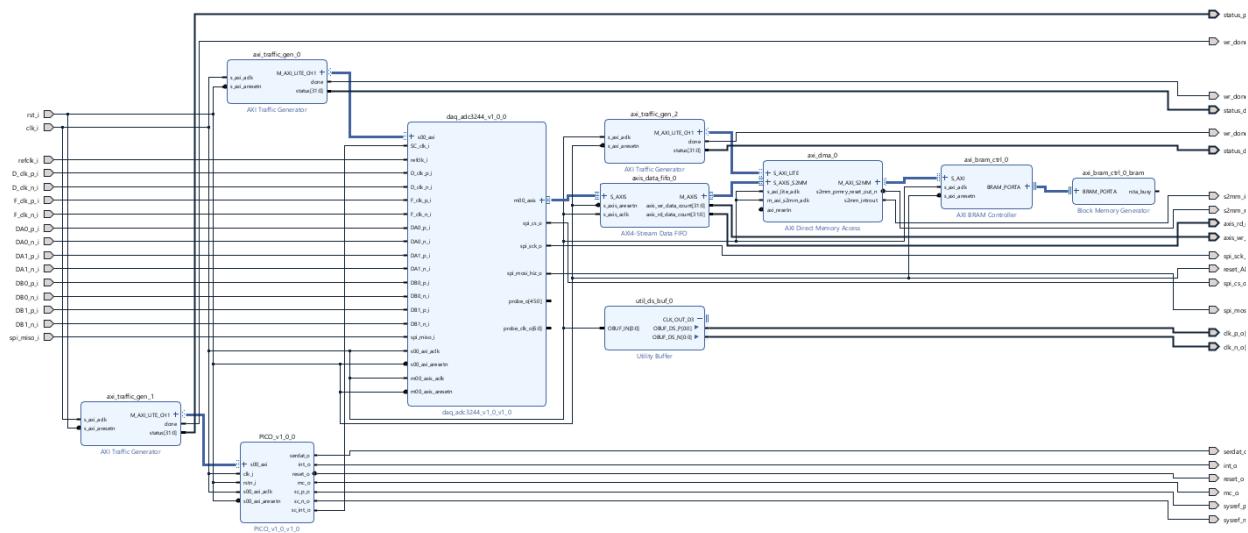


Figura 5.1: Diagrama del Testbench.

Con esta simulación podremos verificar la funcionalidad teórica de las entidades desarrolladas y los tiempos de respuesta. Entre los resultados obtenidos se destacan los siguientes puntos:

- Resultados del ADC3244 IP en canal A.

En la **Figura 5.2** podemos ver como la señal analógica introducida al ADC es un contador. Tras la adquisición de los bits en los registros de desplazamiento ‘c_even_reg_Fhigh’ y ‘c_odd_reg_Fhigh’, para el caso de ‘Frame clock’ a nivel alto, y ‘c_even_reg_Flow’ y ‘c_odd_reg_Flow’, para el caso de nivel bajo, las tramas del contador se ven reflejadas en el registro de salida de 14 bits, ‘c_adc_reg’. (Véase la **Figura 5.3**. (Véase la **Figura 5.3**. Error! No se encuentra el origen de la referencia. para entender el funcionamiento de la adquisición). Finalmente, se comprueba también el modo One-Wire en la **Figura 5.3**.

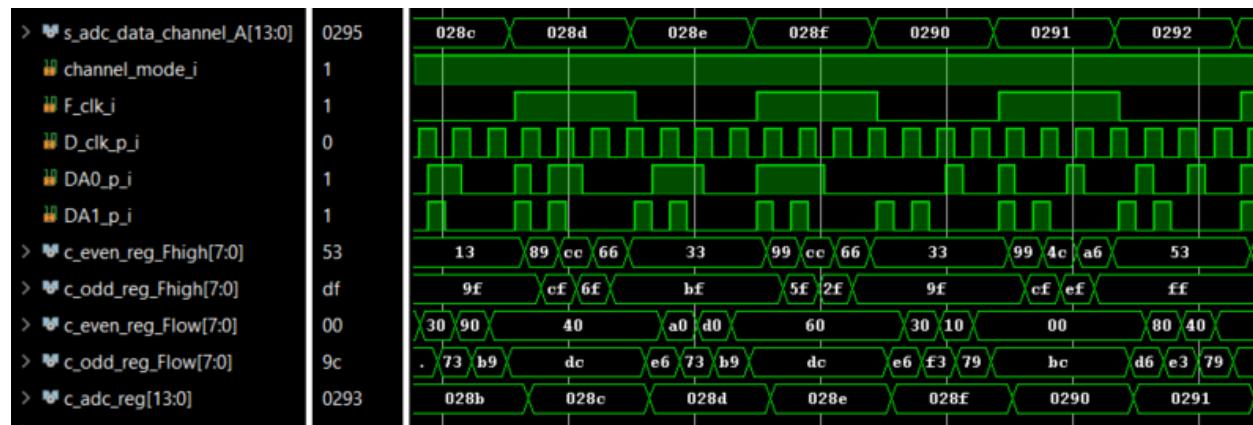


Figura 5.2: Prueba 1. Resultados del ADC3244 IP para Two-Wire.

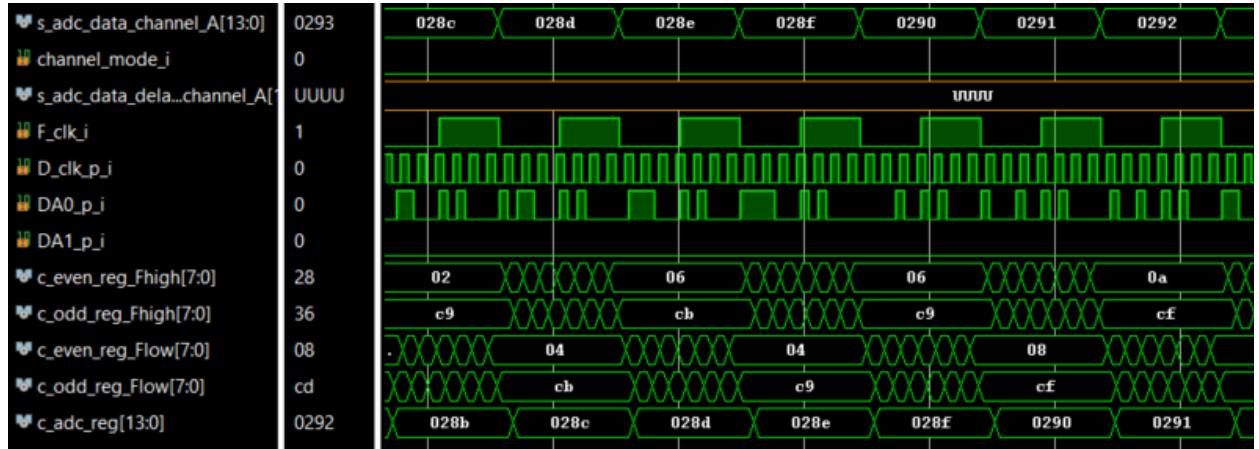


Figura 5.3: Prueba 1. Resultados del ADC3244 IP para One-Wire.

- **Resultados del PICO IP**

En los resultados se busca verificar los distintos estados descritos en el diagrama de la **Figura 3.9**. Por lo tanto, en la **Figura 5.4**, podemos observar el envío de los datos para la configuración del componente según el protocolo definido en el punto **2.2.2 Configuración**. Luego, en la **Figura 5.5** podemos observar el estado de operación descrito en el punto **2.2.3 Operación**. Por último, se verifica en la **Figura 5.6** el correcto apagado del dispositivo tal y como se indica en la **Figura 2.12**.

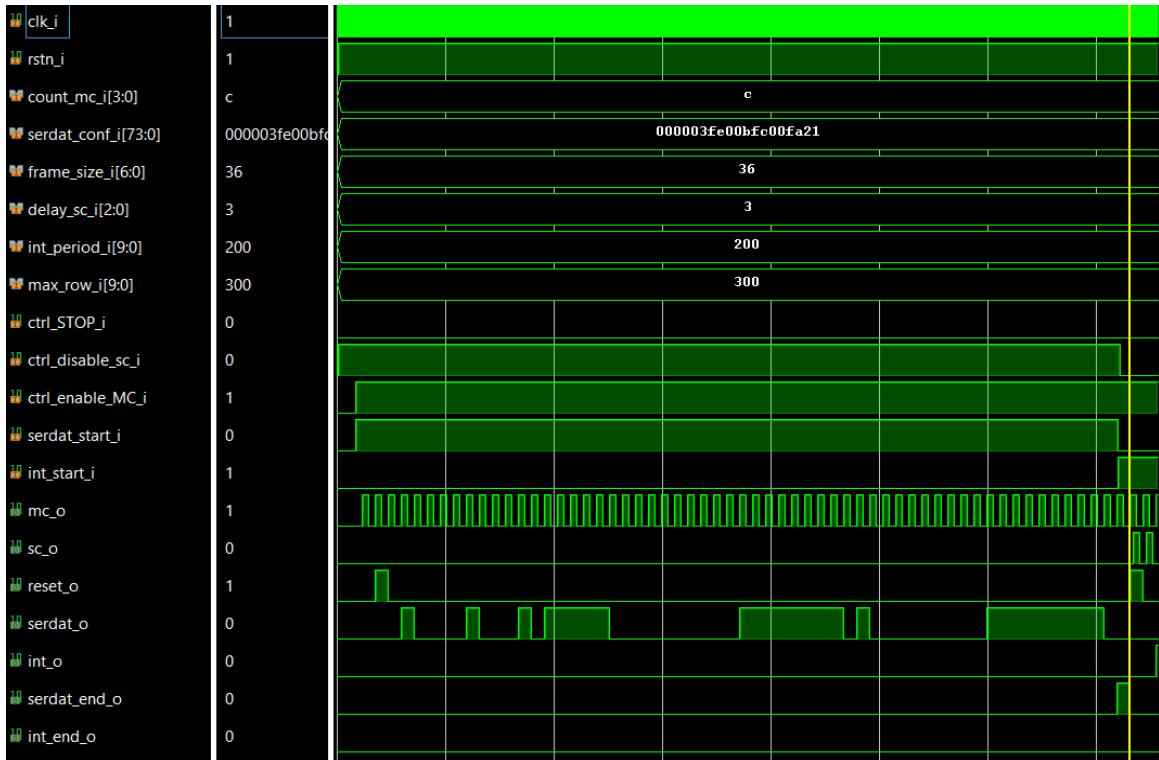


Figura 5.4: Prueba 1. Resultados del protocolo de configuración del PICO IP.

En la **Figura 5.5**, se muestra el comportamiento del sistema en el estado de operación.

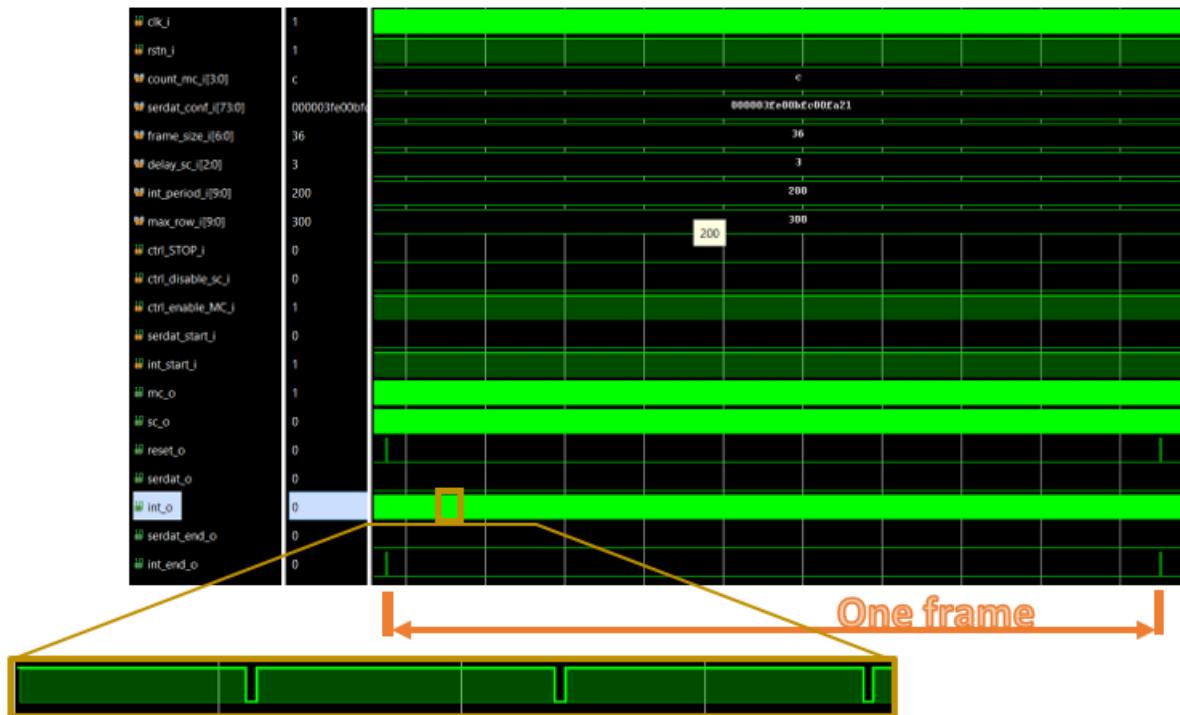


Figura 5.5: Prueba 1. Resultados del modo de operación del PICO IP.

En la **Figura 5.6** se muestra el comportamiento del sistema tras indicar una parada y arranque.

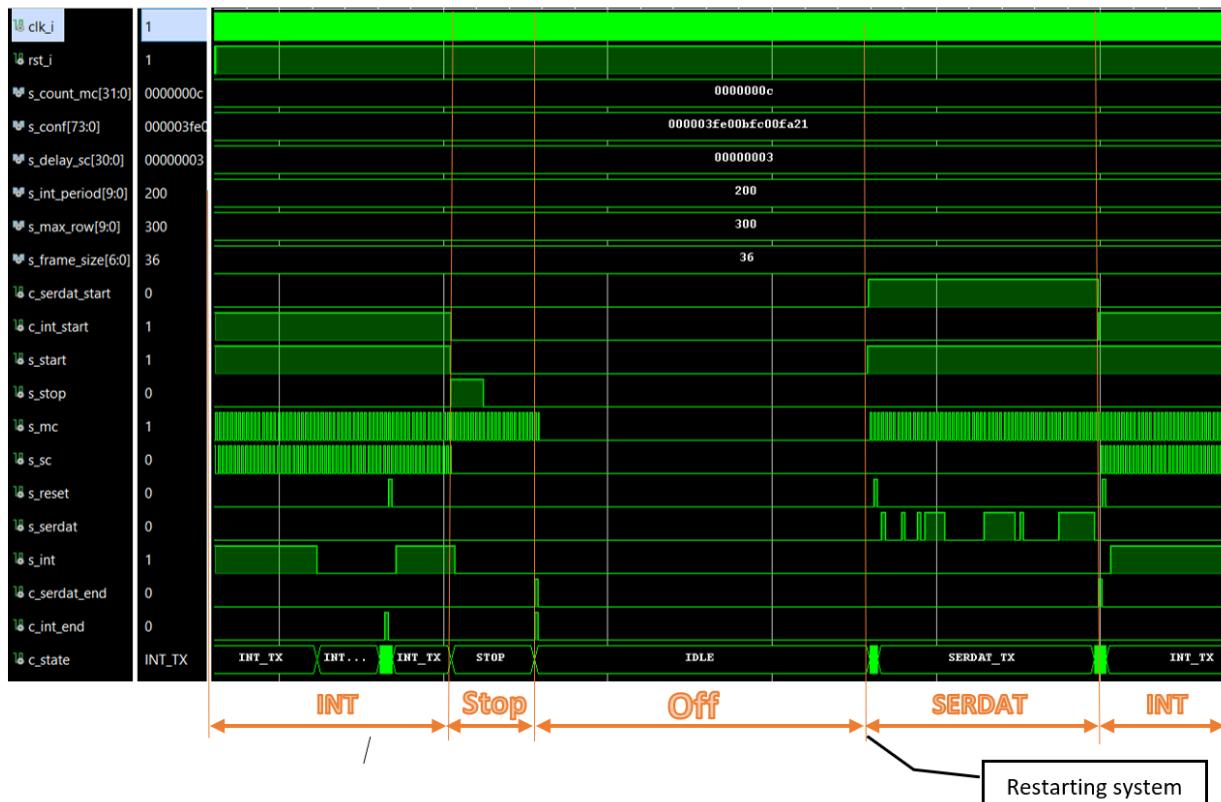


Figura 5.6: Prueba 1. Resultados del stop del PICO IP.

- **Resultados de la DMA**

En los resultados se busca comprobar que la DMA envía 32 paquetes de 32 tramas de 32 bits cada una y, además, que no pierda ningún dato durante el proceso. Para verificar esto último, en la simulación se dispone de un contador en el canal A y un valor constante en hexadecimal 0x1222 en el canal B. Primero, se observa en la **Figura 5.7** que la DMA está leyendo los datos de la FIFO hasta completar la cantidad de bytes solicitados (en este caso $32 \times 32 \times 4 = 4096$ bytes).

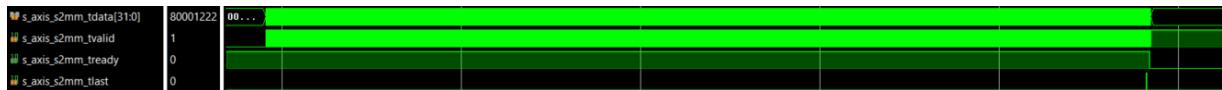


Figura 5.7: Prueba 1. Comunicación DMA – FIFO.

Se muestra a continuación, en la **Figura 5.8**, una ampliación de la **Figura 5.7** para apreciar con más detalle como los datos de las tramas entre el canal A y canal B se alternan.

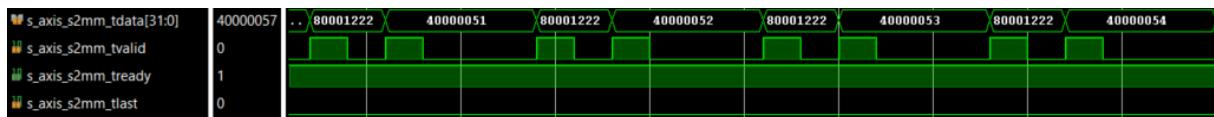


Figura 5.8: Prueba 1. Comunicación DMA – FIFO (Ampliación).

Luego, en la **Figura 5.9** se puede confirmar que se envían 32 paquetes a la dirección 0x0F000000 y una señal de interrupción en el momento en el que finaliza la transmisión. Además, realizando una ampliación en uno de los paquetes, véase la **Figura 5.11**, se verifica que envía 32 datos correctamente alternados entre canal A y canal B.



Figura 5.9: Prueba 1. Resultado del envío de paquetes en la DMA IP.

Se realiza una ampliación de la **Figura 5.10** donde se puede apreciar las 32 tramas.

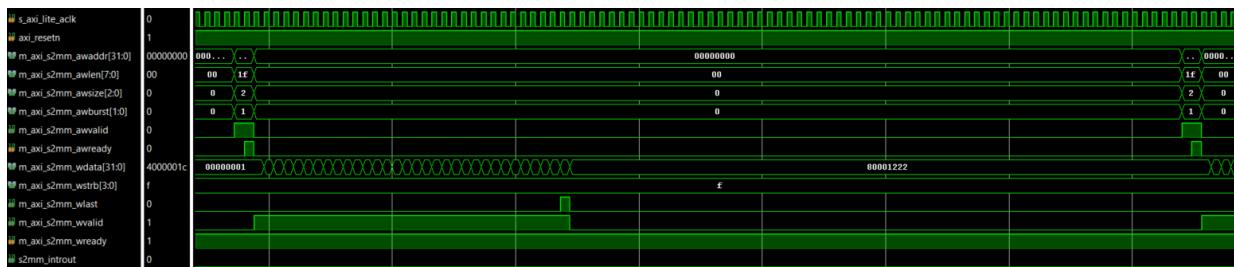


Figura 5.10: Prueba 1. Resultados del envío de paquetes en la DMA IP (Un paquete).

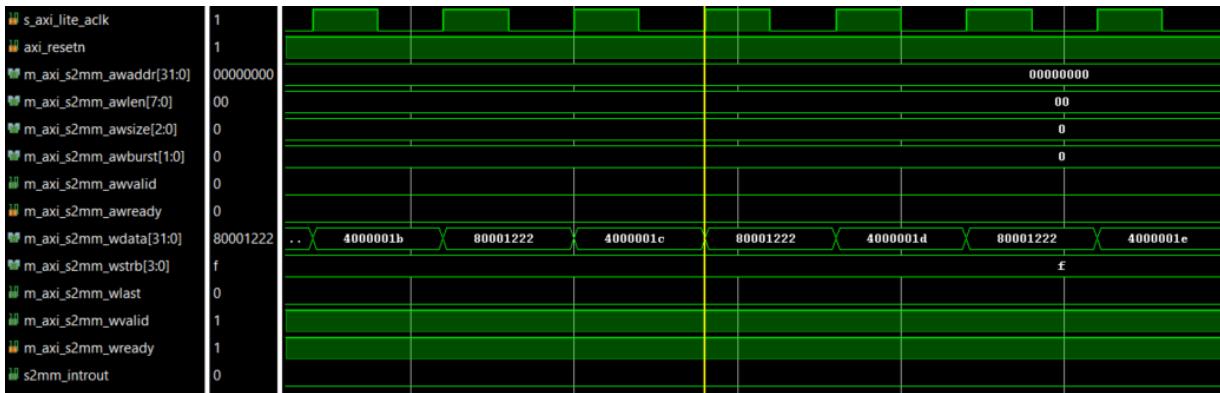


Figura 5.11: Prueba 1. Resultados del envío de paquetes en la DMA IP (Ampliación).

5.2 Prueba 2: Lectura/Escritura de los registros del ADC3244

Para esta prueba, se dispone de la imagen en Petalinux cargada en una tarjeta SD y correctamente arrancada en la ZC702. El objetivo será realizar una escritura en los registros del ADC3244 IP para enviar tramas SPI tanto de escritura como lectura.

El resultado obtenido muestra que el sistema ha sido capaz de reiniciar, escribir y leer registros internos del dispositivo ADC3244. Se muestra una pequeña imagen del proceso manual en la **Figura 5.12**.

```

ptlnx_pico_zc702:~$ sudo devmem 0x40001004 32
Password:
0x00000001
ptlnx_pico_zc702:~$ sudo devmem 0x40000004 32
0x00000141
ptlnx_pico_zc702:~$ sudo devmem 0x40000000 32 0x00000050
ptlnx_pico_zc702:~$ sudo devmem 0x40000008 32 0x00400601
ptlnx_pico_zc702:~$ sudo devmem 0x40000008 32 0x00C00600
ptlnx_pico_zc702:~$ sudo devmem 0x40000004 32
0x00000441
ptlnx_pico_zc702:~$ sudo devmem 0x40000000 32
0x00000000
ptlnx_pico_zc702:~$ sudo devmem 0x40000008 32 0x00400602
ptlnx_pico_zc702:~$ sudo devmem 0x40000008 32 0x00C00600
ptlnx_pico_zc702:~$ sudo devmem 0x40000004 32
0x00000441
ptlnx_pico_zc702:~$ sudo devmem 0x40000000 32
0x00000002
ptlnx_pico_zc702:~$ █

```

Leemos el estado actual:
No hay datos en cola

¡Hay datos en cola!

Leemos el dato recibido por SPI

¡Hay datos en cola!

Leemos el dato recibido por SPI

Escribimos 0x01 en el registro interno 0x006 para indicar el reset

Leemos en el registro interno 0x006

Escribimos 0x02 en el registro interno 0x006

Leemos en el registro interno 0x006

Figura 5.12: Prueba 2. Resultado de la escritura/lectura de registros internos del ADC3244.

5.3 Prueba 3: Lectura de tramas personalizadas del ADC3244

En esta prueba, se busca configurar los registros internos del ADC para manejar tramas personalizadas, incluyendo tipos constantes, de contador y senoidales.

Inicialmente, es necesario ajustar el ADC para que transmita la trama deseada a través del canal A, modificando los registros internos 0x06, 0x0A, 0x0E y 0x0F, tal como se ilustra en la **Figura 5.13** (la función de estos registros se detalla en la sección **2.2.2 Configuración**). El valor leído en la dirección 0x400000010 debe coincidir con la concatenación del patrón escrito en los registros '0x0E' y '0x0F', desplazados 2 bits a la derecha. Por ejemplo, para las tramas en color naranja, al escribir '0x68' en el registro 0x0E y '0xF0' en el registro 0x0F, el resultado sería la concatenación de estos valores, '0x68F0', y desplazado dos posiciones a la derecha, resultando en '0x1A3C'.

```
ptlnx_pico_zc702:~$ sudo devmem 0x40000000 32 0x00000050
ptlnx_pico_zc702:~$ sudo devmem 0x40000008 32 0x00400601
ptlnx_pico_zc702:~$ sudo devmem 0x40000008 32 0x00400A05
ptlnx_pico_zc702:~$ sudo devmem 0x40000008 32 0x00400902
ptlnx_pico_zc702:~$ sudo devmem 0x40000008 32 0x00400E68
ptlnx_pico_zc702:~$ sudo devmem 0x40000008 32 0x00400FF0
ptlnx_pico_zc702:~$ sudo devmem 0x40000008 32 0x00400602
ptlnx_pico_zc702:~$ sudo devmem 0x40001000 32 0x40000000
ptlnx_pico_zc702:~$ sudo devmem 0x40000010 32
0x00001A3C
ptlnx_pico_zc702:~$ sudo devmem 0x40000010 32
0x00001A3C
ptlnx_pico_zc702:~$ sudo devmem 0x40000010 32
0x00001A3C
ptlnx_pico_zc702:~$ sudo devmem 0x40000008 32 0x00400EA3
ptlnx_pico_zc702:~$ sudo devmem 0x40000010 32
0x000028FC
ptlnx_pico_zc702:~$ sudo devmem 0x40000010 32
0x000028FC
ptlnx_pico_zc702:~$ sudo devmem 0x40000008 32 0x00400EE9
ptlnx_pico_zc702:~$ sudo devmem 0x40000010 32
0x00003A7C
ptlnx_pico_zc702:~$ sudo devmem 0x40000008 32 0x00400E0A
ptlnx_pico_zc702:~$ sudo devmem 0x40000008 32 0x00400F50
ptlnx_pico_zc702:~$ sudo devmem 0x40000010 32
0x00000294
ptlnx_pico_zc702:~$ sudo devmem 0x40000010 32
0x00000294
```

Figura 5.13: Prueba 3. Resultados del ADC3244 con un patrón constante.

Adicionalmente, se repite la prueba usando la aplicación para configurar el patrón '0x1222' en ambos canales, como se muestra en la **Figura 5.14**.

```
[ADC32xx] Register: 0x534 Read: 0x0 Writing: 0x0
[ADC32xx] Register: 0x539 Read: 0x0 Writing: 0x0
[ADC32xx] Register: 0x608 Read: 0x0 Writing: 0x0
[ADC32xx] Register: 0x70a Read: 0x0 Writing: 0x1
[ADC32xx] Register: 0x6 Writing: 0x2
[ADC32xx] Reg 0x0: Writing: 0x50
[TEST] ADC_data_A = 0x1222 || ADC_data_B = 0x1222 || Response time: 1824.00 ns
[TEST] ADC_data_A = 0x1222 || ADC_data_B = 0x1222 || Response time: 1728.00 ns
[TEST] ADC_data_A = 0x1222 || ADC_data_B = 0x1222 || Response time: 1710.00 ns
[TEST] ADC_data_A = 0x1222 || ADC_data_B = 0x1222 || Response time: 1674.00 ns
[TEST] ADC_data_A = 0x1222 || ADC_data_B = 0x1222 || Response time: 1668.00 ns
[TEST] ADC_data_A = 0x1222 || ADC_data_B = 0x1222 || Response time: 1650.00 ns
[TEST] ADC_data_A = 0x1222 || ADC_data_B = 0x1222 || Response time: 1680.00 ns
[TEST] ADC_data_A = 0x1222 || ADC_data_B = 0x1222 || Response time: 1788.00 ns
[TEST] ADC_data_A = 0x1222 || ADC_data_B = 0x1222 || Response time: 1740.00 ns
[TEST] ADC_data_A = 0x1222 || ADC_data_B = 0x1222 || Response time: 1734.00 ns
[TEST] ADC_data_A = 0x1222 || ADC_data_B = 0x1222 || Response time: 1716.00 ns
[TEST] ADC_data_A = 0x1222 || ADC_data_B = 0x1222 || Response time: 1704.00 ns
[ADC32xx] Register: 0x6 Writing: 0x1
[ADC32xx] Reg 0x0: Writing: 0xc0000050
Test 2_2 Finish
Press 'e' to exit:
```

Figura 5.14: Prueba 3. Resultados del ADC3244 con un patrón constante (Aplicación).

Para las pruebas con el patrón de contador, se emplea la herramienta de Xilinx, ILA, que permite analizar las señales y asegurarse de que no se produzcan errores en los bits. Esto se visualiza en la **Figura 5.15**.

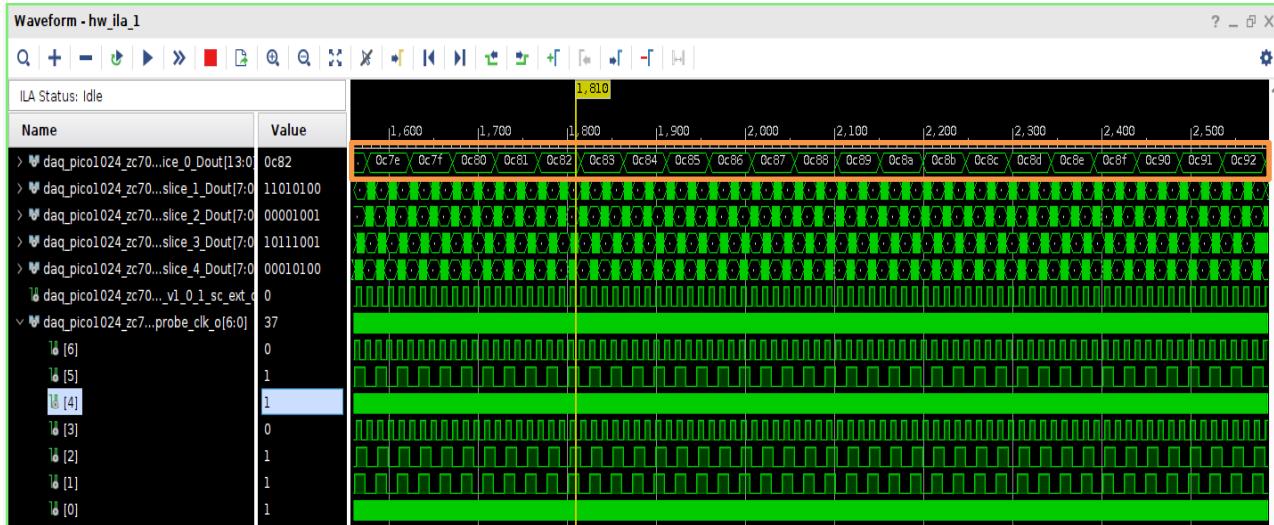


Figura 5.15: Prueba 3. Resultados del ADC3244 con el patrón de contador.

Finalmente, el patrón senoidal generado por el ADC3244 es visualizado utilizando de nuevo el ILA, esta vez configurando la visualización en modo analógico para un análisis más detallado, como se puede ver en la **Figura 5.16**.

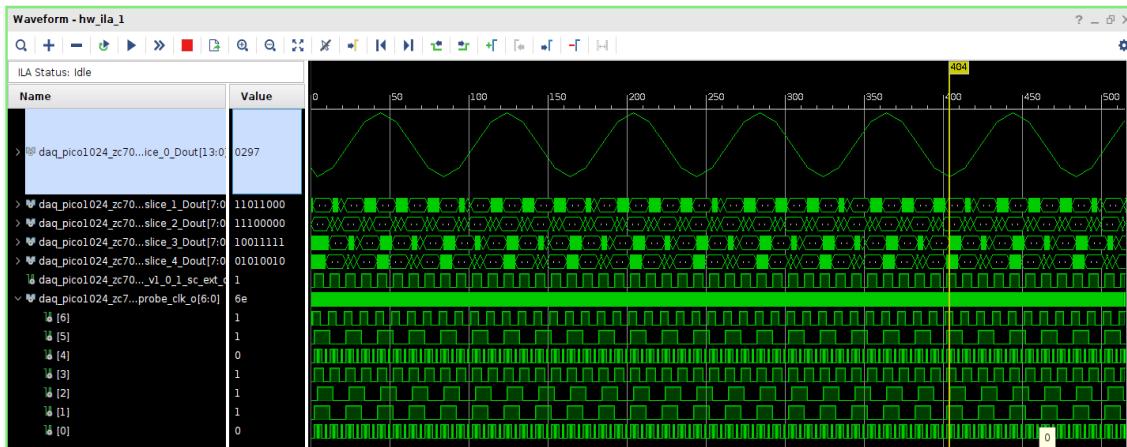


Figura 5.16: Prueba 3. Resultados del ADC3244 con el patrón senoidal.

5.4 Prueba 4: Uso de la DMA para la lectura de un contador programado en el ADC3244

El propósito de esta prueba es comprobar el correcto funcionamiento de la DMA, así como la configuración y gestión de los datos efectuada por la aplicación con el apoyo del módulo kernel diseñado. Para esto, replicaremos la prueba del contador mencionada en la sección 5.3, utilizando la aplicación especificada en el punto 4.3 y el módulo kernel descrito en el punto 4.2. En la **Figura 5.17**, se muestran los datos capturados por la DMA, los cuales han sido almacenados en un archivo de texto por la aplicación y se visualizan en el terminal. Finalmente,

mediante Matlab, se analiza un archivo de texto generado por la aplicación que contiene 160 paquetes de 32 tramas de datos cada uno, lo que totaliza 10240 muestras de un contador, con el fin de verificar la continuidad de los datos. Véase la **Figura 5.18**.

```
[DMA] Writing S2MM DMA
[DMA] DMA Reseted. State S2MM DMA: (0x00000001@0x0d): halted
[DMA] DMA destination stablish. Destination S2MM DMA: e000000
[DMA] DMA initiated. State S2MM DMA: (0x00000000@0x0d): running
[DMA] Length Data S2MM DMA: 0200
[DMA] Status S2MM DMA: (0x00005011@0x0d): halted DMAIntErr IOC_Irq Err_Irq
[DMA] Reg: 0xb6fa8000 ; Value: 00000005
[DMA] Reg: 0xb6fa8004 ; Value: 00000006
[DMA] Reg: 0xb6fa8008 ; Value: 00000007
[DMA] Reg: 0xb6fa800c ; Value: 00000008
[DMA] Reg: 0xb6fa8010 ; Value: 00000009
[DMA] Reg: 0xb6fa8014 ; Value: 0000000a
[DMA] Reg: 0xb6fa8018 ; Value: 0000000b
[DMA] Reg: 0xb6fa801c ; Value: 0000000c
[DMA] Reg: 0xb6fa8020 ; Value: 0000000d
[DMA] Reg: 0xb6fa8024 ; Value: 0000000e
[DMA] Reg: 0xb6fa8028 ; Value: 0000000f
[DMA] Reg: 0xb6fa802c ; Value: 00000010
[DMA] Reg: 0xb6fa8030 ; Value: 00000011
[DMA] Reg: 0xb6fa8034 ; Value: 00000012
[DMA] Reg: 0xb6fa8038 ; Value: 00000013
[DMA] Reg: 0xb6fa803c ; Value: 00000014
[DMA] Reg: 0xb6fa8040 ; Value: 00000015
[DMA] Reg: 0xb6fa8044 ; Value: 00000016
[DMA] Reg: 0xb6fa8048 ; Value: 00000017
[DMA] Reg: 0xb6fa804c ; Value: 00000018
[DMA] Reg: 0xb6fa8050 ; Value: 00000019
[DMA] Reg: 0xb6fa8054 ; Value: 0000001a
[DMA] Reg: 0xb6fa8058 ; Value: 0000001b
[DMA] Reg: 0xb6fa805c ; Value: 0000001c
[DMA] Reg: 0xb6fa8060 ; Value: 0000001d
[DMA] Reg: 0xb6fa8064 ; Value: 0000001e
[DMA] Reg: 0xb6fa8068 ; Value: 0000001f
[DMA] Reg: 0xb6fa806c ; Value: 00000020
[DMA] Reg: 0xb6fa8070 ; Value: 00000021
```

Figura 5.17: Prueba 4. Resultados de la DMA con un contador.

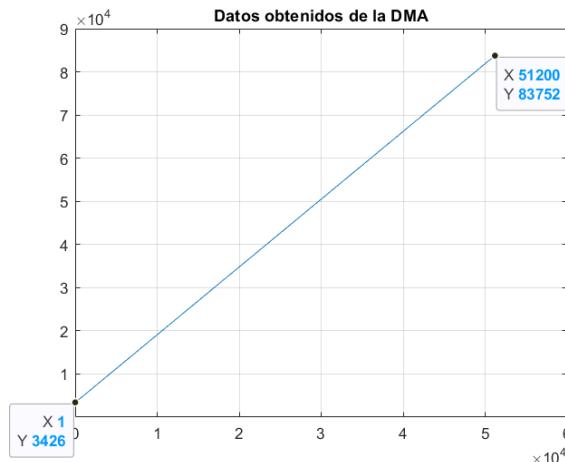


Figura 5.18: Prueba 4. Resultados de la DMA con un contador (Matlab).

5.5 Prueba 5: Aplicación de la configuración al PICO1024

Dado que la configuración del sensor es inaccesible, no ha sido posible verificar directamente si la metodología

empleada en el proceso está correctamente diseñada. Para evaluar de manera indirecta si se han implementado ciertos aspectos de la configuración que confirmen la correcta aplicación del protocolo descrito en la sección 2.2.2, se utilizó como referencia el bit 64. Este bit ajusta el parámetro “MODE_4OUTPUTS”, que cuando se activa en nivel alto, permite el funcionamiento de las salidas OUT3 y OUT4. Normalmente, estas salidas están en alta impedancia y al ser activadas, registran una tensión media de Vtemp. Se observó que antes de configurar, estas salidas tenían un valor aproximado de 32 mV, y después de la configuración, su tensión ascendió a aproximadamente 1.73 V. Para realizar este ajuste, es necesario configurar el sensor en su modo nominal, ya que, por defecto, comienza en el modo de compatibilidad UL05251. Esto último también nos permite verificar que el bit 54 ha sido aplicado correctamente.

5.6 Prueba 6: Señal sinusoidal a la entrada del ADC3244 para verificar la etapa de acondicionamiento

Para verificar la etapa de acondicionamiento y el comportamiento del sistema con una señal real, se ha decidido suministrar una señal sinusoidal de valor 1.7 V con amplitud de 400 mV. Primero con 500 kHz de frecuencia y seguidamente con 100 kHz. En esta prueba se utiliza la DMA para almacenar los datos en un fichero de texto que posteriormente ha sido procesado por la plataforma de Matlab. Véase los resultados de esta prueba en la Figura 5.19.

Se verifica que las características de la onda senoidal visualizada mediante la herramienta de Matlab coincide con los datos esperados. Cabe destacar que el ruido detectado en la señal resultante se ha comprobado que es resultado del ruido introducido por el generador de onda utilizado y el hardware diseñado para acondicionar la señal para los pines de entrada del ADC3244.

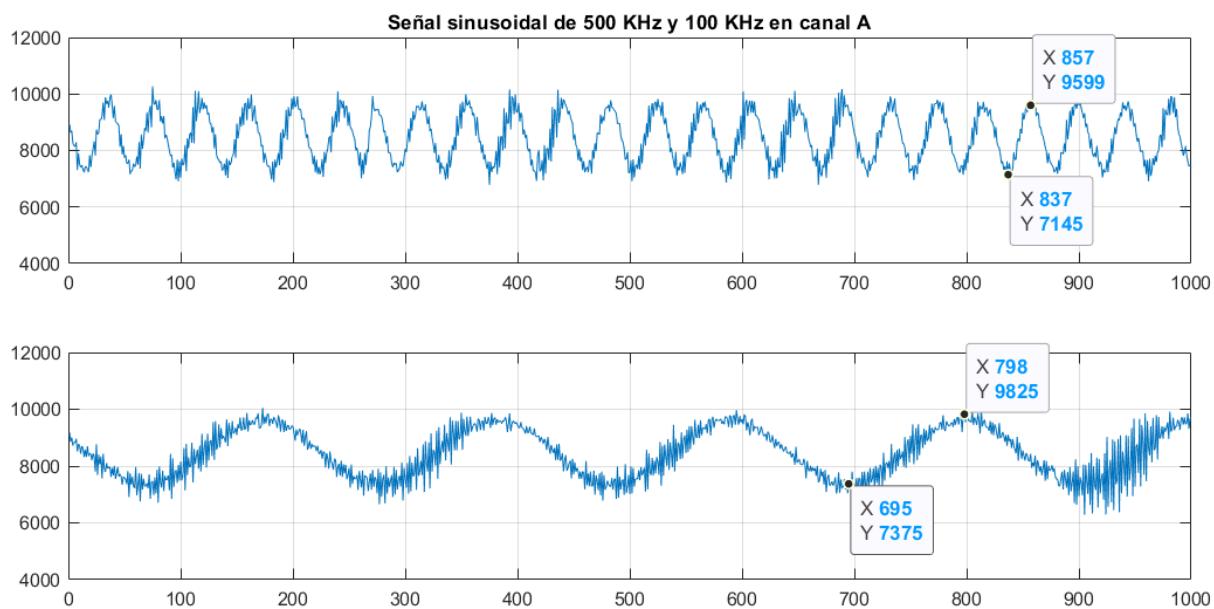


Figura 5.19: Prueba 6. Resultado de una señal sinusoidal.

6 CONCLUSIONES

Cabe resaltar que este proyecto corresponde con un trabajo de investigación realizado en colaboración con la empresa Alter Technology TÜV. Para ello, se abordó una primera fase de este desarrollo enfocada en el sensor analógico PICO 1024, suministrado por la compañía.

Para lograr un sistema de adquisición de datos con la suficiente adaptabilidad para un gran abanico de sensores, se plantearon los siguientes objetivos en este Trabajo Fin de Grado:

- Diseño e implementación de una plataforma modular que permita la rápida y fácil adaptación a nuevos sensores de imagen.
- Desarrollo e implementación de los módulos necesarios, mediante un lenguaje de descripción hardware (HDL), para la configuración y control del sensor de imagen PICO1024 y para la gestión del convertidor analógico-digital.
- Desarrollo y despliegue de un sistema operativo que permita la ejecución una aplicación de alto nivel de control.
- Desarrollo de una aplicación de control que sea capaz de adquirir y procesar los datos generados por el sensor de imagen. Además, esta herramienta deberá permitir la verificación tanto de la correcta configuración del sensor como la integridad de los datos adquiridos.

Para lograr la consecución del objetivo principal, ha sido necesario el desarrollo de diferentes módulos hardware para realizar la gestión y control tanto del convertidor analógico-digital involucrado como el sensor de imagen elegido. Cabe indicar que estos módulos se han diseñado con una arquitectura modular que permite su reutilización en futuros proyectos. Asimismo, como se puede observar en el Capítulo 5, las pruebas, tanto a nivel de simulación como de laboratorio, han demostrado el correcto funcionamiento de estos. No obstante, es importante resaltar que, durante la fase de pruebas en el laboratorio, se ha tenido que hacer frente a múltiples desafíos técnicos relacionados con la operación de la tarjeta de adaptación. En concreto, la red de adaptación de señal responsable de adaptar la salida del sensor PICO1024 a la entrada del convertidor Analógico -Digital ha requerido varias fases de rediseño que han impedido la correcta adquisición de datos. Debido a estos problemas, la obtención de imágenes reales utilizando el sensor PICO1024 no ha sido posible.

Además del desarrollo hardware, otro de los objetivos marcados en el presente Trabajo era el desarrollo de una imagen funcional de un sistema operativo capaz de ejecutarse en la FPGA seleccionada. Para lograr la consecución de este objetivo, se hizo uso de la herramienta Petalinux, suministrada por el fabricante de Xilinx, siendo necesario su correcta configuración, así como el desarrollo de un módulo del kernel, capaz de procesar las interrupciones hardware generadas por los bloques HDL desarrollados.

Por último, se desarrolló una aplicación de control para facilitar el uso y verificación de la plataforma implementada por parte del usuario. Esta herramienta, además de configurar correctamente los diferentes bloques, se encarga de procesar la información recibida del sensor y almacenarla en un fichero de texto para su posterior utilización.

En conclusión, aunque no ha sido posible realizar pruebas con imágenes reales capturadas por el sensor PICO1024 debido a las mencionadas limitaciones de hardware, el Trabajo desempeñado ha logrado cumplir con los objetivos marcados al inicio; originando una plataforma con un gran potencial a futuro y que está siendo actualmente empleada por la compañía en la caracterización del sensor elegido.

6.1 Líneas futuras de trabajo

A continuación, se describen algunas de las posibles líneas para continuar el trabajo desempeñado:

1. Pruebas de laboratorio con el sensor PICO1024 para validar y verificar el correcto funcionamiento del módulo HDL desarrollado.

2. Ampliación a sensores de imágenes digitales, desarrollado la etapa de control correspondiente.
3. Diseño de una aplicación con interfaz gráfica.
4. Implementación de algoritmos de procesamiento de imágenes en tiempo real

BIBLIOGRAFÍA

- [1] P. Kogut, «Teledetección Satelital: Tipos, Usos Y Aplicaciones,» EOS Data Analytics, 22 Enero 2024. [En línea]. Available: <https://eos.com/es/blog/teledeteccion/>. [Último acceso: 19 Marzo 2024].
- [2] Lynred, «PICO1024Gen2 | Lynred,» [En línea]. Available: <https://www.lynred.com/products/pico1024gen2>. [Último acceso: 27 marzo 2024].
- [3] AMD, «AMD Zynq 7000 SOC ZC702 Evaluation Kit,» [En línea]. Available: <https://www.xilinx.com/products/boards-and-kits/ek-z7-zc702-g.html>. [Último acceso: 27 marzo 2024].
- [4] Texas Instrument, «Hoja de datos de ADC3244, información de producto y soporte | TI.com,» [En línea]. Available: <https://www.ti.com/product/es-mx/ADC3244>. [Último acceso: 27 Marzo 2024].
- [5] AMD, «AMD Technical Information Portal,» [En línea]. Available: https://docs.amd.com/r/en-US/pg021_axi_dma/AXI-DMA-v7.1-LogiCORE-IP-Product-Guide.
- [6] AMD, «PetaLinux Tools,» [En línea]. Available: <https://www.xilinx.com/products/design-tools/embedded-software/petalinux-sdk.html>. [Último acceso: 1 abril 2024].
- [7] A. Powell y D. Silage, «Statistical performance of the ARM cortex A9 accelerator coherency port in the xilinx zynq SoC for real-time applications,» 1 diciembre 215. [En línea]. Available: <https://ieeexplore.ieee.org/abstract/document/7393362>. [Último acceso: 27 marzo 2024].

