

Data acquisition system

Sebastián Montoya Hernández

September 22, 2024

La instrumentación científica ha sido un pilar fundamental en el avance de la ciencia y la tecnología, al permitir a los investigadores observar y analizar fenómenos que trascienden los límites de la percepción humana. Desde la explotación del cosmos hasta la manipulación de partículas subatómicas, la capacidad de medir, registrar y procesar datos con precisión ha ampliado de manera significativa los horizontes del conocimiento en múltiples disciplinas. Los avances en tecnología de medición han facilitado la captura de señales complejas, que al ser transformadas en datos utilizables, proporcionan información crítica para la comprensión de fenómenos naturales y artificiales. En este contexto, los sistemas de adquisición de datos (DAQ, por sus siglas en inglés) desempeñan un rol esencial, actuando como el enlace entre los fenómenos físicos observados y su representación digital. Estos sistemas permiten registrar una vasta gama de magnitudes físicas—como temperatura, presión, aceleración o radiación—a partir de sensores que las convierten en señales eléctricas, que pueden ser procesadas y analizadas con precisión. La eficiencia con la que estos sistemas capturan y procesan dichas señales es crucial, especialmente en situaciones donde los eventos ocurren a escalas temporales extremadamente breves o cuando la precisión es determinante para la interpretación fiable de los datos obtenidos.

Los sistemas DAQ no solo son indispensables en la investigación científica básica, sino que también tienen aplicaciones vitales en sectores industriales, médicos y tecnológicos avanzados. En campos como la astrofísica, la física de partículas o la ingeniería de materiales, la capacidad de registrar con exactitud eventos transitorios o señales de baja intensidad es crucial para obtener información precisa sobre los procesos subyacentes. Por ejemplo, en experimentos de colisionadores de partículas, los sistemas DAQ permiten monitorear interacciones subatómicas en tiempo real, donde las señales generadas por los detectores ocurren en escalas temporales minúsculas y requieren ser capturadas con alta resolución tanto temporal como espacial. Asimismo, en la detección de radiación, ya sea en contextos de protección radiológica o de investigación nuclear, los DAQ juegan un papel fundamental al registrar las interacciones ionizantes y convertirlas en datos procesables. Esto permite analizar fuentes de radiación o eventos de baja probabilidad, que de otro modo pasarían desapercibidos en el ruido de fondo. La importancia de los DAQ radica en su capacidad para gestionar grandes volúmenes de datos, procesando señales con rapidez y precisión.

para obtener información que de otra forma sería inaccesible o ininterpretada.

Dentro de las tecnologías empleadas para el desarrollo de estos sistemas, los dispositivos basados en Field Programmable Gate Arrays (FPGA) han adquirido una posición destacada. Su capacidad para procesar datos a gran velocidad y en tiempo real, con una latencia mínima, los convierte en una opción ideal para aplicaciones que requieren un tratamiento eficiente de señales complejas. Además, los FPGA ofrecen una flexibilidad notable en su configuración, permitiendo la implementación de arquitecturas optimizadas para tareas específicas, como el procesamiento de señales de alta velocidad o la reducción de ruido en entornos de alta sensibilidad. Esta versatilidad es especialmente relevante en la detección de partículas y radiación, donde la precisión y la velocidad de procesamiento son elementos cruciales para asegurar que las interacciones físicas sean correctamente registradas y analizadas.

En este trabajo, se presenta el desarrollo de un marco teórico que explica los conceptos fundamentales involucrados en la cadena de manejo de información en un sistema de adquisición de datos (DAQ). Se aborda desde las interacciones de los sistemas físicos con los sensores, hasta las etapas electrónicas involucradas en el tratamiento y digitalización de las señales. Además, se profundiza en las plataformas comúnmente utilizadas para el procesamiento de datos, incluyendo las tecnologías más avanzadas, como los sistemas basados en FPGA, y su posterior comunicación con el usuario a través de diversas interfaces. Este capítulo también incluye una sección dedicada al montaje experimental, en la cual se describen detalladamente los componentes y la configuración implementada, así como las herramientas de hardware y software empleadas para desarrollar un DAQ optimizado para aplicaciones de alto rendimiento, como la adquisición de datos de detectores de radiación. Finalmente, se presenta una sección de resultados, en la que se valida el funcionamiento del sistema mediante la adquisición de señales sintéticas. A través de esta fase experimental, se ponen a prueba las distintas etapas y configuraciones del DAQ, seguido de un análisis de los datos obtenidos. Este análisis no solo busca verificar el desempeño del sistema, sino que también se plantean conclusiones y perspectivas para el desarrollo futuro en este campo.

1 Marco teórico

En esta sección, se abordarán los fundamentos necesarios para comprender el funcionamiento y la implementación de sistemas de adquisición de datos, basándose en una descripción detallada de los principales componentes y procesos involucrados. La explicación comenzará con una visión general de los sistemas físicos e interacciones, proporcionando el contexto necesario para entender cómo los fenómenos físicos son detectados e interpretados a través de sistemas instrumentales. A continuación, se discutirá el papel crucial de los sensores y detectores en la conversión de magnitudes físicas en señales eléctricas, seguido

por un análisis de las señales eléctricas de sensores, que detalla cómo se generan y las características que definen estas señales. Posteriormente, se explorará el tratamiento y acondicionamiento de señales, un proceso fundamental para asegurar que las señales obtenidas sean adecuadas para su procesamiento. El procesamiento digital (digital processing) se abordará a continuación, cubriendo las técnicas y plataformas comúnmente utilizadas para convertir las señales acondicionadas en datos útiles y analizables. La discusión también incluirá la interfaz de comunicación (communication interface), que es clave para la transferencia eficiente de datos entre el sistema de adquisición y otros dispositivos, así como su impacto en la eficiencia general del sistema. Finalmente, se revisará la interfaz de usuario (user interface), destacando la importancia de una representación clara y accesible de los datos para facilitar el análisis y la toma de decisiones. Para facilitar la comprensión de estos conceptos, se presenta la figura 1, que ilustra el diagrama de los principales componentes de un sistema de adquisición de datos genérico. Esta figura proporciona una visión integral de la cadena de etapas, desde la adquisición de señales físicas hasta su presentación final al usuario, y servirá como referencia visual a lo largo de la discusión.



Figure 1: Diagrama de los principales componentes de un sistema de adquisición de datos para la medición de un sistema físico. El flujo de información comienza con un sistema físico que interactúa con un sensor o detector, generando una señal eléctrica. Esta señal es adquirida y procesada por la electrónica de lectura, la cual incluye subetapas como el acondicionamiento de la señal y el procesamiento digital. Posteriormente, los datos procesados son transferidos a través de una interfaz de comunicaciones hacia el usuario, quien accede a ellos mediante una interfaz de usuario, permitiendo su visualización y análisis.

1.1 Sistemas físicos e interacciones

En el campo de la instrumentación científica, un sistema físico se refiere a cualquier entidad u objeto cuyas propiedades y comportamientos son susceptibles de estudio mediante técnicas de medición y observación. Estos sistemas

pueden abarcar desde las partículas subatómicas, que siguen las leyes de la mecánica cuántica, hasta sistemas macroscópicos como materiales sólidos, fluidos, y sistemas biológicos, además de fenómenos astrofísicos de gran escala.

Un sistema físico se caracteriza por poseer ciertas propiedades mensurables, como la temperatura, la presión, el campo eléctrico o magnético, y la energía, entre otras. Dichas propiedades pueden ser modificadas o perturbadas por interacciones externas, lo que permite que sean explotadas para generar señales que puedan ser medidas. Es en este contexto donde los sistemas físicos se vuelven de interés para la adquisición de datos: a través de la manipulación o el monitoreo de las interacciones que estos sistemas tienen con su entorno, es posible extraer información relevante que caracterice su comportamiento.

Para lograr esta interacción, es necesario comprender de manera exhaustiva la naturaleza del sistema físico y cómo sus propiedades responden ante estímulos específicos. Esta comprensión es clave para el diseño de sensores y detectores que, mediante la transformación de las propiedades físicas en señales eléctricas, permiten la extracción sistemática de información. Es fundamental que el sistema físico interactúe de manera controlada con el sensor para que la información obtenida sea representativa de las magnitudes físicas de interés y no de variables externas o perturbaciones no deseadas.

En particular, un ejemplo notable lo constituyen las partículas fundamentales y el transporte de energía a partir de estas, conocido como radiación. Dependiendo de sus propiedades físicas —como la carga, masa o energía—, es posible diseñar experimentos que permitan su interacción controlada con medios materiales, lo que facilita su estudio. Por ejemplo, las partículas cargadas, como los electrones, protones o muones, pueden provocar ionización en el medio por el que atraviesan, liberando pares de electrones e iones. Esta recolección de carga, medida a través de señales eléctricas, permite extraer información sobre la energía, trayectoria o tipo de partícula. Detectores de ionización, como cámaras de gas o semiconductores, explotan este principio para realizar análisis precisos de las propiedades físicas de las partículas.

Además de la ionización, otras propiedades como el momento o la energía de las partículas pueden inferirse a partir de las señales que generan durante su interacción con el sistema de detección. Esto permite no solo estudiar fenómenos como transiciones de estado y decaimientos radiactivos, sino también obtener datos que contribuyen al entendimiento de la composición y dinámica de las partículas en diversas condiciones experimentales. Así, la interacción entre sistemas físicos y los detectores es clave para extraer información que impulse el conocimiento en áreas como la física de partículas y la astrofísica.

1.2 Sensores y detectores

Como se expuso anteriormente, un sensor es un dispositivo que responde a un estímulo físico, químico o biológico específico, y genera una señal de tipo eléctrico, que tiene una relación funcional con la magnitud del estímulo. Los sensores suelen estar compuestos por dos componentes principales: el elemento sensible o transductor, que interactúa directamente con el estímulo (por ejemplo, temperatura, presión, luz) y un mecanismo de conversión que traduce la respuesta del transductor en una señal utilizable (como una corriente o voltaje eléctrico). La precisión, sensibilidad, resolución y rango de operación son características clave que determinan el rendimiento de un sensor [1].

Los sensores se pueden clasificar según diversos criterios: por el tipo de magnitud que miden, el principio de funcionamiento (resistivos, capacitivos, inductivos, piezoelectrinos), el tipo de señal de salida (analógicos, digitales), la forma de operación (activos, pasivos), y la naturaleza del contacto con la magnitud medida (de contacto, sin contacto). Además, según su contexto de uso, los sensores pueden ser clasificados como domésticos (utilizados en electrodomésticos y dispositivos de consumo), industriales (empleados en automatización y control de procesos), o científicos (destinados a investigación y aplicaciones especializadas) [2].

En particular, estos últimos destacan por su alta precisión, exactitud, sensibilidad, robustez y fiabilidad. Su objetivo es proporcionar mediciones extremadamente precisas y reproducibles, ya que los resultados son cruciales para investigaciones avanzadas. Además, suelen poseer un amplio rango dinámico, lo que les permite detectar tanto variaciones muy pequeñas como extremadamente grandes en las magnitudes de interés. Un ejemplo destacado en este grupo son los detectores de radiación o partículas, que sobresalen por su capacidad para interactuar con corpúsculos a nivel atómico y subatómico, generando señales pulsadas que revelan propiedades esenciales de las partículas incidentes, como su energía, momento lineal o trayectoria, entre otras [3].

Existen diferentes tipos de detectores de partículas, cada uno optimizado para aplicaciones específicas. Los detectores de estado sólido, como los semiconductores de silicio, son ampliamente utilizados por su alta resolución espacial y temporal. Estos dispositivos funcionan mediante la creación de pares electrón-hueco cuando una partícula cargada atraviesa el material, generando una señal eléctrica proporcional a la energía depositada. Por otro lado, los detectores gaseosos, como las cámaras de ionización o los contadores proporcionales, operan en medios gaseosos donde la ionización del gas produce electrones libres y iones positivos que se colectan para formar una señal. Estos detectores son valiosos por su capacidad para cubrir grandes volúmenes, lo que es esencial en experimentos que requieren la detección de partículas en áreas extensas. Además, existen otros tipos de detectores como los cintiladores, que emiten luz cuando son excitados por radiación ionizante, y los detectores Cherenkov, que detectan

partículas cargadas moviéndose a velocidades superiores a la velocidad de la luz en un medio dado [4].

1.3 Señales eléctricas de sensores

Dado que las señales eléctricas son esenciales en este contexto como portadoras de la información física que se busca extraer, es fundamental describir sus principales características para comprender cómo las etapas electrónicas del sistema de adquisición de datos (DAQ) las afectan. Aunque las señales pueden ser digitales o análogas, generalmente las producidas por un sensor o detector son de tipo análogo.

Una señal analógica proveniente de un sensor se caracteriza principalmente por su variación continua en el tiempo y su capacidad para representar información en un rango infinito de valores. La amplitud de la señal refleja la magnitud del parámetro medido, como temperatura o presión, y puede variar suavemente en respuesta a cambios en el entorno. La frecuencia de la señal puede no ser relevante en todos los casos, pero el tiempo de respuesta y la estabilidad son cruciales para asegurar mediciones precisas. Además, la señal puede estar afectada por ruido y errores, lo que requiere técnicas de filtrado y calibración para obtener datos confiables. La forma de onda de la señal es continua y puede ser analizada en el dominio del tiempo para evaluar su comportamiento y en el dominio de la frecuencia para identificar componentes relevantes [2]. En la figura 2, se ilustra una señal en el dominio del tiempo proveniente de un sensor de aceleración.

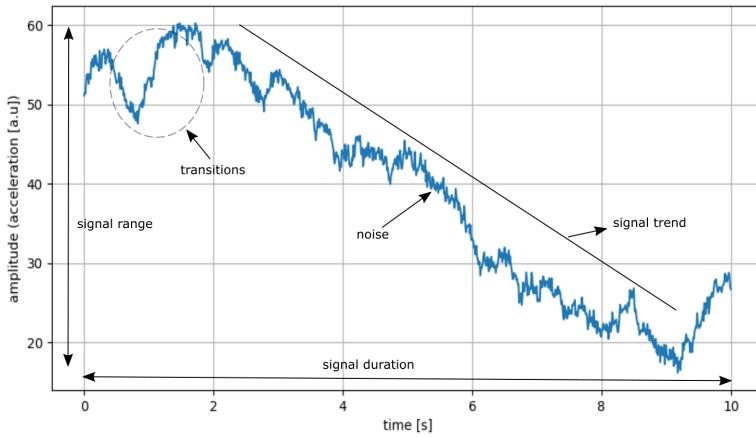


Figure 2: Señal no periódica en el dominio del tiempo proveniente de un sensor de aceleración. En ella es posible observar algunas características como duración, rango de la magnitud de interés en el lapso registrado, ruido electrónico, tendencia y transitorios. La amplitud de la señal solo puede ser definida con base a un nivel de referencia.

Un caso de particular interés son las señales producidas por los detectores de partículas, que en general son de naturaleza pulsada. En este contexto, se puede asociar la detección de un evento con una perturbación localizada, denominada pulso, que se define sobre la línea base de la señal de salida. La figura 3 muestra un ejemplo genérico de un pulso individual, utilizado para ilustrar algunas de sus características principales.

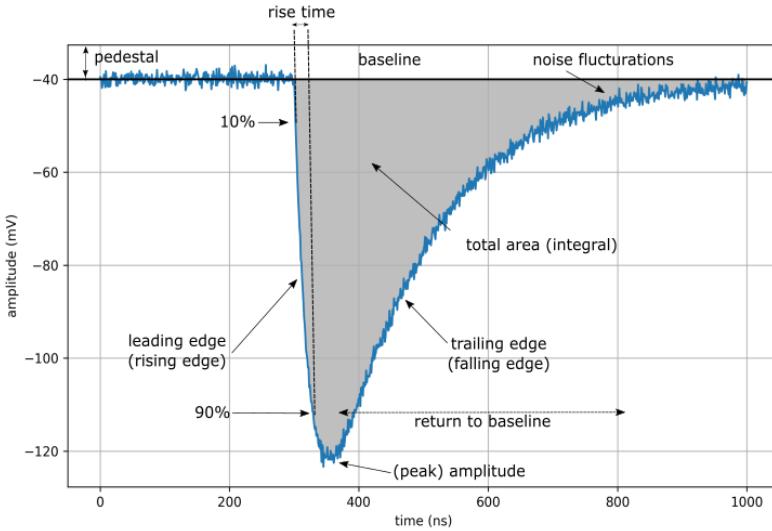


Figure 3: Diagrama ilustrativo de las principales características de un pulso típico de un detector de radiación. Reproducido a partir de [4]

Según [3], suponiendo que la señal generada por el detector es corta comparado con los tiempos típicos de procesamiento de la electrónica de lectura, las cantidades que caracterizan un pulso son:

- La amplitud máxima del pulso: también conocida como altura del pulso, es el valor máximo que alcanza la señal. En sistemas lineales, este valor es proporcional a la energía primaria depositada en el detector.
- Tiempo de pico: Es el momento en el tiempo en el que se alcanza la amplitud máxima del pulso.
- Área o integral del pulso: Representa el área bajo la curva del pulso. En sistemas lineales y para señales cortas tipo δ , su valor debería ser proporcional a la energía primaria depositada.
- Ancho del pulso: Se refiere a la duración del pulso, que generalmente se define como el ancho total a la mitad de la altura máxima (FWHM, por sus siglas en inglés).
- Bordes de subida y bajada: Son las pendientes ascendente y descendente del pulso.
- Tiempo de subida: Caracteriza la rapidez con la que el pulso aumenta. Comúnmente se define como el tiempo necesario para que el pulso pase del 10% al 90% de su amplitud máxima, aunque existen otras definiciones.

- Tasa de variación (Slew rate): Indica el cambio de voltaje por unidad de tiempo dV/dt y se expresa en unidades de V/s .
- Línea base o valor de pedestal: Es el valor de salida cuando no hay ninguna señal de entrada. Define el nivel 'cero' desde el cual se mide la altura de la señal. Aunque generalmente la línea base tiene un valor fijo, pueden ocurrir desviaciones durante un cierto (y breve) período de tiempo. Estas desviaciones se conocen como desplazamientos de la línea base.
- Tiempo de retorno a la línea base: Es el tiempo necesario para que la amplitud del pulso vuelva al valor de la línea base.
- Suboscilación: Se refiere a la parte de la amplitud de un pulso que tiene un signo opuesto (con respecto a la línea base) en comparación con la amplitud principal.
- Señal unipolar: Es una forma de pulso en la que, salvo por las fluctuaciones de ruido, el valor de la amplitud se mantiene por encima o por debajo de la línea base en todo momento t . Por lo general, también se incluyen en esta definición las señales con pequeñas suboscilaciones.
- Señal bipolar: Es una forma de pulso en la que la parte del pulso que ocurre más tarde en el tiempo tiene un signo opuesto al de la parte que ocurre primero.

Por otro lado, las señales digitales, esenciales en el procesamiento de información, se caracterizan por su naturaleza discreta tanto en el tiempo como en la amplitud, en tanto que solo pueden adoptar dos estados, representados por los valores 1 y 0. Es de resaltar, que las señales analógicas tratadas anteriormente, son convertidas a digitales mediante un dispositivo denominado conversor análogo digital o ADC. Adicionalmente, a partir de una señal digital base con frecuencia constante, conocida como reloj, se puede codificar información en señales digitales mediante código binario y operaciones de conteo. Básicamente, se mide cuántos ciclos de reloj la señal permanece en un estado u otro. Esta metodología es altamente versátil, ya que permite realizar operaciones matemáticas utilizando el sistema binario [5]. En la figura 4 se puede observar un ejemplo de señal digital para ilustrar.

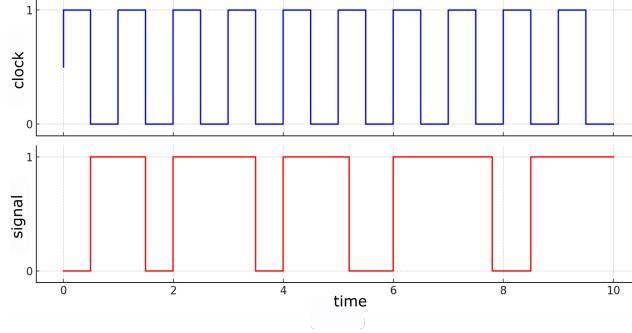


Figure 4: Ejemplo de diagrama de tiempo de una señal digital. En la parte superior se muestra una señal típica de reloj y en la parte inferior una señal genérica con información codificada en su estado (amplitud) o en su duración (ancho).

1.4 Tratamiento y acondicionamiento de señales

Próxima al sensor, se encuentra la etapa de tratamiento y acondicionamiento, que típicamente comprende electrónica analógica para la amplificación, shaping y digitalización de la señal. Esta etapa es esencial para filtrar y modular la señal, otorgándole las características adecuadas para ser transferida a las etapas de procesamiento digital posteriores. En la figura 5 se observa un esquema genérico de esta sección de la cadena de adquisición de datos.

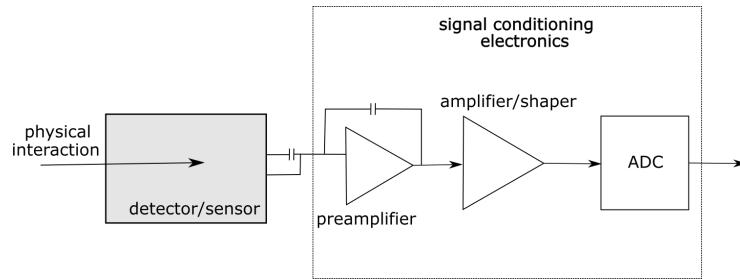


Figure 5: Un esquema de una etapa de electrónica de acondicionamiento típico, utilizado a menudo para la lectura de un detector, que incluye amplificación, conformación de impulsos y digitalización (representada aquí por un ADC).

Los sistemas involucrados en esta etapa, deben cumplir con tres características: (a) ser causales, (b) invariantes en el tiempo, y (c) lineales al menos en la primera etapa de amplificación. Un sistema se considera causal si, en cualquier momento, solo depende del valor de su entrada en ese instante. Es invariante en el tiempo si la relación entre la salida y la entrada, por ejemplo, el ratio $\text{señal}_{\text{entrada}}/\text{señal}_{\text{salida}}$ no varía con el tiempo [4]. La linealidad del sistema

implica que la señal de salida (por ejemplo, $v_{out(t)}$) no depende del tamaño de la señal de entrada (por ejemplo, $i_{in(t)}$), esto es,

$$v_{out} (\alpha \times i_{in} (t)) = \alpha \times v_{out} (i_{in} (t)) \quad (1)$$

En muchos casos, la magnitud a la salida de un sensor puede ser muy pequeña, por lo que es necesario implementar etapas de preamplificación y amplificación que aumenten la amplitud de la señal (ganancia) hasta que esta se encuentre en el rango de sensibilidad de las etapas posteriores. Para llevar a cabo tal función, es posible utilizar distintos dispositivos, sin embargo, los más comunes son los amplificadores operacionales y transistores. Usualmente las señales de entrada provenientes del sensor o detector al preamplificador son transformadas en señales de voltaje (V) o de corriente (I). Dependiendo de los tipos de entrada y salida, es posible distinguir [4]:

- amplificador de voltaje: $V \rightarrow V$,
- amplificador de corriente: $I \rightarrow I$,
- amplificador de transconductancia: $V \rightarrow I$,
- amplificador de transimpedancia: $I \rightarrow V$,
- amplificador de carga: $Q \rightarrow V$ (o I).

Como expone [3], La función principal del preamplificador es captar la señal del detector sin deteriorar notablemente la relación señal-ruido (SNR) inherente. Por ello, el preamplificador se ubica generalmente lo más cerca posible del detector para reducir la carga capacitiva (C) sobre este. Por ejemplo, un preamplificador de voltaje amplifica directamente la señal de voltaje V_{in} , manteniendo una alta impedancia de entrada Z_{in} y una baja impedancia de salida Z_{out} para asegurar una transferencia eficiente y reducir la influencia del ruido. En ambos casos, la linealidad del preamplificador es crucial para que la relación V_{out}/V_{in} se mantenga constante, lo cual es esencial para la precisión en la medición [6].

Por otro lado, un preamplificador de carga convierte una señal de carga Q generada por el detector en un voltaje proporcional V , dada la relación:

$$V = \frac{Q}{C} \quad (2)$$

La figura 6 ilustra su comportamiento típico sobre la señal del detector.

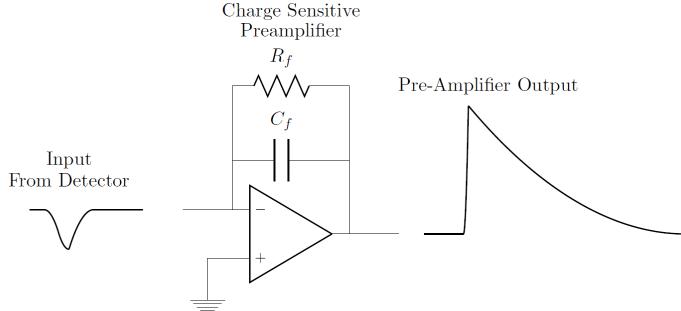


Figure 6: Diagrama que ilustra la estructura y comportamiento típico de un preamplificador de carga. En este se observa la señal de salida de un detector de carga, los componentes electrónicos del preamplificador y la señal típica de salida de este. Tomado de [4].

La amplificación por etapas de una señal ofrece numerosas ventajas significativas. Principalmente, permite reducir el ruido generado en cada etapa individual, resultando en una señal final más limpia y menos propensa a la distorsión. Este método también mejora la estabilidad del sistema al distribuir la amplificación, evitando la saturación que podría ocurrir con una amplificación intensa en una sola etapa. Además, facilita el control preciso de la ganancia y la adaptación de impedancias entre diferentes componentes, lo cual mejora la eficiencia y la transferencia de la señal. Esta amplificación gradual es particularmente útil para manejar señales débiles, amplificándolas sin riesgo de distorsión. Asimismo, distribuye la carga térmica generada, disminuyendo el riesgo de sobrecalentamiento de los componentes [1].

La etapa de amplificación principal generalmente se realiza mediante un amplificador operacional con una resistencia en realimentación. Como modelo matemático, el detector, representado como una capacitancia a descargar, suministra la señal de corriente i_s a través de la resistencia R_s a un nivel de referencia en un tiempo Δt . Si el tiempo de descarga es grande comparado al tiempo de duración de la señal

$$(\tau = R_s C_D \gg \Delta t) \quad (3)$$

en cierto sentido, el detector integra la señal de corriente en la capacitancia del detector

$$(V_D = Q_S / C_D) \quad (4)$$

y a la entrada del amplificador se tiene el voltaje

$$v_{\text{in}}(t) = V_D \exp(-t/R_s C_D) \quad (5)$$

El voltaje de salida es proporcional a v_{in} y el sistema opera como un amplificador de voltaje [4]:

$$v_{out}(t) = -\frac{R_f}{R_S} v_{in}(t) = -\frac{R_f V_D}{R_S} \exp(-t/R_S C_D). \quad (6)$$

1.4.1 Tratamiento y acondicionamiento de señales: Signal shaping

En la cadena de tratamiento de señales, la etapa de *shaping* es un circuito electrónico fundamental que modifica la forma de una señal para optimizar su calidad y adaptarla a los requisitos del procesamiento subsiguiente. Esta etapa generalmente se integra en la fase de preamplificación o amplificación y es especialmente relevante en sistemas de detección de partículas. Su función principal es transformar señales pulsadas, que pueden presentar formas diversas, en pulsos uniformes y bien definidos. Esta uniformidad es crucial para evitar la superposición de pulsos (*pile-up*) (fig. 7) y disminuir el ruido mediante un filtrado eficiente de frecuencia [6].

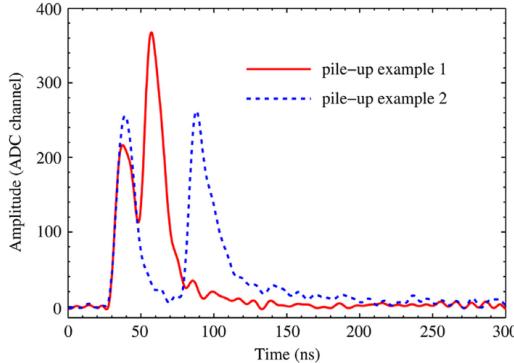


Figure 7: Ejemplos de eventos de pile-up en un detector. El gráfico muestra dos casos de superposición de señales donde dos pulsos ocurren en rápida sucesión, lo que provoca la combinación de sus amplitudes. El ejemplo 1 (línea roja) y el ejemplo 2 (línea azul punteada) ilustran cómo la proximidad temporal de los pulsos afecta la forma y altura de las señales medidas, causando distorsión en la resolución temporal y energética. Tomado de [7]

Los pulsos electrónicos generados por el preamplificador tienen tiempos de decaimiento típicos que varían entre unos pocos nanosegundos y varios microsegundos. Si llegan señales adicionales durante el tiempo de decaimiento, puede ocurrir pile-up a pesar de que el capacitor de retroalimentación del preamplificador se descargue. Para mitigar esto, se utilizan filtros pasa-altos y pasa-bajos, que permiten separar las señales superpuestas y moldear los pulsos de salida en formas más Gaussianas. Además, los filtros ayudan a reducir el ruido blanco que afecta a todas las frecuencias, mejorando así la SNR [4]. La figura 8 ilustra

las formas de pulsos típicos en la salida del preamplificador que ingresan una configuración de shaping básica compuesta por un circuito CR-RC. Usualmente se observa una caída (undershoot) en la salida del shaper, la cual se corrige con la cancelación polo-cero.¹

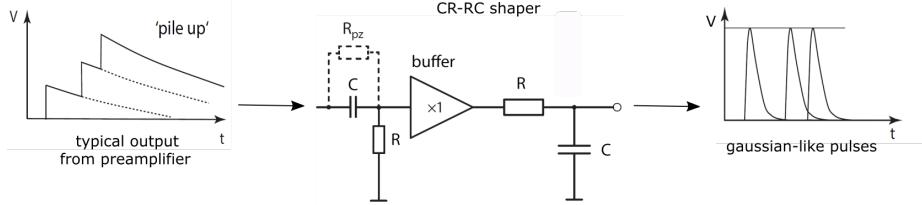


Figure 8: Funcionamiento típico de un shaper. La resistencia R_{pz} paralela al condensador del filtro de paso alto se utiliza para la corrección del undershooting. El amplificador ($\times 1$) entre las secciones del filtro de banda sirve como un buffer, impidiendo que el filtro de paso bajo aplique una carga significativa al filtro de paso alto. Adaptado de [4].

Es importante señalar que el proceso de shaping puede realizarse en la etapa de procesamiento digital, adaptándose a las necesidades específicas de la aplicación. Formas de pulso, como la trapezoidal, que resultan complejas de generar mediante electrónica analógica, pueden implementarse de manera más eficiente en el dominio digital utilizando filtros con parámetros ajustables en plataformas configurables como las FPGA [8]. Este enfoque se explorará en mayor detalle en la sección 1.5. Asimismo, dependiendo de los requisitos del usuario, otros subsistemas, como los discriminadores de señales y los circuitos de trigger, pueden implementarse tanto mediante comparadores analógicos como a través de algoritmos de comparación en la etapa de procesamiento digital.

1.4.2 Tratamiento y acondicionamiento de señales: Digitalización

De acuerdo con [9], la digitalización de las señales provenientes de sensores es crucial debido a la precisión y flexibilidad que ofrece frente a los métodos analógicos tradicionales. Con el avance de los convertidores analógico-digitales (ADC) de alta velocidad y buena resolución desde los años 90, la posibilidad de procesar digitalmente las señales de los sensores se ha consolidado.

Las ventajas de este enfoque incluyen una flexibilidad ilimitada en la elección de parámetros de procesamiento, mayor estabilidad al eliminar el riesgo de derivas debido a cambios de temperatura o voltaje, y la capacidad de realizar análisis más detallados con múltiples salidas de un mismo sensor. Además, la manipulación digital no introduce ruido adicional y permite la implementación

¹Técnica de diseño de circuitos que anula efectos indeseados al colocar un cero en la misma frecuencia que un polo, mejorando la respuesta del sistema.

precisa de formas de señal que serían difíciles o imposibles de lograr en circuitos analógicos. Sin embargo, una desventaja potencial es la limitación en la precisión temporal, ya que los sistemas digitales están restringidos a la frecuencia de muestreo más cercana, lo que puede ser menos exacto que los métodos analógicos en aplicaciones que requieren una temporización muy rápida.

La función principal de un ADC (Convertidor Analógico-Digital) es transformar una señal de tensión analógica en un código digital proporcional, manteniendo esta conversión de manera continua a una frecuencia determinada por el reloj del sistema. Una ilustración genérica de su principio de funcionamiento se observa en la figura 9.



Figure 9: Representación del principio de funcionamiento de un ADC. Una señal eléctrica analógica ingresa al convertidor analógico-digital ADC y genera una señal digital, que codifica mediante representación binaria la información de la señal original.

Por ejemplo, un reloj operando a 500 MHz permite una velocidad de muestreo de 500 MSPS (Megamuestras por segundo), lo que corresponde a una muestra cada 2 ns. En la figura 10 se muestra la simulación de una señal senoidal proveniente de un sensor, digitalizada mediante un ADC con estas características.

La resolución del ADC, que se determina por el número de bits del convertidor, afecta directamente a la precisión de estas conversiones. Un número binario con n bits puede representar 2^n valores. Durante la digitalización, el rango de voltaje desde V_{\min} (usualmente $V_{\min} = 0$) hasta V_{\max} se subdivide en $2^n - 1$ intervalos, a cada uno de los cuales se le asigna un valor binario. Si la conversión es lineal, el rango se divide en intervalos de tamaño igual, donde el paso de voltaje más pequeño corresponde al bit menos significativo (LSB, por sus siglas en inglés), que se encuentra más a la derecha en un número binario. En un convertidor analógico-digital (ADC) de n bits, un LSB corresponde a un paso de voltaje de:

$$1 \text{ LSB} = \frac{V_{\max} - V_{\min}}{2^n - 1} \quad (7)$$

En un ADC ideal, cada conversión de voltaje de entrada a código de salida es independiente, perfectamente lineal y ocurre instantáneamente. Sin embargo, las imperfecciones en los ADCs reales limitan tanto la frecuencia máxima de muestreo como la linealidad y la precisión de la conversión [4].

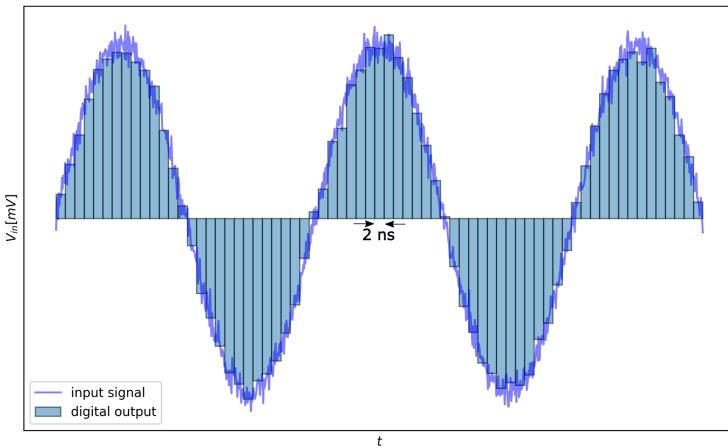


Figure 10: Representación de la digitalización a 500 MSPS de una señal proveniente de un sensor. La señal sinusoidal con una componente de ruido representa la señal analógica generada por un sensor, mientras que las barras azul claro dan cuenta de los niveles digitales equivalentes a la señal original con una resolución temporal de 2 ns.

1.5 Digital processing

El procesamiento digital de señales (DSP) implica la aplicación de operaciones matemáticas a datos digitales, es decir, números en representación binaria, para analizar, modificar o transformar señales. Este proceso se basa en algoritmos matemáticos que pueden ser implementados en diferentes plataformas de hardware, cada una con sus propias características y ventajas [10]. En la figura 11 se ilustra un esquema genérico de la etapa de procesamiento digital de señales en un DAQ.

Las plataformas comunes para el procesamiento digital de señales incluyen varios tipos de sistemas embebidos como microcontroladores (μ Cs), CPUs, FPGAs, GPUs, y DSPs especializados. Cada una de estas plataformas ofrece distintas capacidades en términos de velocidad de procesamiento, latencia, consumo de energía y portabilidad, entre otros factores enumerados a continuación [9]:

- Microcontroladores: Generalmente utilizados para aplicaciones con requisitos de procesamiento relativamente bajos. Ofrecen una buena relación entre costo y funcionalidad, ideal para tareas de procesamiento en tiempo real con bajo consumo de energía.
- CPUs: Adecuadas para aplicaciones que requieren procesamiento general

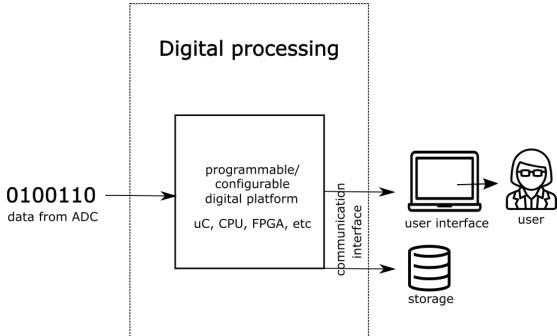


Figure 11: Ilustración de la etapa de procesamiento digital. A la izquierda, los datos provenientes del ADC ingresan al sistema de procesamiento. A continuación, son operados de acuerdo a los algoritmos implementados en el sistema embebido, seguido de la información procesada que se dirige hacia las etapas posteriores de la cadena de información.

y flexibilidad. Son versátiles y poderosas, pero pueden no ser las mejores para tareas que requieren alta velocidad de procesamiento o baja latencia.

- **GPUs:** Excelentes para el procesamiento paralelo de grandes volúmenes de datos, como en el procesamiento de imágenes o aprendizaje automático. Son capaces de manejar tareas complejas con alta eficiencia, aunque pueden consumir más energía y ser más costosas.
- **DSPs:** Diseñados específicamente para el procesamiento de señales digitales, están optimizados para realizar operaciones matemáticas complejas de manera eficiente. Sin embargo, su producción suele ser costosa debido a su implementación en formato ASIC (Application-Specific Integrated Circuit), lo que puede incrementar el costo total del sistema.
- **FPGAs:** Ofrecen flexibilidad y alta velocidad de procesamiento al permitir la implementación de circuitos personalizados. Son útiles para aplicaciones que requieren procesamiento paralelo o que deben adaptarse a cambios en los requisitos de procesamiento.

1.5.1 Digital processing: Filters

Si bien existen diversos tipos de operaciones que se pueden ejecutar en el dominio digital, en el contexto de las señales, las más comunes son los filtros digitales. Estos filtros se utilizan para modificar o extraer información de una señal al atenuar, amplificar o modular ciertas características de la misma, como su amplitud, frecuencia y forma. Los filtros digitales pueden clasificarse en varias categorías, entre ellas [10]:

- **Filtros FIR (Finite Impulse Response):** Estos filtros tienen una respuesta al impulso finita y son conocidos por su estabilidad y capacidad para

diseñar respuestas precisas. Son adecuados para aplicaciones donde se requiere un control exacto sobre la respuesta en frecuencia.

- Filtros IIR (Infinite Impulse Response): A diferencia de los filtros FIR, los filtros IIR tienen una respuesta al impulso infinita. Estos filtros son generalmente más eficientes en términos de recursos computacionales, pero pueden ser más complejos de diseñar debido a la posibilidad de inestabilidad.
- Filtros adaptativos: Estos filtros ajustan sus parámetros en tiempo real para adaptarse a las características cambiantes de la señal o del entorno. Son útiles en aplicaciones donde las condiciones de la señal varían, como en la cancelación de ruido y la ecualización de canales.
- Filtros de paso bajo, paso alto, paso banda y elimina banda: Cada uno de estos filtros está diseñado para permitir o bloquear rangos específicos de frecuencias en una señal. Los filtros de paso bajo permiten pasar frecuencias por debajo de un umbral, mientras que los filtros de paso alto permiten pasar frecuencias por encima de un umbral. Los filtros de paso banda permiten un rango específico de frecuencias y los filtros elimina banda bloquean un rango específico.

Por ejemplo, de acuerdo a [10], el filtro de media móvil es una técnica fundamental en el procesamiento digital de señales, utilizada para suavizar series temporales de datos. Su objetivo principal es reducir el ruido y las fluctuaciones, proporcionando una estimación más estable y continua de la señal subyacente. Este filtro opera promediando los valores de la señal en una ventana de tiempo móvil. Dependiendo del tipo de media móvil utilizado, este proceso puede implicar el cálculo del promedio simple o ponderado de los valores dentro de la ventana. En este caso, la función de respuesta al impulso $h[n]$ es una secuencia de valores constantes que suman a uno, específicamente:

$$h[n] = \frac{1}{N} \cdot \text{rect}\left(\frac{n}{N}\right) \quad (8)$$

donde rect es una función rectangular que es 1 dentro del intervalo de la ventana de tamaño N y 0 fuera de él. Esta respuesta al impulso tiene una longitud de N , que determina el número de muestras de entrada utilizadas para calcular el promedio en cada punto de salida. La salida del filtro de media móvil en tiempo discreto se calcula mediante la convolución de la señal de entrada $x[n]$ con la función de respuesta al impulso $h[n]$:

$$y[n] = (x * h)[n] = \sum_{k=0}^{N-1} x[n-k] \cdot \frac{1}{N} \quad (9)$$

Debido a su respuesta al impulso finita y su implementación directa en términos de convolución, el filtro de media móvil se clasifica como un filtro FIR. Este tipo

de filtro es ampliamente utilizado debido a su estabilidad y facilidad de implementación tanto en hardware digital como en software.

El filtro trapezoidal es una herramienta clave en la instrumentación nuclear, especialmente valorada por su capacidad para optimizar la resolución energética en sistemas de adquisición de datos, como aquellos utilizados en detectores de radiación. Este filtro se aplica a la señal de entrada con el objetivo de obtener su amplitud, la cual es proporcional a la energía depositada en el detector. En términos simples, el filtro funciona como un algoritmo que transforma la típica señal de decaimiento exponencial generada por un preamplificador sensible a la carga en una forma trapezoidal. La diferencia de altura entre la línea base de la señal y la parte superior plana del trapezoide refleja la amplitud del pulso inicial, proporcionando una medición directa de la energía del evento.

Además, este filtro tiene la capacidad de mitigar el efecto de pile-up, que ocurre cuando dos o más pulsos de señal se superponen en un corto intervalo de tiempo debido a altas tasas de eventos. Esto provoca la distorsión de los pulsos y deteriora la resolución energética. El filtro trapezoidal, al modificar la forma de los pulsos y reducir su duración, permite distinguir mejor entre eventos consecutivos, minimizando el impacto del pile-up y garantizando mediciones más precisas. Esta capacidad lo convierte en una herramienta fundamental en la instrumentación nuclear y de detección, especialmente en entornos de alta actividad donde la superposición de señales es frecuente.

Es fundamental resaltar que el filtro asume que la señal de entrada se puede aproximar como una función escalón, lo cual es adecuado para modelar los pulsos del preamplificador en términos del tiempo de subida. Sin embargo, esta aproximación no considera el decaimiento natural del pulso. Para corregir este efecto y evitar distorsiones en la medición, se implementa una técnica de corrección polo-cero. Esta técnica ajusta las características del filtro, compensando la caída de la señal y garantizando una mayor precisión en la determinación de la energía.

El filtro trapezoidal digital se basa en un algoritmo recursivo que convierte un pulso exponencial digitalizado $v(n)$ en un pulso trapezoidal simétrico $s(n)$ [11]. El proceso comienza con el cálculo de $d^{k,l}(n)$, una combinación de diferencias de muestras de la señal $v(n)$ en distintos momentos del tiempo. Esta fórmula resta valores de $v(n)$ separados por k y l muestras para obtener la diferencia temporal de la señal:

$$d^{k,l}(n) = v(n) - v(n - k) - v(n - l) + v(n - k - l) \quad (10)$$

Posteriormente, las siguientes ecuaciones definen cómo se calcula el pulso resultante $s(n)$ en función de las sumas acumuladas de $d^{k,l}(n)$, lo que genera una forma trapezoidal. Los valores iniciales de $p(n)$, $r(n)$ y $s(n)$ son iguales a cero para $n < 0$.

$$p(n) = p(n - 1) + d^{k,l}(n), \quad n \geq 0 \quad (11)$$

$$r(n) = p(n) + M \cdot d^{k,l}(n) \quad (12)$$

$$s(n) = s(n - 1) + r(n), \quad n \geq 0 \quad (13)$$

donde $p(n)$, $r(n)$ y $s(n)$ son las variables acumuladas. El parámetro M depende del tiempo de decaimiento τ del pulso exponencial y del período de muestreo T_{clk} , y se define como:

$$M = \frac{1}{\exp\left(\frac{T_{clk}}{\tau}\right) - 1} \quad (14)$$

Cuando $\frac{\tau}{T_{clk}} > 5$, M puede aproximarse como:

$$M \approx \frac{\tau}{T_{clk}} - 0.5$$

El cálculo de $d^{k,l}(n)$ se puede expresar mediante dos procedimientos secuenciales:

$$d^k(n) = v(n) - v(n - k) \quad (15)$$

$$d^{k,l}(n) = d^k(n) - d^k(n - l) \quad (16)$$

Este algoritmo implementa de manera eficiente la conversión de un pulso exponencial en una señal trapezoidal, mejorando la precisión en la medición de la energía del evento detectado al eliminar ruido de alta frecuencia y adaptar la forma de la señal para su procesamiento digital.

Además de los filtros para señales, otros subsistemas de procesamiento pueden implementarse en la etapa de procesamiento digital. Ejemplos de estos incluyen los sistemas de trigger, que generan una señal digital en respuesta a eventos específicos, activando acciones en otras etapas del sistema de adquisición o en sistemas de adquisición conectados y son esenciales en experimentos de detección de partículas. Del mismo modo, en el contexto de la instrumentación científica para mediciones de radiación, se encuentran sistemas digitales complejos de gran utilidad para extraer información física de interés como son:

- Analizador de Canal Único (SCA): Se emite una señal digital si la señal de energía está entre un rango de canales designado por un Discriminador de Nivel Inferior (LLD) y un Discriminador de Nivel Superior (ULD). Normalmente se muestra el número de cuentas por segundo.
- Escalado Multicanal (MCS): Similar al modo SCA, pero los datos se registran en un histograma de tiempo, en el que se registran sucesivas tasas de conteo.

- Analizador de Altura de Pulso (PHA): También conocido como MCA, las energías se registran en un histograma de energía, en el que las alturas de los pulsos se categorizan por canal, y el número en cada canal representa la cantidad de veces que se registró una altura de pulso particular durante el tiempo de medición.
- Escalado Multiespectral (MSS): Similar al modo PHA, pero se guarda un histograma de energía en intervalos de tiempo cortos, definidos por el tiempo de residencia del sistema.
- Modo de Lista con Marca de Tiempo (TLIST): En este modo, se registra el momento en que ocurre un evento, con una precisión de hasta 100 ns, junto con la energía del evento. Estos datos se almacenan como una lista de pares tiempo-energía, que luego pueden procesarse en cualquiera de los cuatro tipos de datos mencionados anteriormente. Aunque esta es la opción más general de almacenamiento de datos, también es la que utiliza la mayor cantidad de memoria.

Como se ha mencionado anteriormente, diversas plataformas de hardware pueden utilizarse para aplicaciones de procesamiento digital, incluidos los ejemplos presentados. Sin embargo, es fundamental identificar los requerimientos específicos de cada aplicación para lograr una implementación eficiente. Por ejemplo, un microcontrolador de bajo costo como el ESP32 puede emplearse para desarrollar un PHA, como el descrito por [12]. No obstante, esta implementación enfrenta limitaciones inherentes a la lógica secuencial de los microcontroladores, lo que implica que la lectura de los canales de altura se realice de manera secuencial. Esto puede resultar en la pérdida de eventos en los canales que no se están leyendo en un momento dado, especialmente si la frecuencia de generación de eventos es alta en comparación con el tiempo de muestreo. Esta situación pone de manifiesto la necesidad de realizar la lectura de los canales de altura de forma simultánea, lo que requiere un sistema de procesamiento con capacidad de paralelización y, en consecuencia, la necesidad de utilizar plataformas de hardware más avanzadas.

Con esto presente y dada la necesidad de procesar grandes volúmenes de datos generados por ADCs de alta velocidad mientras se minimiza la latencia, para el desarrollo de DAQs de alto rendimiento, se hace imprescindible la implementación de sistemas de procesamiento basados en FPGA. Además, esta tecnología permite la evolución continua de los sistemas digitales gracias a su capacidad de reconfiguración.

1.5.2 Digital Processing: FPGA

Una FPGA es un tipo de circuito integrado reconfigurable que permite a los usuarios personalizar su arquitectura interna para realizar tareas específicas, diferenciándose de los microporcesadores tradicionales, que operan con un conjunto de instrucciones fijas. Estas matrices de puertas programables en campo

son ampliamente utilizadas en aplicaciones que requieren procesamiento en paralelo y alta flexibilidad, como en sistemas de telecomunicaciones y procesamiento de señales digitales. La capacidad de reprogramación de las FPGA les otorga una ventaja significativa en términos de adaptabilidad a nuevas necesidades sin requerir modificaciones físicas en el hardware. Para programar una FPGA, se utilizan lenguajes de descripción de hardware (HDL) como VHDL o Verilog, que permiten definir circuitos personalizados que se cargan directamente en el dispositivo, configurando su comportamiento según el diseño especificado [5].

Como se ilustra en la figura 12, internamente una FPGA se compone de una matriz de bloques lógicos programables (CLBs), interconexiones configurables, y recursos adicionales como bloques de memoria (BRAM). Los CLBs contienen LUTs (Look-Up Tables) y flip-flops, que implementan funciones lógicas y almacenan estados. Las interconexiones programables permiten la comunicación entre los bloques lógicos a través de una red de enrutamiento configurable. Además, suelen incluir bloques dedicados para funciones específicas, como multiplexores y controladores de entrada/salida, lo que proporciona una gran flexibilidad y capacidad de adaptación a distintas aplicaciones [13].

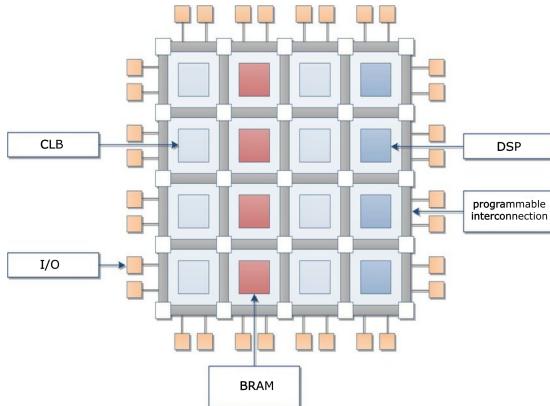


Figure 12: Diagrama de la arquitectura típica de una FPGA. En este se pueden observar las partes fundamentales del dispositivo como los CLBs, puertos de entrada y salida I/O, módulos de BRAM, etapas dedicadas al procesamiento de señales DSP y las interconexiones programables. Tomado de [14].

Esta arquitectura ofrece baja latencia, lo cual es esencial para manejar el gran volumen de datos generado por ADCs de alta velocidad. Su capacidad de procesamiento paralelo permite gestionar eficientemente altas tasas de muestreo, asegurando un procesamiento de datos proveniente de sistemas de sensado sin pérdida de información. En particular, las FPGA ofrecen una flexibilidad considerable en el diseño e implementación de algoritmos de procesamiento de señales, incorporando módulos de hardware dedicados como los DSPs (Digi-

tal Signal Processors) [9].

1.5.3 Digital processing: FPGA SoC

No obstante, como señala Bravo [13], los avances en tecnología FPGA han impulsado el desarrollo de sistemas híbridos, como los SoC (System on Chip). Según el fabricante AMD [15], un FPGA SoC es un dispositivo que integra en un solo chip la flexibilidad programable de una FPGA, también conocida como Lógica Programable (PL, por sus siglas en inglés), con las capacidades de procesamiento de un sistema basado en un procesador (PS, Processing System), generalmente de arquitectura ARM. Como se muestra en la figura 13, existe un subsistema de intercomunicación o bus entre el PS y el PL, donde típicamente el PS actúa como maestro y el PL como esclavo.

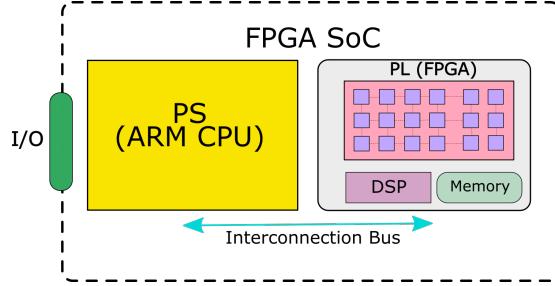


Figure 13: Arquitectura típica de un FPGA SoC. En su estructura fundamental se aprecia el procesador (PS), la FPGA (PL) y el bus de interconexión entre estos dos.

Esta configuración permite aprovechar lo mejor de ambos mundos: la capacidad de realizar tareas complejas y variables mediante software y, al mismo tiempo, la ejecución de operaciones intensivas en paralelo y en tiempo real mediante la lógica programable de la FPGA. Además, esta integración incluye funcionalidades adicionales como procesamiento digital de señales (DSP), dispositivos de señal mixta, y la posibilidad de reemplazar otros componentes dedicados como los ASICs (Application-Specific Integrated Circuits) o ASSPs (Application-Specific Standard Products), todo en un solo dispositivo, optimizando así el rendimiento y la eficiencia energética para aplicaciones específicas.

Es importante destacar las interfaces de memoria disponibles en estos dispositivos. Las FPGA suelen integrar varios bloques de memoria, como registros, RAM y FIFOs, cada uno con funciones específicas. Los registros son pequeños bloques de memoria que almacenan datos temporales para operaciones inmediatas, lo que permite un acceso rápido y eficiente. La RAM (Memoria de Acceso Aleatorio) se utiliza para el almacenamiento de datos volátiles, ofreciendo mayor

capacidad que los registros, pero con tiempos de acceso más largos. Las FIFOs (Colas de Primeros en Entrar, Primeros en Salir) son estructuras de almacenamiento que gestionan flujos de datos de manera ordenada, lo que resulta fundamental para el procesamiento secuencial y la sincronización entre diferentes módulos [13]. Por su parte, el procesador tiene acceso a estos recursos de memoria y periféricos a través de la interfaz DMA (Direct Memory Access), permitiendo la administración de los datos almacenados en las memorias de la FPGA a través del bus de intercomunicación. De esta manera, el PS ofrece al usuario la capacidad de acceder, modificar o extraer estos recursos de memoria, facilitando la adquisición de datos y el control de manera efectiva.

Debido a la naturaleza de los SoC FPGA, estos no cuentan con un conjunto de instrucciones predefinido de fábrica. Para implementar aplicaciones en esta tecnología, es necesario diseñar circuitos lógicos y desarrollar algoritmos que configuren y controlen la plataforma de acuerdo a los requerimientos específicos del proyecto. En arquitecturas de sistemas de adquisición de datos (DAQs) de alto rendimiento, especialmente en el contexto de la detección de partículas, es común dividir las funciones en "fast control" y "slow control" [4]. El "fast control", generalmente implementado en FPGA, gestiona procesos de alta velocidad y tiempo real, como el procesamiento digital de señales en pipeline, mientras que el "slow control", basado típicamente en plataformas de microprocesadores, se encarga de los algoritmos de supervisión del sistema. La tecnología SoC con FPGA permite integrar ambas funciones en un solo hardware, distribuyendo la lógica de fast control en el PL y los algoritmos de supervisión de niveles de voltaje, gestión de comunicaciones y acceso a memoria en el PS.

1.6 Communication interface

Si bien las plataformas mencionadas pueden realizar operaciones en línea, procesando las señales a medida que ingresan al sistema, es esencial enviar la información a etapas externas del sistema de adquisición de datos (DAQ) para realizar actividades adicionales, como almacenamiento, posprocesamiento, interacción con otros sistemas de detección (como triggers) y transferencia a interfaces de alto nivel de abstracción, como computadoras de escritorio, para que el usuario pueda acceder a los datos. La etapa encargada de dicha transferencia es la interfaz de comunicación.

Es importante señalar que la mayoría de la tecnología de comunicaciones se basa en señales digitales, es decir, en la manipulación de bits (1s y 0s). Esto implica que el sistema de procesamiento digital debe implementar un conjunto de instrucciones en forma de algoritmos o subsistemas para codificar la información procesada en un formato específico, asegurando así una transmisión efectiva. Además, la interfaz de comunicaciones se compone principalmente de un estándar de hardware y un protocolo digital particular y dependiendo de la complejidad de la aplicación y del DAQ, se pueden emplear diferentes tipos. Algunos ejemplos de estos protocolos, que abarcan desde tecnología de consumo

hasta grandes sistemas de servidores, incluyen: RS-232, RS-485, USB, GPIB, Ethernet, Wi-Fi, Modbus/TCP, CAN Bus, Profibus, Profinet, SCADA, EtherCAT, MMS, MQTT, OPC UA, LHC Computing Grid e INFN-Tier-1 [16] [17].

Aunque las comunicaciones paralelas fueron comunes en el pasado, su implementación en largas distancias se ha vuelto impráctica debido a la necesidad de una línea de transmisión para cada bit de información, lo que incrementa la complejidad, el costo y los riesgos de interferencia y desincronización. Por esta razón, la mayoría de las interfaces de comunicación alámbricas modernas son de tipo serial. Las interfaces seriales, como USB, Ethernet y UART, permiten la transmisión de datos a altas velocidades utilizando solo unas pocas líneas, lo que las hace más eficientes, fiables y económicas para una amplia gama de aplicaciones, desde sistemas de adquisición de datos hasta comunicaciones en redes de larga distancia [18].

En términos generales, la transmisión de datos en serie se puede clasificar según la manera en que se realiza la transmisión:

- Simplex: el hardware está diseñado de manera que la transferencia de datos ocurre únicamente en una dirección. No es posible transferir datos en la dirección opuesta. Un ejemplo típico es la transmisión desde una computadora hacia una impresora.
- Half Duplex: La transmisión half duplex permite la transferencia de datos en ambas direcciones, pero no de manera simultánea. Un ejemplo común es el uso de un walkie-talkie.
- Full Duplex: La transmisión full duplex permite la transferencia de datos en ambas direcciones de manera simultánea. Un ejemplo típico son las líneas telefónicas.

Es importante destacar que, aunque hasta ahora se ha presentado el concepto de DAQ como un sistema en el que la información fluye desde el sistema físico hacia el usuario, como se ilustra en la figura 1, en la mayoría de los dispositivos o instrumentos de medición es común que también exista un flujo de información en la dirección opuesta. Este subsistema, conocido como sistema de control, abarca todas las acciones que el usuario puede realizar para configurar parámetros en el sistema, ya sea para controlar actuadores o para modificar modos de operación en cualquiera de las etapas descritas. En los dispositivos que cuentan con esta funcionalidad, es crucial disponer de interfaces de comunicación half duplex o full duplex, que permitan la modificación de parámetros en el sistema de manera simultánea al flujo de información de medición.

De acuerdo con [18], los datos en la interfaz de comunicación en serie pueden enviarse en dos formatos principales: asíncrono y síncrono. En el formato asíncrono, orientado a caracteres, los bits de un carácter o palabra de datos se envían a una tasa constante. Sin embargo, los caracteres pueden llegar

a cualquier ritmo (asíncronamente), siempre que no se solapen. Durante los períodos en que no se envían caracteres, una línea se mantiene en nivel alto (lógica 1), conocido como "mark" (marca), mientras que la lógica 0 se denomina "space" (espacio). El inicio de un carácter se señala mediante un bit de inicio, que siempre es bajo, y se utiliza para sincronizar el transmisor con el receptor. Despues del bit de inicio, se envían los bits de datos comenzando por el bit menos significativo, seguido de uno o más bits de parada, que son activos en alto y marcan el final del carácter. Dependiendo del sistema, se pueden utilizar 1, 1 1/2 o 2 bits de parada. La combinación del bit de inicio, el carácter y los bits de parada se conoce como "frame" (trama). Aunque los bits de inicio y de parada no transportan información útil, son esenciales debido a la naturaleza asíncrona de la transmisión de datos. Además, la tasa de transmisión puede expresarse en bits por segundo (bits/seg) o caracteres por segundo (caracteres/seg), siendo el término bits/seg también conocido como la tasa de baudios. Este formato asíncrono es comúnmente utilizado en transmisiones de baja velocidad, típicamente menores a 20 Kbits/seg.

Por otro lado, los bits de inicio y de parada en cada trama del formato asíncrono representan bytes de sobrecarga que reducen la tasa general de caracteres. Estos bits de inicio y parada pueden eliminarse al sincronizar el receptor y el transmisor, la cual se puede lograr mediante una señal de reloj común.

Independientemente de si el dispositivo receptor de la información del DAQ es una computadora de escritorio o un clúster de servidores, es crucial contar con un método sistemático para la decodificación de los datos. Usualmente, se diseñan programas informáticos con algoritmos específicos para reorganizar la información que llega empaquetada en formatos definidos por el protocolo utilizado. Este software actúa como una capa de primer nivel en la gestión de la información proveniente del DAQ y debe integrarse con otros programas para transferir los datos a capas de abstracción que permitan al usuario acceder a la información relevante adquirida por el sistema. Este tipo de software, comúnmente conocido como controlador de comunicaciones, debe ser instalado o implementado junto con otros controladores del DAQ en el dispositivo de recepción.

Para ilustrar el uso de las interfaces de comunicación en la instrumentación científica y los sistemas de adquisición de datos (DAQs), consideremos dos instrumentos de medición con diferentes requerimientos y niveles de complejidad: un sistema de monitoreo de temperatura ambiental y un sistema de detección de partículas radiactivas en una central nuclear. En el primer caso, dado que las variaciones de temperatura debidas a cambios ambientales ocurren en un orden temporal de minutos o incluso horas, es suficiente adquirir las señales del sensor de temperatura con una granularidad de segundos, lo que corresponde a una frecuencia de muestreo del orden de los Hz. Esto permite implementar una plataforma de procesamiento digital de bajo rendimiento, como un microcontrolador de bajo consumo, similar a los utilizados en tarjetas de desarrollo como

Arduino. En consecuencia, la interfaz de usuario podría ser un bus serial USB, fácilmente configurado en el microcontrolador y conectado a un ordenador. Con un software simple, los datos pueden ser decodificados y representados en una interfaz de usuario, permitiendo el análisis de los cambios de temperatura ambiental, que es el objetivo de esta aplicación.

Por otro lado, en el caso de un sistema de detección de radiación, no solo es posible que las tasas de generación de eventos alcancen el orden de los kHz, sino que los eventos individuales pueden durar del orden de μ s o incluso ns, dependiendo del tipo de detector [3]. Estas características exigen que el sistema de digitalización, el procesamiento digital, y la correspondiente interfaz de comunicaciones tengan un rendimiento significativamente superior al del ejemplo anterior. En este caso, es probable que un bus de comunicación serial USB se sature, lo que resultaría en la pérdida de información valiosa debido a la latencia de la interfaz. Por lo tanto, sería necesario considerar interfaces diseñadas para gestionar y transmitir grandes volúmenes de datos, como Ethernet. Si además se garantiza que el dispositivo receptor, como un ordenador, cuenta con el software adecuado para administrar y decodificar la información proveniente de la interfaz Ethernet, se podrá representar la información de manera útil para el usuario, por ejemplo, como un espectro de energías de la fuente radiactiva.

1.7 User interface

La interfaz de usuario en un sistema de adquisición de datos (DAQ) es un componente clave que conecta al operador con el sistema de control y análisis de datos. Esta interfaz puede ser tan simple como una terminal de comandos, también conocida como interfaz de línea de comandos (CLI), donde el usuario interactúa mediante instrucciones textuales, o bien, puede adoptar un enfoque más sofisticado mediante una interfaz gráfica de usuario (GUI).

Mientras que una CLI ofrece control directo y a menudo preciso mediante comandos específicos, las GUI están diseñadas para ser más intuitivas y accesibles, permitiendo a los usuarios interactuar con el sistema a través de elementos gráficos como botones, menús y ventanas. Las GUI son particularmente útiles cuando se requiere visualizar grandes volúmenes de datos o realizar configuraciones complejas de manera visual.

La función principal de estas interfaces, en especial de una GUI, es representar la información física adquirida por el sistema de manera lógica y sistemática, basada en las variables físicas estudiadas. A través de tablas, gráficos estáticos o dinámicos que se actualizan en tiempo real, o histogramas que permiten visualizar distribuciones de datos, la interfaz facilita el análisis de los resultados obtenidos. Estas representaciones visuales permiten a los usuarios interpretar rápidamente el comportamiento del sistema físico bajo estudio, detectando tendencias, anomalías o eventos significativos de forma eficiente.

Además de la representación de los datos, tanto las CLI como las GUI permiten al usuario controlar y configurar las diferentes etapas del DAQ. En una CLI, esto se realiza a través de comandos textuales, mientras que en una GUI se logra mediante recursos gráficos como botones, deslizadores o menús desplegables. Estas opciones de control permiten ajustar parámetros críticos, como la frecuencia de muestreo, los umbrales de detección o la selección de canales de adquisición, lo que otorga al usuario flexibilidad para adaptar el sistema a los requisitos específicos de la aplicación.

La importancia de esta etapa radica en que una interfaz bien diseñada no solo mejora la experiencia del usuario, sino que también maximiza la eficiencia y precisión del sistema. Un diseño intuitivo reduce la posibilidad de errores, simplifica el acceso a configuraciones avanzadas y optimiza el flujo de trabajo del investigador. Las GUI, en particular, permiten una mayor interacción y manipulación en tiempo real de los datos, lo cual es crítico en experimentos que requieren monitoreo constante y ajustes inmediatos.

En el ámbito de la instrumentación científica, la GUI también juega un papel crucial al proporcionar al usuario herramientas de análisis visuales y personalizables. A través de estas interfaces, es posible profundizar en los datos adquiridos, identificar patrones o irregularidades, y generar reportes gráficos que faciliten la toma de decisiones experimentales. Las opciones avanzadas de automatización, como la capacidad de definir secuencias de comandos o macros, permiten al usuario programar el DAQ para realizar tareas específicas de manera automática, lo que es especialmente útil en experimentos de larga duración o con grandes volúmenes de datos.

2 Metodología y montaje experimental

Este trabajo presenta la implementación de un sistema de adquisición de datos (DAQ) monocanal de alto rendimiento, que integra una etapa de digitalización de alta velocidad, un sistema de procesamiento digital basado en FPGA SoC, una interfaz de comunicación de alta capacidad y una interfaz de usuario desarrollada en Python para la visualización y control de datos. La arquitectura está optimizada para adquirir de manera eficiente señales procedentes de diversos sistemas de sensado, complementados por su correspondiente electrónica de acondicionamiento. Además, el sistema es altamente adaptable a una amplia variedad de aplicaciones, que abarcan desde el ámbito biomédico hasta la detección de partículas subatómicas. En este trabajo, sin embargo, se emplea un generador de señales para validar su funcionamiento.

A continuación, se describe la metodología de implementación, que incluye el hardware utilizado, los módulos de adquisición y procesamiento de señales configurados en el PL o FPGA, y el bloque de comunicaciones ComBlock, que permite el acceso a la memoria desde el PS. También se detalla la configuración

de la interfaz de comunicaciones entre el PS y el usuario, denominada UDMA, y la implementación del software de control en Python, que actúa como interfaz de usuario.

Se utiliza el instrumento Digilent Analog Discovery 2 (AD2) (fig. 14) como generador de señales, aprovechando la versatilidad de este dispositivo multifuncional, que también incluye un osciloscopio, una fuente de poder regulable y un analizador lógico, entre otros instrumentos integrados. El AD2 no solo permite acceder a funciones predefinidas, sino que también facilita la programación de formas de señal personalizadas e incluso la creación manual de estas a través de la interfaz gráfica de su software de control, WaveForms (fig. 15). Además, ofrece la opción de añadir ruido y otras características que permiten la generación de señales sintéticas, emulando la respuesta real de sensores o detectores, lo cual es fundamental para la validación del sistema de adquisición de datos (DAQ) implementado.

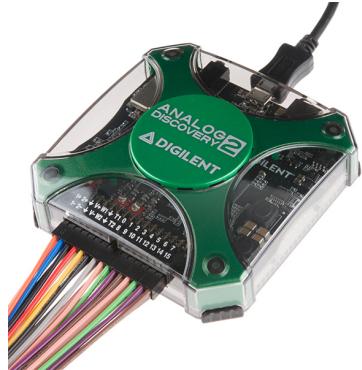


Figure 14: Fotografía del hardware del instrumento multifuncional Digilent Analog Discovery 2 (AD2). Este dispositivo cuenta con generador de señales arbitrarias, fuente regulada y osciloscopio, entre otros.

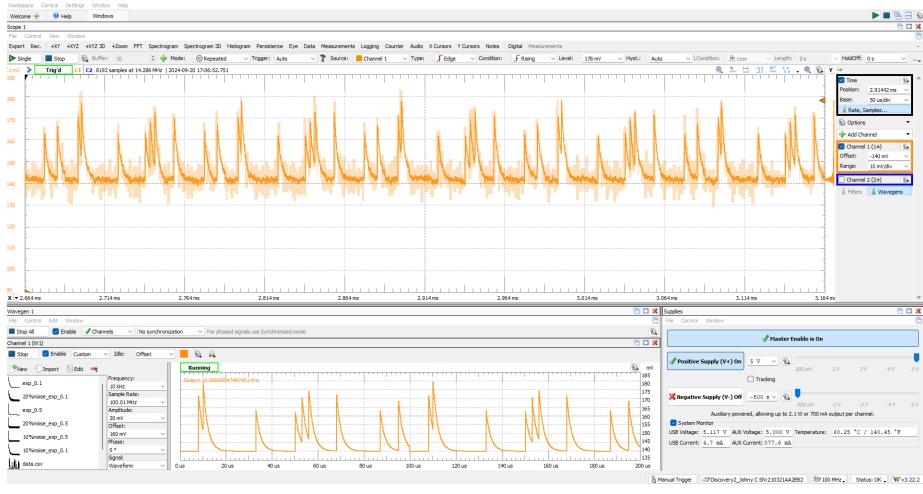


Figure 15: Panel instrumental de WaveForms para el control y adquisición del AD2, donde se identifica la ventana de osciloscopio en la parte superior, el generador de señales en la parte inferior izquierda y la fuente de alimentación en la parte inferior derecha

Como plataforma de digitalización, se hace uso de la tarjeta ICTP-INFN ADC500, en adelante referida como ADC500 (fig 16). Esta tarjeta está basada en un chip Texas Instruments ADC08500 [19] monocanal de alta velocidad, con una frecuencia de muestreo de hasta 500 MHz, 8 bits de resolución y capacidad de digitalización de señales de hasta 1.2 V. Cumple con las especificaciones ANSI/VITA 57.1 y está equipada con un conector de tarjeta mezzanine FPGAs (FMC) de bajo número de pines (LPC) [20].

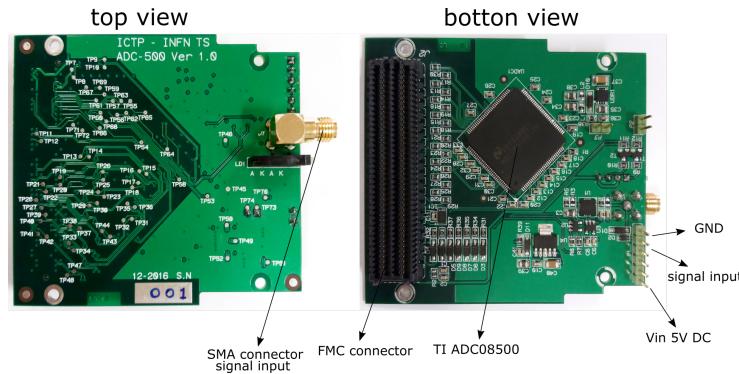


Figure 16: Plataforma de digitalización de señales ICTP-INFN ADC500. Se identifican sus principales partes: chip TI08500, conector FMC, conector SMA y pines de alimentación. Modificado de [21].

Para garantizar el correcto funcionamiento de la tarjeta, es necesario suministrarle 5 V y 1200 mA. La fuente de alimentación utilizada para este fin es la integrada en el AD2, que se conecta a una fuente auxiliar, ya que el puerto USB solo admite 500 mA. La conexión se ilustra en la figura 17.

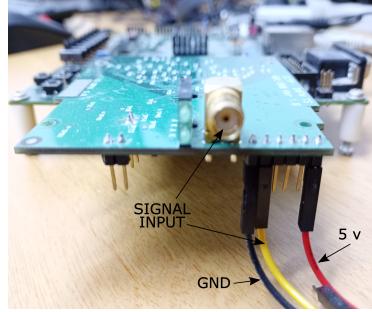


Figure 17: Conexión entre ADC500 y Analog Discovery. Se describen los pines de alimentación de 5 V y GND, así como los puertos de entrada de señal del ADC500, que comprenden un conector SMA conectado en paralelo a un pin de conexión rápida.

Dado que la tarjeta ADC500 está diseñada para integrarse con un sistema de procesamiento basado en FPGA, se implementa un controlador de hardware descrito en lenguaje HDL, que mapea las conexiones para gestionar las funciones del chip. Estas funciones incluyen varios modos de calibración y rangos de digitalización [19]. El controlador para el ADC500, desarrollado por el ICTP-MLAB, se utiliza atribuyendo la autoría correspondiente según su licencia [22]. En la figura 18 se observa la representación en diagrama de bloques del software AMD Xilinx Vivado [23] del controlador mencionado.

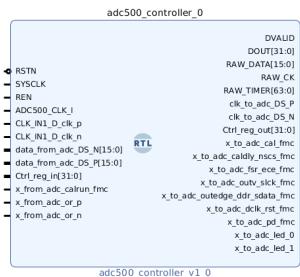


Figure 18: Representación en diagrama de bloques del controlador del ADC500 diseñado en VHDL [22].

Teniendo en cuenta lo expuesto en la sección 1.5 y las características de alto rendimiento del ADC500 como sistema de digitalización, se utiliza una plataforma

de procesamiento digital con capacidad de paralelización de procesos, alta disponibilidad de recursos de computación, reconfigurabilidad y flexibilidad. En este caso se trata de la tarjeta de desarrollo Avnet Digilent Zedboard (figura 19) [24], basada en el SoC (System on Chip) Xilinx Zynq-7000 [15], que integra una FPGA Artix-7 y un procesador ARM Cortex-A9. Esta plataforma ofrece una amplia gama de periféricos tanto de propósito general como específico, incluyendo pines GPIO, pulsadores, interruptores, conectores de audio, interfaces de video HDMI y VGA, Ethernet, puertos USB, y, de particular relevancia para este trabajo, un conector FMC. Además, la Zedboard cuenta con interfaces de comunicación serial y JTAG para la configuración y programación del SoC FPGA.

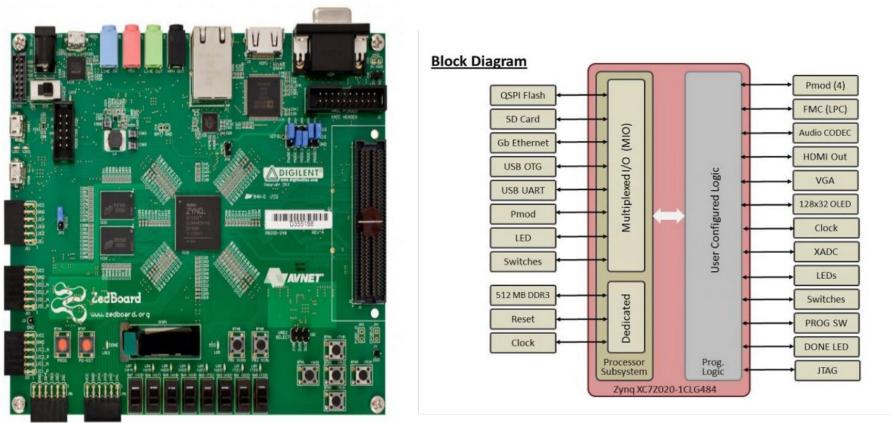


Figure 19: Fotografía de la tarjeta de desarrollo FPGA SoC Avnet Digilent Zedboard y descripción en diagrama de bloques de la distribución de los periféricos en el PS y PL [24].

Para programar y configurar la plataforma FPGA SoC Xilinx Zynq-7000, se emplea el entorno de desarrollo unificado AMD Vitis [25]. Este entorno abarca tanto Vivado [23] para el diseño y configuración de la FPGA (PL), como Vitis SDK para la programación del procesador embebido (PS). En este trabajo, el software Vivado se utiliza para configurar el hardware de control del ADC, los filtros para el procesamiento de señales y la gestión de la comunicación de alta velocidad. Este software permite visualizar scripts de descripción de hardware en bloques con entradas y salidas, permitiendo una abstracción modular de la lógica implementada.

Para interconectar el PL y el PS, se utiliza el ComBlock [26] (fig. 20), un IP portable y altamente configurable diseñado para acceder a interfaces como registros, RAM y FIFOs, evitando la complejidad del bus del Sistema del Procesador (PS), que en el caso del Zynq-7000 es el bus de interconexión AXI y el

AXI DMA como control de memoria [27].

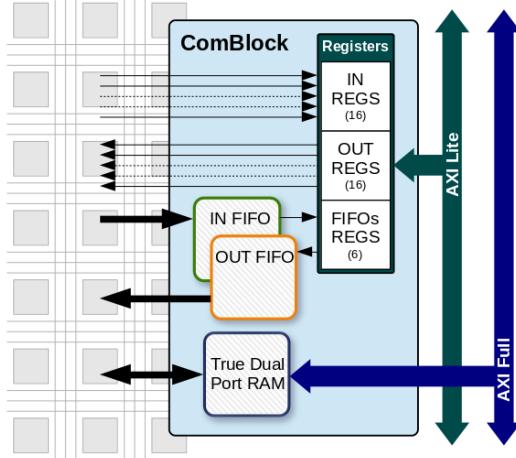


Figure 20: Ilustración de la arquitectura de ComBlock. En este se observa la utilidad del ComBlock como intermediador entre el PL y los buses AXI que interconectan con el PS. Adicionalmente se observan las interfaces de acceso a memoria que este bloque ofrece como registros, FIFO y RAM. Tomado de [26]

Dado que el acceso a la información procesada por el PL se realiza a través del ComBlock, los módulos y filtros implementados en el PL, que se describen a continuación, se conectan de forma bidireccional al ComBlock durante la integración del sistema. Los registros de escritura del ComBlock se emplean para controlar el estado de los registros de entrada y control de los distintos módulos, mientras que las salidas de señal se envían a los elementos de lectura. Estos pueden ser registros para adquirir valores estáticos o FIFO y RAM para señales continuas, como las generadas por los filtros. Tanto los registros de lectura como escritura se configuran con un tamaño de 32 bits para permitir el manejo de valores de alto orden de magnitud. Por otro lado, la FIFO se configura con una profundidad de 4096 datos, valor comparable al manejado por los osciloscopios comerciales.

Teniendo en cuenta que la aplicación de este trabajo se centra en la adquisición y procesamiento de señales de sensores o detectores para extraer medidas de interés en procesos físicos, es fundamental contar con un sistema de visualización que permita inspeccionar las características de estas señales de forma similar a un osciloscopio (véase fig. 21). Este módulo, implementado en VHDL, compara la señal de salida del ADC500 con un umbral configurado por el usuario. Cuando la señal supera dicho umbral, se activa una señal tipo bandera, la cual habilita el registro de escritura de la FIFO del ComBlock, permitiendo el almacenamiento de datos a partir de la muestra en que se cumple la condición.

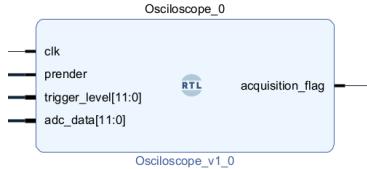


Figure 21: Diagrama de bloques de Vivado de un módulo de trigger simple escrito en VHDL. Entre sus señales de entrada se observan el reloj del sistema, señal de reset, nivel del umbral de amplitud de la señal y señal digitalizada. A su salida se observa la señal para activar la escritura de la FIFO del ComBlock.

Sin embargo, esta lógica presenta una limitación al adquirir señales pulsadas, ya que solo se registra en la memoria la parte del pulso posterior al cruce del umbral. Esto impide la visualización completa de la forma del pulso, lo cual es problemático en aplicaciones como la detección de partículas, donde la forma completa del pulso es un parámetro de interés crucial. Para mitigar este inconveniente, se implementa la lógica de "pre-trigger time". Este sistema acumula muestras digitalizadas en un búfer y, una vez que se cumple la condición de disparo, tanto la porción posterior del pulso como una sección anterior, previamente almacenada, son concatenadas y enviadas a la memoria, permitiendo así la visualización integral del pulso. La lógica se estructura en tres etapas: la lógica de disparo, el temporizador de trigger y un búfer FIFO circular.

El primer módulo en VHDL (fig. 22) implementa una lógica de disparo basada en cruce de nivel de amplitud, que detecta cuando una señal de entrada (`dataIn`) cruza un umbral (`threshold`) predefinido. La lógica permite seleccionar entre activación por cruce ascendente o descendente, controlado por el valor de `edge_Select`. Si `edge_Select` está en 0, el disparo se produce cuando `dataIn` pasa de un valor inferior al umbral a uno igual o superior. Si `edge_Select` está en 1, el disparo ocurre cuando `dataIn` pasa de un valor superior al umbral a uno igual o menor. Este módulo está diseñado para procesar señales con un bus de datos configurable mediante el parámetro genérico `DATA_BUS_WIDTH`, sincronizándose con el reloj (`clk`) y una señal de reinicio asíncrono (`aresetn`).

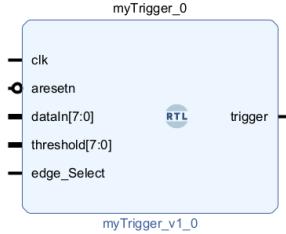


Figure 22: Diagrama de bloques de Vivado del módulo de trigger.

El módulo de temporizador de la señal de disparo en VHDL (fig. 23) genera una señal de disparo (**trig_out**) que se mantiene activa durante un tiempo determinado tras la detección de un evento en la entrada (**trig_in**). La longitud del pulso de salida está controlada por la señal **pulse_len**, que especifica el número de ciclos de reloj durante los cuales el disparo estará activo. Al detectar un flanco ascendente en **trig_in**, se inicia un contador que incrementa en cada ciclo de reloj. Mientras el contador sea menor al valor de **pulse_len**, la señal de disparo permanece en 1. Una vez que el contador alcanza este valor, se reinicia y **trig_out** vuelve a 0.

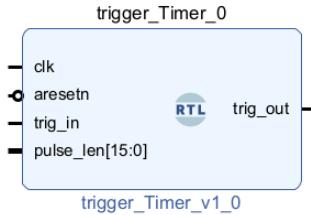


Figure 23: Diagrama de bloques de Vivado del módulo de timer del trigger.

Finalmente, el código que describe un módulo VHDL llamado **circular_FIFO** (figura 24), implementa un buffer circular, utilizado comúnmente para almacenar y gestionar datos de manera eficiente. Este módulo tiene dos puertos: uno para escritura y otro para lectura, controlados mediante las señales **wr_en** (habilitación de escritura) y **rd_en** (habilitación de lectura). El tamaño del buffer está determinado por los parámetros genéricos **RAM_WIDTH** y **RAM_DEPTH**, que definen el ancho y la profundidad de la memoria.

El módulo utiliza dos punteros: **head**, que indica la posición de escritura, y **tail**, que indica la posición de lectura. Cuando se escribe un dato (siempre que el buffer no esté lleno), el puntero **head** se incrementa y el dato se almacena en la posición correspondiente de la memoria (**ram**). De manera similar, cuando se

habilita la lectura y el buffer no está vacío, el puntero `tail` se incrementa y el dato leído se extrae de la memoria. Las señales `empty_i` y `full_i` indican si el buffer está vacío o lleno, respectivamente, y se basan en el valor de un contador (`fill_count_i`) que rastrea cuántos elementos hay almacenados en el buffer.

El procedimiento `incr` garantiza que los punteros de lectura y escritura se encuelen correctamente cuando alcanzan el final del buffer, reiniciándose a la primera posición para mantener el comportamiento circular. La arquitectura incluye procesos para actualizar los punteros y manejar la escritura/lectura en la memoria del buffer, así como para gestionar el conteo de elementos almacenados.

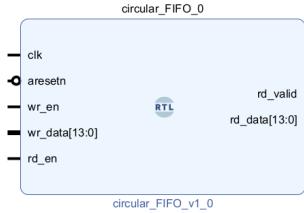


Figure 24: Diagrama de bloques de Vivado del módulo de FIFO circular.

La interconexión entre los tres módulos está organizada de la siguiente manera: la salida del módulo de disparo (trigger) alimenta la entrada del temporizador (timer), y la salida del temporizador activa la escritura en el FIFO circular. Además, el FIFO circular debe incluir un módulo de lógica de registro de desplazamiento (shift register) para configurar el número de muestras a adquirir antes de que se produzca el disparo.

Para visualizar una señal con mayor detalle o, por el contrario, abarcar una ventana de tiempo más amplia, es necesario implementar una etapa de control temporal en la tasa de muestreo. Con este propósito, se diseña un módulo de decimación (fig. 25), el cual actúa como un divisor de la frecuencia de muestreo. Este módulo recibe como entradas la señal de reloj y reset, los datos provenientes del ADC, y el valor de división (N). Este último está conectado a un registro de escritura del comblock, que el usuario puede configurar desde la interfaz de Python. Para garantizar un escalamiento coherente con la decimación aplicada, es fundamental ajustar el eje temporal de la gráfica de visualización de las señales en la interfaz de Python utilizando la relación

$$\frac{N_{\text{data}} * d}{f_{\text{ADC}}} \quad (17)$$

donde N_{data} es el número de datos leídos de la FIFO, d es el factor de decimación y f_{ADC} es la frecuencia de muestreo del ADC.

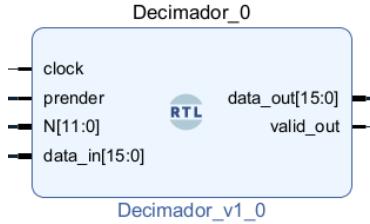


Figure 25: Diagrama de bloques de Vivado de un módulo de decimación escrito en VHDL.

El procesamiento de señales implementado incluye un módulo de media móvil de ventana fija en VHDL (fig. 26), que se utiliza para suavizar señales mediante el filtrado de las mismas. Este filtro realiza un promedio de los valores de una ventana deslizante sobre la señal de entrada, lo que ayuda a reducir el ruido de alta frecuencia y a estabilizar la señal. El filtro está diseñado para procesar datos de entrada en un formato de vector de bits (*i_data*) y generar una señal de salida suavizada (*o_data*). El tamaño de la ventana de promedio se define mediante el parámetro genérico *G_AVG_LEN_LOG*, que determina el número de muestras a considerar en cada cálculo de promedio. El módulo mantiene un acumulador (*r_acc*) que suma los valores de las muestras actuales y un arreglo (*p_moving_average*) que almacena las muestras de la ventana actual. Con cada nuevo dato de entrada, el filtro desplaza la ventana y actualiza el promedio. La señal de salida (*o_data*) se calcula dividiendo el valor acumulado entre el número de muestras en la ventana, logrando así la suavización de la señal. Este enfoque ayuda a eliminar las fluctuaciones rápidas y el ruido de alta frecuencia, proporcionando una representación más estable de la señal. Esto permite analizar directamente su componente de baja frecuencia o ingresarla a un filtro posterior que requiera una señal con bajo nivel de ruido electrónico, como es el caso del shaper trapezoidal.

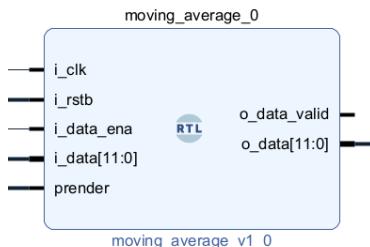


Figure 26: Diagrama de bloques de Vivado de un módulo de media móvil simple escrito en VHDL. Entre sus señales de entrada se observan el reloj del sistema, señal de reset, señal de habilitación y señal digitalizada. A su salida se observa la señal filtrada.

El diseño del filtro trapezoidal comienza con la simulación de los pulsos provenientes de un detector genérico. Se consideran una serie de parámetros clave, como la carga del detector (Q), la capacitancia (C_f), la constante de tiempo (τ) y el período de muestreo (T_s). La constante de tiempo τ se define como el inverso de la frecuencia del sistema, lo que determina la velocidad de decaimiento exponencial de la señal. Asimismo, el período de muestreo genera un número de muestras por segundo igual a $N = 1/T_s$. El término d , calculado como $\exp(-T_s/\tau)$, representa el factor de decaimiento exponencial entre muestras consecutivas. La señal de prueba se genera en un rango de tiempo arbitrario, con un retardo inicial para simular las condiciones reales de detección.

El filtro se basa en una ventana deslizante que realiza una serie de operaciones (fig. 27) sobre las muestras de la señal de entrada, acumulando los resultados en la salida final del filtro. Los tiempos de subida (t_a) y bajada (t_b) definen el comportamiento del filtro, controlando el alisado y la duración del pulso trapezoidal. El algoritmo calcula la salida del filtro trapezoidal utilizando las diferencias entre las muestras de la señal en distintos puntos temporales, ajustadas por el factor de decaimiento d . Esta operación tiene en cuenta los puntos correspondientes a los tiempos t_a , t_b , y un retardo adicional, sumando las contribuciones correspondientes y normalizándolas según el tiempo de subida. Además, la función incorpora términos de corrección para evitar inestabilidades numéricas, asegurando un rendimiento óptimo del filtro en condiciones de detección de señales.

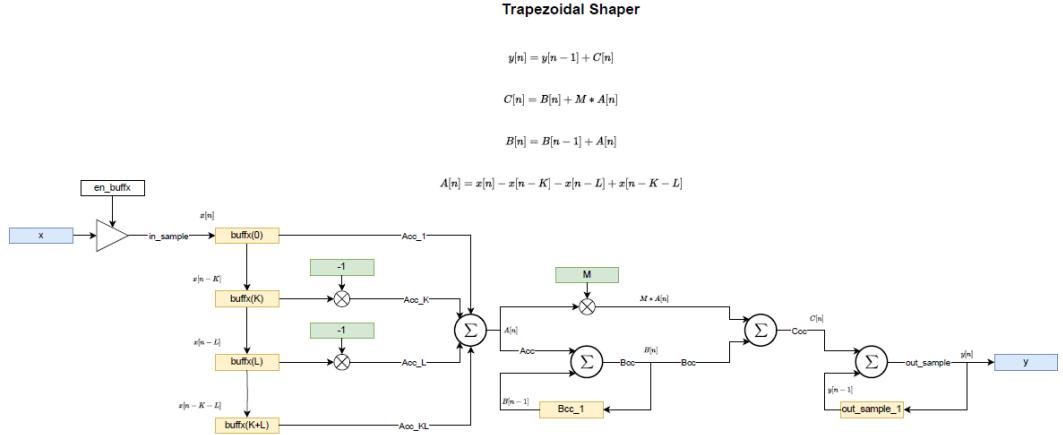


Figure 27: Ecuaciones de recurrencia y diagrama de funciones del filtro trapezoidal implementado.

El filtro FIR se aplica a través de la convolución de la señal sintética con los pesos K , L y M que rigen su comportamiento. Al modificar estos parámetros, se obtienen diferentes respuestas del filtro, optimizándolas según la aplicación. En este caso, se busca que el filtro convierta pulsos exponenciales en pulsos de altura

proporcional para facilitar la extracción de su amplitud y, posteriormente, de la energía depositada. También se pretende que realice *pile-up rejection*, permitiendo diferenciar eventos individuales al ubicar los pulsos trapezoidales sobre la misma línea base. Todo esto con el objetivo de enviar los pulsos trapezoidales al discriminador de eventos y, posteriormente, al MCA para la generación de histogramas de amplitud.

Los parámetros de forma del filtro, K, L y M, se calculan en Python de manera offline a partir de una señal real generada con el AD2 utilizando previamente la señal sintetizada en Python a la que se le añade una componente de ruido, mejorando así la simulación de una señal de detector real. Posteriormente, esta señal se adquiere con el DAQ y se procesa mediante un filtro de media móvil, lo que permite reducir el ruido de alta frecuencia. Considerando la frecuencia de muestreo del ADC500 y el factor de decimación, se utilizan ecuaciones recursivas implementadas en Python para graficar el efecto del filtro sobre la señal, ajustando los parámetros hasta lograr un resultado óptimo.

Posteriormente, se implementa el filtro FIR trapezoidal en VHDL (fig. 28), basado en las ecuaciones de recurrencia descritas y en los parámetros K, L y M optimizados. Este filtro recibe las muestras digitalizadas, previamente filtradas de alta frecuencia mediante una media móvil, y acumula valores según las posiciones de las muestras dentro de un buffer. A través de las operaciones definidas entre las muestras y los parámetros K, L y M, se establecen límites y pesos que controlan aspectos fundamentales del filtro, tales como el tiempo de subida, el flat top y el tiempo de bajada.

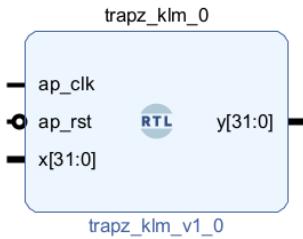


Figure 28: Diagrama de bloques de Vivado del filtro trapezoidal.

Por otro lado, se implementa un Analizador Multicanal (MCA) embebido en el PL, aprovechando las capacidades avanzadas de la memoria True Dual Port RAM a través del ComBlock. Este módulo permite gestionar de manera eficiente y rápida los pulsos provenientes de detectores de radiación, facilitando la construcción de histogramas precisos de las amplitudes de los pulsos. Si el detector utilizado produce pulsos cuya altura es proporcional a la energía depositada por la partícula detectada, estos histogramas pueden ser empleados para generar espectros de distintos tipos, tras una calibración adecuada, según

la técnica experimental utilizada, como Raman o Mössbauer, por ejemplo.

El módulo de VHDL `bram_incr` (fig. 29) implementa una máquina de estados que gestiona operaciones en un bloque de memoria RAM (BRAM). Su funcionalidad principal es leer un valor de una dirección específica del BRAM, incrementar ese valor en uno, y luego almacenar el resultado de vuelta en el BRAM. El módulo recibe señales de entrada para la dirección del BRAM y la disponibilidad de datos, y utiliza una máquina de estados con cuatro etapas para controlar el flujo de operaciones. Las señales internas controlan la lectura, el incremento y la escritura de datos en el BRAM.

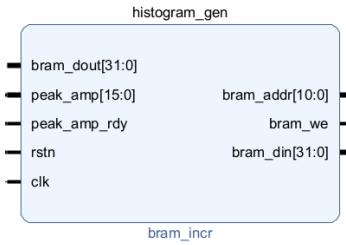


Figure 29: Diagrama de bloques de Vivado del mca.

Como se mencionó anteriormente, los módulos de PL procesan las señales del ADC, enviándolas al ComBlock para su adquisición por el PS y posterior transmisión al usuario. La estructura general del diseño de bloques sigue la topología mostrada en la figura 30. En el recuadro del lado derecho (PL), se encuentran el controlador del ADC500, los diferentes filtros y el ComBlock, mientras que en el lado izquierdo se ubica el controlador del PS.

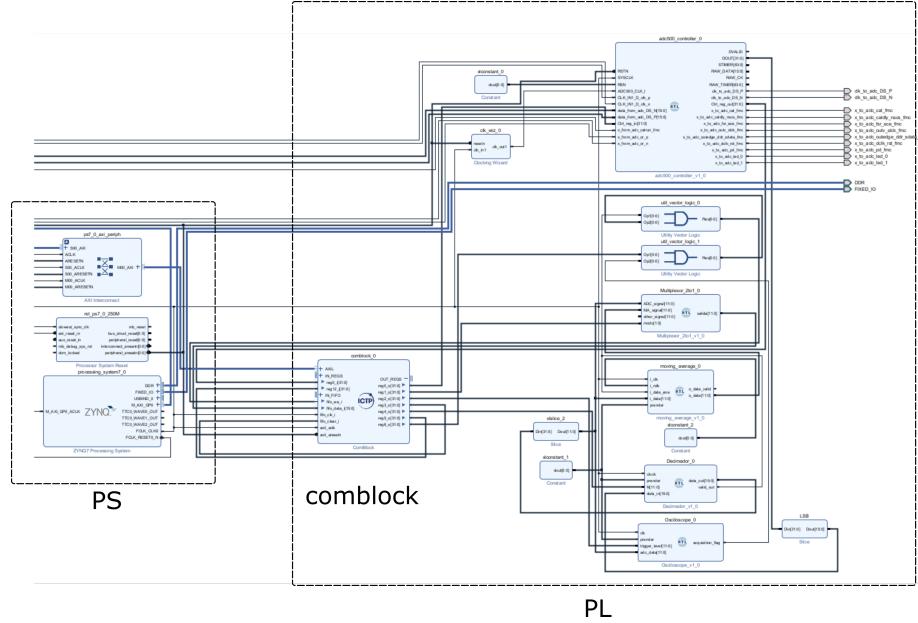


Figure 30: Diagrama de bloques de Vivado donde se observa la interconexión típica entre PL y PS mediada por el ComBlock.

Para la implementación de la interfaz de comunicación bidireccional entre el usuario y el sistema de procesamiento (PS), se emplea el UDMA (Universal Direct Memory Access), un sistema desarrollado en el laboratorio multidisciplinario del ICTP (MLAB) como un conjunto de herramientas de control diseñado para conectar una PC con la lógica personalizada en un SoC-FPGA [20]. El UDMA actúa como un puente entre un generador y un receptor de datos basado en un sistema operativo de tiempo real (RTOS), permitiendo realizar operaciones de lectura y escritura en cualquier memoria dentro del PS por medio del protocolo TCP/IP a través del puerto de Ethernet. Está compuesto por una librería en C y otra en Python, que, al trabajar en conjunto, proporcionan una plataforma de control accesible para el hardware y software integrados en el SoC-FPGA. En particular, el UDMA puede implementarse junto con el ComBlock para permitir al usuario acceder a la información que entra y sale del PL mediante una interfaz de alto nivel basada en Python, proporcionando así el último eslabón de la cadena del DAQ, como se muestra en la figura 31.

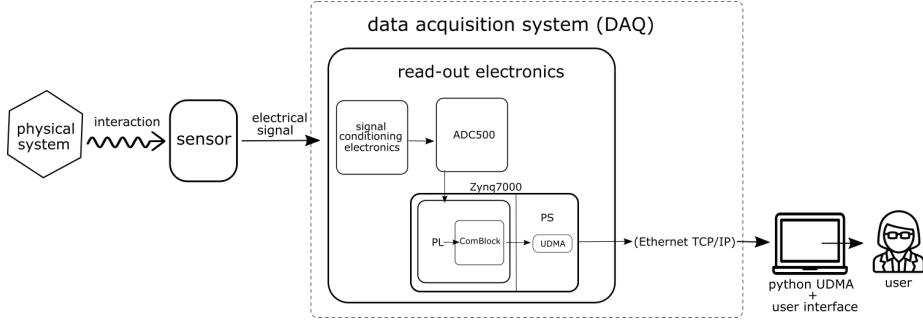


Figure 31: Diagrama esquemático del DAQ implementado en este trabajo, donde se observa el flujo de información desde un sistema físico hasta el usuario, particularizando el diagrama 1 con las etapas descritas en este trabajo como el ADC500, FPGA SoC Zynq7000 y UDMA.

Desde la perspectiva del usuario (PC), se importa la librería UDMA en Python, la cual está basada en una clase que actúa como un envoltorio para un socket TCP y emplea un protocolo simple para gestionar los comandos. Al instanciar la clase, se especifican la dirección IP y el puerto del servidor, estableciendo el socket, lo que permite ejecutar comandos proporcionando los parámetros adecuados [28]. En la figura 32 se muestra un ejemplo del uso de la interfaz UDMA en Python, donde se observa la importación de la librería, la instancia del objeto UDMA y el uso de métodos para la lectura y escritura de los elementos de memoria asignados en el ComBlock.

```
#Libraries
import udma
import socket
#-----

#ComBlock output registers
TRIG_LEVEL = 2
#-----

#Constants
DataNumber= 4000
trigger_level = 890
#-----

#UDMA object instantiation
oudma = udma.UDMA_CLASS("192.168.1.10",7)
#-----

#Write method for ComBlock registers
oudma.write_reg(TRIG_LEVEL,trigger_level)
#-----

#Write method for ComBlock registers
valfifo = oudma.read_fifo(DataNumber)
#-----
```

Figure 32: Interfaz de usuario del DAQ implementado basada en Python. Se observa la importación, instancia y uso del paquete UDMA, que permite la comunicación del usuario con el PS y por tanto con el PL a través del ComBlock.

Cuando se ejecutan los comandos del UDMA, la librería de Python está diseñada para enviar al usuario una estructura de mensaje de confirmación ("acknowledge") que indica si hubo algún error en la ejecución. Si el comando es de escritura, la librería confirma los valores enviados; si es de lectura, proporciona los datos correspondientes a la adquisición. En el caso de la lectura de la memoria FIFO o RAM del ComBlock, los datos obtenidos, ya sea en forma de arreglo o como transmisión en tiempo real, pueden representarse gráficamente o en tablas utilizando librerías especializadas de Python, o procesarse de manera offline para su posterior análisis. En la figura 33 se muestra el hardware utilizado y su conexión.

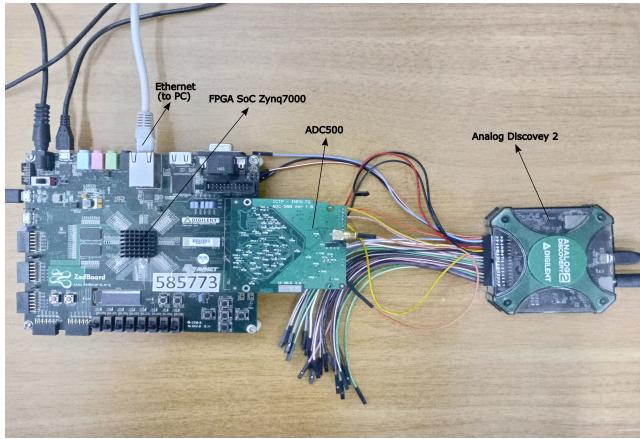


Figure 33: Fotografía del hardware del DAQ implementado en este trabajo. A la derecha se observa el AD2 como instrumento generador de señales arbitrarias y fuente de alimentación para el ADC500, a continuación se aprecia la tarjeta ADC500 acoplada mediante el puerto FMC a la tarjeta de desarrollo del FPGA SoC Zynq7000 y las conexiones con el computador de escritorio para la interacción con el usuario.

3 Resultados

Como se explicó anteriormente en la sección de Montaje experimental y metodología, la implementación efectiva del DAQ requiere la correcta conexión de los bloques de control, adquisición y procesamiento digital al ComBlock, así como la programación del UDMA en el PS como interfaz de comunicación y el desarrollo de una interfaz de usuario en Python (ver fig. 31). Por lo tanto, el primer resultado significativo de este trabajo es la integración de las etapas descritas y la obtención de datos que confirman implícitamente su correcto funcionamiento. Esto se evidencia, por ejemplo, en el encendido de los LEDs de la tarjeta ADC500 (fig.) al enviar el comando

```
CTR_ADC_OUT_REG = 0
```

```

INIT_ADC = 128
oudma.write_reg(CTR_ADC_OUT_REG, INIT_ADC)

```

desde la interfaz de Python, teniendo en cuenta que el valor 128 es una instrucción interna del ADC500 para su encendido y el puerto de control de encendido del bloque de Vivado del controlador del ADC500 18 se encuentra conectado al registro de escritura 0 del ComBlock

INCLUIR FOTO DE ADC LED ENCENDIDOS!!!!!!

Dado que la cadena de flujo de datos desde el ADC500 hasta la interfaz de usuario opera correctamente, se dirige el enfoque de las pruebas y sus resultados hacia los módulos de procesamiento de señales configurados en el PL o FPGA. Para ello, se emplean señales sintéticas que simulan la respuesta real de sensores o detectores, generadas por el AD2. Esto permite validar el funcionamiento de dichos módulos, variando parámetros de la señal como su forma, amplitud, offset, frecuencia y proporción de ruido añadido. En particular, se implementan señales sinusoidales, pulsos exponenciales y trenes de pulsos que presentan características similares a las señales típicamente generadas por sensores o detectores.

Inicialmente, se muestra el modo osciloscopio, donde se valida la capacidad de modificar la ventana temporal de adquisición desde la interfaz de usuario en Python, utilizando señales de prueba para comprobar que el período de la señal corresponde con el adquirido. A continuación, se describe el funcionamiento de los filtros configurados, como la media móvil y el "shaper" trapezoidal. Finalmente, se presenta el funcionamiento del MCA (Analizador Multicanal) o generador de histogramas multifuncional, utilizando un flujo continuo de pulsos de diferentes amplitudes como fuente, simulando la respuesta de un detector de radiación ante una fuente radiactiva o similares.

En la figura 34, se presenta la estructura característica de la interfaz de usuario en Python, donde se pueden observar los comandos de control utilizados para configurar los distintos módulos de hardware en la FPGA, mediante el método de escritura en registros del ComBlock a través de la instancia del UDMA. Asimismo, se muestra el comando para la lectura de la FIFO del ComBlock, y finalmente, la utilización de la herramienta gráfica `matplotlib` para la visualización de los datos obtenidos.

```

#acquisition parameters
DataNumber = 1500
deci = 50
trigger_level = 0

#DAQ control commands
oudma.write_reg(MUX_MODE_REG,RAW_ADC)
oudma.write_reg(DECI_N, deci)
oudma.write_reg(TRIG_LEVEL,trigger_level)

#acquisition FIFO data
valfifo = oudma.read_fifo(DataNumber)

#time axis scaling as a function of decimation factor
tm = np.linspace(0,DataNumber*deci/250,DataNumber)

#plot data
plt.xlabel("t [us]")
plt.ylabel("Amplitude [ADC units]")
lista = list(valfifo[1])
plt.plot(tm, lista)
plt.grid()

```

Figure 34: Interfaz de usuario de Python donde se evidencia el uso de métodos de escritura y lectura de la librería UDMA para el control y adquisición a través del ComBlock.

De especial importancia es el comando `oudma.write_reg(MUX_MODE, RAW_ADC)`, ya que este permite seleccionar la señal que entra a la FIFO del ComBlock para su visualización al controlar un multiplexor instanciado en el hardware. Este multiplexor cuenta con los modos `RAW_ADC` para observar la señal del ADC500 después del decimador, `MA_FILTER` para observar la salida del filtro pasabajos de media móvil y `SHAPER` para observar la salida del filtro trapezoidal. Debe darse por sentado que, para la obtención de cada resultado, se ejecuta el comando que selecciona la salida de cada filtro.

Para probar el módulo de decimación, que permite controlar la ventana temporal de adquisición, se configura una señal periódica sinusoidal sin ruido añadido, ya que esto facilita la comprobación de cómo se modifica la ventana temporal. En este caso, la señal tiene un período de $50 \mu\text{s}$ y una amplitud de 50 mV .

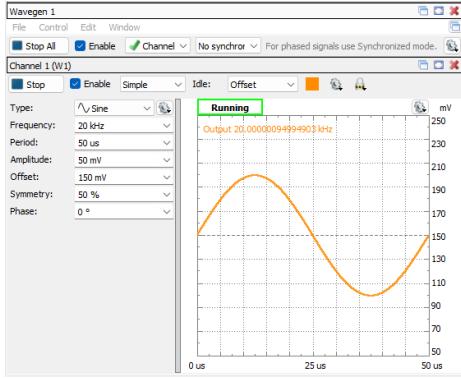


Figure 35: Señal sinusoidal con $T = 50\mu s$ generada con el AD.

En la figura se muestra el resultado de la adquisición de la señal anterior con valores de decimación de 50 y 10 y umbral de trigger configurado en 1500. En ambas gráficas se observa que la señal corresponde a una sinusoidal con un período de $50\mu s$ que inicia en el valor de amplitud de 1500, validando el funcionamiento de la lógica de trigger simple. Sin embargo, en la primera gráfica, un valor de decimación más alto genera una ventana temporal de adquisición mayor, lo que permite observar varios períodos de la señal. En contraste, en la segunda gráfica, se observa solo un período de la señal, pero con mayor nivel de detalle.

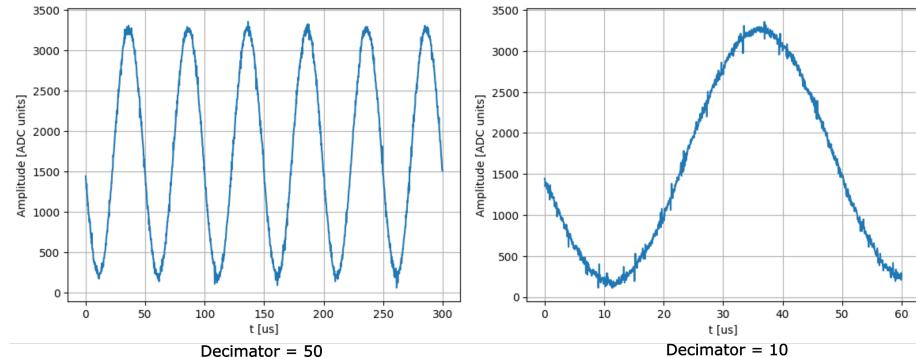


Figure 36: Señal adquiAdquisición de una señal sinusoidal con un período de $50\mu s$ y una amplitud de 50 mV. A la izquierda se observa la señal adquirida con un valor de decimación de 50, donde se capturan varios períodos completos. A la derecha, se muestra la señal con un valor de decimación de 10, permitiendo una mayor resolución temporal en un único período.

Para validar la capacidad del sistema en la adquisición de señales pulsadas, típicas en la detección de partículas y radiación, se generó con el AD2 una señal de pulsos exponenciales. A esta señal, con una frecuencia de 20 kHz, se le

añadió un 10% de ruido sobre su amplitud, simulando así condiciones realistas de medición en entornos experimentales.

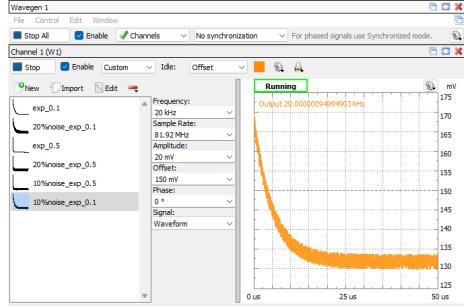


Figure 37: Señal pulsada de tipo exponencial decreciente con ruido añadido de 10% generada con el AD2.

En la figura 38 se observa la adquisición de la señal pulsada utilizando el DAQ implementado. De manera similar al caso de la señal sinusoidal, se verifica que el decimador controla eficazmente la ventana temporal de adquisición al ajustar la tasa de muestreo del sistema. En la gráfica de la izquierda, un valor de decimación mayor permite visualizar un mayor número de pulsos, mientras que en la gráfica de la derecha, un valor de decimación menor permite apreciar con mayor detalle la forma de la señal.

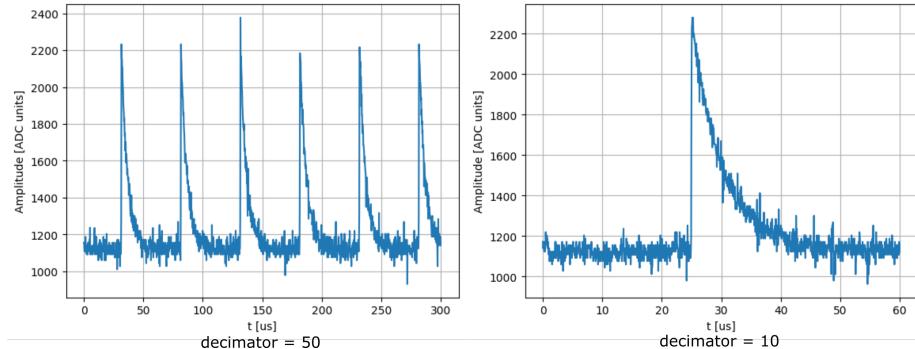


Figure 38: Señal pulsada exponencial adquirida utilizando el sistema DAQ implementado, con valores de decimación de 50 (izquierda) y 10 (derecha). En la gráfica de la izquierda se visualizan múltiples pulsos debido a la mayor ventana temporal de adquisición, mientras que en la derecha, con un valor de decimación menor, se observa con mayor detalle la forma de un solo pulso.

A partir de parámetros típicos de pulsos provenientes de detectores de radiación, se genera un conjunto de datos en Python (ver fig. 39) que simula un tren de pulsos exponenciales decrecientes. Estos pulsos presentan el fenómeno

de pile-up, lo cual resulta útil para el ajuste de los filtros implementados y para la validación de ciertas funcionalidades del sistema de adquisición de datos (DAQ).

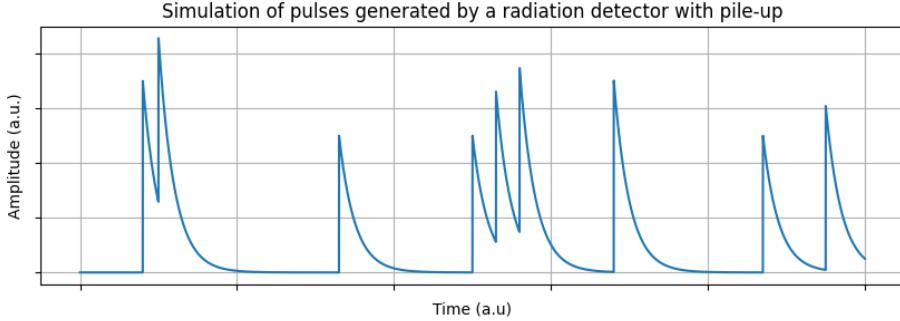


Figure 39: Simulación de pulsos sintéticos en Python que representan la respuesta típica de un detector de radiación. Se aprecia el fenómeno de pile-up. Esta señal se exporta como archivo de datos que puede utilizar el AD2 para generar una señal real con su forma.

El conjunto de datos es exportado a un archivo .CSV, el cual puede ser leído por el software AD2 WaveForms para generar una señal que modula sus características, como se observa en la parte izquierda de la figura 40. A su vez, en la parte derecha se muestra la gráfica de los datos adquiridos por el DAQ con el parámetro de decimación ajustado para observar una ventana temporal del orden de la izquierda.

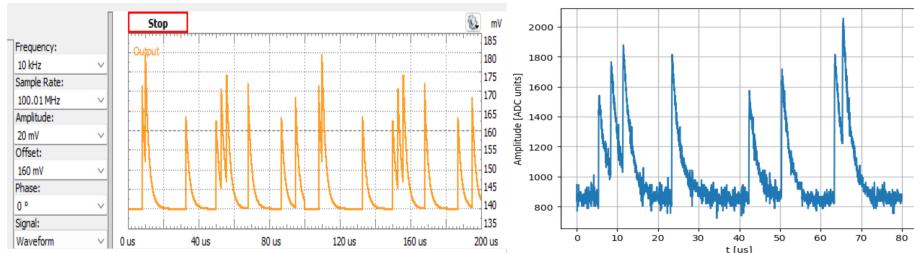


Figure 40: Señal tren pulsos exponencial adquirida

El diseño del filtro de media móvil comienza en Python, donde se exploran diferentes algoritmos basados en el filtro FIR descrito en la sección de procesamiento digital del marco teórico. Durante este análisis, se identifican variaciones en la eficiencia de los algoritmos en términos de procesamiento, tomando como métrica clave la capacidad para atenuar las componentes de alta frecuencia, como el ruido electrónico, utilizando la menor cantidad posible de muestras N . Se concluye que el algoritmo más eficiente es aquel que emplea un acumulador

de muestras y realiza la división por un valor de N que es potencia de 2. Esta elección proporciona una ventaja significativa, ya que simplifica las operaciones en el diseño posterior del filtro en VHDL.

En la figura 41, se muestra la aplicación de este filtro, diseñado en Python, a una señal sinusoidal con un ruido añadido equivalente al 30% de su amplitud, simulando la respuesta de un sensor en condiciones de alto ruido electrónico. Se observa claramente la recuperación de la señal original al eliminar de manera efectiva las componentes de alta frecuencia.

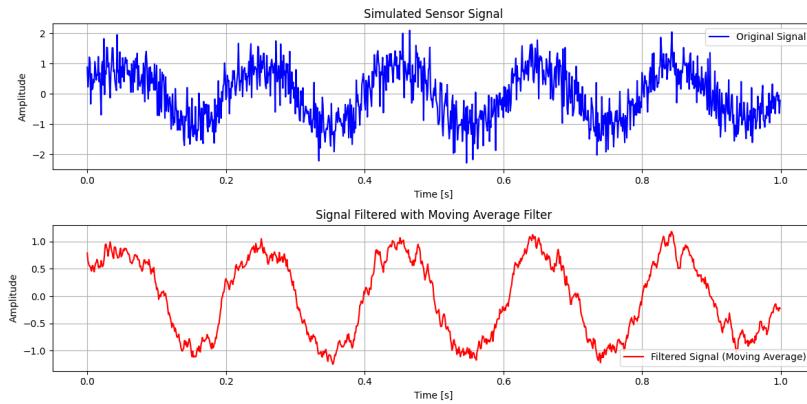


Figure 41: Resultado de diseño de media móvil implementado en Python para probar el concepto

Tras la configuración del filtro diseñado en VHDL en el PL, se genera con el AD2 una señal sinusoidal de 10 kHz con un ruido añadido equivalente al 20% de la amplitud de la señal. Esta señal se ingresa al DAQ, y desde la interfaz de usuario en Python se selecciona el canal de media móvil en el multiplexor para observar el efecto del filtro sobre la señal real, como se muestra en la figura 42.

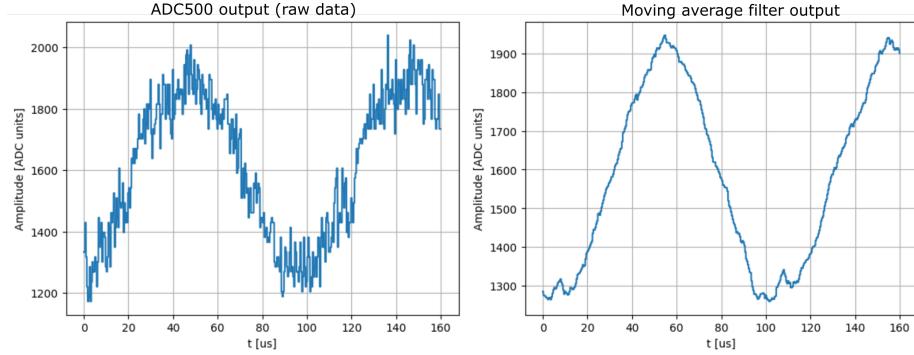


Figure 42: A la izquierda se muestra la gráfica de la señal sinusoidal de 10 kHz con un 20% de ruido añadido sin filtrar, mientras que a la derecha se observa la salida del filtro de media móvil. El efecto del filtro es evidente al reducir las componentes de alta frecuencia y mejorar la representación de la señal original.

De manera similar, se realiza la prueba con una señal sinusoidal de 50 kHz, a la que se añade un 10% de ruido en su amplitud. Se observa la salida del filtro de media móvil, evidenciando su capacidad para atenuar el ruido y recuperar la señal original.

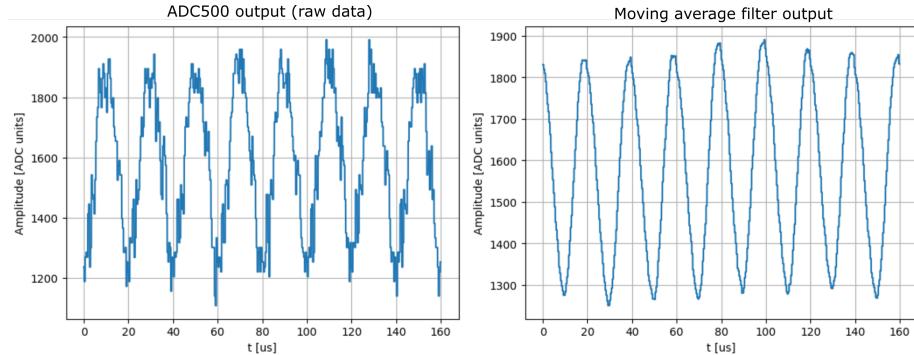


Figure 43: A la izquierda se muestra la gráfica de la señal sinusoidal de 50 kHz con un 10% de ruido añadido, mientras que a la derecha se observa la salida del filtro de media móvil. El filtro atenúa eficazmente las componentes de alta frecuencia, permitiendo una mejor recuperación de la señal original.

Se introduce en el filtro una señal de tipo pulso exponencial decreciente con un 10% de ruido añadido a su amplitud, con el objetivo de evaluar la respuesta del filtro ante señales pulsadas, simulando condiciones reales de filtrado de señales provenientes de detectores. Como se observa en la figura 44, el filtro elimina de manera efectiva gran parte de las componentes de ruido electrónico, permitiendo revelar la forma del pulso y sus características físicas de interés.

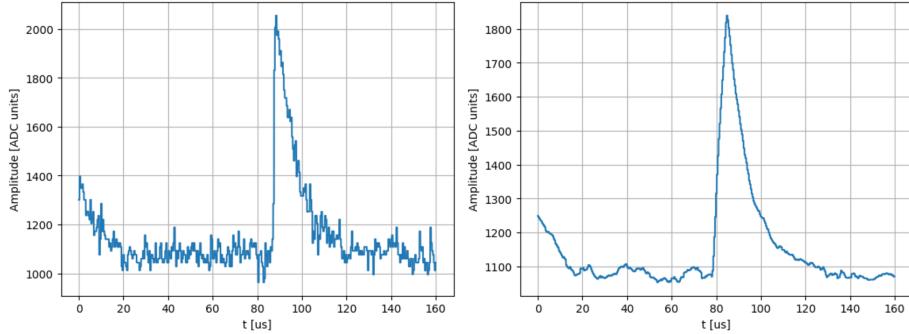


Figure 44: Señal pulsada exponencial con ruido filtrada por MA 10KHz con 10 porciento de ruido sumado

Se repite el mismo procedimiento, aumentando la frecuencia a 100 kHz y la componente de ruido añadido al 20%, con el fin de poner a prueba el sistema ante una mayor frecuencia y la difusión de la señal pulsada. Sin embargo, se observa que el filtro aún puede mejorar la relación señal-ruido, entregando pulsos que pueden ser utilizados en etapas posteriores.

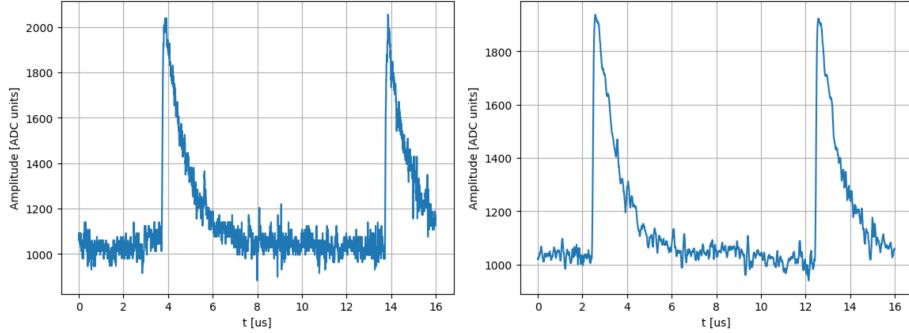


Figure 45: A la izquierda se muestra la gráfica de la señal pulsada de 100 kHz con un 20% de ruido añadido, mientras que a la derecha se observa la salida del filtro de media móvil. A pesar de la mayor frecuencia y el aumento del ruido, el filtro mejora la relación señal-ruido, permitiendo que los pulsos sean adecuados para etapas posteriores.

Haciendo uso de un tren de pulsos exponenciales que simulan un conjunto de eventos de un detector y presentan pile-up (fig. 40), se añade en WaveForms una componente de ruido del 10% de su amplitud para observar el comportamiento del filtro de media móvil ante este tipo de señales. Se encuentra que, al igual que en los casos anteriores, el filtro tiene la capacidad de eliminar componentes de alta frecuencia, permitiendo apreciar mejor la forma y los detalles de la señal.

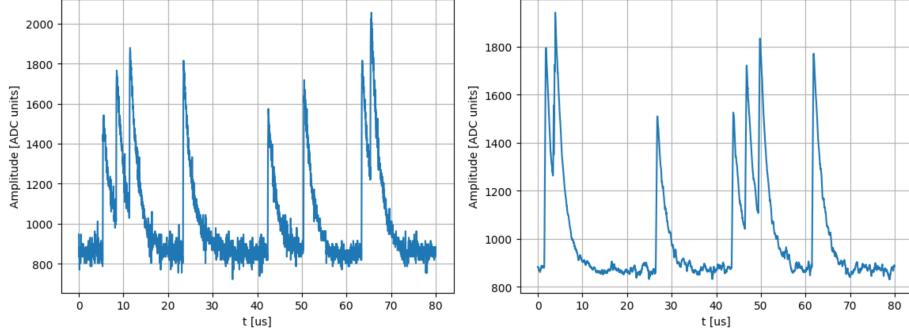


Figure 46: A la izquierda se muestra el tren de pulsos exponenciales simulado con pile-up y un 10% de ruido añadido generados a 10kHz. A la derecha se observa la salida del filtro de media móvil. El filtro demuestra su capacidad para eliminar componentes de alta frecuencia, lo que permite una mejor visualización de la forma y los detalles de la señal.

Como se expuso en la sección anterior, donde se desarrolla la metodología de diseño del filtro trapezoidal, inicialmente se aplica el filtro en Python para calcular los coeficientes K , L y M que generan la mejor respuesta a la señal. El resultado inicial para un tren de pulsos sintetizados, que refleja las características de una señal generada por un detector, se observa en la figura 47.

Se puede apreciar el efecto que tiene el filtro shaper trapezoidal sobre los pulsos exponenciales. La primera observación es que el filtro genera un pulso trapezoidal, similar a un pulso cuadrado, proporcional a la altura del pulso respecto a la línea base. Además, se evidencia su capacidad para realizar *pile-up rejection*, ya que separa pulsos exponenciales adyacentes y los coloca sobre la línea base, revelando la amplitud real de cada uno.

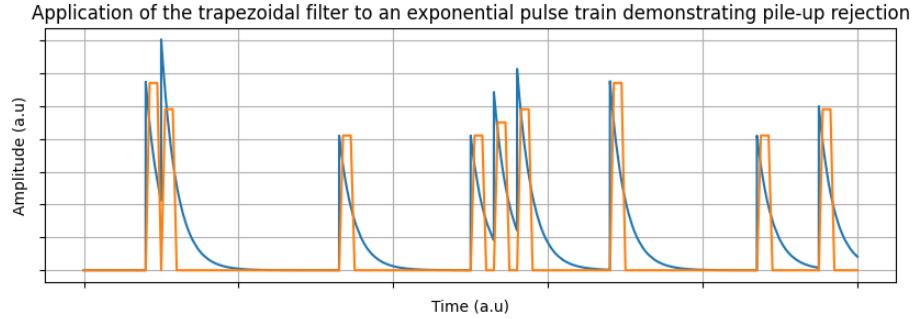


Figure 47: Aplicación del filtro trapezoidal diseñado en Python a los pulsos anteriores usando Python

No obstante, para obtener los coeficientes adecuados que se configuran en el filtro

que se instanciará en el PL para procesamiento continuo, es necesario realizar la calibración con la señal real. Por lo tanto, se ingresa al DAQ la señal de un tren de pulsos exponenciales con ruido, generada con el AD2, y se hace pasar por el filtro de media móvil para eliminar las componentes de alta frecuencia o ruido. Una vez adquirida esta señal con el UDMA en Python, se aplica la función del filtro trapezoidal creada anteriormente, variando los parámetros K , L y M hasta observar una respuesta adecuada del shaper trapezoidal. En particular, se encuentran los valores $K = 57$, $L = 5$ y $M = 100$, con los cuales se observa una buena forma de los pulsos trapezoidales, un adecuado *pile-up rejection* y la ubicación de los pulsos sobre la misma línea base. En la figura 48 se observan tres gráficas: en la primera, se muestra la señal adquirida por el DAQ después del módulo de decimación; en la segunda, se observa la salida del módulo de media móvil que se ingresa al módulo de shaper trapezoidal; y en la tercera, se presenta la respuesta del filtro trapezoidal en Python, optimizado con los parámetros mencionados.

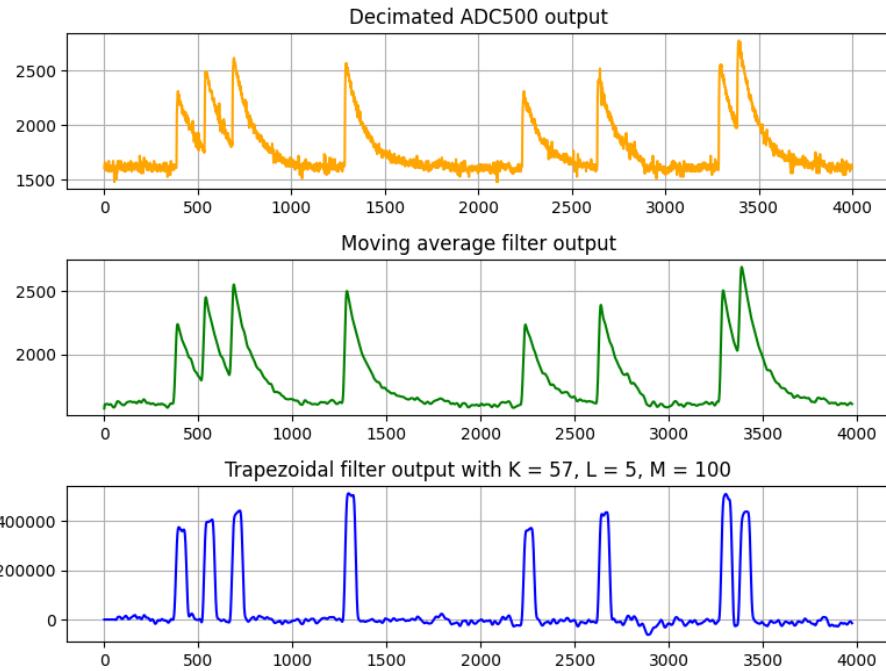


Figure 48: Se muestran tres gráficas relacionadas con el procesamiento de la señal. En la primera gráfica, se presenta la señal adquirida por el DAQ tras el módulo de decimación. La segunda gráfica muestra la salida del módulo de media móvil que se ingresa al módulo de shaper trapezoidal. Finalmente, la tercera gráfica representa la respuesta del filtro trapezoidal en Python, optimizado con los parámetros de forma $K = 57$, $L = 5$, $M = 100$.

Luego de la implementación del módulo de filtro shaper trapezoidal en VHDL, se configuran los parámetros optimizados de forma y se realiza la interconexión con el resto del sistema. A continuación, se lleva a cabo una prueba de funcionamiento utilizando la herramienta Vivado ILA (Integrated Logic Analyzer), conectando pruebas a la señal decimada de salida del ADC500, a la salida del módulo de media móvil y a la salida del filtro trapezoidal. Tras la síntesis del diseño, se obtienen los resultados mostrados en la figura 49, donde se observan las tres señales y se confirma el funcionamiento del filtro trapezoidal a nivel de la FPGA utilizando el ILA.

La señal de salida del filtro trapezoidal, que se muestra en color rojo en la parte inferior de la interfaz de visualización del ILA, al compararla con la señal en color amarillo, que corresponde a la salida del módulo de media móvil, permite observar un resultado similar al presentado en la figura 48. En esta comparación, los pulsos exponenciales son convertidos en trapecios simétricos, ascienden desde la misma línea base y tienen la capacidad de realizar **pile-up rejection**, diferenciando los pulsos que originalmente estaban apilados.

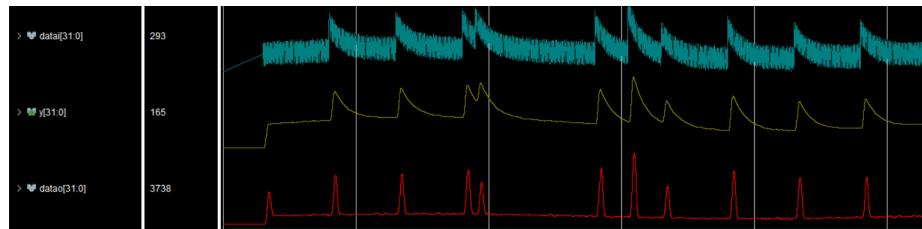


Figure 49: Resultado de la prueba de funcionamiento del módulo de filtro shaper trapezoidal implementado en VHDL. Se muestran las tres señales: la señal decimada de salida del ADC500, la salida del módulo de media móvil y la salida del filtro trapezoidal. Este análisis confirma el correcto funcionamiento del filtro trapezoidal a nivel de la FPGA, utilizando la herramienta Vivado ILA (Integrated Logic Analyzer).

Al realizar la adquisición de la salida del módulo trapezoidal desde la interfaz de usuario en Python a través del UDMA, mientras se ingresa el tren de pulsos de prueba con el AD2, se obtiene el resultado mostrado en la figura 50. De manera similar al resultado anterior, se observa que el filtro convierte los pulsos exponenciales en trapecios, cuya amplitud es proporcional al pulso original respecto a la misma línea base. Adicionalmente, se aprecia el efecto de **pile-up rejection**.

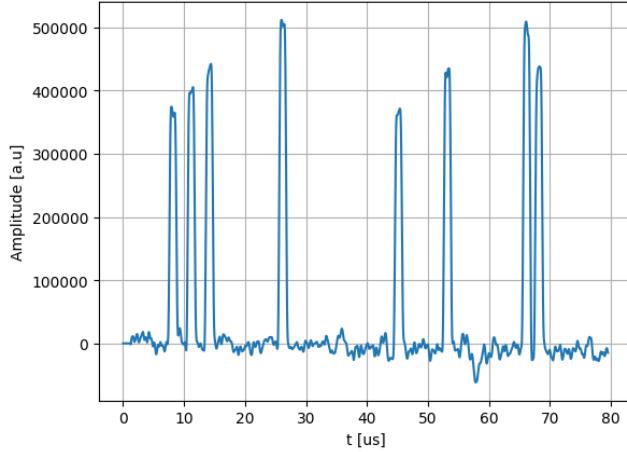


Figure 50: Resultado de la adquisición de la salida del módulo trapezoidal a través de la interfaz de usuario en Python, utilizando el UDMA. Se observa la conversión de los pulsos exponenciales en trapecios, con amplitud proporcional al pulso original respecto a la misma línea base. Además, se evidencia el efecto de pile-up rejection, que permite diferenciar los pulsos originalmente apilados.

4 Análisis y conclusiones

Así como se diseñaron e implementaron esos filtros, se puede montar cualquiera en función de los requerimientos de la aplicación.

References

- [1] J. G. Webster and H. Eren, *Measurement, Instrumentation, and Sensors Handbook: Two-Volume Set*. CRC press, 2018.
- [2] I. Sinclair, *Sensors and transducers*. Elsevier, 2000.
- [3] G. F. Knoll, *Radiation detection and measurement*. John Wiley & Sons, 2010.
- [4] H. Kolanoski and N. Wermes, *Particle Detectors: Fundamentals and Applications*. Oxford University Press, USA, 2020.
- [5] S. D. Brown and Z. G. Vranesic, *Fundamentals of digital logic with VHDL design*, vol. 70125910. McGraw-Hill New York, 2000.
- [6] W. R. Leo, *Techniques for nuclear and particle physics experiments: a how-to approach*. Springer Science & Business Media, 1994.

- [7] X. Luo, V. Modamio, J. Nyberg, J. Valiente-Dobon, Q. Nishada, G. De Angelis, J. Agramunt, F. Egea, M. N. Erduran, S. Ertürk, *et al.*, “Pulse pile-up identification and reconstruction for liquid scintillator based neutron detectors,” *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 897, pp. 59–65, 2018.
- [8] V. Radeka, “Optimum signal-processing for pulse-amplitude spectrometry in the presence of high-rate effects and noise,” *IEEE Transactions on Nuclear Science*, vol. 15, no. 3, pp. 455–470, 1968.
- [9] U. Meyer-Baese and U. Meyer-Baese, *Digital signal processing with field programmable gate arrays*, vol. 65. Springer, 2007.
- [10] J. G. Proakis, *Digital signal processing: principles, algorithms, and applications*, 4/E. Pearson Education India, 2007.
- [11] V. T. Jordanov, G. F. Knoll, A. C. Huber, and J. A. Pantazis, “Digital techniques for real-time pulse shaping in radiation measurements,” *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 353, no. 1-3, pp. 261–264, 1994.
- [12] L. F. Ramirez and S. Montoya, *low cost PHA for Mössbauer Spectroscopy*. Elsevier, 2024.
- [13] I. Bravo-Muñoz, A. Gardel-Vicente, and J. L. Lázaro-Galilea, “New applications and architectures based on fpga/soc,” 2020.
- [14] E. Rucci, *Evaluación de rendimiento y eficiencia energética de sistemas heterogéneos para bioinformática*. 2018.
- [15] AMD, “Zynq-7000 soc.” <https://www.amd.com/en/products/adaptive-socs-and-fpgas/soc/zynq-7000.html>, 2024. Último acceso: 13 de agosto de 2024.
- [16] R. Zurawski, *Industrial communication technology handbook*. CRC press, 2014.
- [17] G. Bortolotti, A. Cavalli, L. Chiarelli, A. Chierici, S. Dal Pra, L. Dell’Agnello, D. De Girolamo, M. Donatelli, A. Ferraro, D. Gregori, *et al.*, “The infn tier-1,” in *Journal of Physics: Conference Series*, p. 042016, IOP Publishing, 2012.
- [18] EEE Guide, “Serial communication interface 8251,” 2024. Accessed: 2024-08-20.
- [19] Texas Instruments, “DC08500 High Performance, Low Power 8-Bit 500 MSPS A/D Converter.” <https://www.ti.com/lit/ds/symlink/adc08500.pdf>, 2024. Último acceso: 07 de agosto de 2024.

- [20] M. L. Crespo, F. Foulon, A. Cicuttin, M. Bogovac, C. Onime, C. Sisterna, R. Melo, W. Florian Samayoa, L. G. García Ordóñez, R. Molina, *et al.*, “Remote laboratory for e-learning of systems on chip and their applications to nuclear and scientific instrumentation,” *Electronics*, vol. 10, no. 18, p. 2191, 2021.
- [21] ICTP Multidisciplinary Laboratory, “Smr3765 repository.” <https://gitlab.com/ictp-mlab/smr3765/-/wikis/uploads/d6f86235c15669356a9567b2aa0363c4/image.png>, 2022. Último acceso: 13 de agosto de 2024.
- [22] ICTP Multidisciplinary Laboratory, “Adc500 repository.” <https://gitlab.com/ictp-mlab/adc500>, 2022. Último acceso: 22 de agosto de 2024.
- [23] X. Inc., “Vivado design suite.” Disponible en: <https://www.xilinx.com/products/design-tools/vivado.html>, 2023. Versión 2023.1.
- [24] Avnet, “Zedboard: Zynq-7000 arm/fpga soc development board.” <https://www.zedboard.org/product/zedboard>, 2012. Recuperado el 07 de agosto de 2024.
- [25] X. Inc., “Vitis unified software platform,” 2024. Accedido: 22-ago-2024.
- [26] R. Melo, “Core comblock.” <https://gitlab.com/rodrigomelo9/core-comblock>. Accessed: 2024-08-22.
- [27] Xilinx, “Vivado design suite user guide: Designing with ip.” https://support.xilinx.com/s/article/1053914?language=en_US. Accessed: 2024-08-22.
- [28] I. M-Lab, “Udma.” <https://gitlab.com/ictp-mlab/udma>, 2021. Accessed: 2024-08-24.