

Data Acquisition System

La medición es un pilar fundamental de la investigación científica; no obstante, la cuantificación de una magnitud física solo es posible mediante la interacción con el sistema y la extracción de la información relevante. En este sentido, el diseño e implementación de tecnologías capaces de llevar a cabo esta tarea constituye una parte esencial del campo de la instrumentación científica. Partiendo de estos conceptos, este capítulo se orienta a explorar los métodos que permiten convertir fenómenos físicos en señales que transporten información, así como las tecnologías que posibilitan la adquisición de dichas señales y su representación en un formato útil para el observador o usuario [1]. Este conjunto de procesos, diseñado para extraer información de un sistema físico, se conoce como sistema de adquisición de datos e incluye varias etapas que se ilustran en la figura 1 y se describen a continuación.

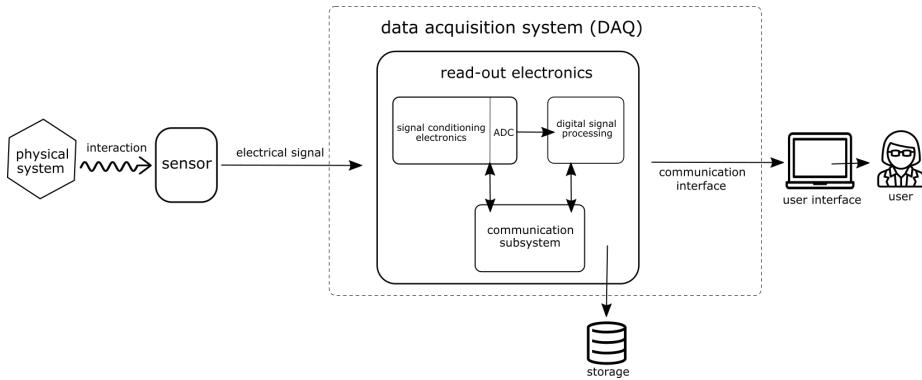


Figure 1: Diagrama de los principales componentes de un sistema de adquisición de datos genérico

En primer lugar, es fundamental identificar un sistema de interés y las variables que se desean medir. Cualquier objeto físico puede ser estudiado siempre que se cuente con el marco teórico y los métodos experimentales adecuados para extraer sistemáticamente sus medidas. Por ejemplo, si se considera el universo observable, es posible medir su tasa de expansión mediante argumentos cos-

mológicos y un sistema de telescopios que observe cambios en las características de la luz proveniente de diversas galaxias, lo que evidencia la expansión del universo [2]. Por otro lado, un tumor cerebral puede evaluarse al analizar los cambios en los espines nucleares de los átomos de hidrógeno en los tejidos cerebrales al aplicar un campo magnético potente para alinear estos espines, y luego pulsos de radiofrecuencia para perturbar esta alineación. La respuesta de los espines al campo magnético y a los pulsos de radiofrecuencia se detecta y se utiliza para crear imágenes detalladas de los tejidos cerebrales [3]. Aunque estos ejemplos pueden parecer desconectados, comparten un elemento clave: son sistemas físicos cuyos atributos pueden medirse mediante la interacción con un medio capaz de extraer información.

El medio mencionado es diseñado sistemáticamente para interactuar con el objeto de interés. Estos medios son comúnmente dispositivos llamados sensores, un tipo de transductor capaz de convertir estímulos físicos en señales eléctricas. Como se ha mencionado, existe una amplia variedad de sensores diseñados para medir todo tipo de magnitudes físicas, los cuales explotan distintos fenómenos en función de los requisitos de la medición [1]. Por ejemplo, para medir la temperatura de un objeto, se pueden utilizar aleaciones metálicas que cambian su resistividad eléctrica ante variaciones de temperatura [4]. Sin embargo, una cámara fotosensible al espectro infrarrojo también podría cumplir la misma función [5]. Estos sistemas se diferencian en la metodología con la que la información de los cambios de temperatura es transferida al usuario. En el primer caso, se requiere la lectura de la resistividad eléctrica y el conocimiento de su tasa de cambio en función de la temperatura, mientras que en el segundo caso, se emplea un sistema óptico equipado con la tecnología necesaria para convertir la radiación electromagnética en señales eléctricas. Finalmente estas señales pueden ser manipuladas para calcular la temperatura del objeto original utilizando la ley de Planck [6].

Para lograr la extracción y transferencia efectiva de información hacia el observador, esta debe ser presentada en un formato que permita su codificación, transporte y almacenamiento sin pérdidas. En este contexto, las señales eléctricas, y por ende los dispositivos electrónicos, desempeñan un papel crucial en los sistemas modernos de adquisición de datos. Estos sistemas integran tanto componentes analógicos como digitales en sus diferentes etapas, tal como se ilustra en la figura 1.

En general, las señales provenientes del sensor o detector deben ser tratadas y acondicionadas según las características que se deseen resaltar y preservar. Por ejemplo, es común que factores externos e internos de los sistemas electrónicos introduzcan señales no deseadas, conocidas como ruido, en la señal de interés, lo que hace necesario el uso de filtros de frecuencia o amplitud. Además, puede ocurrir que la señal generada por el sensor sea muy débil y las etapas subsiguientes no tengan la resolución adecuada para procesarla, por lo que se deben implementar amplificadores de distintos tipos. En otros casos, es necesario

transformar el tipo de señal que transporta la información, por ejemplo, de carga a voltaje o de voltaje a corriente. Estas tareas suelen ser realizadas por circuitos de dispositivos analógicos, conocidos como etapas de acondicionamiento de señales, que algunos autores denominan analog front-end [7].

Sin embargo, ante grandes volúmenes de datos, se requieren sistemas de cómputo con la capacidad de manejar señales con alta precisión, almacenar datos sin degradación y ejecutar operaciones complejas a gran velocidad, predominando los circuitos digitales como la tecnología principal en este ámbito. Como resultado, las etapas de procesamiento en los sistemas de adquisición de datos (DAQ) modernos están generalmente basadas en esta tecnología. Estos sistemas utilizan señales discretas, que representan información en forma de bits (0s y 1s), en contraste con las señales analógicas continuas. Como se mencionó anteriormente, estos sistemas aplican sobre las señales diversas operaciones, que pueden incluir clasificación, discriminación y la ejecución de operaciones matemáticas. Entre los principales exponentes de la electrónica digital se encuentran plataformas como procesadores o CPUs, FPGAs, ASICs, GPUs, DSPs, SoCs y computadoras cuánticas. Cada una de estas plataformas posee características únicas que las hacen especialmente adecuadas para diversas aplicaciones en el procesamiento de datos [8].

Una vez que los datos han sido adquiridos y procesados digitalmente, el siguiente paso crucial en un sistema de adquisición de datos es la transmisión de esta información hacia un computador personal o una estación de control, donde serán analizados y presentados al usuario. La interfaz de comunicación es la etapa que conecta el sistema de adquisición de datos con el dispositivo final, facilitando la transferencia de datos de manera eficiente y confiable. Existen diversas tecnologías de comunicación, cada una con sus ventajas y limitaciones, que pueden ser seleccionadas según los requisitos específicos de la aplicación. Entre las interfaces más comunes en instrumentación científica se encuentran GPIB (General Purpose Interface Bus), USB (Universal Serial Bus), Ethernet, y protocolos inalámbricos, cada uno adecuado para diferentes volúmenes de datos, velocidades de transmisión, y distancias. La elección de la interfaz adecuada no solo impacta el rendimiento del sistema, sino también la facilidad de integración con el software de análisis y visualización utilizado por el usuario final [9].

Finalmente, mediante el uso de programas informáticos diseñados específicamente para decodificar los datos provenientes de la interfaz de comunicación, las cantidades físicas de interés pueden presentarse en formatos comprensibles y útiles para el usuario. Por ejemplo, un osciloscopio que transmite datos a una computadora de escritorio a través de una interfaz GPIB-USB cuenta con una interfaz gráfica de usuario que permite visualizar la señal en el dominio del tiempo, destacando características como amplitud, frecuencia, y valor de la línea base. De manera similar, un sistema de espectroscopía nuclear puede enviar datos a través de Ethernet a un servidor, donde estos se almacenan en una base de datos accesible para usuarios de la red [10]. Con aplicaciones web, es posible

generar histogramas de energía frente a la altura, visualizando así los espectros de energía de una fuente de radiación. En última instancia, el objetivo de cualquier sistema de adquisición de datos, independientemente de su escala o infraestructura, es proporcionar al usuario información física relevante para su análisis e interpretación.

A continuación, se presentarán algunos conceptos clave sobre los sistemas de Adquisición de Datos (DAQ) con el propósito de profundizar en las características de sus etapas generales y establecer una base sólida para abordar el DAQ desarrollado en este trabajo. Posteriormente, se describirá el montaje experimental y la metodología empleada para su implementación.

0.1 Marco Teórico

0.1.1 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 correspondiente, generalmente 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) [11].

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 [12]. 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 voltímenes, 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 [13].

0.1.2 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 [11]. En la figura 2, se ilustra una señal en el dominio del tiempo proveniente de un sensor de aceleración.

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



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 ejemplo genérico de un pulso individual, utilizado para ilustrar algunas de sus características principales.

Según [12], 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.



Figure 3: Pulso genérico para ilustrar las principales características. Reproducido a partir de [13]

- 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 [14]. En la figura 4 se puede observar un ejemplo de señal digital para ilustrar.

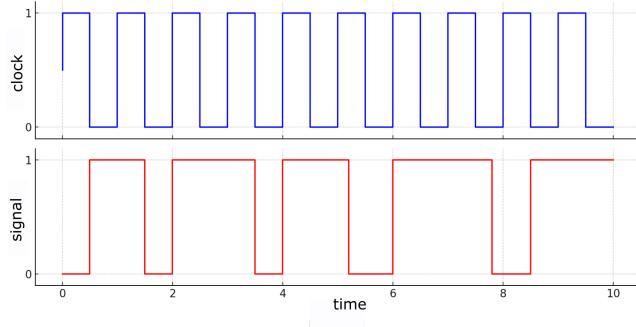


Figure 4: Ejemplo de diagrama de tiempo de una señal digital. En la parte superior se muestra una señal típica de reloj y en la parte inferior una señal genérica con información codificada en su estado (amplitud) o en su duración (ancho).

0.1.3 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.

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

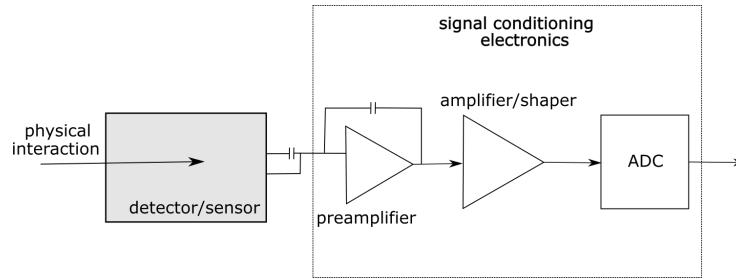


Figure 5: Un esquema de una etapa de electrónica de acondicionamiento típico, utilizado a menudo para la lectura de un detector, que incluye amplificación, conformación de impulsos y digitalización (representada aquí por un ADC).

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 señal_{entrada}/señal_{salida} no varía con el tiempo [13]. 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 [13]:

- 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 [12], La función principal del preamplificador es captar la señal del detector sin deteriorar notablemente la 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 $V_{\text{out}}/V_{\text{in}}$ se mantenga constante, lo cual es esencial para la precisión en la medición [15].

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: Ilustración del comportamiento de un preamplificador de carga. Este tipo de dispositivos son comúnmente utilizados para acondicionar señales pulsadas provenientes de detectores de partículas [13].

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 [13]:

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

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. Generalmente, esta etapa 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) y disminuir el ruido mediante un filtrado de frecuencia eficiente [15].

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 [13]. La figura 7 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

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

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.



Figure 7: 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 paso de banda sirve como un buffer, impidiendo que el filtro de paso bajo aplique una carga significativa al filtro de paso alto. Adaptado de [13].

mediante electrónica analógica, pueden implementarse de manera más eficiente en el dominio digital utilizando filtros con parámetros ajustables en plataformas configurables como las FPGA [16]. Este enfoque se explorará en mayor detalle en la sección 0.1.4. 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.

Tratamiento y acondicionamiento de Señales: Digitalización

De acuerdo con [17], 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 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 8.



Figure 8: Representación del principio de funcionamiento de un ADC.

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 9 se muestra la simulación de una señal senoidal proveniente de un sensor, digitalizada mediante un ADC con estas características.

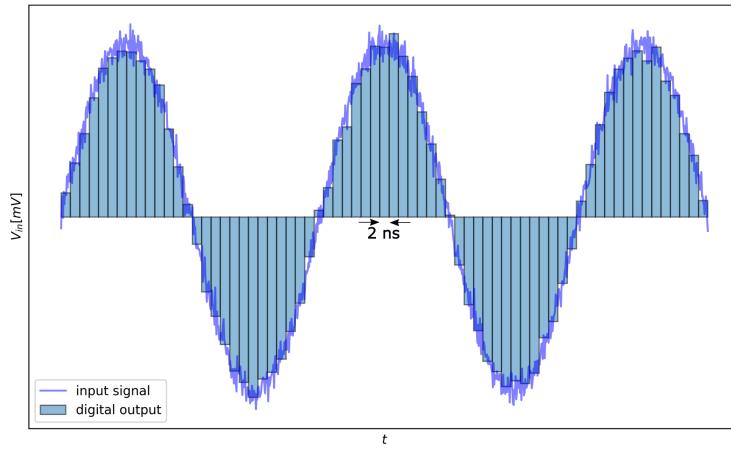


Figure 9: Ilustración de la digitalización a 500 MSPS de una señal proveniente de un sensor.

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 [13].

0.1.4 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 [8]. En la figura 10 se ilustra un esquema genérico de la etapa de procesamiento digital de señales en un DAQ.

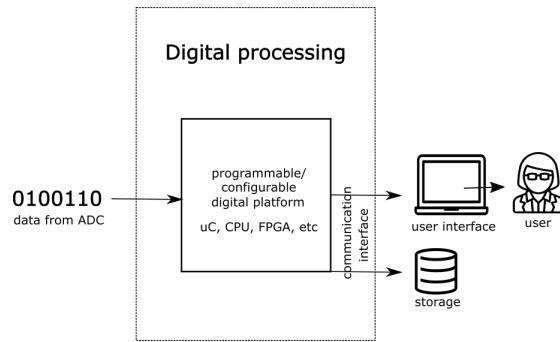


Figure 10: Ilustración de la etapa de procesamiento digital. A la izquierda, los datos provenientes del ADC ingresan al sistema de procesamiento. A continuación, son operados de acuerdo a los algoritmos implementados en el sistema embebido, seguido de la información procesada que se dirige hacia las etapas posteriores de la cadena de información.

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 [17]:

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

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 [8]:

- 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 [8], 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. En la figura 11 se observa la aplicación del filtro a una señal con ruido.

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 para el funcionamiento de los osciloscopios. En el contexto de la instrumentación científica para espectroscopía nuclear, se encuentran los analizadores multicanal (MCA), sistemas digitales que clasifican los pulsos provenientes de un detector de radiación según su amplitud, almacenándolos en memoria para generar un histograma que permita extraer el espectro de energía de la fuente radiactiva [12].

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

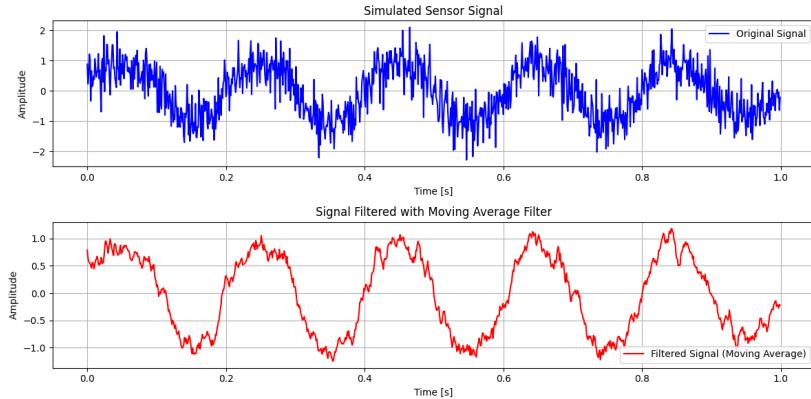


Figure 11: Ejemplo de la aplicación de un filtro de media móvil simple con ventana de tamaño 10 a una señal simulada con ruido. En este caso, el filtro actúa como un filtro pasa bajo, ya que elimina las componentes de alta frecuencia (ruido) y suaviza la señal. El procesamiento digital se implementó utilizando un computador de escritorio (CPU) y la programación del algoritmo mediante el módulo SciPy de Python.

microcontrolador de bajo costo como el ESP32 puede emplearse para desarrollar un sistema de análisis de altura de pulsos con funcionalidades similares a las de un MCA, como el descrito por [18]. 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 parallelización 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.

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 microprocesadores 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 [14].

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 [19].

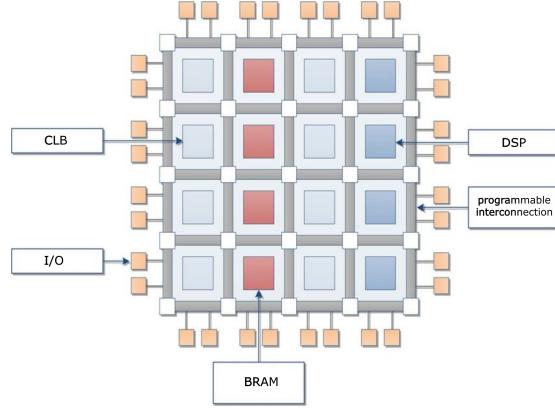


Figure 12: Diagrama de la arquitectura típica de una FPGA. Tomado de [20].

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 (Digital Signal Processors) [17].

Digital Processing: FPGA SoC

No obstante, como señala Bravo [19], los avances en tecnología FPGA han impulsado el desarrollo de sistemas híbridos, como los SoC (System on Chip). Según el fabricante AMD [21], 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.

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 [19]. 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.

0.1.5 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 [22] [23].

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 [24].

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 [24], 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. Después 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 [12]. 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 soft-

ware 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.

0.2 Montaje Experimental

En este trabajo se presenta la implementación de un sistema de adquisición de datos (DAQ) monocanal de altas prestaciones, que incluye una etapa de digitalización, un sistema de procesamiento digital basado en FPGA SoC y una interfaz de comunicación de alta velocidad. Esta arquitectura permite una adquisición efectiva de señales provenientes de diversos sistemas de sensado con su respectiva electrónica de acondicionamiento, cubriendo posibles aplicaciones que van desde el ámbito biomédico hasta la detección de partículas, entre otros. Tanto el hardware utilizado como la metodología de trabajo se fundamentan en los resultados obtenidos en la escuela conjunta ICTP-IAEA sobre aplicaciones de instrumentación científica y nuclear basadas en FPGA SoC [25].

0.2.1 Montaje Experimental: Digitalization Hardware - ADC500

En este trabajo, se utilizó la tarjeta ICTP-INFN ADC500 como plataforma de digitalización, en adelante referida como ADC500 (fig 14). Esta tarjeta está basada en un chip Texas Instruments ADC08500 [26] monocanal de alta velocidad, con una frecuencia de muestreo de hasta 500 MHz y 8 bits de resolución. Cumple con las especificaciones ANSI/VITA 57.1 y está equipada con un conector de tarjeta mezzanine FPGA (FMC) de bajo número de pines (LPC) [10].

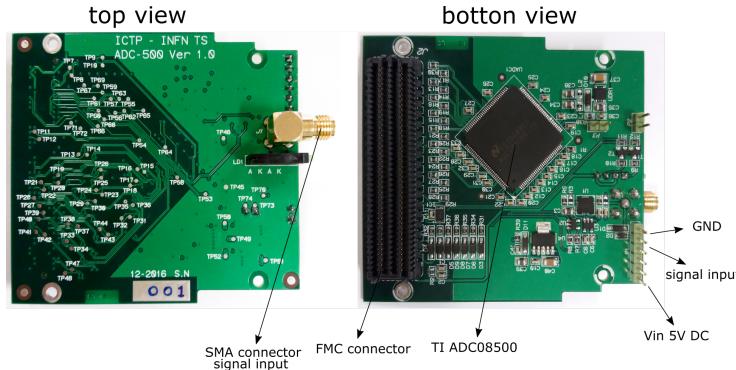


Figure 14: Plataforma de digitalización de señales ICTP-INFN ADC500. Se identifican sus principales partes: chip TI08500, conector FMC, conector SMA y pines de alimentación. Modificado de [25].

Dado que la tarjeta ADC500 está diseñada para integrarse con un sistema

de procesamiento basado en FPGA, es esencial desarrollar un controlador de hardware descrito en lenguaje HDL, que mapee correctamente las conexiones necesarias para gestionar las funciones del chip. Estas funciones incluyen varios modos de calibración y rangos de digitalización [26]. El controlador para el ADC500, desarrollado por el ICTP-MLAB, está disponible para su implementación con los derechos correspondientes según su licencia [27]. En la figura 15 se observa la representación en diagrama de bloques del software AMD Xilinx Vivado [28] del controlador mencionado.

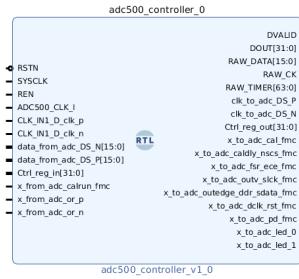


Figure 15: Representación en diagrama de bloques del controlador del ADC500 diseñado en VHDL [27].

0.2.2 Montaje Experimental: Digital Processing Platform based on FPGA SoC

Teniendo en cuenta lo expuesto en la sección 0.1.4 y a las características de alto rendimiento del ADC500 como sistema de digitalización, es necesario utilizar una plataforma de procesamiento digital con capacidad de paralelización de procesos, alta disponibilidad de recursos de computación, reconfigurabilidad y flexibilidad.

En este trabajo se utiliza la tarjeta de desarrollo Avnet Digilent Zedboard (figura 16) [29], basada en el SoC (System on Chip) Xilinx Zynq-7000 [21], 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.

Es importante destacar que, 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

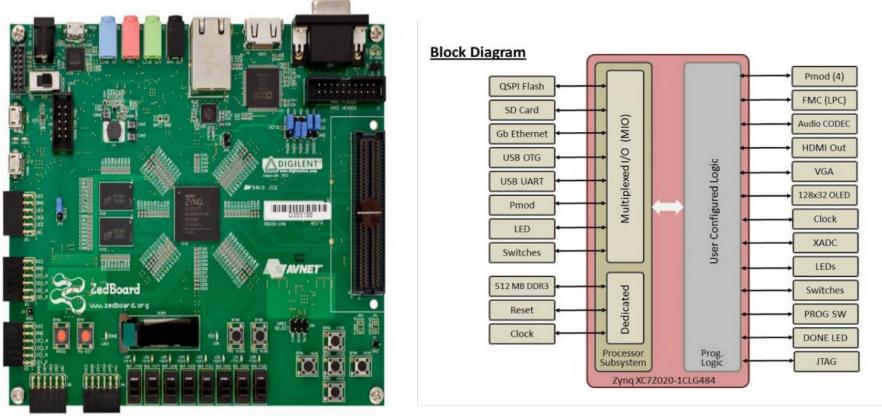


Figure 16: Fotografía de la tarjeta de desarrollo FPGA SoC Avnet Digilent Zedboard y descripción en diagrama de bloques de la distribución de los periféricos en el PS y PL [29].

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" [13]. 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.

Para llevar a cabo esta tarea, es esencial utilizar el software especializado proporcionado por el fabricante del chip. Dado que el desarrollo se basa en el SoC FPGA Xilinx Zynq-7000, se emplea el entorno de desarrollo unificado AMD Vitis [30]. Este entorno abarca tanto Vivado [28] para el diseño y configuración de la FPGA (PL), como Vitis SDK para la programación del procesador embebido (PS).

El flujo general de diseño para configurar la plataforma consta de dos etapas principales:

1. Diseño y Configuración en Vivado:

- Cargar la configuración del mapeo de pines y puertos de la tarjeta de desarrollo (en este caso, la Zedboard).
- Desarrollar el diseño del sistema digital en un lenguaje de descripción de hardware (HDL) como VHDL.

- Generar el archivo bitstream, que contiene toda la información necesaria para configurar la lógica en la FPGA.

2. Programación en Vitis:

- Cargar el archivo bitstream generado en Vivado como plataforma de hardware en Vitis.
- Programar el firmware del procesador embebido en un lenguaje de alto nivel (C o C++).
- Finalmente, cargar la configuración de la FPGA y el firmware del procesador en el hardware, permitiendo que el sistema integral ejecute las acciones previstas.

En este trabajo en particular, el software Vivado se utiliza para configurar el hardware de control del ADC, procesamiento de señales e interfaces de comunicación. Una de sus funcionalidades destacadas es la capacidad de convertir scripts de descripción de hardware en bloques con entradas y salidas, permitiendo una abstracción modular de la lógica implementada, similar al paradigma de la programación orientada a objetos, como se muestra en la figura 17 con un multiplexor.

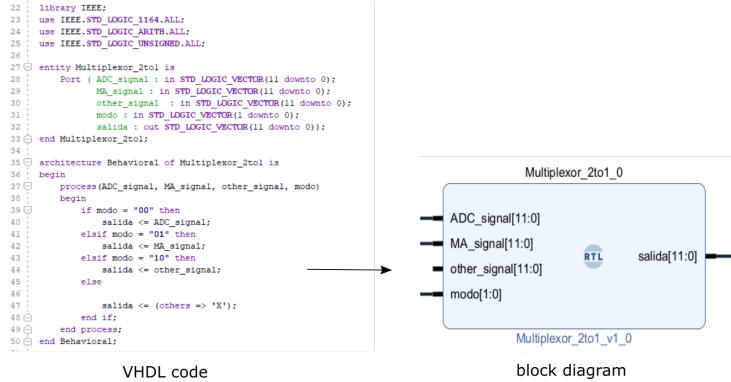


Figure 17: Ilustración de la función de abstracción de código HDL a bloques funcionales en Vivado.

De acuerdo con los requisitos de la aplicación, las entradas y salidas del sistema en el PL, se asocian a componentes de memoria, como registros, FIFOs o RAM, accesibles desde el procesador (PS) mediante protocolos de interconexión entre el PL y el PS (véase la figura 13). Desde el entorno de programación de alto nivel del procesador, en este caso Vitis SDK, es posible interactuar con el hardware sintetizado en la FPGA para configurar parámetros del sistema a través de métodos de escritura y adquirir datos desde la memoria mediante métodos de lectura. No obstante, las herramientas proporcionadas por el fabricante, como los controladores del bus del PS, pueden resultar complejas de implementar para

algunos usuarios debido a la cantidad de parámetros y el nivel de experticia requerido [31].

0.2.3 Montaje Experimental: Subsistema de Intercomunicación PS-PL: ComBlock

Para simplificar este proceso, la colaboración ICTP-INTI desarrolló el ComBlock [32], un IP portable y altamente configurable diseñado para ofrecer interfaces conocidas (como registros, RAM y FIFOs) a los usuarios de la Lógica Programable (PL), 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 [33]. Según sus creadores, el ComBlock cuenta con:

- Hasta 16 registros de entrada y/o salida, configurables de 1 a 32 bits.
- Una memoria RAM de doble puerto, que ofrece una interfaz Simple RAM. La inclusión, el ancho de datos, el ancho de la dirección y la profundidad de la memoria son configurables.
- Dos FIFOs asincrónicos, uno de PL a PS y otro de PS a PL, con indicaciones de vacío/lleno, casi vacío/lleno y condiciones de subdesbordamiento/desbordamiento. La inclusión individual, el ancho de datos y la profundidad de la memoria son configurables.

En términos simples, el ComBlock actúa como una capa de abstracción para los buses AXI. En la versión del paquete IP de Vivado, se utiliza una interfaz AXI Lite para manejar los registros y FIFOs, mientras que la RAM se gestiona mediante una interfaz AXI Full como se ilustra en la figura 18.

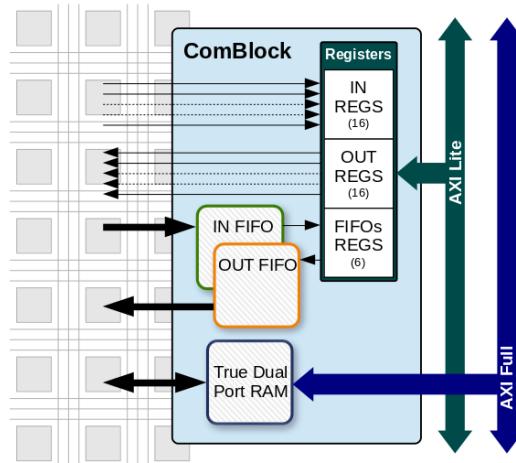


Figure 18: Ilustración de la arquitectura de Comblock. Tomado de [32]

Sin embargo, este proceso es completamente transparente para el usuario, quien no necesita lidiar con las complejas configuraciones de estos buses. Como se ilustra en la figura 19, el ComBlock IP ha sido diseñado de manera modular, lo que permite su incorporación directa en los diagramas de bloques de hardware digital en Vivado como un componente adicional. Esto facilita la interconexión entre el PS, el diseño específico en el PL, y los componentes de memoria, todo de forma intuitiva para el usuario.

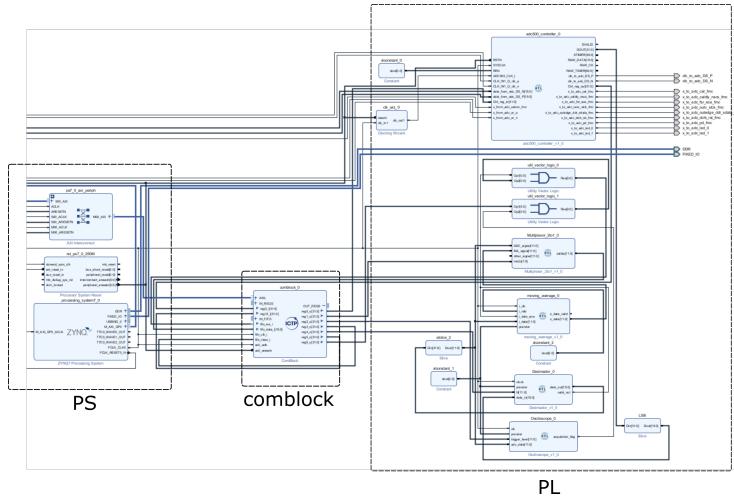


Figure 19: Diagrama de bloques de un diseño real en Vivado donde se identifican los módulos del PS, el diseño específico de PL y la intercomunicación mediante Comblock

Con un diseño de hardware digital configurado en el PL, una interfaz de intercomunicación con el PS y acceso a memoria mediante Comblock, es posible concentrarse en la programación del PS para gestionar las funciones del slow control del DAQ. Esta programación se realiza en Vitis SDK, un entorno de desarrollo integrado (IDE) basado en Eclipse [34], donde se desarrollan los scripts de firmware que controlan las funciones y periféricos del procesador. Además, el proyecto Comblock incluye un controlador en C que puede ser implementado en proyectos Baremetal o basados en FreeRTOS [35], todo ello desde el mismo entorno Vitis SDK.

Como se ha mencionado, es necesario un método de comunicación bidireccional entre el usuario y el PS, tanto para el acceso a la información adquirida y procesada por el DAQ como para el envío de comandos de configuración y control al sistema. Esta interfaz de comunicación debe cumplir con las características descritas en la sección 0.1.5, disponiendo la plataforma Zedboard de varias opciones, como UART USB y Ethernet, entre otras. Inicialmente, la interfaz gráfica de Vitis SDK ofrece una terminal de monitor serial embebida que utiliza la comunicación UART USB, permitiendo el acceso a los recursos de memoria

mediante métodos como "xprint". Sin embargo, este protocolo puede resultar limitado en velocidad para aplicaciones que requieren lectura de memoria, como FIFOs en modo de lectura continua (streaming) a altas velocidades, algo común en la adquisición de datos en sistemas de detección de partículas. Por tanto, es recomendable considerar el uso de la interfaz Ethernet TCP/IP, aunque esto implica desarrollar los controladores de software necesarios para la emisión y recepción de la información.

0.2.4 Montaje Experimental: Communication Interface - UDMA

Una herramienta desarrollada específicamente para solventar este problema es el UDMA (Universal Direct Memory Access). De acuerdo a sus creadores en el laboratorio multidisciplinario del ICTP (MLAB), el UDMA es un conjunto de herramientas de control remoto diseñado para conectar una PC con la lógica personalizada en un SoC-FPGA [10]. Este actúa como un puente entre un generador y un receptor de datos, permitiendo realizar operaciones de lectura y escritura en cualquier memoria dentro del PS. Está compuesto por una librería en C y una de Python, que, al trabajar juntas, proporcionan una plataforma de control accesible para el hardware y software específicos integrados en el SoC-FPGA. En particular, el UDMA puede ser implementado junto al Comblock para permitir al usuario acceso a la información que entra y sale del PL desde una interfaz de alto nivel basada en Python, proporcionando así la última parte de la cadena del DAQ, como se observa en la figura 1.

Para gestionar el protocolo Ethernet TCP/IP, el UDMA se ejecuta en un sistema operativo de tiempo real en el PS, específicamente FreeRTOS. No obstante, es posible implementarlo también en otros sistemas operativos embebidos. Este sistema operativo puede configurarse directamente en el Vitis SDK. Posteriormente, es necesario importar la biblioteca del servidor UDMA, la cual está disponible en el repositorio del proyecto [36]. La interfaz física del sistema utiliza un cable de estándar RJ45.

Desde la perspectiva del usuario (PC), el UDMA proporciona una biblioteca en Python que incluye una clase diseñada como un envoltorio para un socket TCP, la cual emplea un protocolo básico para gestionar los comandos. Al instanciar la clase, es necesario ingresar la IP y el puerto del servidor; una vez establecido el socket, se pueden ejecutar comandos mediante la inclusión de los parámetros necesarios [36].

La figura 20 muestra la arquitectura del DAQ implementado, destacando las diferentes etapas descritas para identificar el flujo de información y concretar el diagrama general presentado en la figura 1.

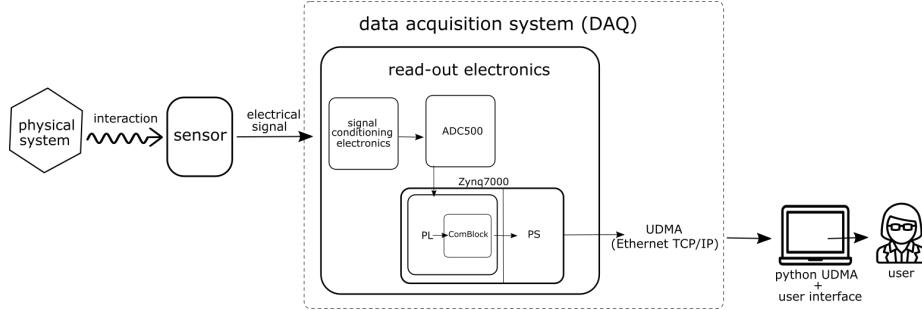


Figure 20: Diagrama esquemático del DAQ implementado donde se describen sus etapas particulares como el ADC500, FPGA SoC Zynq7000 y UDMA

0.2.5 Montaje Experimental: Diseño Implementado en el PL

Aprovechando la estructura modular del diseño implementado en el PL, se procede inicialmente a describir cada módulo por separado, detallando la lógica configurada y sus respectivas funciones. Posteriormente, se abordará el diseño en su conjunto, analizando las interconexiones entre los módulos y destacando las diferentes configuraciones posibles. Esto permitirá demostrar la flexibilidad y versatilidad de esta metodología para el desarrollo de sistemas de adquisición y procesamiento de señales.

Durante la fase de pruebas de los módulos de procesamiento digital de señales, es crucial poder visualizar las señales de salida generadas por estos módulos. Para lograrlo, se utiliza un módulo multiplexor (ver figura 17), que permite seleccionar, desde un registro de salida del ComBlock, la señal que se almacenará en la FIFO y que, a su vez, se mostrará en la interfaz de Python. Esta metodología presenta varias ventajas importantes, como la capacidad de verificar el funcionamiento de cada módulo de procesamiento de señales de forma individual, sin necesidad de modificar el diseño del PL ni realizar una nueva síntesis cada vez que sea necesario corregir o mejorar la lógica del hardware.

Dado que la aplicación específica de este trabajo se centra en la adquisición y procesamiento de señales provenientes de sensores o detectores para extraer medidas de interés en procesos físicos, es esencial contar con un sistema de visualización que permita inspeccionar las características de estas señales, similar a la función de un osciloscopio. Considerando que el ADC500 puede digitalizar señales de manera continua y a alta velocidad, el flujo de datos resultante puede ser significativo, llegando a saturar los recursos de memoria disponibles. Por lo tanto, es crucial implementar un sistema que seleccione las señales a visualizar en función de algún parámetro específico. Por ejemplo, la lógica de trigger, cuyo módulo se observa en la figura 21, habilita el modo de escritura de la FIFO del ComBlock únicamente cuando la amplitud de la señal supera un umbral definido

por el usuario.

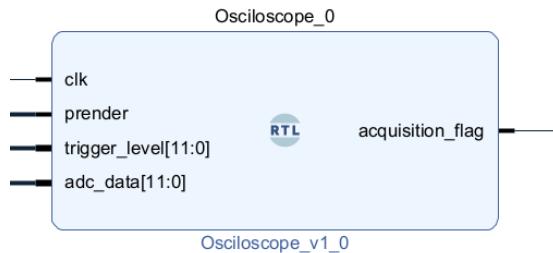


Figure 21: Diagrama de bloques de Vivado de un módulo de trigger simple escrito en VHDL. Entre sus señales de entrada se observan el reloj del sistema, señal de reset, nivel del umbral de amplitud de la señal y señal digitalizada. A su salida se observa la señal para activar la escritura de la FIFO del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

Tanto la señal de reloj (clk) como la de reinicio (reset) están conectadas al gestor de reloj y al sistema de reset, respectivamente. La señal de control del nivel de umbral, por su parte, está vinculada a un registro de salida del ComBlock, el cual puede ser configurado por el usuario a través de la interfaz UDMA en Python. De igual manera, desde esta interfaz es posible habilitar la lectura de la FIFO y extraer los datos almacenados una vez que el módulo de trigger ha activado la escritura en la FIFO. Finalmente, los datos de la señal, a partir del punto en que se cumple la condición de superación del umbral, pueden ser representados gráficamente utilizando un entorno especializado como matplotlib.

No obstante, esta lógica presenta un problema al realizar la adquisición de señales pulsadas, ya que solo se registra en la memoria la parte del pulso posterior al cumplimiento de la condición de umbral. Esto impide la visualización de la forma completa del pulso, lo cual es inconveniente, especialmente en contextos como la detección de partículas, donde la forma completa del pulso es un parámetro de interés crucial. Para solucionar este problema, una posible solución es implementar una lógica de "pre-trigger time". Esta lógica acumula muestras digitalizadas de la señal en un búfer y cuando se cumple la condición de trigger, tanto la parte de la señal posterior al cumplimiento de la condición como una parte anterior, previamente almacenada en el búfer, son concatenadas y enviadas a la memoria, permitiendo visualizar el pulso completo.

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 (ver figura 22), 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 in-

terfaz 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

$$\frac{N_{\text{data}} * d}{f_{\text{ADC}}} \quad (10)$$

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.

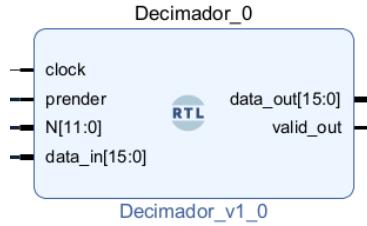


Figure 22: Diagrama de bloques de Vivado de un módulo de decimación escrito en VHDL.

Un filtro comúnmente utilizado en tratamiento de señales es el de media móvil. Como se detalla en la sección 0.1.4, el filtro de media móvil simple es esencial en los sistemas de adquisición de datos (DAQ) por su capacidad para atenuar componentes de alta frecuencia en las señales, funcionando como un filtro contra el ruido electrónico. En este trabajo, se implementa en VHDL un filtro de media móvil de ventana fija, tal como se describe en la figura 23, donde se muestra su representación en bloque en Vivado.

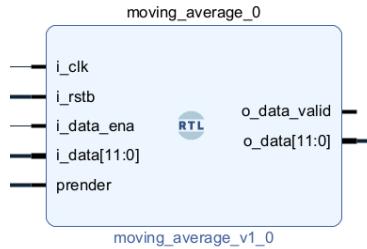


Figure 23: Diagrama de bloques de Vivado de un módulo de media móvil simple escrito en VHDL. Entre sus señales de entrada se observan el reloj del sistema, señal de reset, señal de habilitación y señal digitalizada. A su salida se observa la señal filtrada.

0.3 Resultados

0.4 Análisis y conclusiones

Bibliography

- [1] J. G. Webster and H. Eren, *Measurement, Instrumentation, and Sensors Handbook: Two-Volume Set*. CRC press, 2018.
- [2] S. Weinberg, “The first three minutes,” *FLAMINGO, Fontana*, 1976.
- [3] R. R. Ernst, G. Bodenhausen, and A. Wokaun, *Principles of nuclear magnetic resonance in one and two dimensions*. Oxford university press, 1990.
- [4] R. Scarr and R. Setterington, “Thermistors, their theory, manufacture and application,” *Proceedings of the IEE-Part B: Electronic and Communication Engineering*, vol. 107, no. 35, pp. 395–405, 1960.
- [5] R. G. Driggers, M. H. Friedman, and J. Nichols, *Introduction to infrared and electro-optical systems*. Artech House, 2012.
- [6] K. S. Krane, *Modern physics*. John Wiley & Sons, 2019.
- [7] V. Kledrowetz, R. Prokop, L. Fujcik, and J. Haze, “A fully differential analog front-end for signal processing from emg sensor in 28 nm fdsOI technology,” *Sensors*, vol. 23, no. 7, p. 3422, 2023.
- [8] J. G. Proakis, *Digital signal processing: principles, algorithms, and applications, 4/E*. Pearson Education India, 2007.
- [9] H. Austerlitz, *Data Acquisition Techniques Using PC*. Academic Press, 2014.
- [10] M. L. Crespo, F. Foulon, A. Cicuttin, M. Bogovac, C. Onime, C. Sistera, R. Melo, W. Florian Samayo, 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.
- [11] I. Sinclair, *Sensors and transducers*. Elsevier, 2000.
- [12] G. F. Knoll, *Radiation detection and measurement*. John Wiley & Sons, 2010.

- [13] H. Kolanoski and N. Wermes, *Particle Detectors: Fundamentals and Applications*. Oxford University Press, USA, 2020.
- [14] S. D. Brown and Z. G. Vranesic, *Fundamentals of digital logic with VHDL design*, vol. 70125910. McGraw-Hill New York, 2000.
- [15] W. R. Leo, *Techniques for nuclear and particle physics experiments: a how-to approach*. Springer Science & Business Media, 1994.
- [16] 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.
- [17] U. Meyer-Baese and U. Meyer-Baese, *Digital signal processing with field programmable gate arrays*, vol. 65. Springer, 2007.
- [18] L. F. Ramirez and S. Montoya, *low cost PHA for Mössbauer Spectroscopy*. Elsevier, 2024.
- [19] I. Bravo-Muñoz, A. Gardel-Vicente, and J. L. Lázaro-Galilea, “New applications and architectures based on fpga/soc,” 2020.
- [20] E. Rucci, *Evaluación de rendimiento y eficiencia energética de sistemas heterogéneos para bioinformática*. 2018.
- [21] 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.
- [22] R. Zurawski, *Industrial communication technology handbook*. CRC press, 2014.
- [23] 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.
- [24] EEE Guide, “Serial communication interface 8251,” 2024. Accessed: 2024-08-20.
- [25] ICTP Multidisciplinary Laboratory, “Smr3765 repository.” <https://gitlab.com/ictp-mlab/smr3765/-/wikis/uploads/d6f86235c15669356a9567b2aa0363c4/image.png>, 2022. Último acceso: 13 de agosto de 2024.
- [26] 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.

- [27] ICTP Multidisciplinary Laboratory, “Adc500 repository.” <https://gitlab.com/ictp-mlab/adc500>, 2022. Último acceso: 22 de agosto de 2024.
- [28] X. Inc., “Vivado design suite.” Disponible en: <https://www.xilinx.com/products/design-tools/vivado.html>, 2023. Versión 2023.1.
- [29] Avnet, “Zedboard: Zynq-7000 arm/fpga soc development board.” <https://www.zedboard.org/product/zedboard>, 2012. Recuperado el 07 de agosto de 2024.
- [30] X. Inc., “Vitis unified software platform,” 2024. Accedido: 22-ago-2024.
- [31] R. A. Melo, B. Valinoti, M. B. Amador, L. G. García, A. Cicuttin, and M. L. Crespo, “Study of the data exchange between programmable logic and processor system of zynq-7000 devices,” in *2019 X Southern Conference on Programmable Logic (SPL)*, pp. 3–8, IEEE, 2019.
- [32] R. Melo, “Core comblock.” <https://gitlab.com/rodrigomelo9/core-comblock>. Accessed: 2024-08-22.
- [33] Xilinx, “Vivado design suite user guide: Designing with ip.” https://support.xilinx.com/s/article/1053914?language=en_US. Accessed: 2024-08-22.
- [34] The Eclipse Foundation, “Eclipse IDE: Integrated Development Environment,” 2024. Version 2024-06.
- [35] Real Time Engineers Ltd., “FreeRTOS: Real-Time Operating System for Microcontrollers,” 2024. Version 10.5.1, accessed August 24, 2024.
- [36] I. M-Lab, “Udma.” <https://gitlab.com/ictp-mlab/udma>, 2021. Accessed: 2024-08-24.