UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

DISEÑO E IMPLEMENTACIÓN DE UN ESPECTRÓMETRO DE ALTA RESOLUCIÓN BASADO EN FPGA PARA EL ANÁLISIS DE SEÑALES RADIOASTRONÓMICAS

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO

RAÚL IGNACIO SAPUNAR OPAZO

PROFESOR GUÍA:
RICARDO ALBERTO FINGER CAMUS

MIEMBROS DE LA COMISIÓN:
FAUSTO PATRICIO MENA MENA
CLAUDIO ESTÉVEZ MONTERO

Este trabajo fue realizado gracias al apoyo de CONICYT a través del Centro de Astrofísica y Tecnologías Afines (PBF 06), ALMA-CONICYT 31120005, Gemini-CONICYT 32120004 y FONDECYT 11140428. Agradecemos a Xilinx Inc., por la donación de circuitos integrados y licencias de software.

SANTIAGO DE CHILE
2015

RESUMEN DE LA MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO
POR: RAÚL IGNACIO SAPUNAR OPAZO
FECHA: ENERO 2015
PROF. GUÍA: RICARDO FINGER CAMUS

DISEÑO E IMPLEMENTACIÓN DE UN ESPECTRÓMETRO DE ALTA RESOLUCIÓN BASADO EN FPGA PARA EL ANÁLISIS DE SEÑALES RADIOASTRONÓMICAS

Un espectrómetro de alta resolución se usa en radioastronomía para observar espectros de nubes moleculares de dinámica compleja con un amplio rango de velocidades. Dicho instrumento corresponde a un tipo de *Back End*, el cual es el último componente electrónico de la cadena en un radiotelescopio y está encargado de procesar la señal proveniente de las fuentes astronómicas para obtener la información deseada. El estado del arte en esta materia consiste en utilizar técnicas digitales para procesar los datos, debido a la estabilidad de las mediciones y buen rendimiento que estas permiten.

En este trabajo se presenta el diseño e implementación de un espectrómetro digital de transformada de Fourier rápida (FFT), de alta resolución espectral, utilizando una plataforma ROACH 2, la cual está basada en un chip FPGA (*Field Programmable Gate Array*) y está equipada con ADCs (*Analog-to-digital* coverters) de alta velocidad.

Como metodología de diseño, se propone maximizar el uso de recursos del hardware para obtener el mayor número de canales espectrales posibles para un ancho de banda no menor a 1,5 GHz, manteniendo un alto rango dinámico. Para lograr el correcto funcionamiento del espectrómetro a altas velocidades se aplicaron técnicas de localización física de recursos (*floorplanning)* en el FPGA. Se presenta también una metodología de calibración de los ADCs, los cuales poseen cuatro núcleos que en conjunto muestrean a una tasa máxima de 5 GSps.

El mejor desempeño se obtuvo con un espectrómetro de 1,8 GHz y 32768 canales, es decir con una resolución espectral de 54,9 kHz. Éste posee un rango dinámico libre de espurios (SFDR) superior a 42 dB en toda la banda con caídas a 39 dB en frecuencias puntuales. El aumento de resolución espectral se ve limitado principalmente por problemas de sincronización (*timing*) en el circuito y por la cantidad de memoria disponible.

Como trabajo futuro se propone explorar distintas técnicas para optimizar el uso de recursos del FPGA y así aumentar aún más la resolución espectral del espectrómetro. Ejemplos de estas técnicas son modificaciones de diseño que permitan el uso eficiente de memorias y la reducción de operaciones matemáticas necesarias. También se presenta un cambio mayor en el diseño del espectrómetro, el cual implementa una forma distinta para calcular la FFT y permitiría reducir enormemente el número de memorias utilizadas por esta, logrando llegar a 1 millón de canales espectrales.

AGRADECIMIENTOS

Quiero agradecer a todos los que hicieron posible este momento. A mi familia, por su incondicional apoyo durante el camino recorrido y en especial a mi padre, por siempre motivarme a ponerme metas altas y a dar lo mejor de mí para alcanzarlas. También a mis amigos, por brindarme la distracción y buenos momentos cuando más lo necesité.

A todos los integrantes del Laboratorio de Ondas Milimétricas, por su buena disposición a ayudarme cada vez que tuve dudas, en especial a mi profesor guía, Ricardo Finger, por darme la oportunidad de trabajar en este proyecto y estar presente durante todo el proceso.

Finalmente agradecer a la comunidad de CASPER por su colaboración en el proyecto.

TABLA DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

CAPÍTULO 1

INTRODUCCIÓN

1.1 MOTIVACIÓN Y OBJETIVO GENERAL

Dado el progreso tecnológico en las antenas y receptores radioastronómicos en el último tiempo, se ha abierto la posibilidad de explorar partes del universo que antes eran impensadas. Así surge la necesidad de mejorar también el procesamiento de señales provenientes del espacio, junto con desarrollar herramientas específicas para distintos tipos de observaciones astronómicas.

Antiguamente el procesamiento de señales astronómicas era analógico y desde hace un tiempo que se ha migrado al mundo digital, principalmente debido a la estabilidad y fácil operación que éste permite, sumado a tiempos de diseño y construcción mucho más cortos. Un FPGA (*Field Programmable Gate Array*) entrega la ventaja de su velocidad y capacidad de procesamiento paralelo, lo que da una enorme posibilidad para implementar circuitos digitales tremendamente útiles para aplicaciones radioastronómicas.

Con cielos privilegiados para la observación astronómica, Chile es sede de grandes observatorios internacionales. Esto ha generado oportunidades para entrar al desarrollo de instrumentación radioastronómica, que ha tomado una gran fuerza en los últimos años. En el Observatorio Astronómico Nacional se ha incursionado también en el procesamiento digital de señales radioastronómicas.

En este contexto, el objetivo principal de esta memoria es diseñar e implementar un espectrómetro digital de transformada de Fourier rápida, de alta resolución espectral, utilizando una plataforma ROACH 2 (*Reconfigurable Open Arquitecture Computing Hardware*), la cual está basada en un chip FPGA Xilinx Virtex 6.

1.2 ALCANCES Y OBJETIVOS ESPECÍFICOS

Un espectrómetro de alta resolución utilizará una gran cantidad de recursos del FPGA, por lo que la meta de este trabajo es lograr la mayor cantidad de canales espectrales posibles con el hardware escogido, intentando llegar al estado del arte en el rubro.

Otro punto de interés es obtener un espectrómetro que además de buena resolución, tenga un ancho de banda instantáneo considerable, para lo cual se disponen de ADCs de alta velocidad conectados a la plataforma ROACH 2.

Al no existir mayores antecedentes de diseños implementados en ROACH 2, será parte de este trabajo lograr en una primera instancia la obtención de resultados similares a los obtenidos con la versión anterior del hardware, ROACH 1, de la cual si se tiene una variada bibliografía.

De esta forma, los objetivos específicos de este trabajo son los siguientes:

- Implementar una metodología de calibración para los ADC de alta velocidad de la plataforma ROACH 2
- Diseñar e implementar un espectrómetro digital de transformada de Fourier rápida en ROACH 2.
- Diseñar e implementar un espectrómetro digital de transformada de Fourier rápida con la mayor resolución espectral posible en ROACH 2, para un ancho de banda instantáneo no menor a 1.5 GHz.

1.3 ESTRUCTURA DE LA MEMORIA

La estructura utilizada en este documento para exponer el trabajo realizado es la siguiente:

- **Capítulo 2. Revisión Bibliográfica:** En este capítulo se presenta el marco teórico necesario para entender y contextualizar el trabajo.
- **Capítulo 3. Implementación:** Este capítulo contiene la metodología de diseño del circuito implementado, el hardware utilizado y el diseño final. También se describen las pruebas realizadas y la calibración de los conversores análogo-digital.
- **Capítulo 4. Resultados:** Se presentan y analizan los resultados obtenidos con los diseños y programas implementados, junto con medidas de desempeño de estos.
- **Capítulo 5. Conclusiones:** Se muestran las conclusiones del trabajo realizado junto con recomendaciones y el trabajo futuro a realizar para mejorar los resultados obtenidos.

CAPÍTULO 2

REVISIÓN BIBLIOGRÁFICA

2.1 RADIOASTRONOMÍA

La astronomía es la ciencia que estudia el comportamiento y los fenómenos ligados a los cuerpos celestes del universo, tales como planetas, cometas, estrellas y galaxias. La radioastronomía es la rama de esta ciencia que estudia estos cuerpos mediante la medición de radiación electromagnética en la banda de radio del espectro. Es un método muy utilizado en la observación espacial, ya que en estas longitudes de onda la atmósfera terrestre resulta ser muy transparente, es decir no absorbe las ondas electromagnéticas [1]. En la Figura 2.1 se representa la opacidad de la atmósfera terrestre, donde se aprecian las ventanas atmosféricas para luz visible y ondas de radio.

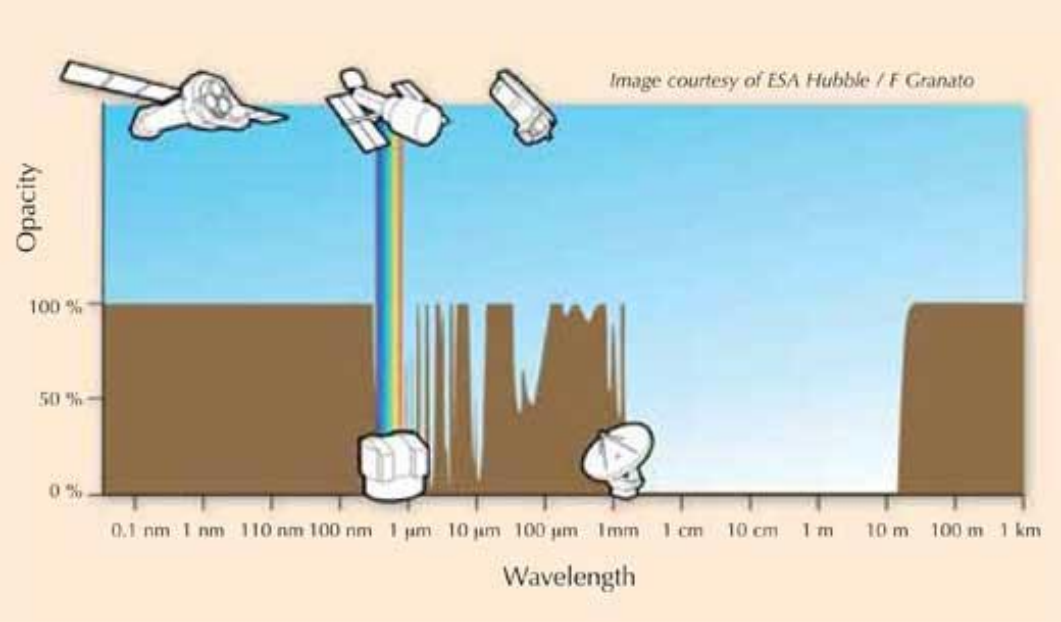


Figura 2.1: Opacidad de la atmosfera.[1]

La banda de radio se extiende entre longitudes de onda de 1 cm a 10 m, aunque existen también algunas bandas relativamente transparentes situadas por debajo del centímetro, las que corresponden a las llamadas ventanas milimétricas y submilimétricas [2].

[1] Figura extraída de Fariña, 2010 [1]

2.1.1 Emisión de ondas de radio

Todos los objetos a temperaturas sobre el cero absoluto emiten energía en forma de ondas electromagnéticas, la que puede presentarse en forma continua o como líneas espectrales. La emisión continua se extiende en una región ancha del espectro electromagnético mientras que las líneas espectrales se hallan centradas en una frecuencia específica.

Toda la materia del universo emite o absorbe radiación electromagnética en forma de líneas espectrales. En la región radio del espectro suelen encontrarse líneas de transición, rotacionales y vibracionales de las moléculas más comunes del universo, lo que resulta de gran interés en radioastronomía ya que su estudio sirve para conocer la composición química de las galaxias [3].

Las señales de radio capturadas en la Tierra resultan ser de muy baja energía, lo que hace que sean muy vulnerables al ruido que generan los receptores electrónicos utilizados en radiotelescopios. Esto hace necesario el diseño de componentes muy especializados para la construcción de dichos receptores, los que varían dependiendo de la banda de frecuencia dentro del radio espectro.

2.1.2 Radiación de cuerpo negro

Los objetos no solamente emiten energía electromagnética sino que también absorben o reflejan la energía incidente en ellos. Kirchoff (1859) demostró que un cuerpo que es buen absorbente de energía es también un buen emisor de esta [2]. Un cuerpo negro es un objeto idealizado que absorbe radiación en todas las longitudes de onda y emite energía sólo como consecuencia de su temperatura. Esta idealización permite obtener una buena aproximación de la distribución espectral de un cuerpo en emisión termal, conociendo únicamente su temperatura de cuerpo negro [1].

La *ley de radiación de Planck,* presentada a continuación, describe la distribución del brillo de la radiación de un cuerpo negro en función de su temperatura T y su frecuencia *v* de radiación.

$$B_v(T) = \frac{2hv^3}{c^2} \frac{1}{e^{hv/kT} - 1} \tag{2.1}$$

donde,

B = intensidad $\left[\frac{W}{m^2 rad^2 Hz}\right]$
h = constante de Planck = $6{,}626 \times 10^{-34} [J\ seg]$
v = frecuencia $[Hz]$
c = velocidad de la luz = $3 \times 10^8 \left[\frac{m}{seg}\right]$
k = constante de Boltzmann = $1{,}38 \times 10^{-23} \left[\frac{J}{K}\right]$
T = temperatura $[K]$

En la región del espectro de ondas de radio el producto *hv* es muy pequeño comparado con el factor *kT* (*hv* << *kT*), lo que hace posible realizar la siguiente aproximación:

$$e^{hv/kT} \approx \frac{hv}{kT} + 1 \tag{2.2}$$

Aplicando esta expresión a la *ley de radiación de Planck* se obtiene la ecuación (2.3), que corresponde a la *ley Rayleigh-Jeans*, la que resulta muy útil para determinar la intensidad de brillo de un cuerpo en la banda de radio.

$$B = \frac{2kT}{\lambda^2} \tag{2.3}$$

donde λ corresponde a la longitud de la onda emitida por el objeto.

2.2 ARQUITECTURA DE RECEPTORES EN RADIOASTRONOMÍA

La mayoría de los receptores utilizados en radioastronomía son similares a los usados en telecomunicaciones. Estos se llaman receptores heterodinos y su principal característica es que convierten la señal de entrada a una de menor frecuencia, conservando la amplitud y fase original. Estos forman parte del *Front End* de un radiotelescopio, el cual recibe la señal radioastronómica capturada previamente por una antena, la que es comúnmente parabólica. La señal resultante es de baja frecuencia y se conoce como "frecuencia intermedia" (IF), la que es procesada por componentes electrónicos, conocidos como *Back End*, para obtener la información deseada [4].

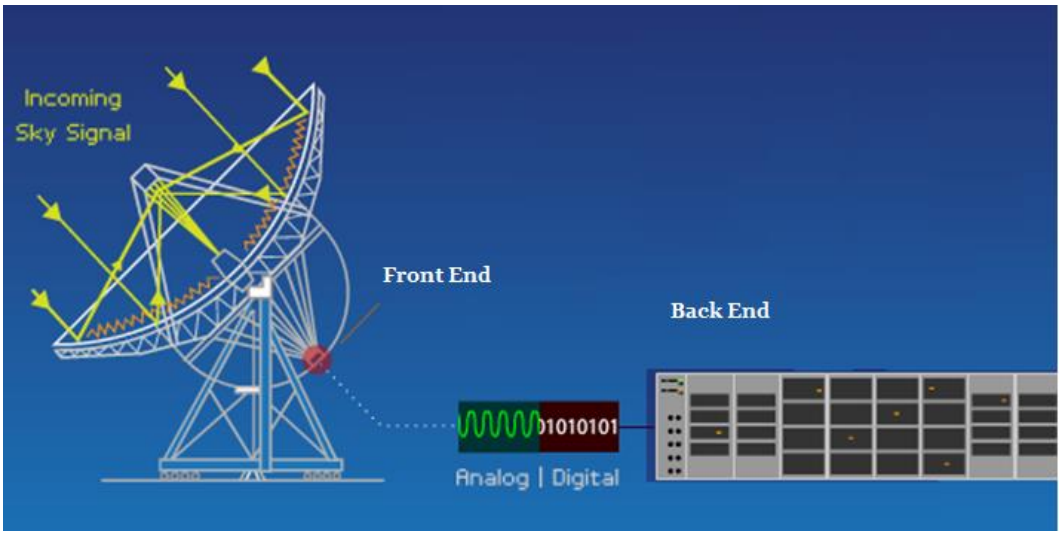


Figura 2.2: Arquitectura general de un radiotelescopio

2.2.1 Antena

Una antena tiene la función de concentrar y realizar la transición de la energía electromagnética desde el espacio libre a líneas de transmisión o viceversa [5]. La mayoría de los radiotelescopios utilizan antenas parabólicas tipo Cassegrain, cuya principal característica es la presencia de un reflector en forma de paraboloide de revolución. Su función es reflejar las ondas de radio

procedentes de fuentes astronómicas concentrándolas en un punto conocido como foco [6]. Mediante un subreflector se logra posicionar el foco en un punto detrás de la antena, donde se encuentra el receptor que captura la señal. Este proceso de reflexiones se puede apreciar en la Figura 2.2.

Para captar la mayor cantidad de energía proveniente del espacio y así optimizar la calidad de la señal recibida es que la superficie de las antenas debe ser perfecta. En general se intenta construir una parábola con un error menor a $\sim\lambda/20$ (alrededor de 25 micrómetros en antenas de ALMA[2]) [7].

2.2.2 Front End

Front End corresponde a los primeros elementos electrónicos (analógicos) por los que pasa la señal proveniente del espacio, los que tienen la función de detectar, amplificar y convertir dicha señal. Para maximizar la detección de la señal original, es sumamente importante que estos componentes tengan el menor ruido posible. Por esta razón, la etapa más crítica del *Front End* se mantiene a temperaturas cercanas al cero absoluto.

En el centro de la cadena de *Front End* se realiza la *downconversión* de la señal, que consiste en mezclar la señal con una señal de frecuencia de referencia para bajar la frecuencia original de esta. Este proceso es realizado utilizando un receptor heterodino. Una configuración simple de un receptor de este tipo se muestra en la Figura 2.3.

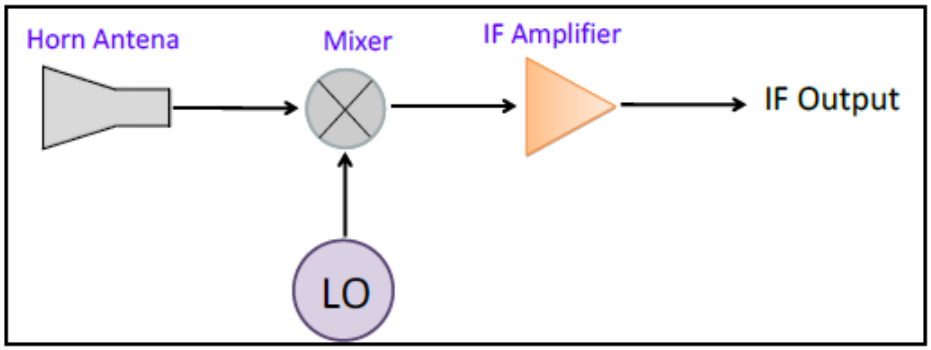


Figura 2.3: Diagrama de bloques de un receptor heterodino.[3]

En todos los receptores heterodinos se utiliza un mezclador, el cual es un dispositivo no lineal que mezcla la señal de entrada (RF) y una señal de referencia producida por un oscilador local (LO). Como resultado se obtienen señales compuestas por la suma y resta de las señales de entrada, así como otras frecuencias de intermodulación de mayor orden y armónicos. Un filtro pasa bajo es utilizado para detectar la señal que tiene una frecuencia igual a la diferencia en frecuencia entre la señal LO y RF, la que se conoce como "frecuencia intermedia" (IF). Este proceso corresponde a la *downconversión* de la señal.

[2] Atacama Large Millimiter/submillimiter Array
[3] Figura extraida de Ricardo Finger, 2013 [4]

2.2.3 Back End

Back End corresponde a la última parte de la cadena de componentes electrónicos de un radiotelescopio, encargada de digitalizar y procesar la señal IF para obtener la información deseada. En general el procesamiento de la señal digital consiste en analizar la polarización de esta, su estructura temporal y propiedades espectrales de la radiación continua capturada [8]. Para hacer esto se implementan detectores de polarización, correladores y espectrómetros.

El hardware utilizado en un *Back* End digital consiste en ADCs (conversores análogo-digital) y un computador o circuitos digitales para el procesamiento de los datos digitales. Por ejemplo en ALMA se utiliza un correlador, diseñado para procesar los datos que llegan de todas las antenas. Este correlador corresponde a una solución ASIC[4] (*Application–specific integrated circuit*), formado por 2192 circuitos impresos [7].

El estado del arte en dispositivos *Back End* de análisis espectral en tiempo real corresponde a espectrómetros de transformada de Fourier rápida (FFTS), con un ancho de banda instantáneo máximo de 2,5 GHz y 65536 canales espectrales [9].

2.3 ANÁLISIS DE SEÑALES RADIOASTRONÓMICAS

Como se mencionó anteriormente, la primera etapa del proceso de *Back End* consiste en digitalizar la señal radioastronómica de frecuencia intermedia (IF), para lo cual se utilizan ADCs. Para hacer esto se debe muestrear la señal tal que con los datos tomados se pueda reconstruir exactamente la señal analógica. Para lograr esto existen ciertos criterios que deben cumplirse en el muestreo de la señal.

2.3.1 Teorema de Nyquist-Shannon

El teorema de muestreo de Nyquist-Shannon es de vital importancia en la digitalización de señales. Éste demuestra que la reconstrucción exacta de una señal continua a partir de sus muestras, es matemáticamente posible si la señal está limitada en frecuencia y la tasa de muestreo es mayor al doble de su ancho de banda [10], es decir

$$F_{muestreo} \geq 2B_{señal} \tag{2.4}$$

Es importante notar que el concepto de ancho de banda no necesariamente es sinónimo del valor de la frecuencia más alta en la señal analógica. Esto sólo se cumple para señales de banda base.

[4] http://en.wikipedia.org/wiki/Application-specific_integrated_circuit

Si este criterio no se cumple al muestrear una señal analógica limitada en banda, ésta no podrá ser reconstruida correctamente a partir de la señal discreta. Eso se debe a que señales de alta frecuencia aparecerán en la banda fundamental como señales de menor frecuencia, produciendo una señal distinta a la original. Este efecto es conocido como *aliasing*.

2.3.2 Espectrómetro

Los espectrómetros son probablemente los dispositivos *Back End* más utilizados en radioastronomía. Estos son diseñados para medir la densidad espectral de una señal, usando métodos para calcular la respuesta en frecuencia de una señal en el tiempo.

El diseño de los espectrómetros se enfoca en obtener la información espectral (líneas espectrales) contenida en la radiación capturada por la antena del radiotelescopio, por lo que la resolución en frecuencia debe ser pequeña (en el rango de los kHz) y además deben tener una estabilidad temporal muy alta.

El tiempo necesario para medir el espectro de la potencia de una señal astronómica puede ser reducido en un factor de *n* si el espectrómetro es construido utilizando *n* filtros e integradores. Esto requiere *n* canales independientes que miden simultáneamente diferentes partes del espectro. En este caso, cada uno de estos canales deberá tener un ancho de banda en el orden de los kHz, pero dependiendo del número de canales se podrá generar un espectrómetro de muy alta resolución, lo que es muy útil para algunas aplicaciones astronómicas [8].

Los espectrómetros se diseñan para medir la densidad espectral de una señal, la que corresponde a la distribución de la potencia de dicha señal sobre las distintas frecuencias por las que está formada, es decir, su espectro. Esta se conoce como densidad espectral de potencia o PSD.

Densidad Espectral de Potencia

En general, la energía contenida en una señal $x(t)$ se define como:

$$E\big(x(t)\big) = \int_{-\infty}^{+\infty} |x(t)|^2 dt \tag{2.5}$$

Utilizando el Teorema de Parseval, la energía de una señal puede ser escrita en términos de su transformada de Fourier,

$$\int_{-\infty}^{+\infty} |x(t)|^2 dt \;=\; \int_{-\infty}^{+\infty} |\hat{x}(f)|^2 df \tag{2.6}$$

Como la integral del lado derecho equivale a la energía de la señal, el integrando puede ser interpretado como una función de densidad de potencia. Se define entonces la densidad espectral de potencia de una señal (PSD) como,

$$S(f) = |\hat{x}(f)|^2 \tag{2.7}$$

Esta definición se extiende a señales discretas en el tiempo, tal y como señales digitales.

Ancho de Banda

El ancho de banda de un espectrómetro, correspondiente al rango en frecuencia del espectro, dependerá de la frecuencia de muestreo de los ADC, esto debido al Teorema de Nyquist-Shannon. Cuando se cumple el teorema con una igualdad, se tendrá que,

$$B = \frac{F_{muestreo}}{2} \tag{2.8}$$

Es importante destacar que cuando se tiene un ADC que muestrea en los cantos de subida y bajada de su reloj, el ancho de banda del espectrómetro será igual a la frecuencia de reloj.

Resolución

Al hablar de la resolución de un espectrómetro, existe la resolución espectral (en frecuencia) y la resolución temporal. Dependiendo de la aplicación radioastronómica para la cual se quiera usar, el diseño de un espectrómetro requerirá enfatizar una de estas.

La resolución espectral corresponde al ancho de los canales espectrales del espectrómetro. Esta depende del ancho de banda y el número de canales espectrales. El ancho en frecuencia de cada canal está dado por:

$$\Delta f = \frac{B}{n° \; de \; canales} \tag{2.9}$$

Δf es una medida de cuan preciso podrá medir frecuencia un espectrómetro. Entre más pequeño sea el ancho de los canales, más alta será la resolución espectral de un espectrómetro para un determinado ancho de banda.

La resolución temporal de un espectrómetro corresponde a la tasa a la cual éste actualiza su espectro, la que depende de la cantidad de datos que haya que procesar. Esto quiere decir que espectrómetros con una gran cantidad de canales, y por lo tanto con una mayor resolución espectral, tendrán una peor resolución temporal.

Generalmente un espectrómetro acumula múltiples espectros para promediar el ruido de salida y así mejorar la relación señal a ruido del espectro. Entre mayor sea la acumulación de datos, menor

será la resolución temporal del espectrómetro. Para un espectrómetro de Fourier, el cual se detalla posteriormente, la resolución temporal está dada por,

$$\Delta t = \frac{FFT_{size}}{n^{\underline{o}}{}_{datos\ paralelos}} \times l_{acc} \times T \tag{2.10}$$

donde se tiene que,

- FFT_{size} = tamaño de la transformada de Fourier del espectrómetro
- l_{acc} = número de espectros completos que se acumulan y promedian
- T = período de reloj del procesador utilizado

Cuando se quieren observar eventos instantáneos de poca duración en el espectro de una señal, se hace necesario tener una resolución temporal alta, en cambio cuando se quiere observar un tono débil dentro de la señal, una larga acumulación temporal de datos es requerida, a costa de una baja resolución temporal.

Espectrómetro de Fourier

Existen varios tipos de espectrómetros, que principalmente varían en el método utilizado para encontrar la densidad espectral de la señal de entrada. El espectrómetro de Fourier, utiliza la transformada de Fourier para hacer esta operación, y es el que se implementa en la mayoría de las aplicaciones radioastronómicas. El espectrómetro de Fourier puede seguir dos caminos para llegar a la PSD (densidad espectral) de una señal en el tiempo. Esto se puede ver gráficamente en la Figura 2.4.

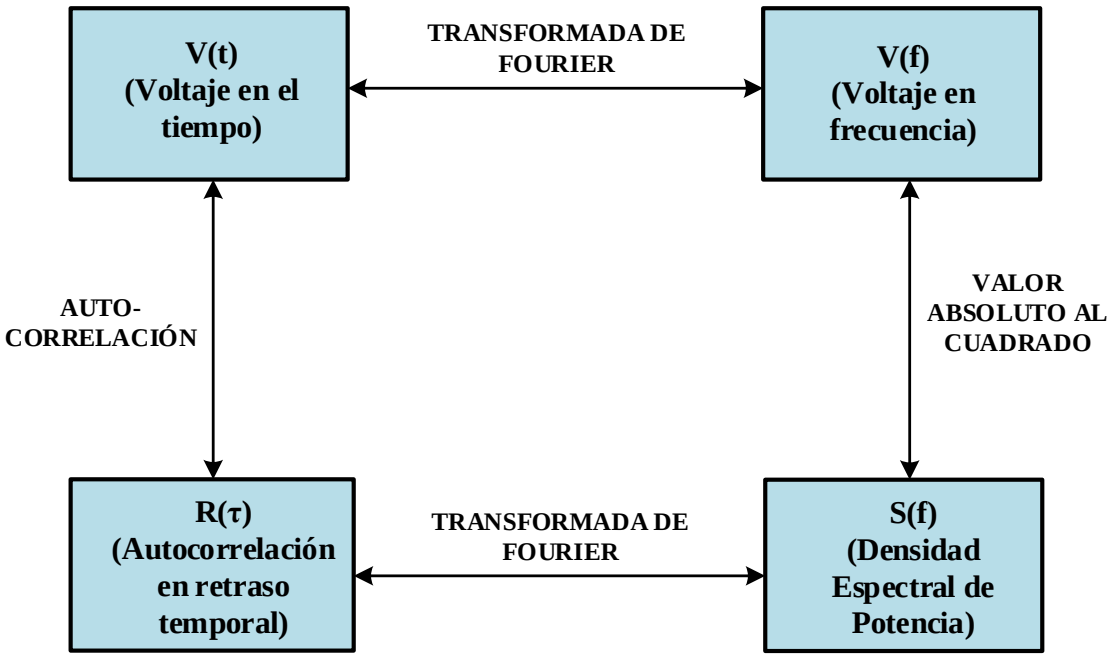


Figura 2.4: Diagrama de funcionamiento de espectrómetro de Fourier

2.3.3 Transformada de Fourier

El análisis de Fourier es una familia de técnicas matemáticas basadas en descomponer señales en sinusoides, lo que se puede utilizar para calcular el espectro de una señal en el tiempo. La transformada de Fourier discreta (DFT) es el miembro de la familia que se utiliza para señales digitales (discretas), que es el caso de la aplicación en un *Back End*.

Transformada de Fourier Discreta (DFT)

Una señal aperiódica discreta $x(n)$ de energía finita, tiene un espectro continuo y periódico de período 2π. Su transformada de Fourier $X(\omega)$ está dada por [11]:

$$X(\omega) = \sum_{n=-\infty}^{\infty} x(n)e^{-j\omega n} \qquad (2.\ 11)$$

Debido a su periodicidad, toda la información de la transformada $X(\omega)$ queda contenida de 0 a 2π. Al evaluar *N* muestras equiespaciadas entre 0 y 2π [12], se obtiene la transformada de Fourier discreta (DFT):

$$X(k) = \sum_{n=0}^{N-1} x(n)e^{-j2\pi kn/N} \qquad \forall k = 0,1,\dots,N-1 \qquad (2.\ 12)$$

La DFT más general es compleja, es decir acepta como entrada muestras con parte imaginaria, produciendo espectros con componente complejo también. Cuando la entrada $x(n)$ es real, es decir que cada una de las *N* muestras no posee parte imaginaria, su transformada $X(\omega)$ es simétrica [12]. Esto quiere decir que toda la información espectral de las *N* muestras reales queda contenida en la mitad de un período, es decir de 0 a π.

De esta forma se introduce la DFT real, la cual es una versión simplificada de la transformada de Fourier discreta, la cual representa la señal de salida utilizando únicamente la mitad de los puntos espectrales, lo que es suficiente cuando se tienen señales digitales en el dominio del tiempo. La DFT real transforma una señal de entrada de *N* muestras en dos señales de salida de *N/2 + 1* puntos, las que contienen amplitudes de ondas sinusoidales (senos y cosenos) [10]. La señal correspondiente a la parte real de la señal en el dominio de la frecuencia se denomina Re X[ ] y contiene las amplitudes del coseno, mientras que la imagen de la señal, denominada Im X[ ] contiene las amplitudes del seno. Esto se puede apreciar en la Figura 2.5.

En conjunto, los *N/2 + 1* puntos reales e imaginarios de la salida de la DFT corresponden al espectro de la señal de entrada, los cuales se utilizan para obtener la densidad espectral de potencia de la señal temporal al calcular su módulo al cuadrado. Cada uno de los *N/2 + 1* puntos obtenidos corresponderán a la información de cada canal espectral del espectrómetro de Fourier. Si se quisiera reconstruir la señal en el tiempo usando el espectro de la señal (DFT inversa), sólo haría falta asignar las amplitudes a los senos y cosenos correspondientes y sumarlos.

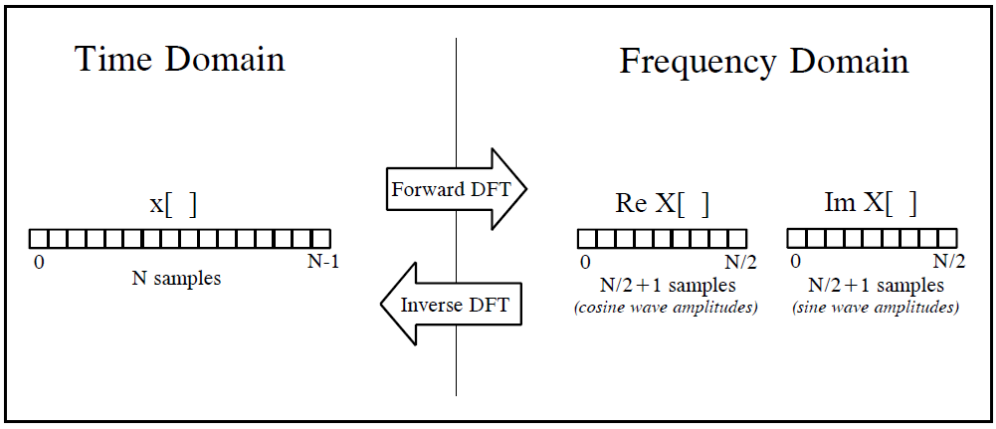


Figura 2.5: Transformada de Fourier discreta real (DFT real).[5]

Para calcular la DFT existen tres maneras totalmente distintas. La primera consiste en resolver un sistema lineal de *N* ecuaciones independientes. El segundo método es el de la "correlación", que se basa en reconocer funciones conocidas en otra señal. La tercera forma se conoce como transformada de Fourier rápida (FFT), que consiste en un algoritmo que descompone la DFT en *N* puntos para luego calcular *N* DFTs de un sólo punto.

Transformada de Fourier Rápida (FFT)

Este método para calcular la DFT de una señal es el más utilizado debido a su eficiencia, siendo mucho más rápido que los demás métodos, pero corresponde a uno de los algoritmos más complejos en el procesamiento de señales digitales.

El algoritmo que implementa la FFT se basa en la DFT compleja. Es por eso que para entender cómo utilizar la FFT para calcular la DFT real de una señal, es necesario saber cómo transferir datos de la DFT real a formato de DFT compleja. La DFT real transforma una señal en el dominio del tiempo de *N* puntos en dos señales de *N/2+1* en el dominio de la frecuencia, mientras que la DFT compleja transforma dos señales de *N* puntos en dos señales de *N* puntos. Esto se puede apreciar gráficamente en la Figura 2.6 [10].

[5] Figura extraída de Smith,1997 [10]

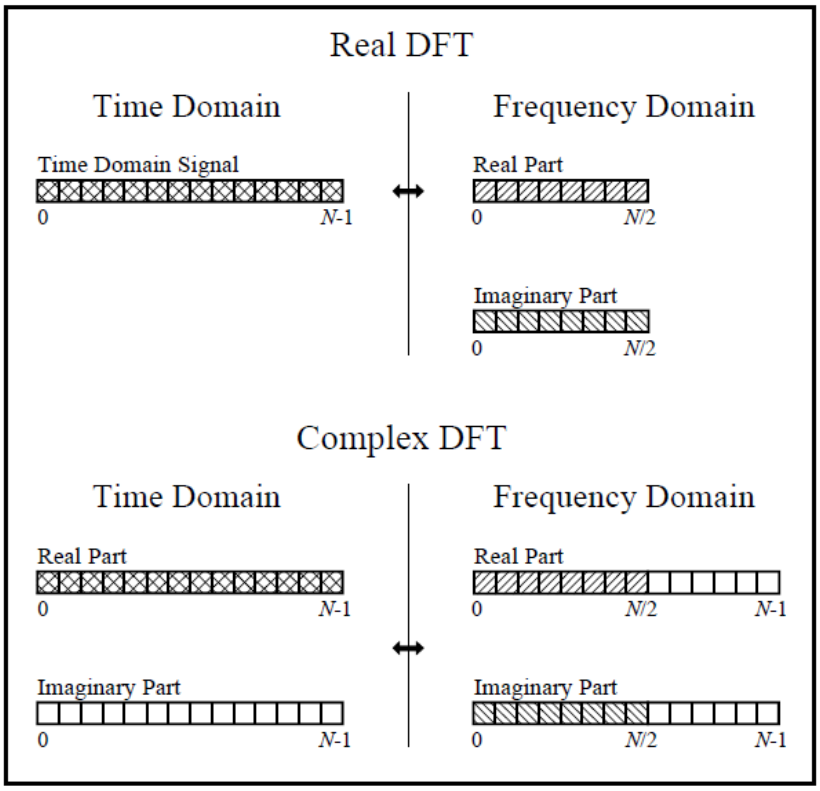


Figura 2.6: Comparación de DFT real y compleja.[6]

Si bien existen varios algoritmos de FFT, en general esta opera descomponiendo la señal en el tiempo de *N* puntos en *N* señales en el tiempo compuestas de un sólo dato cada una. El siguiente paso es calcular los *N* espectros de frecuencia correspondientes a estas señales, para finalmente, a partir de estos, sintetizar un solo espectro de la señal de entrada.

En la primera etapa, el algoritmo descompone las muestras intercaladamente, es decir separando las muestras con numeración impar y par. Cuando el número de muestras es una potencia de dos, el algoritmo se llama 'Radix-2', en el cual la descomposición requiere *N* pasos para descomponer una señal de 2^N puntos. Digitalmente, esta operación es análoga a reordenar las muestras según su representación binaria invertida. En la Figura 2.7 se presenta un ejemplo de una descomposición de una señal de 16 puntos en 4 pasos.

[6] Figura extraída de Smith, 1997 [10]

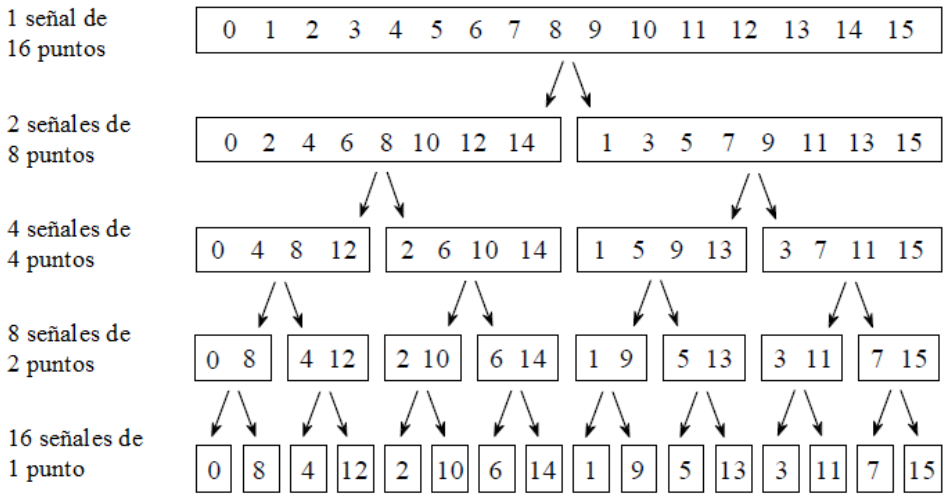


Figura 2.7: Ejemplo de primera etapa de algoritmo FFT.[7]

El siguiente paso del algoritmo resulta directo, ya que el espectro en frecuencia de una señal de un punto es igual a sí misma. Esto quiere decir que en realidad en este paso no se realiza ninguna operación. Finalmente el último paso consiste en combinar los *N* espectros de frecuencia de manera inversa de la misma forma en la que se hizo la descomposición. Para hacer esto no sirve el mismo reordenamiento de bits utilizado en la primera etapa; este proceso es más complejo. A modo de ejemplo, si se tiene una señal en el tiempo de 8 muestras *abdcefgh*, su espectro obtenido al final del algoritmo corresponde a *aebfcgdh*. En la Figura 2.8 se muestra un esquema de cómo se realiza la síntesis de una señal en frecuencia de 8 puntos a partir de 2 señales de 4 puntos espectrales. Los puntos impares se multiplican por una sinusoide de frecuencia escogida apropiadamente para duplicar la señal y luego se suman con los puntos espectrales pares.

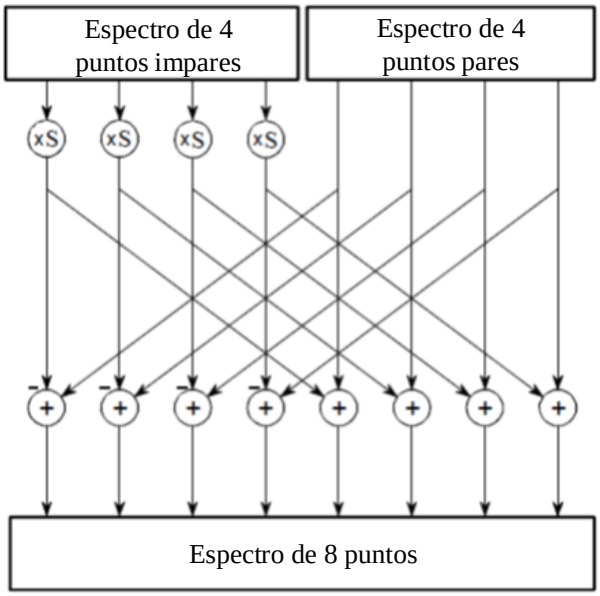


Figura 2.8: Ejemplo de última etapa del algoritmo FFT.[8]

[7] Figura extraída de Smith, 1997 [10]
[8] Figura extraída de Smith, 1997 [10]

2.4 ESPECTROSCOPÍA DE ALTA RESOLUCIÓN

Una importante motivación para estudiar nubes moleculares es entender cómo se formaron las estrellas y los espectrómetros de alta resolución son una poderosa herramienta para hacerlo. Las nubes moleculares son observadas mediante la radiación que emiten debido a transiciones rotacionales de sus moléculas. Esta radiación corresponde a líneas espectrales, las que en realidad corresponden a ondas electromagnéticas en un rango de frecuencias (ancho de la línea espectral) y no en una frecuencia específica.

El ancho de líneas moleculares varía debido a distintos fenómenos, como por ejemplo debido al efecto Doppler, en que la frecuencia de una onda varía para un observador moviéndose relativamente a la fuente de la onda. Este fenómeno es conocido como "*molecular line broadening*". Esto implica que moléculas moviéndose a distintas velocidades producirán líneas espectrales de distinto ancho, necesitando distinta resolución espectral para poder estudiarlas de manera adecuada.

Otra importante aplicación de la espectroscopía de alta resolución es la observación de estructuras finas e hiperfinas de átomos y moléculas. La estructura fina explica la separación de líneas espectrales de átomos debido a la interacción de los momentos magnéticos asociado con el *spin* de los electrones y el *momentum* angular de los orbitales. La estructura hiperfina corresponde a pequeños *shifts* y separaciones de niveles de energía de átomos y moléculas, los que son consecuencia de la interacción de momentos multipolo con campos magnéticos internos. Los efectos de la estructura fina e hiperfina de átomos se aplican a cada núcleo de moléculas, que sumado a algunos otros efectos producen la separación de líneas espectrales moleculares [13].

Las líneas espectrales producidas por estas estructuras se encuentran muy cercanas en frecuencia y tienen un ancho muy pequeño, por lo que es necesario tener una alta resolución espectral para poder diferenciarlas al observarlas. En la Figura 2.9 se muestra una ilustración de la estructura fina e hiperfina del átomo de hidrogeno a modo de ejemplo.

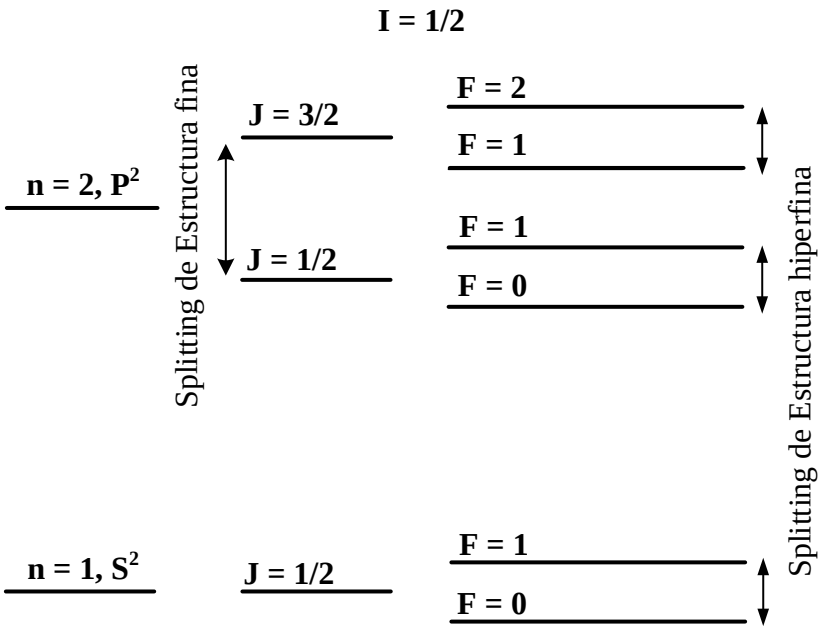


Figura 2.9: Diagrama de estructura fina e hiperfina del átomo de hidrogeno

Para aumentar la resolución de un espectrómetro es posible disminuir el ancho de banda o aumentar el número de canales espectrales, pero la primera práctica no se utiliza ya que junto con una buena resolución siempre se busca tener un ancho de banda instantáneo considerable, para poder observar al mismo tiempo nubes moleculares de gran tamaño. El caso más crucial en el que se necesita tener un gran ancho de banda y buena resolución espectral es cuando se quiere observar la dinámica fina de una nube molecular con un gran rango de velocidades, tal y como el centro galáctico.

2.5 IMPLEMENTACIÓN DIGITAL

Para implementar de forma digital espectrómetros de Fourier de alta resolución con un ancho de banda considerable, se hace necesario un alto poder de procesamiento en paralelo, además de una alta velocidad de muestreo. El estado del arte de este tipo de espectrómetros consiste en utilizar ADCs de alta velocidad y un FPGA para implementar la FFT.

2.5.1 FPGA

Un FPGA es un dispositivo semiconductor que contiene una matriz de bloques lógicos cuya interconexión y funcionalidad puede ser configurada mediante un lenguaje de descripción especializado. Los FPGAs pueden ser reprogramadas para distintas aplicaciones después de su construcción, característica que los distingue de los circuitos integrados de aplicación específica (ASICs), los cuales son diseñados y construidos para tareas particulares.

Una tendencia reciente en el diseño de FPGAs ha sido combinar los bloques lógicos con microprocesadores y periféricos para formar un "sistema programable". En la mayoría de los FPGA

se pueden encontrar también funciones de alto nivel como sumadores o multiplicadores embebidas en la matriz de interconexiones, así como bloques de memoria.

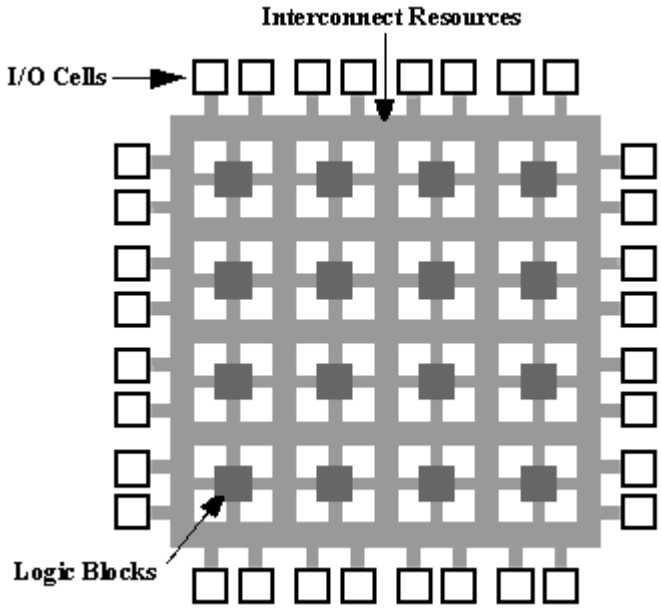


Figura 2.10: Diagrama de arquitectura interna de un FPGA.[9]

La programación de un FPGA no es como en otros dispositivos tales como computadores o microcontroladores, ya que en realidad se estará describiendo el hardware que tendrá el FPGA. Es por esto que la tarea del programador es describir la función lógica que tendrá cada uno de los bloques de la matriz del FPGA. Para programar estos dispositivos se utilizan lenguajes conocidos como HDL (*Hardware Description Language*), siendo los más utilizados los siguientes:

- VHDL
- Verilog
- ABEL

La gran ventaja de utilizar un FPGA para implementar espectrómetros digitales es su gran capacidad de procesamiento en paralelo, dando la posibilidad de procesar grandes anchos de banda en tiempo real. Además, el uso de un FPGA permite desarrollar y probar distintos prototipos en un mismo *chip*, facilitando el proceso de construcción y optimización. También abarata los costos de producción, debido al uso de componentes comerciales.

2.5.2 ADC

Un ADC es un dispositivo que convierte una señal analógica a una señal digital, es decir números binarios que representan amplitudes (generalmente de voltaje). La digitalización de una señal requiere de tres pasos: muestreo, cuantificación (o cuantización) y codificación.

[9] Figura extraída de http://es.wikipedia.org/wiki/Field_Programmable_Gate_Array

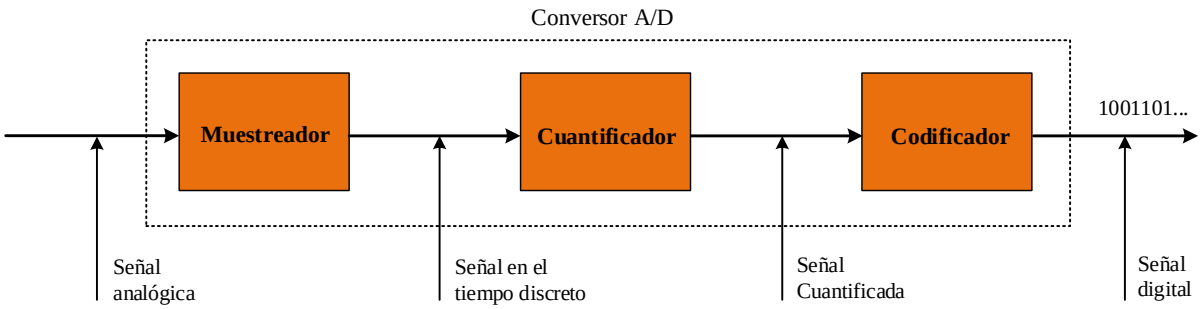


Figura 2.11: Proceso de digitalización de un ADC

- Muestreo: consiste en tomar medidas de amplitud de la señal analógica de forma periódica. La señal muestreada corresponde a una señal discreta en el tiempo.
- Cuantificación: consiste en la asignación de un nivel de salida único a las muestras dentro de un rango de voltaje (discretización en voltaje o nivel).
- Codificación: consiste en la asignación de códigos binarios a los niveles obtenidos en la cuantificación.

El número de niveles disponibles en la cuantificación de la señal entregarán la resolución del ADC y dependerá del número de bits de éste. El número de niveles es igual a 2^n , con *n* el número de bits. Para digitalizar señales de alta frecuencia (más de 1 GHz) se necesitan ADCs que puedan muestrear al doble de esa frecuencia. Estos se conocen como ADCs de alta velocidad y en general poseen varios núcleos que muestrean en forma intercalada.

Para caracterizar el desempeño de ADCs se utilizan medidas tales como su respuesta en frecuencia, el SFDR (*Spurius Free Dinamic Range),* el SINAD (*Signal-to-noise and distortion ratio)* y el ENOB (*Effective number of bits)* [14].

SFDR (*Spurius Free Dinamic Range)*

El SFDR de un ADC con una señal sinusoidal pura de amplitud y frecuencia conocida como entrada, se define como el radio de la amplitud promedio del componente espectral a la frecuencia de entrada sobre la amplitud del componente de ruido o armónico más grande medido en la primera zona de Nyquist. [14]. La expresión para calcular el SFDR se presenta a continuación.

$$SFDR\ (dB) = 20log_{10}\left(\frac{A_{input}}{\max(f_h, f_s)}\right) \qquad (2.\ 13)$$

donde f_h y f_s corresponden a las frecuencias del armónico y espurio de mayor amplitud respectivamente.

El SFDR puede ser expresado relativo a la amplitud de la señal (dBc) o a la amplitud de escala completa del ADC (dBFS). Esta representación se puede apreciar en la Figura 2.12.

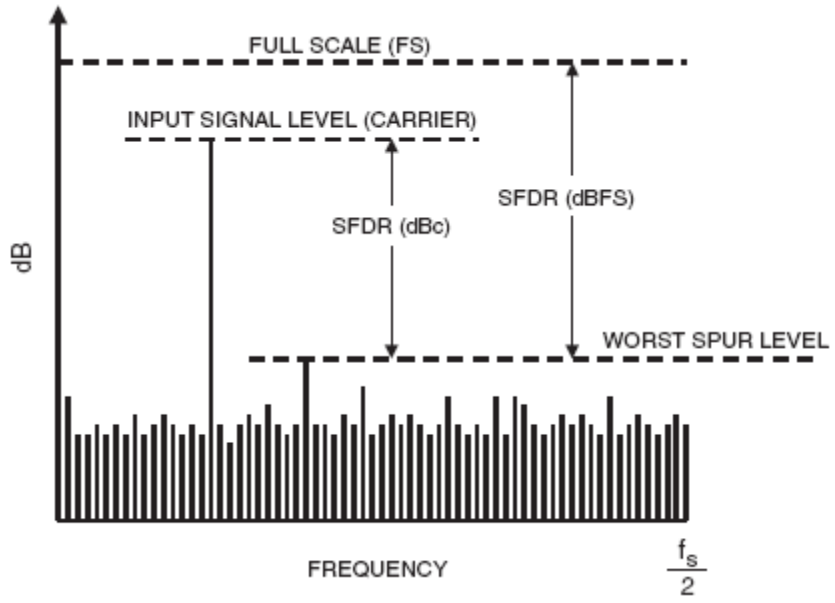


Figura 2.12: SFDR (Spurius Free Dinamic Range).[11]

SINAD (*Signal-to-noise and distortion ratio)*

Corresponde a una medida de la calidad de la señal de salida del ADC. Se define como el radio entre la amplitud rms (*root-mean-square)* de la salida del ADC y la amplitud rms del ruido en la salida (NAD). Este último incluye errores aleatorios, distorsiones no lineales y efectos producidos por errores de sincronización en el muestreo [14]. La expresión para calcular el SINAD se muestra a continuación.

$$SINAD = \frac{A_{rms}}{\text{NAD}} \tag{2.14}$$

Donde,

$$A_{rms} = \frac{Amplitud\ del\ máximo\ de\ la\ salida\ del\ ADC}{\sqrt{2}}$$

Para calcular el NAD se requiere muestrear con el ADC una onda sinusoidal de frecuencia fundamental conocida y además encontrar la mejor onda sinusoidal que se adecue a los datos muestreados. De esta forma se tiene que,

$$NAD = \sqrt{\frac{1}{M}\sum_{n=1}^{M}(\,x[n] - x'[n]\,)^2} \tag{2.15}$$

donde,

[11] Figura extraída de ww.eetimes.com

$x[n]$ = datos reales en la salida del ADC
$x'[n]$ = datos de la mejor onda sinusoidal que se adecua a $x[n]$
M = número de muestras

ENOB (*Effective number of bits*)

El número efectivo de bits (ENOB) es una medida del SINAD, utilizada para comparar el desempeño de un ADC con el de un ADC ideal [14]. Corresponde al número de bits utilizados por el ADC que efectivamente corresponden a información de la señal muestreada y no ruido. Un ADC ideal tiene un ENOB igual a su resolución.

$$ENOB = \frac{SINAD - 1.76}{6.02} \tag{2.16}$$

CAPÍTULO 3

IMPLEMENTACIÓN

Para implementar un espectrómetro digital de alta resolución, con un ancho de banda superior a 1,5 GHz, se hace necesaria la utilización de tecnología que permita el procesamiento en paralelo de señales y de conversores análogo-digital (ADCs) de alta velocidad, es decir con una alta tasa de muestreo. El teorema de Nyquist-Shannon indica que la frecuencia de muestreo del ADC debe ser al menos el doble del ancho de banda de la señal que se quiere reconstruir digitalmente. Dicho esto, para cumplir con los objetivos de diseño se utilizó como hardware principal una plataforma basada en arreglos de compuertas lógicas programables (FPGA).

3.1 HARDWARE

El hardware utilizado consiste en la segunda versión de una plataforma abierta basada en FPGA llamada ROACH[12] (*Reconfigurable Open Arquitecture Computing Hardware*). Este producto fue desarrollado por una colaboración internacional encabezada por el Centro de Procesamiento de Señales Astronómicas de la Universidad de California Berkeley, conocido como CASPER [13] (*Center for Astrononomy Signal Processing and Electronic Research*).

3.1.1 ROACH 2

ROACH 2 posee un diseño similar a la versión anterior de este hardware (ROACH 1), con la gran diferencia de que utiliza un chip FPGA Xilinx Virtex 6, varias veces más poderoso que su predecesor, Virtex 5. En paralelo, un procesador (chip PowerPC) corre Linux y es utilizado para controlar todo el sistema, permitiendo programar el FPGA y hacer la interface entre los registros de software del FPGA/BRAMs/FIFOs y dispositivos externos a través de una conexión Ethernet.

La memoria de la plataforma está compuesta por cuatro QDR (*quad data rate)* SRAMs de 72 Mb, las que proveen capacidad media de memoria de alta velocidad y una DDR3 RDIMM (SDRAM) de hasta 16 GB que provee de memoria de más baja velocidad pero de alta capacidad. El procesador PowerPC tiene una memoria DRAM propia para ejecutar los

[12] https://casper.berkeley.edu/wiki/ROACH-2_Revision_2
[13] https://casper.berkeley.edu/

procesos y una versión especial de Linux llamada BORPH[14] (*Berkeley Operating system for ReProgrammable Hardware*).

BORPH es un sistema operativo diseñado para sistemas reconfigurables basados en FPGA. Corresponde a una extensión de un Linux *kernel* que maneja FPGAs como si fueran CPUs. BORPH introduce el concepto de un "*hardware process*", que consiste en un diseño de hardware (circuito digital) que corre en un FPGA pero que se comporta de la misma forma que un programa de usuario normal.

La ROACH 2, al igual que su versión anterior, posee dos ranuras (*Z-DOK connectors)*, que permiten conectar tarjetas tales como ADCs y DACs al FPGA. En la Figura 3.1 se muestra el diagrama de bloques de la ROACH 2.

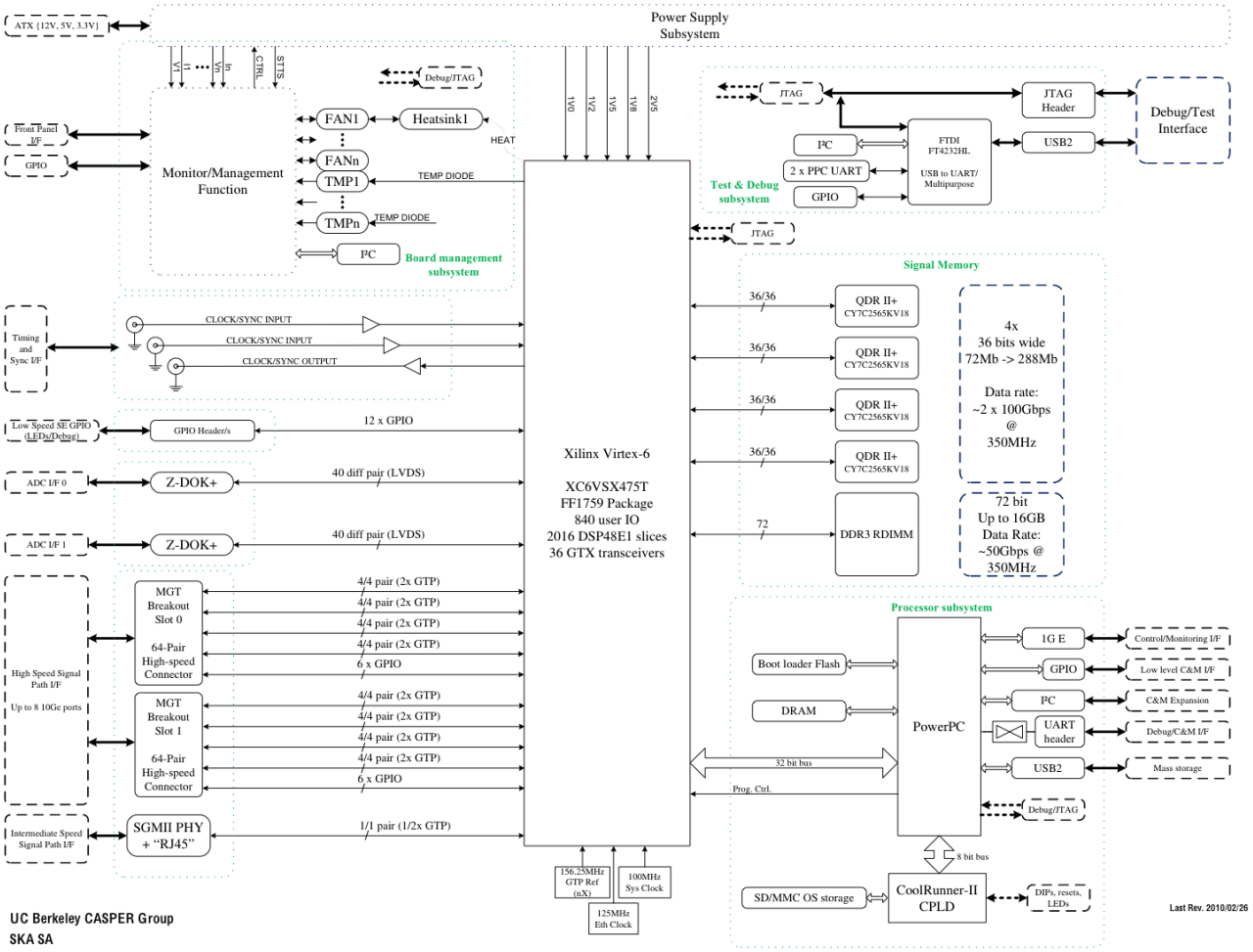


Figura 3.1: Diagrama de bloques de la ROACH 2.[15]

[14] https://casper.berkeley.edu/wiki/BORPH
[15] Figura extraída de https://casper.berkeley.edu/wiki/ROACH-2_Revision_2

Los principales recursos disponibles en el FPGA Virtex 6 consisten en los que proveen de memoria y de funciones de DSP (*digital signal processing)* a los diseños, tales como multiplicación y suma. La unidad básica de memoria del chip consiste en un 'Block RAM', el cual puede ser utilizado para el almacenamiento eficiente de datos, para implementar máquinas de estado, registros, *look-up tables* y ROMs. Una BRAM puede almacenar hasta 36 kbits y puede ser configurada como dos RAMs independientes de 18 kbits o como una RAM de 36 kbits.

El chip Virtex 6 posee una unidad de lógica destinada a las operaciones DSP, llamada DSP48E1. Esta puede implementar variadas funciones independientes, tales como multiplicación, multiplicación con acumulación (MACC), suma de hasta 3 entradas, desplazamientos de bits (*shitfs)*, multiplexación, comparador de magnitud y otras. La arquitectura que se tiene también permite utilizar multiples DSP48E1 en cascada para implementar funciones matemáticas complejas, filtros y aritmética de números complejos sin utilizar la lógica de propósito general del FPGA. Esta última corresponde a los llamados 'slices', los cuales consisten en compuertas lógicas que se pueden programar como variadas compuertas combinacionales básicas.

3.1.2 ADC 5 GSps

La tarjeta ADC seleccionada fue desarrollada por ASIAA (*Academia Sinica Institute for Astronomy and Astrophysics*) y es compatible con plataformas de CASPER. Está basada en el circuito integrado EV8AQ160 de e2v [15], el cual es un dispositivo de cuatro núcleos. Cada uno de estos corresponde a un ADC de 8 bits de 1,25 GSps, lo que permite 256 niveles en la amplitud de la señal muestreada y una tasa de muestreo de hasta 5 GSps al muestrear de forma intercalada una sola señal de entrada. Esta propiedad hace necesaria una calibración de los núcleos del ADC para poder utilizarlo. Este chip permite, además, seleccionar la desmultiplexación de las salidas (1:1 o 1:2), pero la tarjeta fija esta opción, generando dos versiones de hardware. Para este proyecto se tiene la versión *dmux* 1:1, en la cual se generan muestras de 8 bits.

El ADC posee una interface SPI (*Serial Peripheral Interface*), mediante la cual es posible escribir registros en el circuito integrado, los cuales son utilizados para fijar correcciones de ganancia, *offset*, fase y otros. La ganancia posee un rango de control de $\pm 18\%$, el *offset* de $\pm 50\ mV$ y la fase de $(\pm 14\ ps)$. En la Figura 3.2 se muestra el diagrama interno del chip ADC.

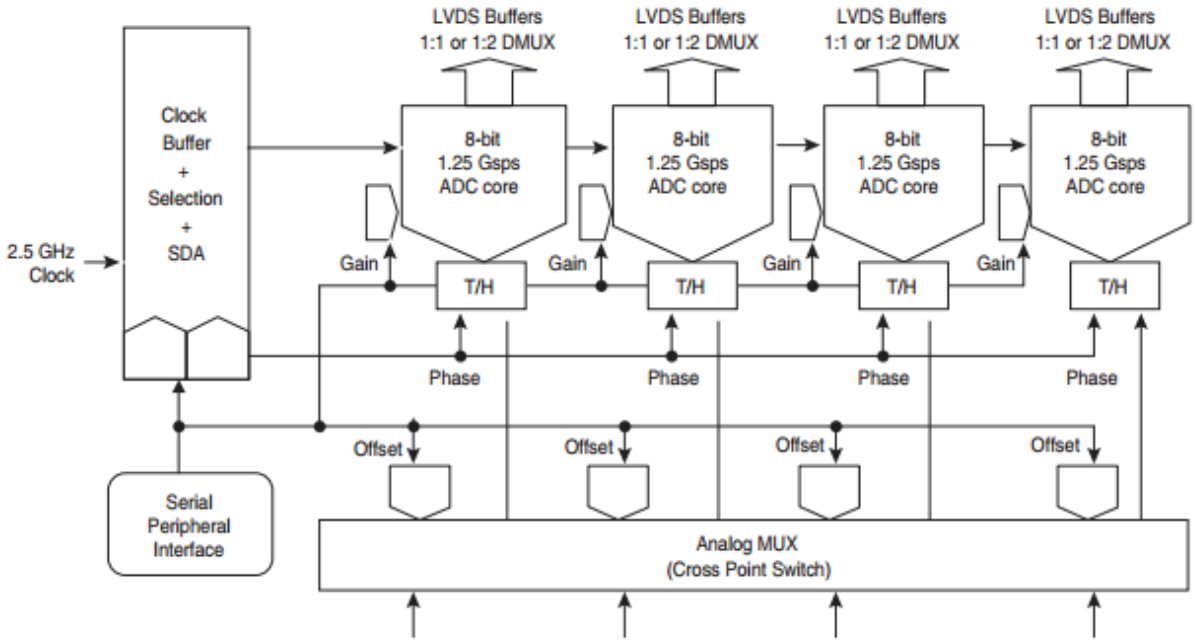


Figura 3.2: Diagrama de bloques interno del chip ADC 5 GSps.[17]

Los cuatro núcleos del ADC utilizan la misma señal externa de reloj, la que puede ser de una frecuencia máxima de 2,5 GHz. Para generar esta señal se utiliza un sintetizador de reloj externo. El muestreo sucede en los flancos de subida y bajada de dicha señal, generando una tasa de muestreo de 5 GSps. El reloj es dividido por 8 antes de entrar en el FPGA, generando 16 muestras paralelas (de 8 bits cada una) cuando el ADC se encuentra en modo de un canal de entrada.

El chip ADC puede trabajar en modo de 1 canal de entrada, 2 canales y 4 canales, pero sólo las dos primeras opciones están disponibles en la PCB utilizada, ya que esta tiene sólo 2 entradas SMA[19] independientes. Cuando se utilizan las dos entradas, se puede considerar que se tienen dos ADCs independientes, cada uno funcionando con 2 núcleos muestreando intercaladamente. Esto genera que cada uno de estos ADC pueda tener una frecuencia de muestreo máxima de 2,5 GSps y 8 muestras paralelas simultáneas. Para la realización de este trabajo, se utilizará el ADC en modo de una entrada solamente. En la Figura 3.3 a continuación, se muestra una fotografía de la ROACH 2 utilizada, con dos ACDS 5 GSps conectados en los conectores ZDOK.

[17] Figura extraída del *datasheet* del ADC [15]
[19] http://es.wikipedia.org/wiki/SMA_%28conector%29

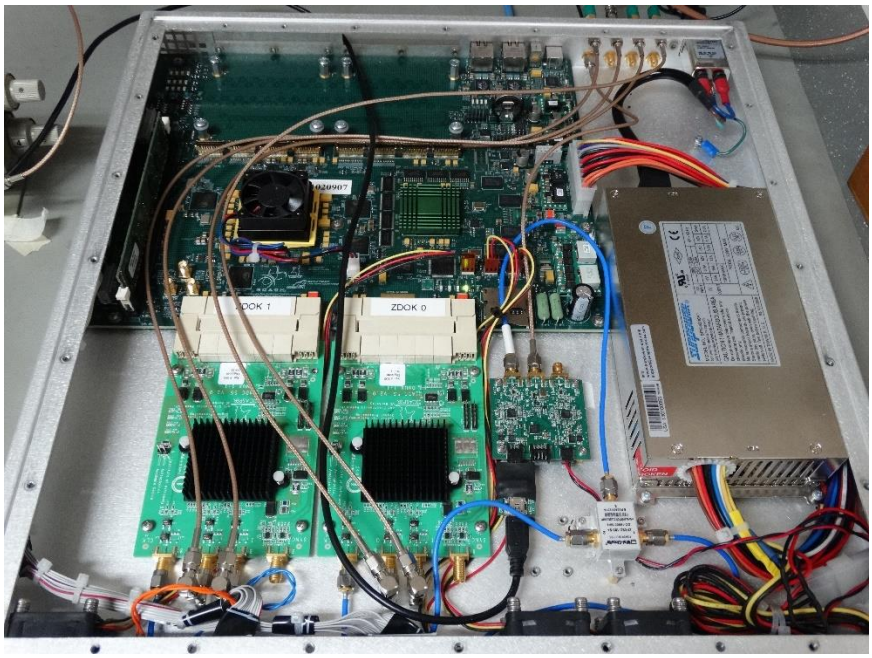


Figura 3.3: ROACH 2 con dos ADCs 5 GSps conectados en los conectores ZDOK

3.1.3 Sintetizador de reloj

Para generar la señal de reloj del ADC se utiliza un módulo Valon 5007[20], el cual genera una señal de referencia de alta calidad, de frecuencias entre 137,5 y 4000 MHz. Este sintetizador posee una memoria *FLASH* que permite que la programación de éste sea no volátil, es decir guarda las opciones escogidas después de apagar el módulo.

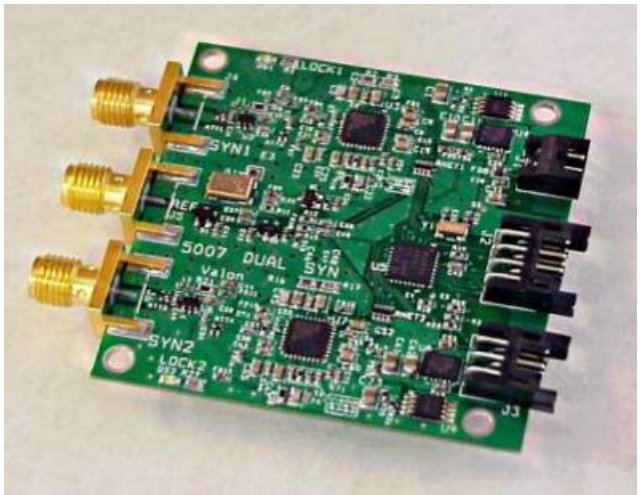


Figura 3.4: Sintetizador de señal de reloj Valon 5007

[20] http://www.valontechnology.com/5007%20synthesizer.html

3.2 CALIBRACIÓN ADC

El circuito integrado del ADC ASIAA 5 GSps posee registros de control para alinear los cuatro núcleos y así reducir el impacto de espurios que podrían surgir producto del desalineamiento en *offset*, ganancia y fase (OGP) entre ellos, o también debido a su no linealidad (INL). También es necesario realizar una calibración en la interface entre el ADC y el FPGA para prevenir *glitches* en la transferencia de datos.

Para implementar todas las calibraciones del ADC mencionadas anteriormente se escribieron códigos en Python, utilizando librerías y funciones proporcionadas en repositorios de CASPER. Estos programas se corren desde un computador Linux, el cual se comunica vía Ethernet con la ROACH 2, la cual a su vez envía los comandos correspondientes al ADC de forma serial. En la Figura 3.5 se presenta un diagrama del *setup* general utilizado para la calibración y pruebas del ADC.

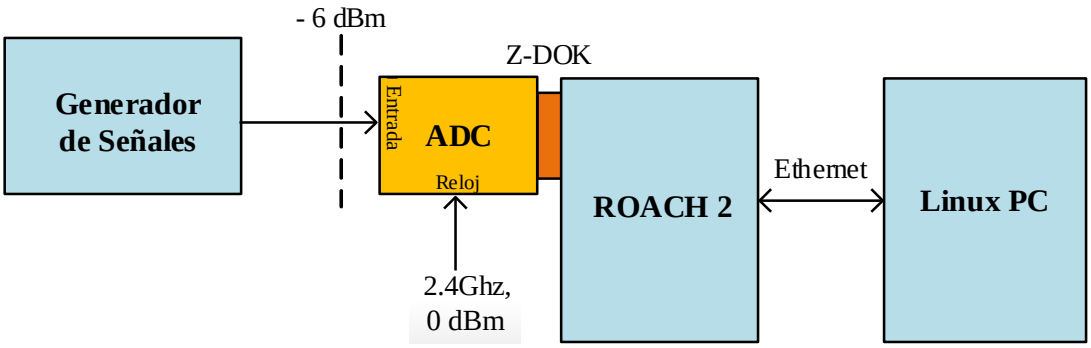


Figura 3.5: *Setup* de calibración y prueba del ADC.

3.2.1 Calibración OGP

El chip del ADC posee un circuito que genera las señales de reloj de cada uno de los cuatro núcleos utilizando una señal de reloj externa de hasta 2,5 GHz. Al trabajar con un canal de entrada, una señal de reloj en fase de 1,25 GHz es enviada al primer núcleo, denominado “ADC A”, una señal invertida en fase es enviada al ADC B y señales desfasadas en 90° y 270° son enviadas a los núcleos C y D respectivamente. La Figura 3.6 muestra un esquema de este modo de operación, el cual genera una frecuencia de muestreo equivalente de 5 GSps.

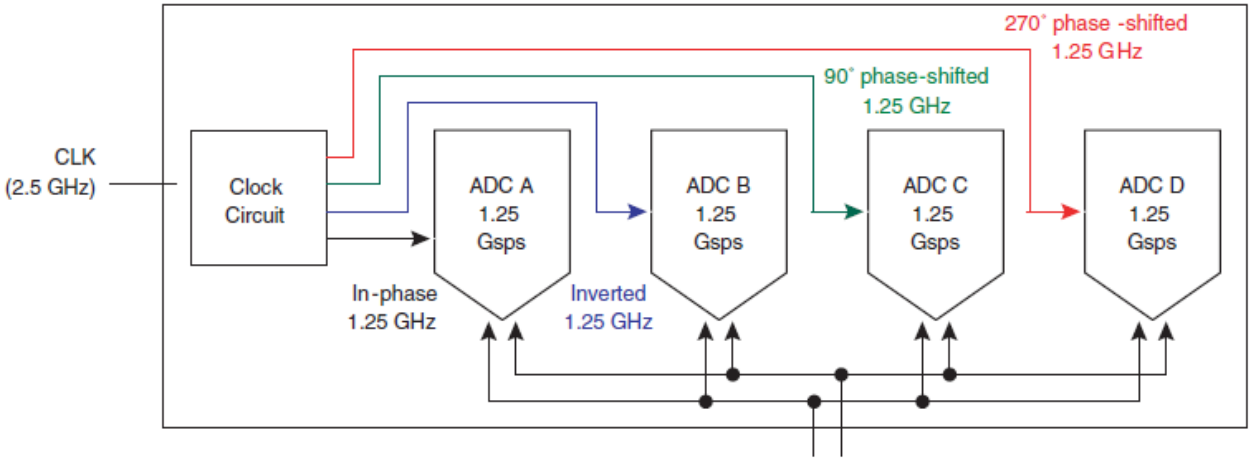


Figura 3.6: Esquema de señal de reloj del ADC funcionando con un canal de entrada.[21]

Si se toma un *snapshot* (valores medidos por el ADC) de N muestras, las muestras de los núcleos A, B, C y D tendrán las siguientes secuencias:

A: N, N + 4, N + 8…
B: N + 1, N + 5, N + 9…
C: N + 2, N + 6, N + 10…
D: N + 3, N + 7, N + 11…

Dado el modo en que muestrean los núcleos del ADC, el más mínimo desfase de alguno de ellos en el muestreo o diferencias de *offset* o ganancia generarán un error en la toma de datos. El objetivo de la calibración OGP (*offset-gain-phase*) es obtener coeficientes de corrección de *offset*, ganancia y fase para cada uno de los 4 núcleos del ADC y así hacer que queden alineados para que muestreen de forma intercalada como si fueran el mismo ADC. Los coeficientes son transferidos a los registros SPI del ADC, los cuales son volátiles, por lo que se requiere cargarlos cada vez que se encienda la ROACH 2.

El método utilizado para obtener los coeficientes de corrección fue tomado del artículo de la comunidad CASPER de N.A. Patel [16]. Una onda sinusoidal de frecuencia conocida es ajustada usando mínimos cuadrados a la salida de cada núcleo. El valor de *offset* sin corrección es aproximadamente 127 (de 256 niveles), el cual es tomado como referencia. Los valores de referencia de ganancia y fase son obtenidos del promedio de las ondas sinusoidales ajustadas a cada uno de los 4 núcleos. Las correcciones en *offset*, ganancia y fase de cada núcleo se obtienen tomando la diferencia de dichos parámetros de la señal ajustada y los valores de referencia, indicados anteriormente. Haciendo este procedimiento varias veces se converge a los coeficientes de corrección finales (3 para cada núcleo), los cuales se cargan en los registros del ADC (escalados apropiadamente). Es importante destacar que la calibración OGP depende de varios factores, tales como la temperatura del ADC y la frecuencia del reloj externo.

[21] Figura extraida del *datasheet* del ADC [15]

3.2.2 Calibración INL

A cada voltaje de entrada en el ADC, le corresponderá una palabra binaria en la salida de éste. El número de palabras digitales disponibles (pasos o niveles) para representar la entrada analógica depende del número de bits con que muestrea el ADC y será igual a 2^n, siendo *n* el número de bits de las muestras del ADC. Para un ADC ideal, la relación entre la señal de entrada y su representación digital es una función "escalera" que puede ser aproximada de forma exacta por una recta. Esta relación es conocida como la función de transferencia del ADC. En la Figura 3.7, se presenta la función de transferencia de un ADC ideal de 12 bits, en la que el ancho de los pasos son exactamente iguales a 1 LSB, que corresponde al valor de voltaje que representa el bit menos significativo ($1\ LSB = V_{FSR}/2^n$, donde V_{FSR} es igual al rango de voltaje de entrada del ADC y *n* al número de bits del ADC) [17].

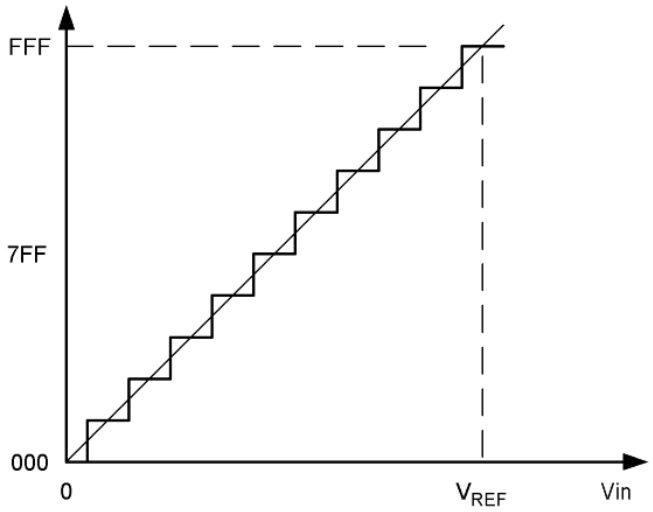


Figura 3.7: Función de transferencia de ADC ideal.[22]

En un ADC real, sobre todo en uno de alta velocidad, aparecen no linealidades entre los pasos de cuantización. Para expresar esta no-linealidad se define la mejor recta que se adecua a la función de transferencia del ADC. La máxima desviación entre dicha recta y la línea de un ADC ideal se conoce como INL, la cual se mide en LSB. En la Figura 3.8 se presenta la función de transferencia de un ADC de 12 bits con una pequeña no-linealidad.

[22] Figura extraída www.masteringelectronicsdesign.com [17]

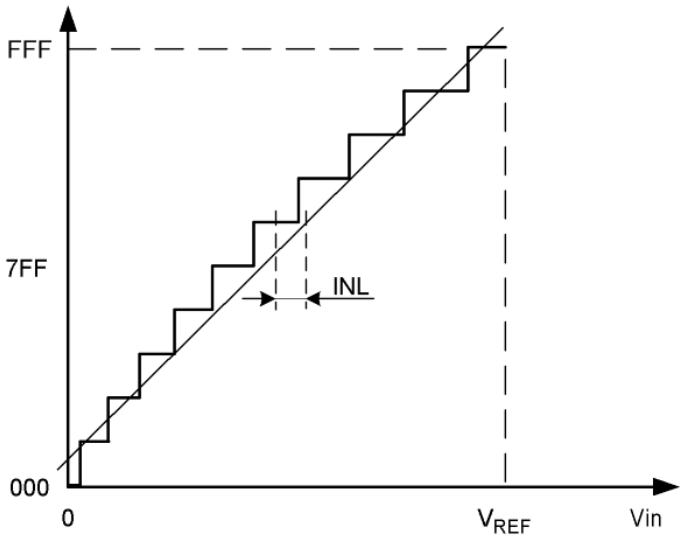


Figura 3.8: Función de transferencia de ADC real[23]

Para compensar esta no-linealidad presente en los ADC, se realiza una calibración, la cual consiste en encontrar coeficientes de corrección para cada uno de los 17 puntos de corrección INL disponibles de cada núcleo del ADC. Estos puntos se encuentran distribuidos uniformemente a través de los 256 niveles de cuantización del ADC. Entonces, cuando se encuentra un máximo INL ($\pm$ 0.5 LSB) alrededor de un nivel especifico, es posible reducirlo en $\pm$ 0.15 LSB o $\pm$ 0.45 LSB escribiendo un "1" en el registro correspondiente. De ser necesario es posible realizar correcciones de $\pm$ 0.3 LSB o $\pm$ 0.6 LSB combinando las correcciones disponibles [15].

Para encontrar los coeficientes de corrección, se implementó un programa en Python utilizando las librerías y funciones disponibles. Éste sigue los siguientes pasos [16]:

- Utilizando una señal de entrada de prueba, se promedia la diferencia entre el valor medido por el ADC (*snapshot*) y el valor obtenido haciendo una aproximación de mínimos cuadrados de la misma señal medida. Esto se hace para cada uno de los 256 niveles de cada núcleo del ADC.
- El proceso anterior se repite usando múltiples *snapshots* de la señal de prueba.
- Cada uno de los 17 coeficientes de corrección se calcula como el promedio ponderado de las diferencias obtenidas en los 31 niveles más cercanos a dicho nivel (15 niveles hacia cada lado), con un coeficiente de ponderación (peso) inversamente proporcional a la distancia al nivel de corrección.

3.2.3 Calibración de interface ADC - FPGA

Esta calibración tiene el objetivo de prevenir *glitches* en la transferencia de datos desde el ADC al FPGA, los cuales corresponden a transiciones indeseadas en la señal (errores de un bit). El FPGA Virtex 6 posee un componente llamado MMCM (*Mixed-Mode Clock Manager*), el cual funciona como sintetizador de reloj, es decir se encarga de generar la señal de reloj del FPGA utilizando como referencia el reloj proveniente del ADC. El reloj del FPGA (*MMCM clk*) tiene una frecuencia

[23] Figura extraída www.masteringelectronicsdesign.com [17]

igual a 1/8 de la frecuencia del reloj del ADC en la configuración utilizada, esto ya que el FPGA no puede lograr la velocidad del ADC, lo que compensa con el procesamiento paralelo de 16 muestras temporales.

Los *glitches* en los datos se producen cuando existe un retraso entre el reloj del ADC y el reloj MMCM. Éste último posee un atributo que permite retrasar o adelantar la señal de reloj generada, siendo posible hacerlo en incrementos de $1/56$ de T_{VCO}, período que depende de la configuración del MMCM.

Para realizar la calibración se utiliza una función de Python, la cual produce un ciclo en el cual va cambiando la fase del reloj MMCM (pasando por los 56 pasos posibles) y encuentra el número total de *glitches* en el vector de prueba para cada núcleo, en cado paso. Entonces encuentra la fase con menor cantidad de *glitches* y la fija escribiéndola en el registro disponible del FPGA . El vector de prueba corresponde a una función rampa conocida, la cual es generada en el ADC al ponerlo en '*test mode'*. Esta es una funcionalidad propia del chip ADC.

Es importante tomar en cuenta que esta calibración puede variar para distintos diseños cargados en el FPGA, ya que podría cambiar la sincronización entre la señal de reloj del ADC y el FPGA, necesitando una corrección de fase distinta en el reloj producido por el MMCM.

3.2.4 Metodología de calibración y consideraciones de uso

Utilizando las tres calibraciones descritas anteriormente se implementó un programa en Python de calibración automática del ADC, el cual posee parámetros para escoger la frecuencia de reloj del ADC, la frecuencia de la señal de prueba y el conector de ROACH 2 donde se encuentra el ADC que se desea calibrar. Para utilizar el programa es necesario usar un modelo de ROACH 2 donde se obtengan *snapshots* de las muestras del ADC. Este programa implementa la calibración en el siguiente orden:

1. Calibración de interface ADC - FPGA
2. Calibración OGP
3. Calibración INL

Para realizar la calibración es necesario que el ADC esté funcionando a escala completa, es decir que la potencia de entrada de la señal de prueba sea suficiente para que la cuantización utilice los 8 bits (256 pasos) disponibles, o al menos lo más cercano posible a dicho valor. Es importante también que no se produzca saturación, es decir que la potencia de entrada sea tal que los 8 bits se llenen con 1s. Para verificar de manera exacta el nivel de entrada de los ADCs, se implementó un diseño digital en ROACH 2, el cual junto con un programa de Python entrega el porcentaje del nivel máximo del ADC. En la Figura 3.9 se muestra el diseño de alto nivel de dicho circuito, el que incluye también un bloque de *snapshot*, utilizado para graficar en tiempo real la onda sinusoidal muestreada por el ADC.

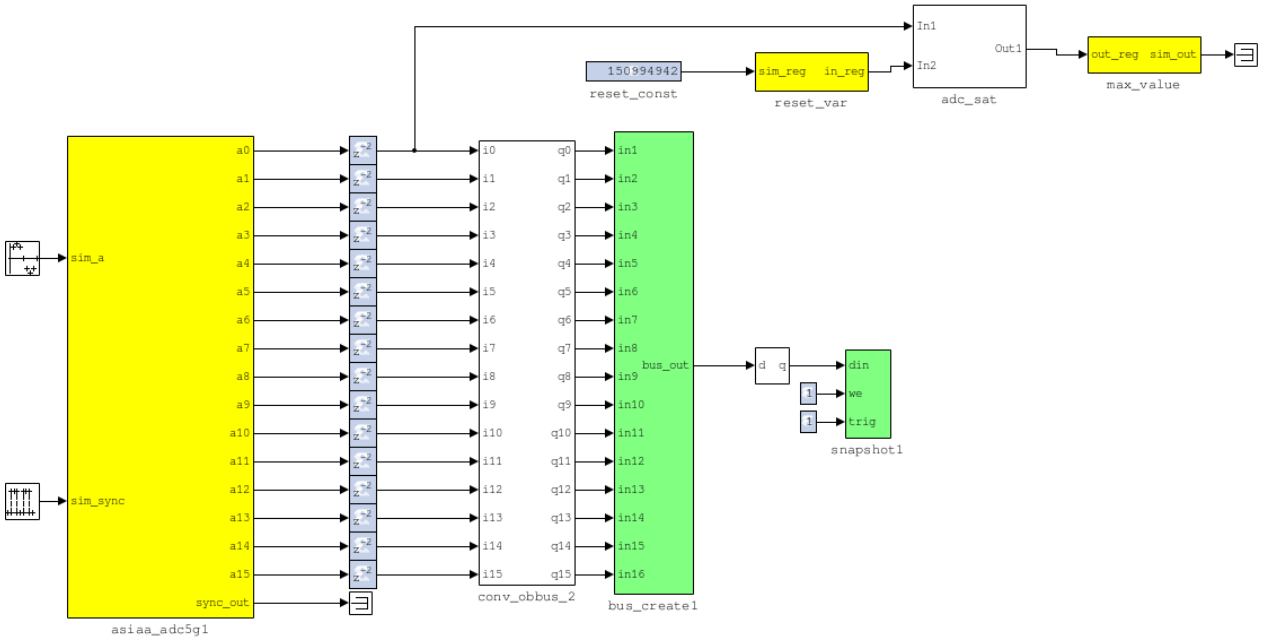


Figura 3.9: Circuito de medición del nivel máximo del ADC

Para medir el nivel del ADC el circuito guarda en un registro el mayor valor registrado en el muestreo. Esto sucede en el bloque del circuito llamado "adc_sat". En la Figura 3.10 se presenta el circuito interno de dicho bloque. Luego en un programa Python se lee este valor y se le asocia un porcentaje. En este mismo programa se va reseteando continuamente el registro para así poder detectar cuando el porcentaje del nivel del ADC baja.

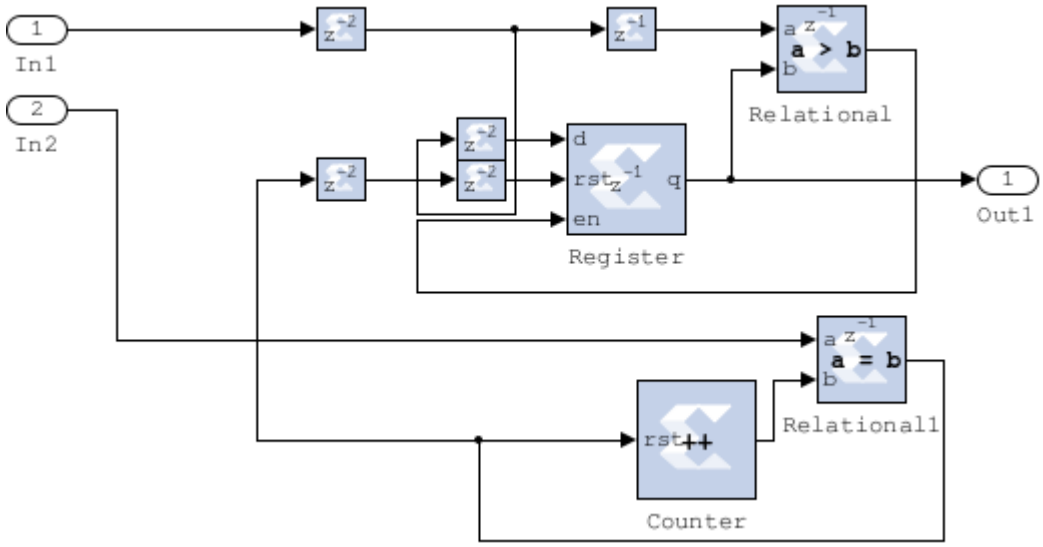


Figura 3.10: Circuito que registra el nivel máximo del ADC

Debido a que los registros OGP e INL que se escriben en el ADC son volátiles, es necesario realizar la calibración cada vez que se encienda la ROACH 2. Además, la señal de reloj del FPGA producida por el MMCM puede cambiar de fase para cada diseño, o incluso para un mismo diseño si es que se le aplica optimización a nivel físico usando el programa 'PlanAhead'. Esto hace que sea necesario incluir una rama de '*snapshot*' en cualquier diseño que se implemente en el FPGA

para realizar una calibración de la interface al comienzo del uso de cada diseño (utilizando el diseño que se va a probar para esto). La calibración OGP depende fuertemente de la frecuencia de reloj del ADC, por lo tanto es necesario calibrar a la frecuencia a la que se utilizará posteriormente.

3.3 ESPECTRÓMETRO DE ALTA RESOLUCIÓN

El primer paso en el diseño del espectrómetro fue implementar un espectrómetro simple de transformada de Fourier rápida (FFTS) en ROACH 2, el cual se basó en el espectrómetro de separación de banda lateral (Ricardo Finger, 2013) implementado en ROACH 1, con algunas modificaciones. La Figura 3.11 muestra el diseño de alto nivel del espectrómetro en ROACH 2, implementado en el *software* Simulink.

El primer bloque amarillo de la izquierda corresponde al ADC ASIAA 5GSps, el cual produce 16 muestras paralelas de 8 bits, seguido de un bloque que convierte las muestras de formato binario *offset* a binario con signo, antes de procesarlas en el FPGA. El bloque verde llamado "snapshot", se utiliza para obtener los datos muestreados antes de ser procesados, los que se utilizan para la tercera etapa de calibración del ADC (interface con el FPGA), la cual debe realizarse con el mismo diseño de cada espectrómetro.

La parte más importante del espectrómetro es el *Polyphase Fir Filter Bank (PFB)*, conformado por un filtro FIR y una transformada de Fourier rápida con *pipeline*, ambos representados por los bloques verdes del centro del diseño. La FFT utilizada computa muestras reales (sin parte imaginaria) e implementa un algoritmo Radix-2 con *pipeline*[24], que permite procesar los datos de manera secuencial, permitiendo un procesamiento en tiempo real mucho más rápido que el algoritmo Radix-2 regular utilizado para implementar la FFT [4].

Del bloque FFT, salen 8 muestras paralelas en el dominio de la frecuencia, las que entran en los bloques "power_dsp", los que calculan la potencia espectral de cada muestra usando bloques DSP48E. Seguido de estos se tiene un bloque de recuantización, utilizado para el control de amplitud y para aumentar el número de bits de los canales espectrales y así prevenir *overflow*.

Para la integración y almacenamiento del espectro se utilizan bloques "bram_vacc" y memorias BRAM (*Xilinx Block RAM*) respectivamente. Se utiliza además un bloque de control de datos que entrega un pulso a los bloques de acumulación para habilitar la escritura en ellos, esto con el objetivo de controlar la escritura y lectura de las memorias desde un PC y prevenir colisiones en el acceso o corrupción de los datos.

[24] Bin Zhou, et al. (2009). Pipeline FFT Architectures Optimized for FPGAs.

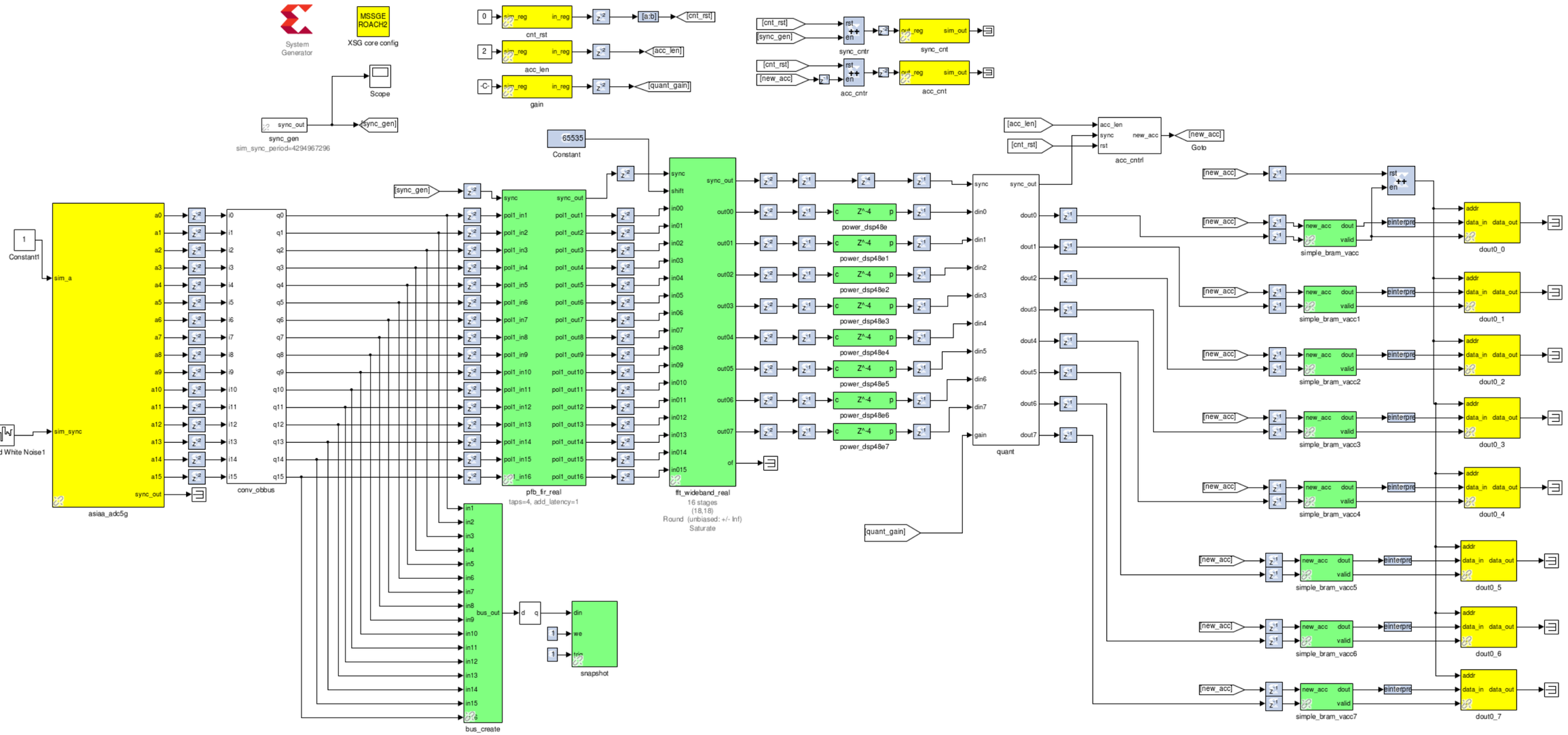


Figura 3.11: Diseño de alto nivel de espectrómetro en ROACH 2

La respuesta de un filtro FIR a una entrada finita tiene una duración finita, por lo tanto la salida *y[n]* se hace cero en un tiempo finito después de que la entrada *x[n]* se hace cero. La salida *y[n]* de un filtro FIR es la suma ponderada de la entrada actual y las pasadas. El orden del filtro es el número de muestras temporales que son usadas por el filtro para calcular la salida [4]. Un filtro de 4to orden es utilizado en el diseño del espectrómetro, el cual mejora la respuesta en frecuencia de cada canal espectral, haciéndola más plana y aguda en los bordes, además de reducir fuertemente la filtración de potencia espectral entre canales. En la Figura 3.12 se puede apreciar los beneficios en la respuesta en frecuencia de un canal espectral al aplicar esta técnica.

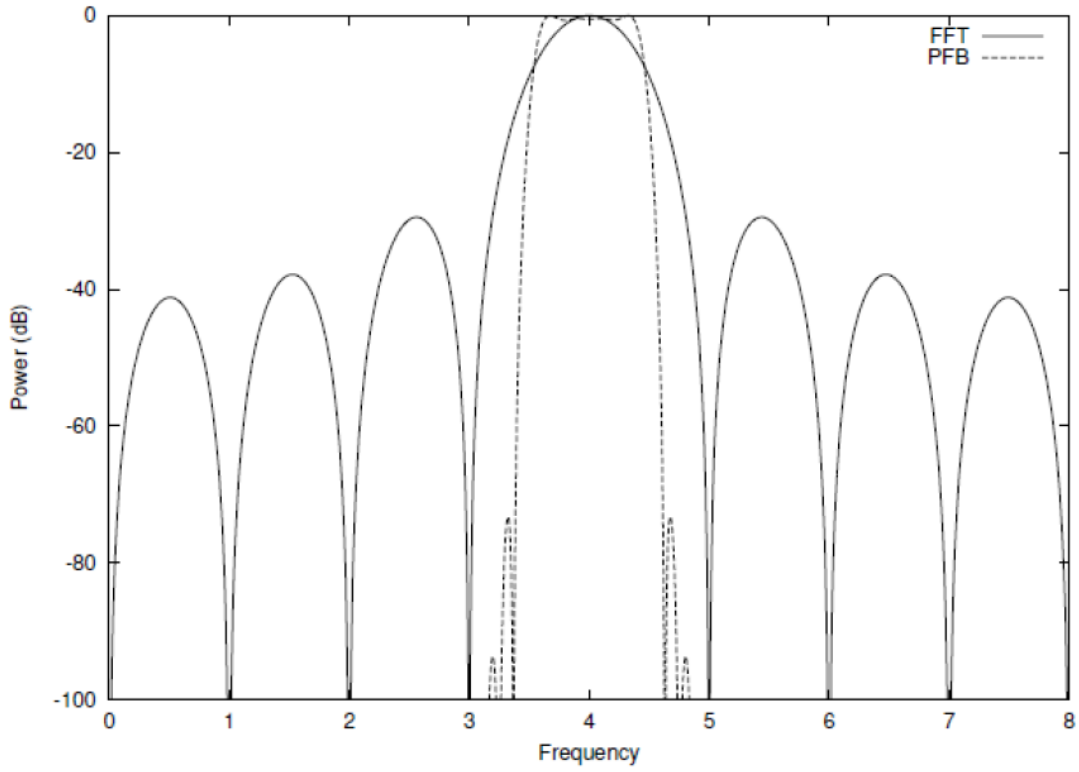


Figura 3.12: Comparación de la respuesta en frecuencia de un canal espectral de un PFB de 4to orden con una FFT directa.[25]

3.3.1 Aumento de resolución espectral

La metodología de diseño que se siguió para aumentar la resolución espectral del espectrómetro en ROACH 2 fue ir aumentando progresivamente el tamaño de la FFT para un ancho de banda no menor a 1,5 GHz. El número de canales espectrales será igual a la mitad del tamaño de la FFT. Como la FFT utilizada usa un algoritmo Radix-2 el tamaño de ésta debe ser una potencia de dos. Es importante destacar que el diseño del espectrómetro podrá funcionar a todas las velocidades menores o iguales a la escogida. Otras restricciones de diseño que se impusieron fueron:

- Utilizar al menos un ancho de 18 bits para los datos de frecuencia que salen de la FFT, esto para asegurar un buen rango dinámico y bajo ruido del espectrómetro.

[25] Figura extraída de Ricardo Finger, 2013 [4]

- Para evitar *overflow* y también la recuantizacion, se utilizaron vectores acumuladores de 64 bits y memorias BRAM de 64 bits.
- Realizar un *down shitft* (división por 2) en cada etapa de la FFT, y en consecuencia la constante en la entrada *shift* del bloque FFT será siempre igual a $2^n - 1$, con *n el* número de etapas de la FFT.

El objetivo es intentar compilar el espectrómetro de mayor resolución posible utilizando la mayor cantidad de recursos disponibles en el FPGA. Esto sumado a las altas velocidades generará, en algunos diseños, problemas de *timing*, que corresponden a problemas de sincronización entre ramas del circuito.

Para mejorar la sincronización de los diseños se aumentaron las latencias internas de los bloques PFB y FFT, tal que tuvieran una mayor holgura para trabajar a altas velocidades. Cuando agregando latencias en el circuito no es suficiente para que el diseño funcione correctamente, es necesario implementar una optimización física de éste, es decir aplicar técnicas de *floorplanning* para mejorar el rendimiento del circuito. La etapa de *floorplanning* es la última en el proceso de compilación de los diseños en FPGA y consiste en ubicar los componentes del circuito diseñado en las distintas partes del chip (*placement*). Esta optimización se realizó utilizando el programa de Xilinx PlanAhead.

El programa PlanAhead permite rehacer el *placement* de un diseño compilado, reubicando los componentes del circuito digital implementado en los recursos disponibles del chip FPGA. Es posible asignar cada componente manualmente o bien definir *P-blocks*, los cuales corresponden a áreas dentro del chip que se utilizan para restringir el posicionamiento de componentes.

La estrategia de reubicación de componentes utilizada fue generar *P-blocks* tal que la propagación de datos desde el ADC a las memorias sigan la misma ruta que el diseño de alto nivel [18] presentado en la Figura 3.12. Esto quiere decir que componentes cercanos en el diseño lógico se posicionaran cerca a nivel físico también. Esto mejorará la sincronización de los distintos caminos del diseño.

Otra estrategia que se utilizó en el posicionamiento físico fue incluir únicamente BRAMs y bloques DSP en las restricciones de los *P-blocks*, ya que son los principales componentes del diseño y el FPGA, y además, al ubicar estos componentes en lugares específicos, el compilador posicionará los componentes restantes que se comunican con estos cerca de ellos [19]. El chip FPGA Virtex 6 utilizado posee 1064 memorias BRAM de 36 kbits, 2128 BRAM de 18 kbits y 2016 DSP48E.

Cuando la optimización física en base a restricciones de áreas generales no soluciona todos los problemas de sincronización del circuito, se utiliza la herramienta *Timing Analizer* de Xilinx, la cual proporciona información específica de cada camino crítico del diseño que no cumplió los requerimientos de tiempo en la compilación. Utilizando dicha información es posible reubicar específicamente los componentes involucrados en caminos críticos.

En la Figura 3.13 se muestra la representación física del chip FPGA en PlanAhead con un diseño de un espectrómetro. Las partes celestes corresponden a los componentes del chip que están siendo utilizados por el circuito implementado.

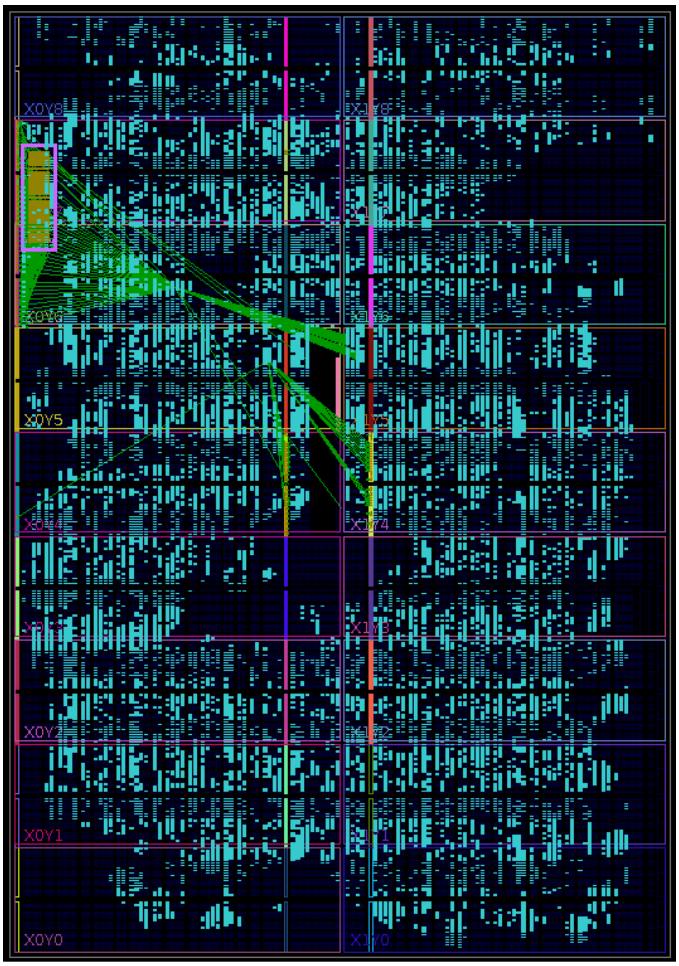


Figura 3.13: Representación física de Planahead del chip FPGA

3.3.2 Pruebas

Para probar los espectrómetros diseñados se utilizó un generador de señales para generar un tono puro en frecuencias dentro del ancho de banda y así verificar que el rango dinámico se mantuviera y fuera lo suficientemente grande.

También se realizó una prueba de los espectrómetros sumando un piso de ruido analógico al tono de entrada, el cual fue generado utilizando una fuente de ruido y sumado utilizando un *splitter* en sentido inverso. El objetivo es fijar el piso de ruido analógico por sobre el piso de ruido numérico (producido digitalmente) pero no por sobre los espurios del espectro. Esto sirve para simular el ruido que existiría si se tuviera un receptor radioastronómico en vez de un generador de señales antes de la ROACH 2.

En la Figura 3.14 se presenta un diagrama del *setup* utilizado para sumar el piso de ruido analógico a la señal de entrada. La señal de ruido es producida por una fuente de ruido de 28 volts DC y luego amplificada utilizando un amplificador de 65 dB. La señal amplificada pasa por atenuadores regulables antes de entrar al *splitter* (de forma inversa) para ser sumada al tono de entrada. La señal que contiene el tono puro pasa por un atenuador de 6 dB para evitar reflexiones en el *splitter*. Los componentes y equipos utilizados son los siguientes:

- Agilent 346b Noise Source
- Atenuador Agilent 8495B / 70 dB
- Atenuador Agilent 8494B / 11 dB
- Amplificador Miteq AMF-6F-00100400-20-10p
- Mini-Circuits Power Splitter ZFRSC-183-S+
- Agilent EXG Analog Signal Generator

Al utilizar este *setup* para probar un espectrómetro, éste debe ser usado también para realizar las calibraciones del ADC previas a las pruebas de los diseños.

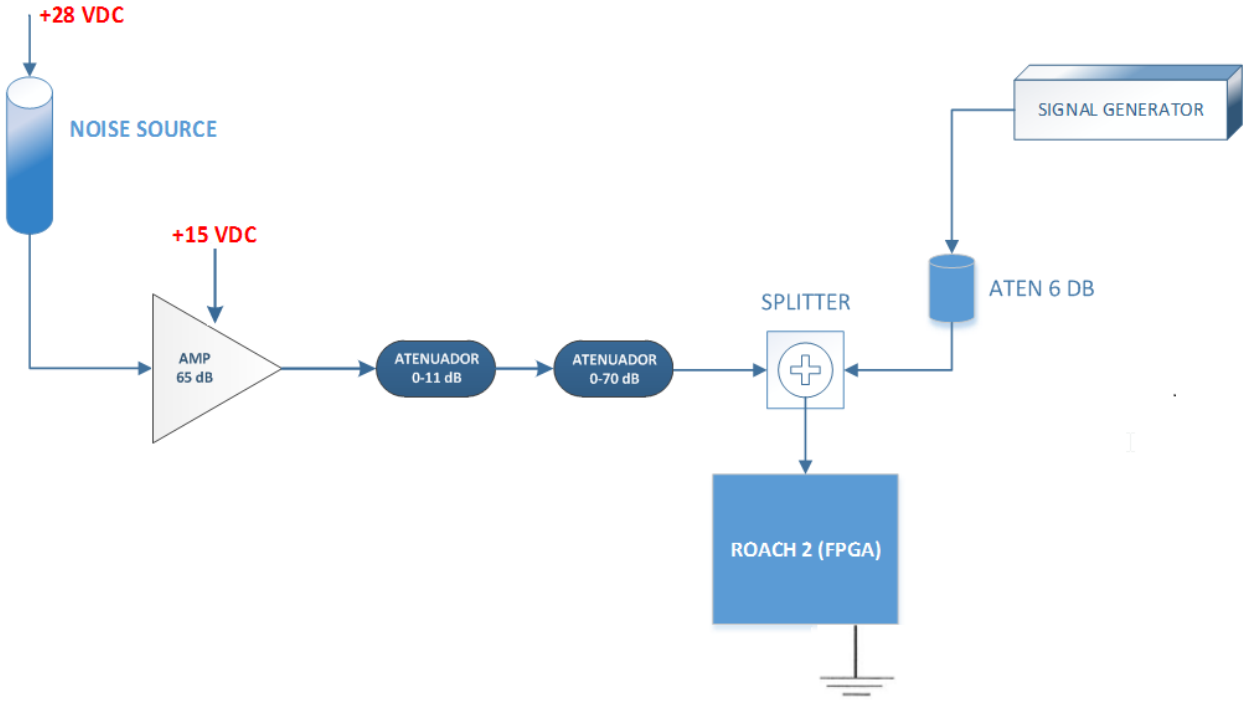


Figura 3.14: Diagrama de bloques de *setup* de prueba con piso de ruido analógico

El *splitter* posee aproximadamente una caída de 6 dB de potencia, es decir que la amplificación de la señal de ruido será aproximadamente igual a 59 dB menos la atenuación escogida.

En la Figura 3.15 se presenta una fotografía del *setup* correspondiente al esquema de la Figura 3.14, donde se indica a que corresponde cada componente.

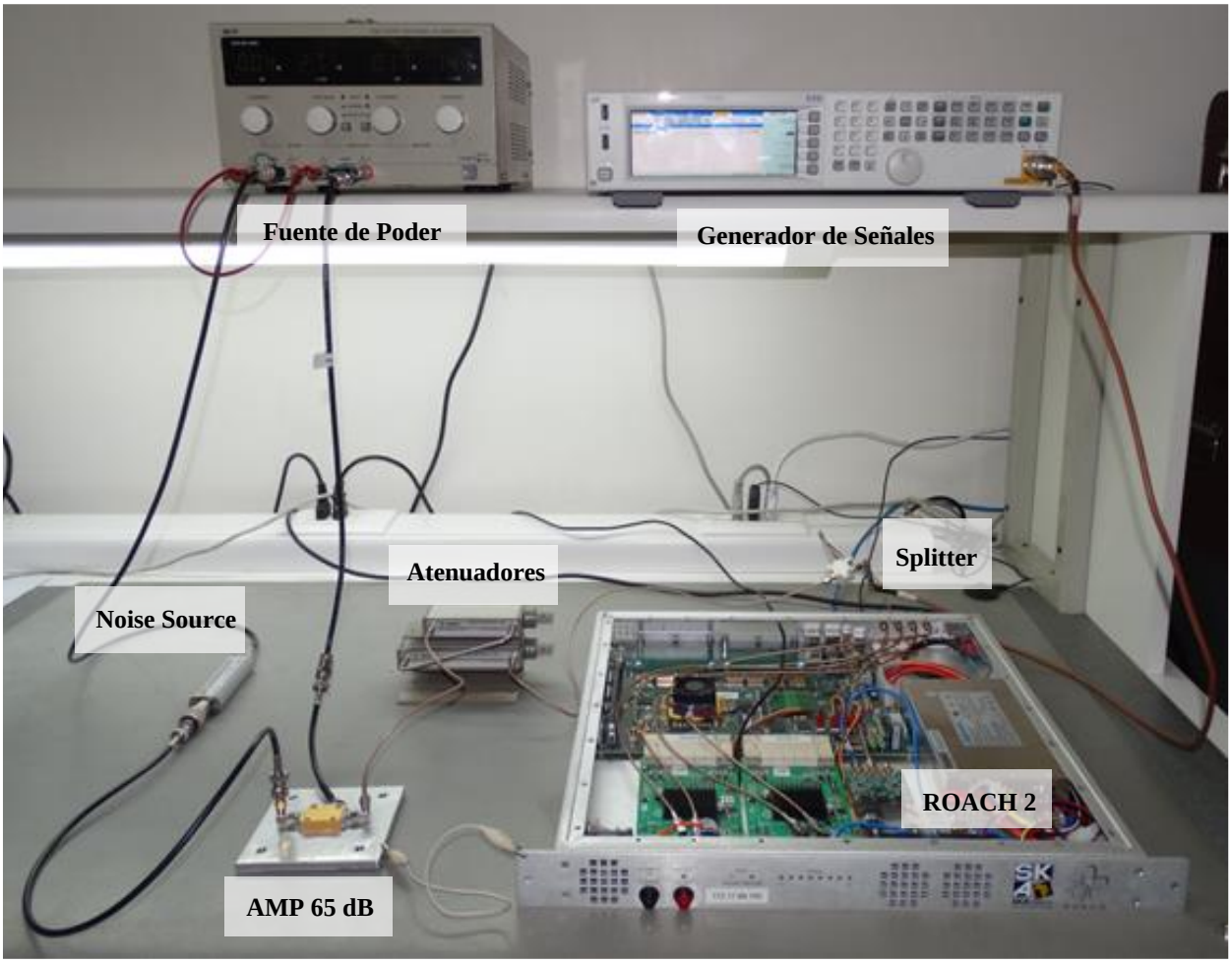


Figura 3.15: Fotografía de *setup* de prueba con piso de ruido analógico

3.3.3 Software

Se escribieron varios códigos en Python para escribir y leer datos de los espectrómetros en ROACH 2, los que difieren en algunos parámetros y funciones pero cumplen los mismos objetivos. Al utilizar uno de estos programas, lo primero que éste hace es ejecutar el diseño que se desea correr en el FPGA, el cual ya se encuentra cargado en la memoria FLASH de la ROACH 2. Posteriormente se realiza la calibración de la interface del ADC con la FPGA y se escriben valores en los registros correspondientes para fijar parámetros del espectrómetro tales como la ganancia digital y el largo de acumulación. Una vez que el espectrómetro está funcionando en el FPGA, se entra en un ciclo en el cual se leen las memorias que contienen los datos espectrales (salida del espectrómetro) y luego se grafican, todo esto en tiempo real.

CAPÍTULO 4

RESULTADOS

4.1 CALIBRACIÓN ADC

Utilizando el circuito de medición de nivel (Figura 3.9), se obtuvo que al utilizar una señal de baja frecuencia con -6 dBm de potencia en la entrada del ADC, se tiene aproximadamente un 92% del nivel máximo de éste, lo que es apropiado para efectuar la calibración. Una vez ejecutado el programa de calibración, para corroborar el efecto de esta en el muestreo del ADC, se utiliza un programa en Python que grafica los datos muestreados en tiempo real obtenidos de un bloque *snapshot* en el diseño cargado en el FPGA, tal y como el utilizado en el circuito de control de nivel (Figura 3.9).

En la Figura 4.1 se muestra un *snapshot* parcial (300 muestras) de una señal de prueba de 10 MHz con el ADC no calibrado, funcionando con una frecuencia de reloj de 2400 MHz. En esta se puede observar que el muestreo de la señal no es perfectamente continuo, lo que es producido principalmente por el desfase existente entre los cuatro núcleos del ADC. Esto indica preliminarmente que al usar el ADC no calibrado para implementar un espectrómetro, el rango dinámico libre de espurios (SFDR) se verá afectado.

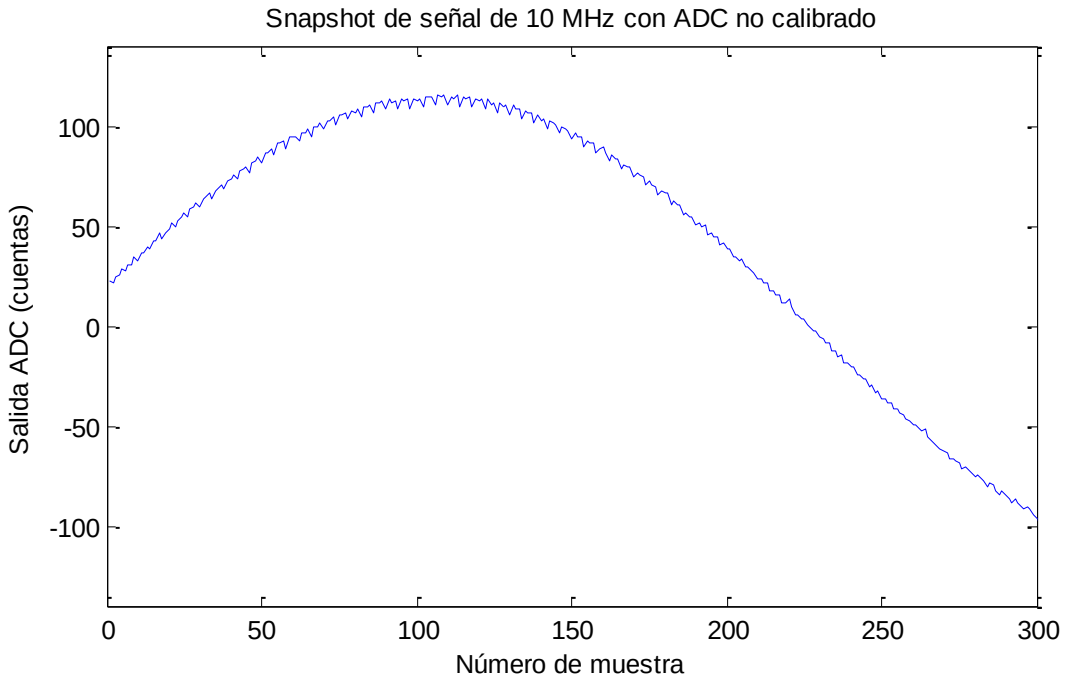


Figura 4.1: *Snapshot* de señal sinusoidal de baja frecuencia (10 MHz) con el ADC no calibrado

En la Figura 4.2, se presenta el *snapshot* parcial de la misma señal de 10 MHz pero después de haber hecho una calibración completa del ADC, es decir habiendo calibrado la interface con el FPGA, el *offset*, ganancia y fase y la no linealidad de cada núcleo. En la figura se puede observar que el muestreo del ADC calibrado mejora considerablemente, generando una onda sinusoidal continua. Esto mejorará el rendimiento de los espectrómetros implementados, lo que se puede corroborar caracterizando el SFDR del ADC.

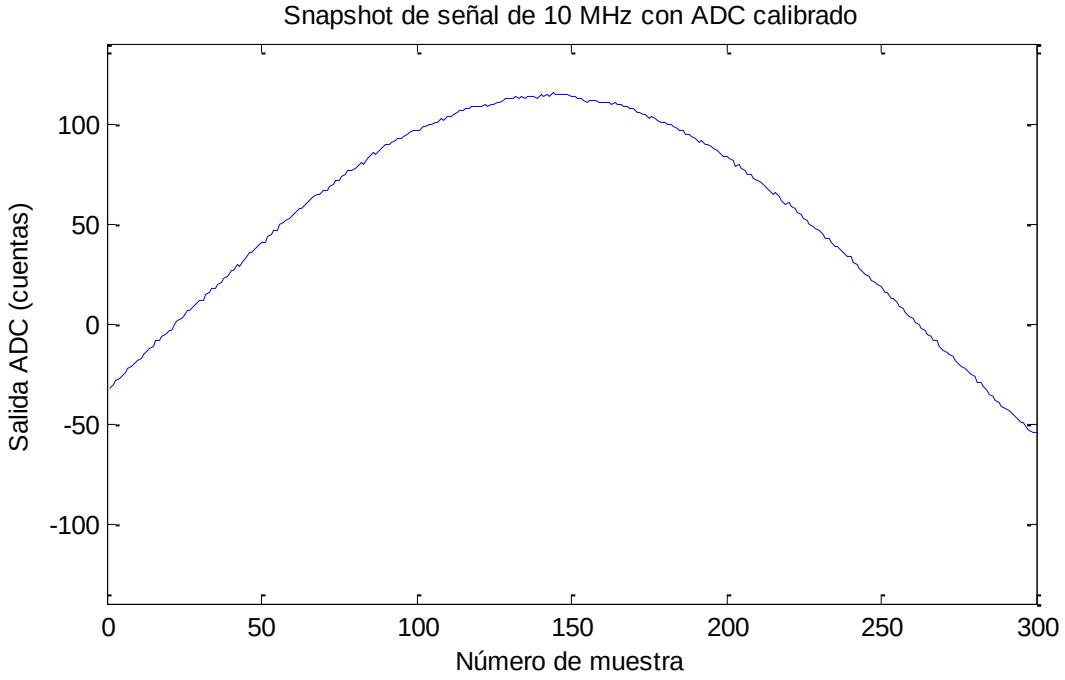


Figura 4.2: *Snapshot* de señal sinusoidal de baja frecuencia (10 MHz) con el ADC calibrado

Como la calibración del ADC es volátil y depende de variados factores, los coeficientes de corrección calculados serán distintos en cada calibración, por lo tanto no tiene relevancia indicarlos. Lo que si se mantendrá en todas las calibraciones, es la forma de la onda calibrada.

Además, se descubrió que la calibración implementada depende de la frecuencia de la señal de prueba utilizada. Esto implica que el alineamiento de los núcleos del ADC podría empeorar al muestrear señales de alta frecuencia (más de 1 GHz), afectando el desempeño del ADC.

4.1.1 Medidas de desempeño del ADC

Para caracterizar el ADC que se utilizará en la implementación del espectrómetro y también para escoger la frecuencia de la señal de prueba para la calibración, se usaron dos medidas de desempeño, el ancho de banda analógico y el SFDR (*spurius free dinamic range)*. Estas medidas son utilizadas entre otras por el equipo que diseñó la tarjeta del ADC [20] y usadas en general en la literatura para caracterizar conversores análogo-digital.

Para obtener estas medidas se utilizó el mismo *setup* de la Figura 3.5, controlando la frecuencia de la señal de entrada desde un programa Python vía Ethernet entre 100 MHz y 2,4 GHz con intervalos de 50 MHz. Para calcular el espectro de la señal en cada frecuencia, se utilizó la función "get_psd" de Matlab, la cual utiliza como entrada las 16384 muestras del bloque *snapshot.*

Ancho de banda analógico

La Figura 4.3 muestra la respuesta en frecuencia del ADC calibrado en todo su ancho de banda (no se muestra la curva del ADC no calibrado ya que prácticamente no existe variación). Existe una caída en ganancia de aproximadamente 5,5 dB a 2 GHz (contra 3 dB que indica el *datasheet* del chip del ADC [15]) y de 6,1 dB a 2,35 GHz. La caída de potencia observada se conoce como *el roll-off* del ADC. La curva de este ADC en particular indica que no podrá ser utilizado para muestrear en zonas de Nyquist de mayor orden.

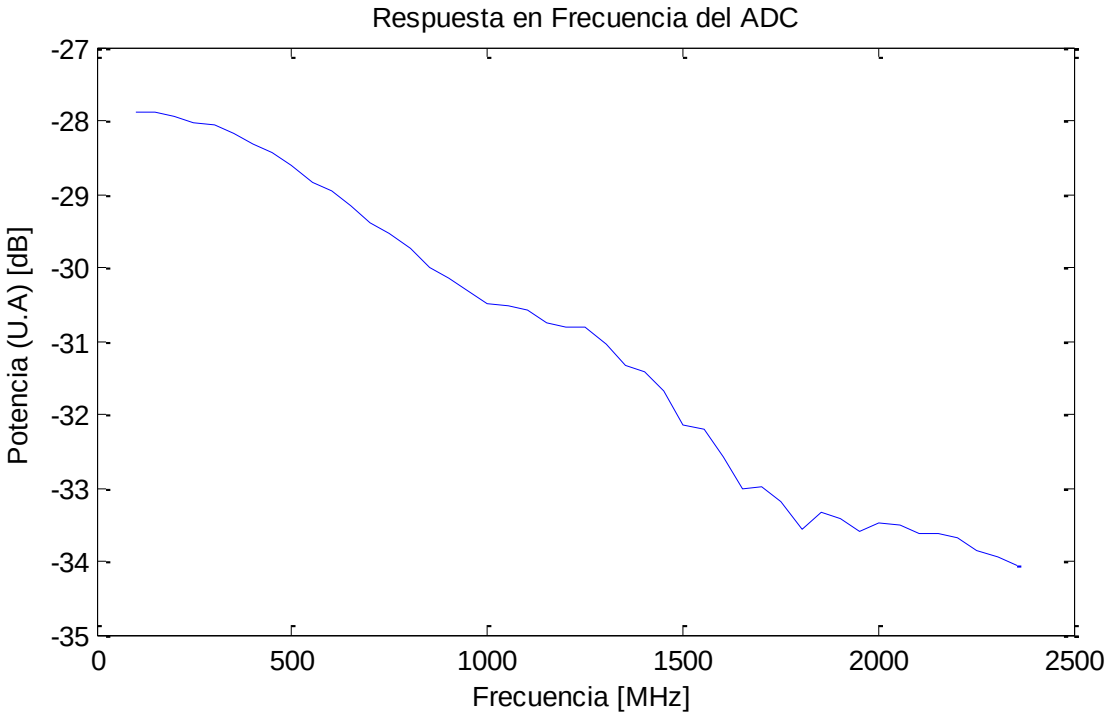


Figura 4.3: Respuesta en frecuencia del ADC calibrado sobre todo el ancho de banda

SFDR

En la Figura 4.4 se muestra el SFDR del ADC en todo su ancho de banda luego de ser calibrado utilizando distintas frecuencias. Se encontró que al utilizar una señal de 10 MHz para calibrar el ADC, el SFDR del ADC disminuye considerablemente al aumentar la frecuencia de muestreo. Como se puede ver en la figura, el mejor desempeño se obtuvo calibrando con una frecuencia de 100 MHz. Para dicho caso, el SFDR del ADC cae a un mínimo de 41,83 dB al llegar a una frecuencia de entrada de 2,4 GHz aproximadamente.

Alrededor de 250 MHz, el SFDR del ADC tiene una brusca caída (a 44,8 dB) para todas las calibraciones, y luego vuelve a estabilizarse. Este comportamiento es propio del tipo de calibración realizada, ya que también sucede en los resultados del artículo de CASPER [16].

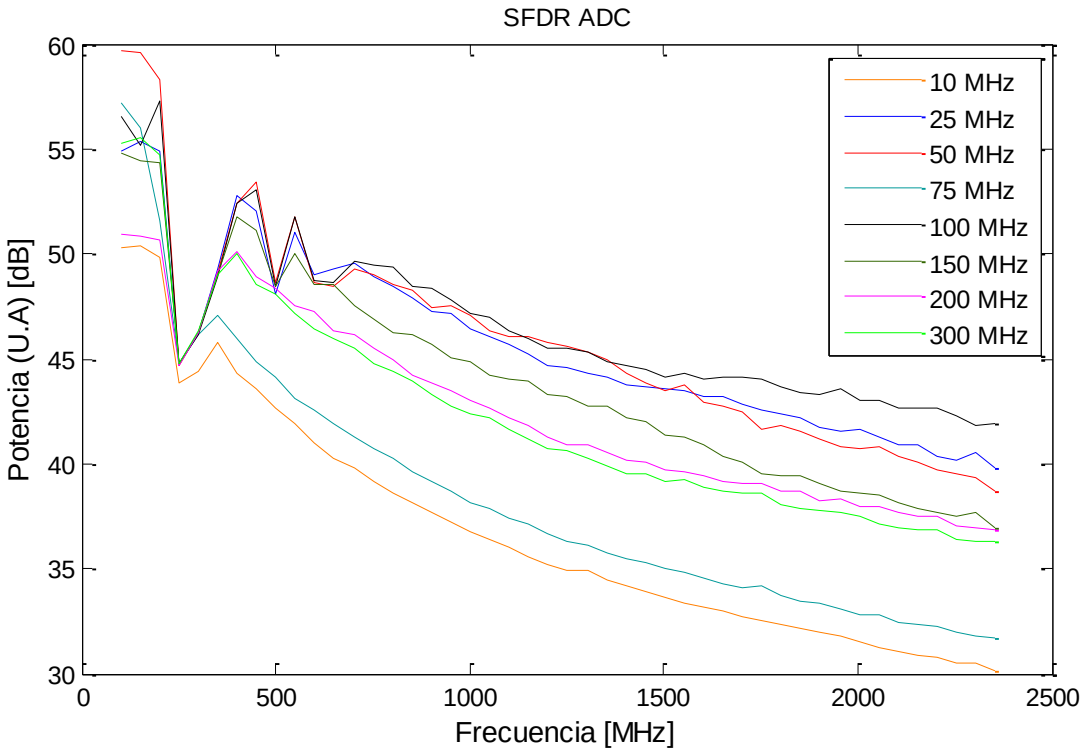


Figura 4.4: Medida del *Spurius Free Dinamic Range (*SFDR) del ADC calibrado usando distintas frecuencias de entrada, en todo su ancho de banda

En la Figura 4.5, se observa el SFDR del ADC calibrado a 100 MHz en todo su ancho de banda comparado con el SFDR del ADC no calibrado. Con el ADC calibrado se tiene un SFDR de 56,59 dB a 100 MHz y 48,72 dB a 600 MHz (contra 58 dB y 56 dB respectivamente según el *datasheet* del chip del ADC para estas frecuencias [15]).

Se observa que la calibración completa del ADC mejora considerablemente el SFDR de éste. Se tiene un aumento de aproximadamente 16 dB a 100 MHz y de 10 dB a 600 MHz. Esta mejora va creciendo a medida que aumenta la frecuencia de entrada, esto debido a que el SFDR del ADC no calibrado depende fuertemente de la frecuencia, disminuyendo de forma casi lineal, mientras que el SFDR del ADC calibrado es más constante frente al aumento de frecuencia.

En la calibración del ADC realizada en el artículo de CASPER [16], el SFDR del ADC calibrado se mantiene sobre 50 dB prácticamente en todo su ancho de banda, mientras que en el artículo de ASIAA [20], el SFDR se mantiene siempre sobre 44 dB. En esta calibración se logró mantener el SFDR por sobre 41 dB en todo el ancho de banda, lo que se puede considerar consistente con los demás resultados.

Una explicación a los resultados obtenidos, los cuales son un poco peores que los valores presentados en el *datasheet* del ADC y los artículos mencionados, es que la calibración implementada no es totalmente perfecta, manteniendo una leve dependencia de la frecuencia de entrada. Al aumentar la frecuencia el desalineamiento de los núcleos del ADC aumenta, empeorando el SFDR del ADC. Esto es algo que es sugerido por el grupo de CASPER [16] pero

negado por los diseñadores de la placa del ADC [20]. Este comportamiento en el rango dinámico del ADC afectará el desempeño de los espectrómetros que se implementen, limitando el SFDR de estos.

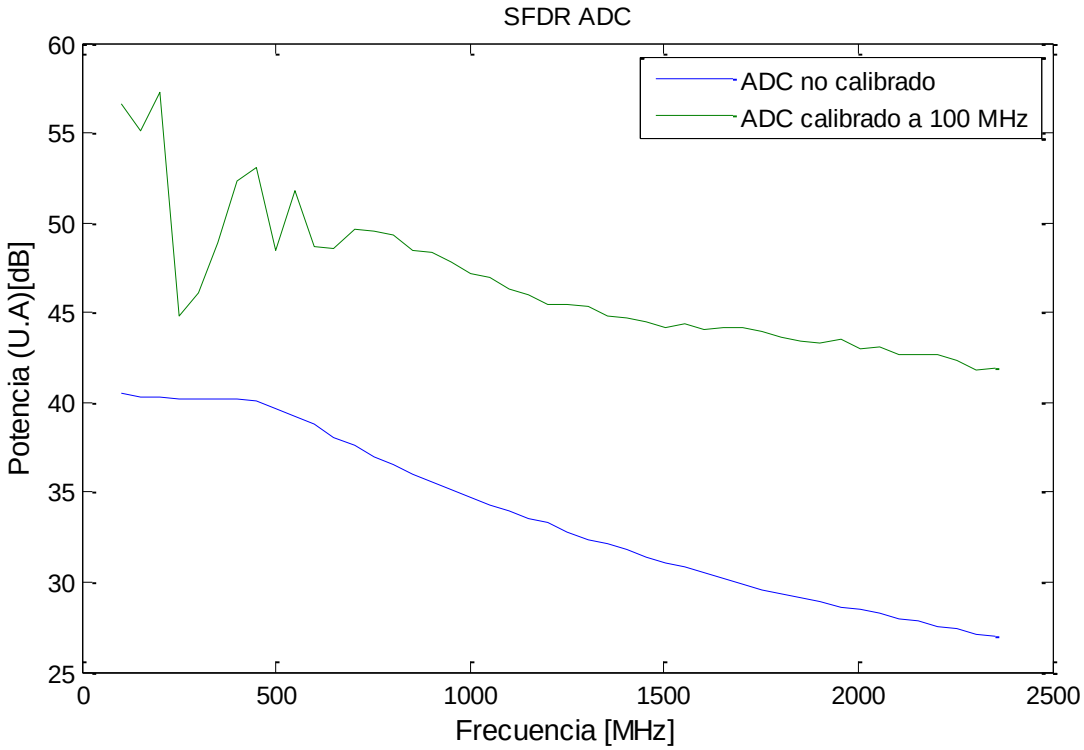


Figura 4.5: Medida del *Spurius Free Dinamic Range (*SFDR) del ADC no calibrado y calibrado (a 100 MHz) en todo su ancho de banda

4.2 ESPECTRÓMETRO DE ALTA RESOLUCIÓN

A continuación se presentan los resultados de dos espectrómetros implementados en ROACH 2 con un ancho de banda de 1,8 GHz: un espectrómetro de 4096 canales espectrales y un espectrómetro de 32768 canales, que fue el de mayor resolución obtenido. Ambos siguen el diseño de alto nivel de la Figura 3.12. El primer espectrómetro servirá como referencia para analizar los cambios en el desempeño de éste al aumentar su resolución espectral y para una eventual comparación con espectrómetros desarrollados en ROACH 1, los cuales tienen una cantidad similar de canales.

Para caracterizar el desempeño de los espectrómetros se presentan espectros utilizando tonos de distintas frecuencias de entrada, escogidas de tal manera que se permita analizar distintos factores importantes, tales como el rango dinámico (SFDR) del espectrómetro en toda la banda y la estabilidad temporal. Las frecuencias escogidas fueron las siguientes:

- 450 MHz y 1,35 GHz: la frecuencia de reloj del ADC es un múltiplo exacto de estas frecuencias.
- 659,1797 MHz y 1647,9492 MHz: frecuencias elegidas aleatoriamente, tales que caigan exactamente en el centro de un canal espectral.

- 450,1099 MHz y 1350,1099 MHz: frecuencias correspondientes a 450 + $\Delta f/2$ MHz y 1350 + $\Delta f/2$ MHz, donde Δf es la resolución espectral del espectrómetro. Es decir los tonos de estas frecuencias se ubicarán entre dos canales espectrales.

El largo de acumulación utilizado para probar ambos espectrómetros es de 65536 y la ganancia digital fue fijada en $\mathrm{F0000000}_{16}$, lo que es igual a 4026531840_{10}.

4.2.1 Espectrómetro de 4096 canales

Este espectrómetro posee una resolución espectral de 439,45 kHz. El diseño utiliza el 6% de las memorias BRAM de 36 kbits del FPGA Virtex 6, el 6% de las BRAM de 18 kbits y el 33% de los bloques DSP48E. Para su implementación no fue necesario implementar técnicas de *floorplaning* para conseguir los requerimientos de tiempo del circuito, siendo suficiente el posicionamiento de los componentes realizado por el compilador.

En la Figura 4.6 se presentan dos espectros con tonos de 450 MHz y 659,18 MHz introducidos directamente en la entrada del ADC, es decir sin sumarles un piso de ruido analógico. El tono de 450 MHz, correspondiente a ¼ del ancho de banda, tiene una amplitud de 82,39 dB y un rango dinámico de 47,18 dB, el cual es determinado por un espurio de una amplitud de 35,21 dB ubicado en 900 MHz. Dicho espurio corresponde a una suma entre un espurio real de la señal de entrada (correspondiente al segundo armónico del tono principal, ubicado en el doble de la frecuencia de éste) y un tono producido por el ADC en la mitad del ancho de banda, el cual estará siempre presente. El otro espurio presente en 1350 MHz es producido digitalmente, lo que se verificó observando la misma señal de entrada utilizando un analizador de espectro. Dichos resultados se presentan en el Apéndice B.

El tono de 659,18 MHz corresponde a una frecuencia cualquiera que cae justo en el centro de del canal 1500 de este espectrómetro. La amplitud del tono es de 81,69 dB y la del espurio más grande, ubicado aproximadamente en 1559,2 MHz es de 34,22 dB, imponiendo un rango dinámico de 47,47 dB. Este espurio es producido digitalmente y corresponde al mismo que aparece en la Figura 4.6a, el cual se corrió exactamente en 209 MHz a la derecha, producido por el aumento de esa cantidad en la frecuencia fundamental de la señal. El segundo espurio más fuerte está ubicado aproximadamente en 241 MHz y también es producido digitalmente. El 3er armónico más fuerte corresponde al 2do armónico de la frecuencia fundamental (en 1318 MHz) y el 4to espurio con mayor amplitud corresponde al espurio intrínseco del ADC. Todos los demás espurios corresponden a ruido numérico, lo que quiere decir que son producidos digitalmente producto del error de muestreo.

Se puede observar que el espectro con el tono de 450 MHz (Figura 4.6a) tiene mucho menos ruido numérico que el espectro con el tono de 659,18 MHz (Figura 4.6b). Esto se debe a que la frecuencia de muestreo es un múltiplo de 450 MHz. Aun así, el rango dinámico es mayor para el segundo espectro.

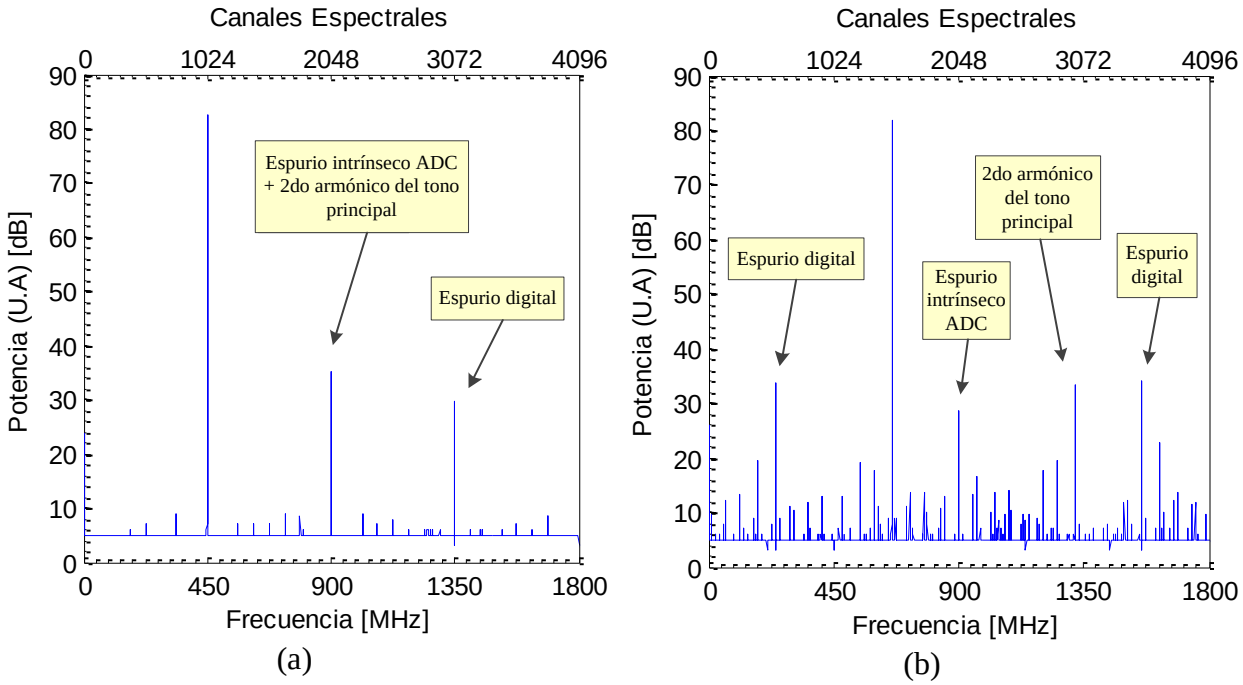


Figura 4.6: Espectro de espectrómetro de 1.8 GHz, 4096 canales. (a) Tono de 450 MHz. (b) Tono de 659.18 MHz.

En la Figura 4.7, se presentan espectros con los mismos tonos que la Figura 4.6, pero esta vez agregándole un piso de ruido analógico a la entrada, esto usando el *setup* de la Figura 3.14. El atenuador después del amplificador se fijó en 25 dB, es decir la señal de ruido se amplifica en aproximadamente 34 dB, lo que genera un aumento en potencia del piso de ruido del espectro en 15 dB, que al aumentar la frecuencia va cayendo, debido a la caída de ganancia intrínseca del amplificador.

El tono de 450 MHz (Figura 4.7a) tiene una amplitud de 82,21 dB y un SFDR de 48,77 dB, es decir 1,3 dB más que sin el piso de ruido, mientras que el tono de 659,18 MHz (Figura 4.7b) tiene una amplitud de 81,59 dB y un rango dinámico de 46,41 dB es decir alrededor de 1 dB menos que sin el piso de ruido analógico, lo que es algo inusual.

Se puede ver que al agregarle un piso de ruido analógico al espectrómetro, el rango dinámico de éste mejora en algunos decibeles para el tono de 450 MHz y se mantiene casi constante para el tono de 659,18 MHz. Las pruebas con las frecuencias de entrada restantes se realizaron únicamente usando el piso de ruido.

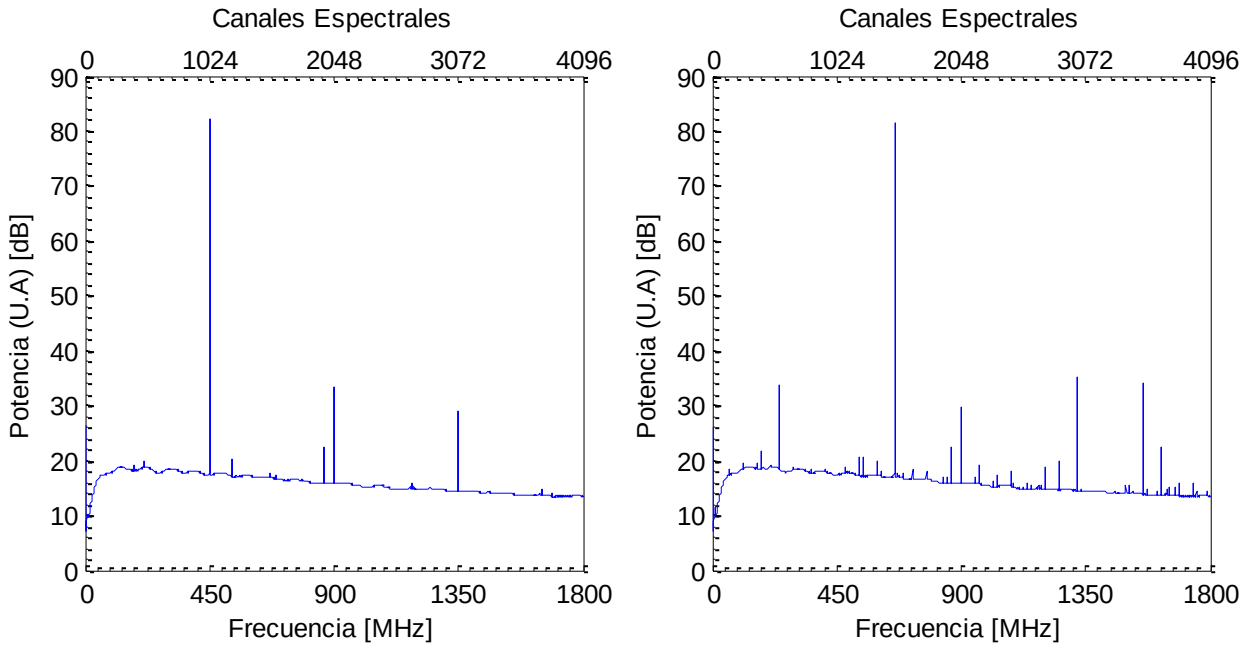
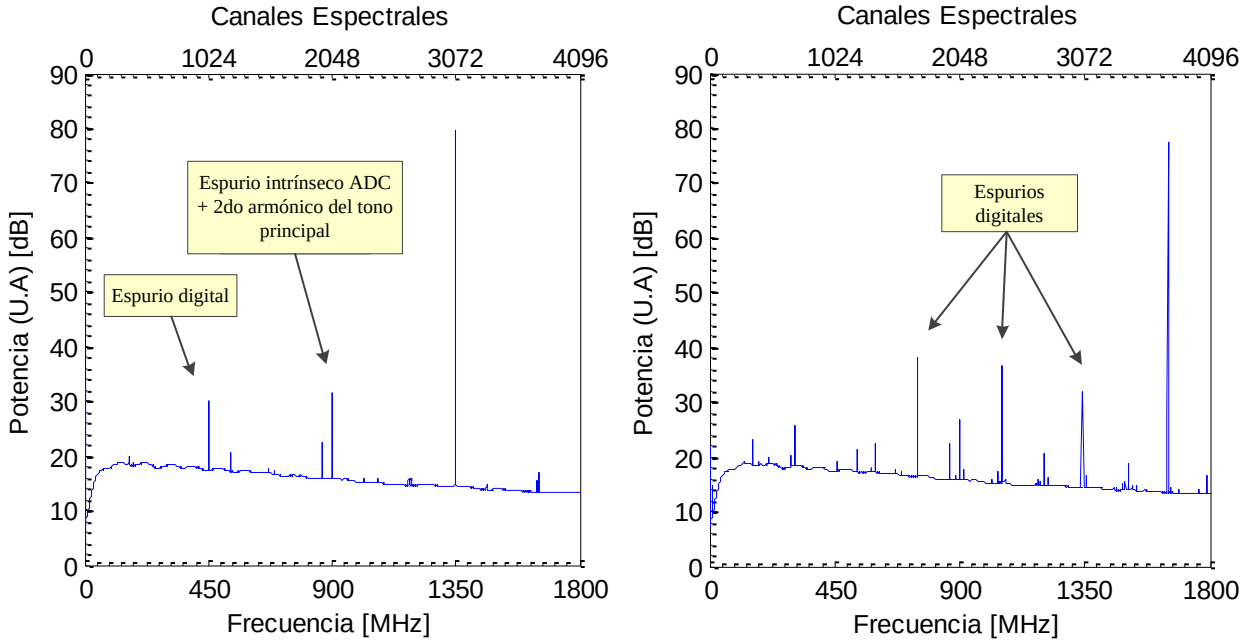


Figura 4.7: Espectro de espectrómetro de 1.8 GHz, 4096 canales, con piso de ruido. (a) Tono de 450 MHz. (b) Tono de 659.18 MHz.

En la Figura 4.8 se presentan espectros con tonos de entrada de 1350 MHz (Figura 4.8a) y 1648 MHz (Figura 4.8b), los cuales tienen amplitudes de 79,57 dB y 77,52 dB respectivamente. El SFDR del tono de 1350 MHz es de 46,52 dB y está determinado por el espurio presente en 900 MHz, mientras que el SFDR del tono de 1648 MHz es de 39,33 dB, el cual está determinado por un espurio digital de 38,19 dB. Este último se encuentra ubicado aproximadamente en 748 MHz.

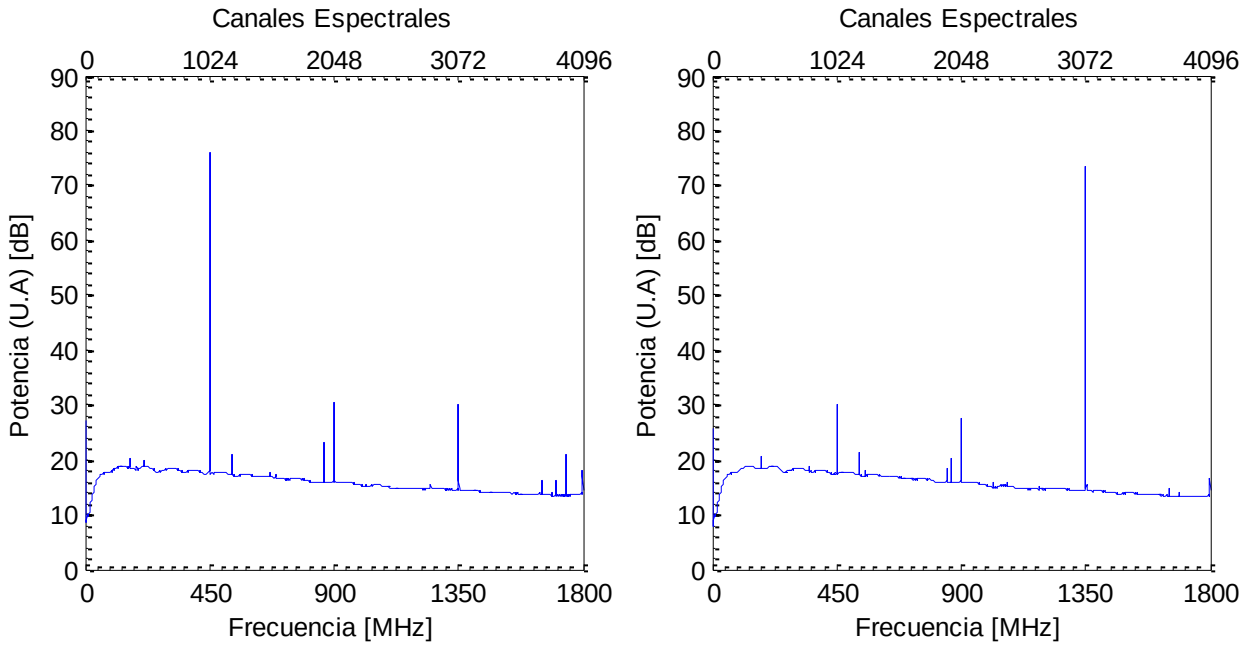
En la Figura 4.8b se muestra que todos los espurios de mayor amplitud son producidos digitalmente, lo que quiere decir que este tipo de espurios aumentó en cantidad con respecto a tonos de otras frecuencias. Además, los espurios producidos por otras razones (intrínseco del ADC y armónico de la frecuencia fundamental) disminuyeron en potencia. Esta situación se debe a que para frecuencias más altas, el desalineamiento de los núcleos del ADC disminuye, disminuyendo su SFDR.



(a) (b)

Figura 4.8: Espectro de espectrómetro de 1.8 GHz, 4096 canales, con piso de ruido. (a) Tono de 1350 MHz. (b) Tono de 1648 MHz.

En la Figura 4.9 se presentan los espectros de señales de 450,22 MHz (Figura 4.9a) y 1350,22 MHz (Figura 4.9b), las cuales posicionan al tono principal entre los canales 1024 y 1025, y 3072 y 3073 respectivamente. Estos tonos tienen amplitudes de 76,13 dB y 73,44 dB, y un SFDR de 45,81 dB y 43,87 dB respectivamente. Es decir que, al situar el tono principal entre canales espectrales, la amplitud de éste baja del orden de 4 dB, lo cual era esperable.



(a) (b)

Figura 4.9: Espectro de espectrómetro de 1.8 GHz, 4096 canales, con piso de ruido. (a) Tono de 450,22 MHz. (b) Tono de 1350,22 MHz.

4.2.2 Espectrómetro de 32768 canales

Este espectrómetro posee una resolución espectral de 54,93 kHz, es decir 8 veces mayor a la del espectrómetro de 4096 canales. Aun así, su desempeño es similar al espectrómetro mostrado anteriormente, teniendo espectros muy parecidos para las frecuencias de prueba que se utilizaron (los espurios que se tienen para cada tono de entrada están localizados en las mismas frecuencias y sólo cambian en amplitud en algunos casos).

El diseño utiliza el 42% de las memorias BRAM de 36 kbits del FPGA Virtex 6, el 1% de las BRAM de 18 kbits y el 38% de los bloques DSP48E. Para lograr el correcto funcionamiento de éste se aplicó *floorplanning*, pero aun aplicando todas las técnicas de *placement* descritas en la sección 3.3.1, el espectrómetro quedó con 1 error de *timing* de -0,002 ns entre dos compuertas del circuito, el cual es tan pequeño que influirá en el desempeño del circuito. El *floorplanning* definitivo con que se compiló el espectrómetro, utiliza tan solo un área de restricciones (*P-block*) para posicionar los componentes del ADC cercanos a las entradas del FPGA. En la Figura 4.10 se presenta la representación visual del *placement* del espectrómetro en el chip FPGA, donde el *P-block* se muestra como un rectángulo color morado.

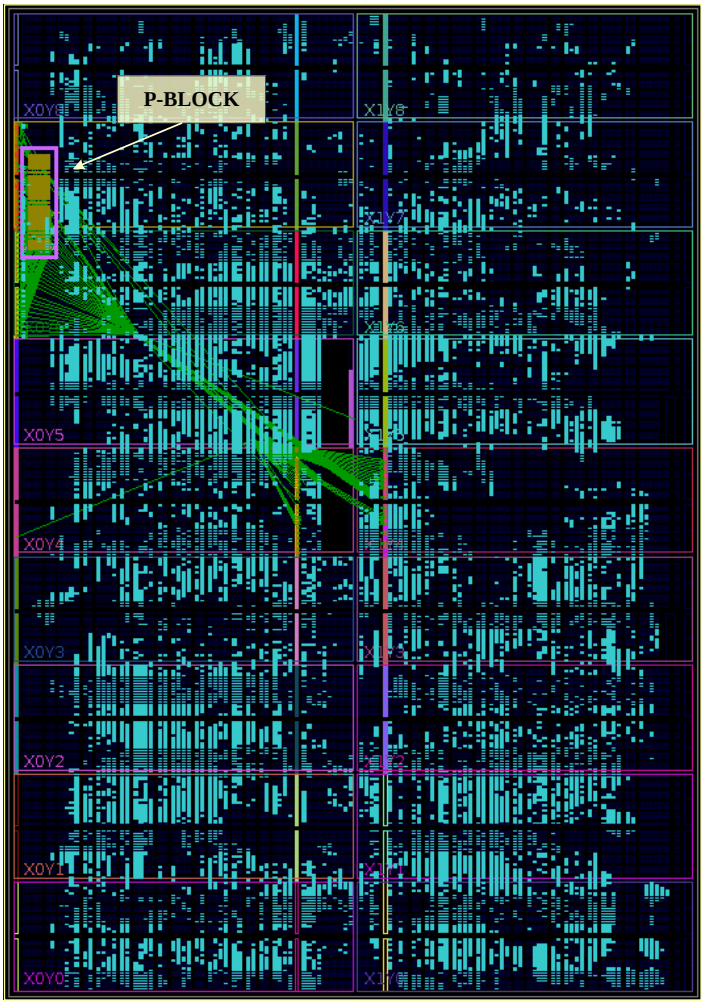


Figura 4.10: *Floorplanning* de espectrómetro 1.8 GHz, 32k canales

En la Figura 4.11 se muestran espectros de señales de entrada de 450 MHz (Figura 4.11a) y 659,18 MHz (Figura 4.11b) sin un piso de ruido analógico, los cuales tienen una amplitud de 82,42 dB y 81,69 dB respectivamente. Los espurios que aparecen en los espectros son los mismos que en el espectrómetro de 4096 canales pero con distintas amplitudes. Aun así, el rango dinámico en ambas señales es determinado por el mismo espurio que en el espectrómetro de 4k canales. El SFDR del tono de 450 MHz es de 49,52 dB y con un tono de 659,19 MHz es de 47,45 dB.

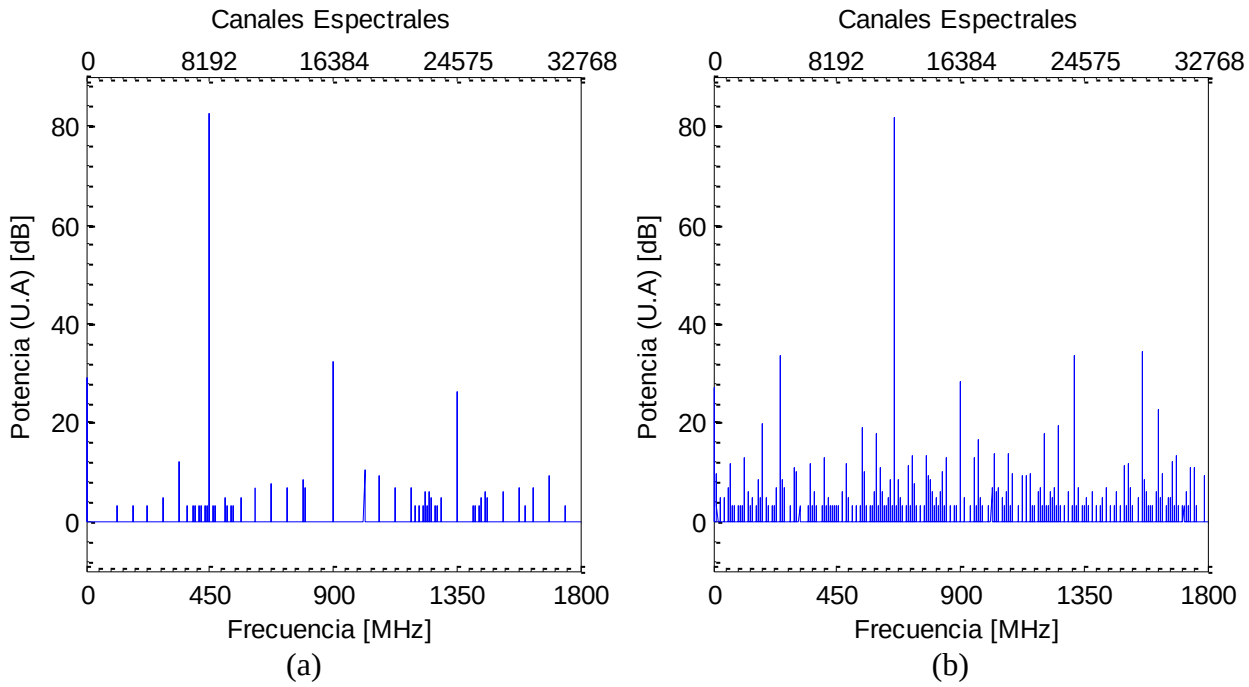


Figura 4.11: Espectro de espectrómetro de 1.8 GHz, 32768 canales. (a) Tono de 450 MHz. (b) Tono de 659.18 MHz.

En la Figura 4.12 se tienen las mismas señales de entrada de la Figura 4.11 pero se utilizó el *setup* de la Figura 3.14 para agregar un piso de ruido analógico. La atenuación después de la amplificación de la fuente de ruido fue fijada en 20 dB, generando una amplificación de 39 dB en la entrada de la ROACH 2. Esto implicó que el piso de ruido del espectro subiera en 15 dB aproximadamente.

La amplitud del tono de 450 MHz (Figura 4.12a) es de 82,24 dB con un SFDR de 49,06 dB, mientras que para el tono de 659,18 MHz (Figura 4.12b) la amplitud de éste es de 81,56 dB con un rango dinámico de 46,23 dB. Se puede ver que al agregarle ruido a la señal de entrada, para ambos tonos el SFDR del espectrómetro disminuye alrededor de 1 dB.

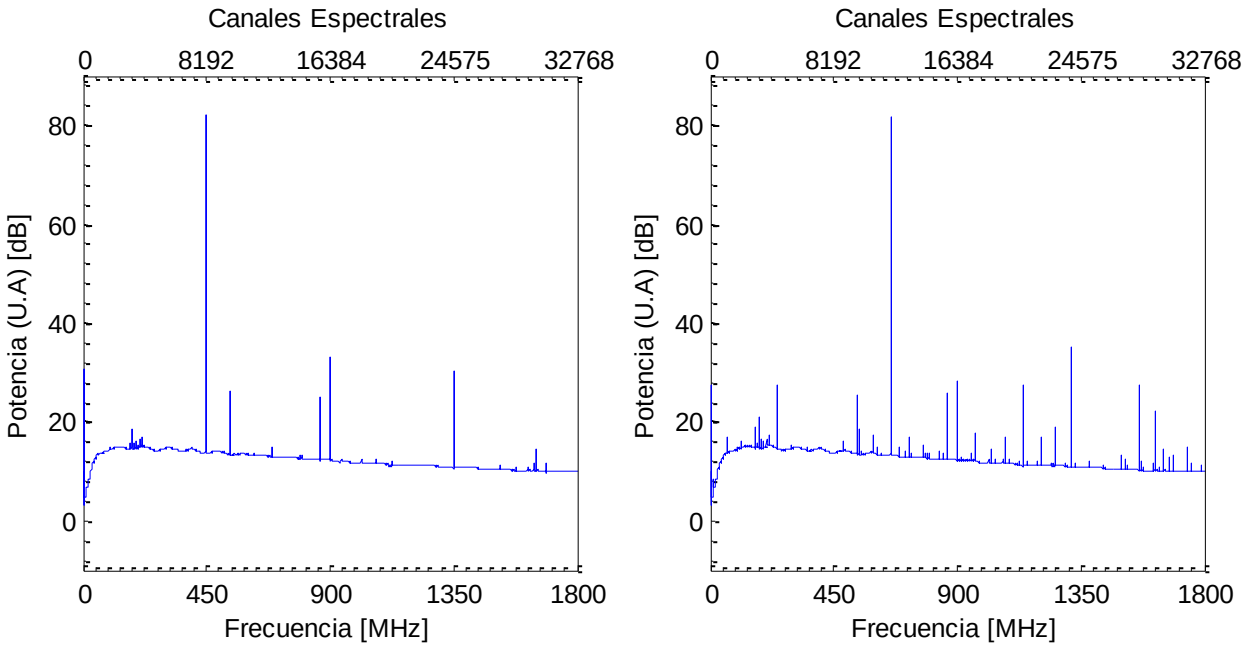


Figura 4.12: Espectro de espectrómetro de 1.8 GHz, 32768 canales, con piso de ruido. (a) Tono de 450 MHz. (b) Tono de 659.18 MHz.

Al igual que con el espectrómetro de 4k canales, las demás señales se probaron únicamente con un piso de ruido analógico. En la Figura 4.13 se tienen los espectros de las señales con frecuencias de 1350 MHz (Figura 4.13a) y 1648 MHz (Figura 4.13b), los cuales tienen una amplitud del tono principal de 79,43 dB y 77,45 dB respectivamente y un rango dinámico de 42,81 dB y 44,18 dB respectivamente.

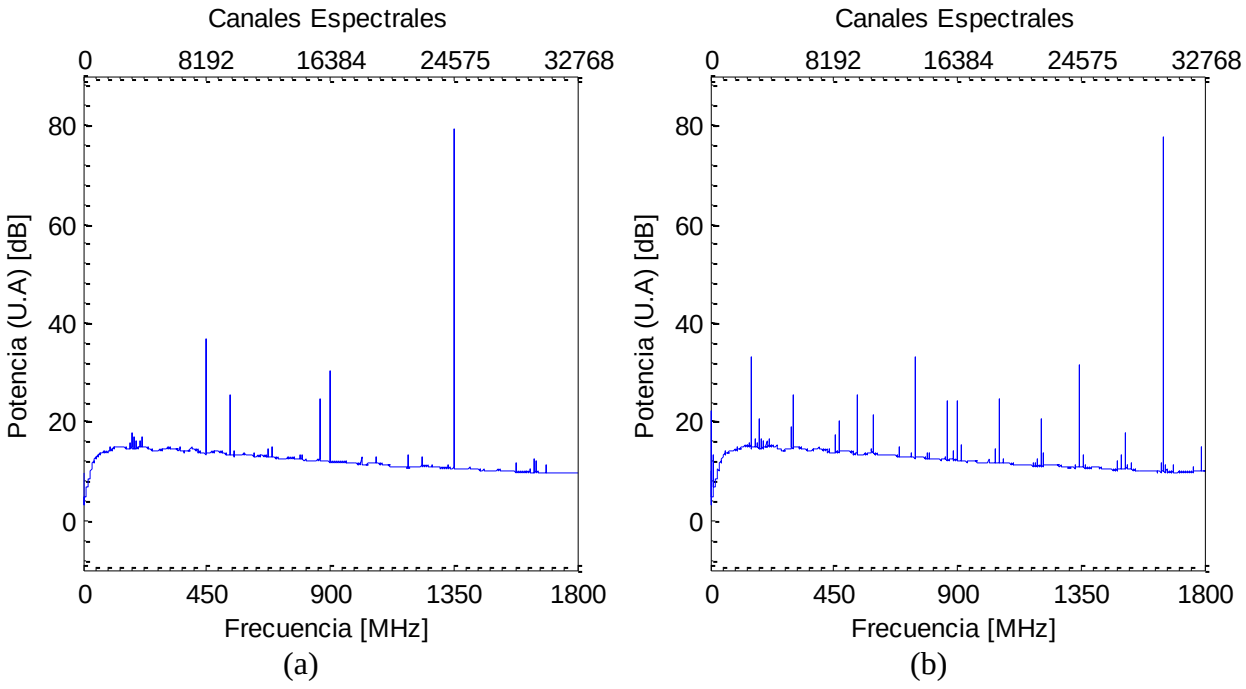


Figura 4.13: Espectro de espectrómetro de 1.8 GHz, 32768 canales, con piso de ruido. (a) Tono de 1350 MHz. (b) Tono de 1648 MHz.

En la Figura 4.14 se presentan espectros de señales de frecuencias de 450,03 MHz (Figura 4.14a), la cual posiciona el tono principal entre los canales espectrales 8192 y 8193, y 1350,03 MHz (Figura 4.14b), la cual posiciona el tono principal entre los canales 24576 y 24577. La amplitud del tono de 450,03 MHz es de 76,16 dB con un rango dinámico de 46,95 dB, mientras que la amplitud del tono de 1350,03 MHz es de 73,38 dB con un rango dinámico de 39,19 dB.

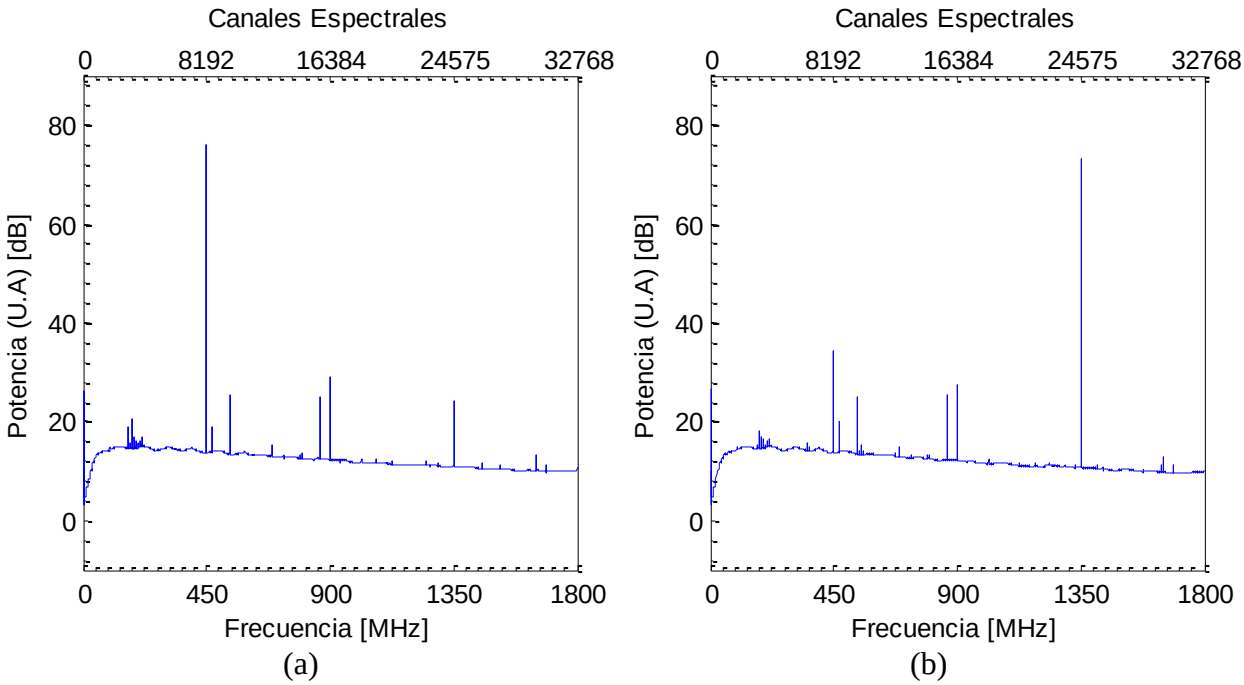


Figura 4.14: Espectro de espectrómetro de 1.8 GHz, 32768 canales, con piso de ruido. (a) Tono de 450,03 MHz. (b) Tono de 1350,03 MHz.

4.2.3 Análisis

En la Tabla 4.1 se presenta el SFDR de ambos espectrómetros para las distintas frecuencias de entrada utilizando el *setup* con piso de ruido analógico (Figura 3.14).

	450 MHz	659,18 MHz	1350 MHz	1647 MHz	450 + Δf / 2 MHz	1350 + Δf / 2 MHz
4096 ch.	48,38 dB	46,41 dB	46,52 dB	39,33 dB	45,81 dB	43,87 dB
32768 ch.	49,06 dB	46,23 dB	42,81 dB	44,18 dB	46,95 dB	39,19 dB

Tabla 4.1: SFDR de espectrómetros para distintas frecuencias de entrada

De la tabla anterior, se puede ver que el SFDR de ambos espectrómetros es superior a 39 dB en toda la banda, por lo que ambos espectrómetros tienen un desempeño similar respecto a SFDR. También se puede ver que para las dos frecuencias menores (450 MHz y 659,18 MHz) el SFDR de ambos espectrómetros es mayor a 46 dB. Por otro lado, para las frecuencias mayores, el SFDR es

siempre mayor a 42 dB, con una caída puntual por debajo de los 40 dB en cada uno de los espectrómetros.

En el espectrómetro de alta resolución, la caída del SFDR a 39,19 dB se da cuando el tono de entrada queda situado entre canales espectrales. Además, el segundo valor de SFDR más bajo (42,81 dB) se obtuvo para el tono de 1350 MHz, frecuencia que corresponde a un múltiplo de la frecuencia de reloj del ADC. Para frecuencias con este tipo de relación a la frecuencia de reloj, el ruido númerico desaparece casi por completo pero los armónicos a veces aumentan en potencia.

Ambos espectrómetros presentan los mismos espurios en sus espectros, donde los principales son los siguientes:

- Espurio en 900 MHz, generado por el ADC en la mitad de la frecuencia de reloj, que en este caso es 1,8 GHz. Este espurio está presente incluso cuando no se tiene una señal de entrada en el ADC.
- 2do armónico del tono principal, ubicado en el doble de la frecuencia de éste, o bien introducido en el espectro por efecto *aliasing* cuando la frecuencia del armónico supera el ancho de banda del espectro. Este espurio es producido por el generador de señales utilizado.
- Espurios producidos digitalmente en distintas frecuencias del espectro, los cuales se mueven junto con el tono fundamental. Estos son los que limitan el SFDR de ambos espectrómetros.

En el Apéndice B se pueden ver los espectros de las señales de entrada utilizadas en ambos espectrómetros obtenidos a partir de un Spectrum Analyzer[26]. En estos se aprecia el 2do armónico del tono principal, aunque para las dos señales de más alta frecuencia no se ven, debido a que el equipo utilizado no posee efecto *aliasing*. Por tanto los armónicos de frecuencias mayores al ancho de banda no se reflejan en el espectro (banda base).

Debido a que los valores de SFDR del ADC fueron obtenidos con medidas independientes a los espectrómetros y son consistentes con los valores del SFDR de los espectrómetros, es posible deducir que los espurios digitales de los espectrómetros son generados por el ADC. Concluimos esto por la pequeña desincronización restante en el muestreo de los núcleos del ADC después de la calibración. Dicha desincronización, generará señales de alta frecuencia, las cuales ingresarán a la FFT de los espectrómetros y entrarán por efecto *aliasing* al espectro de la señal.

En la Tabla 4.2, se presenta la amplitud del tono fundamental de ambos espectrómetros para todas las frecuencias de entrada que se probaron.

[26] http://en.wikipedia.org/wiki/Spectrum_analyzer

	450 MHz	659,18 MHz	1350 MHz	1647 MHz	450 + Δf / 2 MHz	1350 + Δf / 2 MHz
4096 ch.	82,21 dB	81,59 dB	79,57 dB	77,52 dB	76,13 dB	73,44 dB
32768 ch.	82,24 dB	81,56 dB	79,43 dB	77,45 dB	76,16 dB	73,38 dB

Tabla 4.2: Amplitud de tono principal de espectrómetros para distintas frecuencias de entrada

En la tabla anterior se puede apreciar que el aumento en la resolución espectral del espectrómetro prácticamente no afecta la amplitud del tono principal, variando muy poco para todas las frecuencias probadas. Este hecho muestra la alta estabilidad que tiene el espectrómetro diseñado. La caída de potencia del tono principal en ambos espectrómetros es la esperada, debido al *roll-off* del ADC (caída de aproximadamente 5 dB a 1800 MHz).

En el espectrómetro de alta resolución (32k canales) se observó un problema con la señal de sincronización (*sync)* del diseño, ya que el tono principal se corre 8 canales a medida que va pasando el tiempo de observación. Este corrimiento significa que la señal de *sync* está entrando desfasada en un canal, error que se acumula para la siguiente sincronización. Debido a que se utilizan 8 memorias BRAM para guardar los datos espectrales, es que todo el espectro se corre en 8 canales cada vez que se usa la señal de *sync*. Este problema quedó pendiente de solucionar, pero es independiente del desempeño del espectrómetro, ya que el error está en un bloque externo que produce la señal de *sync*.

Aumentar el número de canales del espectrómetro de 1,8 GHz a 65536 canales no fue factible debido a que no se logra optimizar la distribución de recursos dentro del FGPA (*floorplanning)* tal que se logren los requerimientos de tiempo del circuito. Este diseño utiliza el 86% de las BRAM de 36 kbits y 40% de los DSP48E, por lo que por cantidad de recursos disponibles, un espectrómetro de 128k canales de este tipo (*pipeline, Radix*-2) no se podría compilar en el FPGA utilizada.

CAPÍTULO 5

CONCLUSIONES

La calibración del ADC implementada logra que el muestreo funcione correctamente pero no de forma perfecta. Esto hace posible utilizarlo para construir espectrómetros con anchos de banda de hasta 1,8 GHz en ROACH 2, pero produciendo algunos espurios como consecuencia de señales de alta frecuencia. Estas aparecen en el espectro de la señal muestreada producto de efecto *aliasing*. El SFDR del ADC calibrado es superior a 41 dB en todo el ancho de banda, valor que limitará el desempeño de los espectrómetros implementados. Eventualmente podrían implementarse espectrómetros de pocos canales espectrales de hasta 2,2 GHz de ancho de banda, pero teniendo en cuenta que la caída en potencia del ADC en su salida es superior a 6 dB a 2 GHz.

El espectrómetro de alta resolución obtenido, con un ancho de banda de 1,8 GHz y 32768 canales, posee una resolución espectral de 54,9 kHz. Esto corresponde a un ancho de canal 8 veces menor que lo que se tenía con los espectrómetros implementados en ROACH 1 en el Observatorio Astronómico Nacional, los cuales son de 1 GHz y 2048 canales. Además, el espectrómetro diseñado, tiene un SFDR mayor a 42 dB en toda la banda, con una caída a un mínimo de 39 dB en frecuencias puntuales. Estos valores son suficientes para su utilización en observaciones radioastronómicas.

El aumento de resolución espectral de un espectrómetro en ROACH 2 con ancho de banda superior a 1,8 GHz se ve limitado en primer lugar por problemas de sincronización del diseño. Estos problemas se producen principalmente en los bloques acumuladores (*vector accumulators*), los cuales no funcionan correctamente a altas velocidades. La segunda limitación, es la gran cantidad de recursos de memoria que requiere una cantidad grande de canales espectrales, lo que sugiere una optimización de los tipos de memoria del FPGA que utiliza el diseño.

Este trabajo presenta un avance considerable respecto a los anteriores espectrómetros del Observatorio Astronómico Nacional, tanto en ancho de banda como en resolución espectral y forma parte del comienzo del desarrollo de este tipo de herramientas en Chile. En particular, el espectrómetro diseñado tiene aplicaciones específicas en la observación de grandes nubes moleculares de dinámica compleja, siendo de gran utilidad para conocer la composición química de estas fuentes. Queda demostrado además, que el uso de FPGAs para aplicaciones radioastronómicas es tremendamente útil, proporcionando una fuerte herramienta de procesamiento en paralelo y dejando en claro que forman parte del estado del arte en espectroscopía digital.

TRABAJO FUTURO

Para aumentar el SFDR del espectrómetro implementado hay que eliminar o al menos disminuir la amplitud de los espurios de alta frecuencia producidos por el ADC. Como no es posible implementar un filtro *anti-aliasing* después de que la señal de entrada es digitalizada, la única opción factible es mejorar la calibración del ADC. Para realizar esto se proponen dos caminos:

- Realizar una calibración haciendo un barrido de frecuencias de entrada, de forma tal de encontrar los coeficientes de corrección para cada una de estas frecuencias y finalmente calcular el promedio de estos para escribirlos en los registros del ADC [16].
- Implementar un programa de optimización que vaya variando los coeficientes de corrección OGP del ADC y verificando el SFDR para así encontrar la combinación de coeficientes que implica la mínima cantidad de espurios en el espectro de la señal muestreada. Este procedimiento podría hacerse únicamente para la frecuencia de entrada con peor SFDR.

Se propone también implementar un espectrómetro de 65k canales con un ancho de banda de 1,5 GHz, ya que con el ADC funcionando a una menor velocidad se hace más factible lograr implementar correctamente el *floorplanning* del diseño.

Para disminuir los problemas de sincronización generados al correr diseños a altas velocidades en el FPGA, se propone diseñar un nuevo bloque de acumulación (*Vector Accumulator*). El que se tiene actualmente utiliza un multiplexor y un sumador. Se podría cambiar por uno basado en un DSP48E (Suraj, 2010), lo que solucionaría los problemas de sincronización que se producen dentro del bloque [21].

Para aumentar aún más la resolución espectral del espectrómetro, se propone optimizar los recursos del FPGA que utilizan los bloques de alto nivel del diseño. El diseño completo utiliza prácticamente solo las BRAM de 36 kbits, dejando libres las BRAM de 18 kbits. Se propone entonces programar bloques que utilicen también las BRAM de 18 kbits, con tal de optimizar el uso de todos los elementos de memoria disponibles en el FPGA. Al usar todas las BRAM del FPGA sería posible llegar a un espectrómetro de 128k canales. Es posible también optimizar el uso de los módulos DSP48E del FPGA para que estos realicen múltiples operaciones matemáticas independientes en un mismo componente, disminuyendo el uso de recursos del chip [21].

Finalmente, se propone implementar un espectrómetro de Fourier con un algoritmo distinto para implementar la FFT, el cual permitiría disminuir el número de operaciones matemáticas necesarias para calcular la DFT y a su vez el número de memorias utilizadas por esta. El *trade-off* que tiene este algoritmo es la necesidad de almacenar los datos temporales y espectrales no finales en matrices, generando un gran requerimiento de memoria. Pero es posible almacenar esta información en memorias externas al FPGA que contiene la ROACH 2, tales como 4 QDR SRAMs de 72 Mb cada una y una SDRAM de 16 GB. Este espectrómetro recibe el nombre de "*Corner Turner Spectrometer*" debido a la utilización de una matriz *Corner Turner* utilizada en su diseño [22] y usa el algoritmo Cooley-Turkey FFT [11] para calcular la DFT. Este espectrómetro podría llegar a tener 1 millón de canales espectrales. Esto se explica en mayor detalle en el Apéndice C.

GLOSARIO

ASIC: Del ingles, *application-specific integrated circuit.* Circuito integrado diseñado para una aplicación específica.

Back End: Último elemento electrónico en la cadena de un radiotelescopio, encargado de realizar el procesamiento de las señales radioastronómicas.

CASPER: Del inglés, *Collaboration for Astronomy Signal Processing and Electronics Research*.

DFT: Del inglés, *Discrete Fourier Transform*. Versión de la transformada de Fourier para señales discretas.

ENOB: Del inglés, *Effective number of bits, c*orresponde a una medida del desempeño dinámico de un ADC.

FFT: Del inglés, *Fast Fourier Transform.* Algoritmo eficiente para calcular la transformada de Fourier discreta.

Floorplanning: Corresponde a la distribución física de los recursos lógicos dentro de un chip FPGA, lo cual sirve para minimizar los retardos entre compuertas tal de mejorar la sincronización del circuito y así poder maximizar la frecuencia de operación de un diseño.

FPGA: Del inglés, *Field Programmable Gate Array*. Dispositivo semiconductor que contiene una matriz de componentes lógicos cuya interconexión y funcionalidad puede ser programada.

Front End: Primer grupo de elementos electrónicos en la cadena de un radiotelescopio después de la antena.

INL: Del inglés, *integral non-linearity.* No linealidad presente en el muestreo de ADCs reales.

MMCM: Del inglés, *Mixed Clock-Mode Manager*, correspondiente a un módulo del FPGA Virtex 6.

OGP: Del inglés, *Offset-Gain-Phase.*

Overflow: Error que se da en un chip o computador cuando se requiere guardar un dato muy grande para la cantidad de bits disponibles. Es posible manejar este error aproximando al mayor número posible en la representación (saturación).

Pblock: Área dentro de un chip FPGA definida para ubicar recursos lógicos específicos en ella.

Placement: Se refiere a la ubicación física de componentes lógicos dentro de un chip FPGA. Se utiliza para implementar *flooplanning* en un diseño.

ROACH: Del inglés, *Reconfigurable Open Arquitecture Computing Hardware*. Plataforma abierta basada en un chip FPGA.

Roll-Off: Corresponde a la pérdida de potencia en la salida de un componente electrónico al aumentar la frecuencia de operación.

SFDR: Del inglés, *Spurius free dinamic range*, corresponde al rango dinámico libre de espurios de una señal.

SINAD: Del inglés, *signal-to-noise and distortion ratio*, corresponde a una medida de la calidad de una señal.

Snapshot: Consiste en capturar un conjunto de datos normalmente en el dominio del tiempo y guardarlos en memoria para hacerlos accesibles por la CPU.

Timing: Se refiere a la sincronización de las compuertas lógicas de un circuito digital. El *timing* debe cumplir ciertos requerimientos para que los datos se propaguen correctamente a través de un diseño. En ocasiones se utilizó como sinónimo de este término, las palabras en español ‘sincronización’ y ‘tiempo’.

BIBLIOGRAFÍA

[1] J. Fariña, Diseño y Construcción de la Etapa Analógica de un Interferómetro de dos Atenas, 2010.

[2] J. D. Kraus, Radio Astronomy. 2da edición, 1996.

[3] «Wikipedia» [En línea]. Available: http://es.wikipedia.org/wiki/Radioastronomia. [Último acceso: 1 Diciembre 2014].

[4] R. Finger, Design and Construction of Digital Sideband Separating Spectrometer for the 1.2-Meter Southern Radio Telescope. PhD Thesis, Facultad de Ciencias Físicas y Matemáticas, U. de Chile, 2013.

[5] V. V. Tapia, Diseño, construcción y medición de una antena tipo bocina para el receptor heterodino de banda 1 de Alma. Memoria de Ingeniero Civil Eléctrico, Facultad de Ciencias Físicas y Matemáticas, U. de Chile, 2013.

[6] P. J. Astudillo, Medición del patrón de radiación del telescopio Mini. Memoria de Ingeniero Civil Eléctrico, Facultad de Ciencias Físicas y Matemáticas, U. de Chile, 2014.

[7] «ALMA,» [En línea]. Available: www.almaobservatory.org. [Último acceso: 7 Noviembre 2014].

[8] T. Wilson, Tools of Radio Astronomy. 5ta edición, 2009.

[9] B. Klein, *Back-ends for THz heterodyne systems: Fast Fourier Transform Spectrometer (FFTS), Apex Conference,* 2014.

[10] S. W. Smith, The Scientist and Engineer's Guide to Digital Signal Processing, 1997.

[11] J. G. Proakis, Digital Signal Processing. 3era edición, 1996..

[12] E. Huaracán, Diseño e implementación de un espectrómetro basado en FPGA, de ancho de banda seleccionable para aplicaciones astronómicas. Memoria de Ingeniero Civil Eléctrico, Facultad de Ciencias Físicas y Matemáticas, U. de Chile, 2014.

[13] «Wikipedia» [En línea]. Available: http://en.wikipedia.org/wiki/Hyperfine_structure.

[14] «IEEE Standard for Terminology and Test Methods for Analog-to-Digital Converters,» IEE Std 1241-2010, 2010.

[15] «EV8AQ160 Quad ADC Datasheet» [En línea]. Available: http://www.e2v.com/resources/account/download-datasheet/1798.

[16] N. Patel, Characterizing the performance of a high-speed ADC for the SMA digital backend, *Journal of Astronomical Instrumentation,* 2014.

[17] [En línea]. Available: http://masteringelectronicsdesign.com/an-adc-and-dac-integral-non-linearity-inl/.

[18] A. A. Alvear, Extensión del ancho de bvanda de espectrometros radio astronómicos basados en FPGA, Memoria de Ingeniero Civil Electrónico, Pontificia Universidad Católica de Valparaíso, 2014.

[19] A. P. R. j. D. W. Suraj Gowda, Automated Placement for Parallel, Pipelined, FPGA-based FFTs, 2010. [En línea]. Available: http://www.eecs.berkeley.edu/~sgowda/GowdaEtAl_FFTAutoplacement.pdf.

[20] H. Jiang, H. Liu, K. Guzzino, D. Kubo, C.-T. Li, R. Chang y M.-T. Chen, A 5 Giga Samples Per Second 8-bit Analog to Digital Printed Circuit Board for Radio Astronomy, *Astronomical Society of the Pacific,* 2014.

[21] S. Gowda, A 3 Gigahertz Bandwidth FPGA Spectrometer, 2010. [En línea]. Available: https://casper.ssl.berkeley.edu/wiki/images/3/30/3ghz_spec_library_design.pdf.

[22] «CASPER, Seti Spectrometer,» [En línea]. Available: https://casper.berkeley.edu/wiki/SETI_Spectrometer#DDC. [Último acceso: 3 Diciembre 2014].

APÉNDICE A

CÓDIGOS PYTHON

Calibracion.py

```
# Script written by Raul Sapunar (rasapunar@gmail.com)
# Is used to perform the calibration of the ADC ASIAA 5GSPs(OGP, INL and
ADC-FPGA Interface)
# It works with the adc5g_snapshot_zdok1_2014_Aug_06_2400.bof
# You need to have KATCP, CORR and NUMPY installed.
# Functions and libraries used in the code are from a CASPER repository:
https://github.com/sma-wideband/mlib_devel

import numpy as np
import adc5g as adc
import corr
import time
import sys
import struct
import pylab
import fit_cores
import rww_tools
import scipy.optimize
import scipy

################   START OF MAIN   ################

if __name__ == '__main__':
    from optparse import OptionParser

    p = OptionParser()
    p.set_description(__doc__)
    p.add_option('-p',                                        '--skip_prog',
dest='prog_fpga',action='store_false', default=True,
        help='Skip  FPGA  programming  (assumes  already  programmed).
Default: program the FPGAs')
    p.add_option('-v',    '--verbosity',    dest='verbosity',type='int',
default=0,
        help='Verbosity level. Default: 0')
    p.add_option('-r',        '--roach',        dest='roach',type='str',
default='172.17.89.155',
        help='ROACH IP address or hostname. Default: 172.17.89.155')
```

```
    p.add_option('-b','--boffile',dest='boffile',type='str',
default='adc5g_snapshot_zdok1_2014_Aug_06_2400.bof',
        help='Boffile to program. Default: ami_fx_sbl_wide.bof')
    p.add_option('-N',    '--n_trials',    dest='n_trials',type='int',
default=2,
        help='Number of snap/fit trials. Default: 2')
    p.add_option('-c',  '--clockrate',  dest='clockrate',  type='float',
default=None,
        help='Clock rate in MHz, for use when plotting frequency axes.
If none is given, rate will be estimated from FPGA clock')
    p.add_option('-f',  '--testfreq',  dest='testfreq',  type='float',
default=10.0,
        help='sine wave test frequency input in MHz. Default = 10')
    p.add_option('-z', '--zdok', dest='zdok', type='int', default=1,
        help='ZDOK where the ADC is connected. Default = 1')

    opts, args = p.parse_args(sys.argv[1:])

    # Connect to ROACH 2
    print 'Connecting to %s'%opts.roach
    fpga = corr.katcp_wrapper.FpgaClient(opts.roach,7147)
    time.sleep(0.2)
    print 'ROACH is connected?', fpga.is_connected()

    # Parameters for rwwtools.py
    rww_tools.roach2 = fpga
    rww_tools.freq = opts.testfreq
    rww_tools.snap_name='snapshot1'    # snapshot block name in the model
    rww_tools.samp_freq=3600.0         # Sampling rate of the ADC
    FNAME = 'snapshot_adc1_raw.dat'
    rww_tools.zdok = opts.zdok         # ZDOK connector of the ADC

    # Program FPGA with the bof file
    if opts.prog_fpga:
        print 'Programming ROACH with boffile %s'%opts.boffile
        fpga.progdev(opts.boffile)
        time.sleep(0.5)

    # Estimation of the ADC clock speed
    print 'Estimating clock speed...'
    clk_est = fpga.est_brd_clk()
    print 'Clock speed is %d MHz'%clk_est
    if opts.clockrate is None:
        clkrate = clk_est*16     # Debe dar igual a la freq de muestreo
del ADC en interleave mode (5gs)
    else:
        clkrate = opts.clockrate


    ############################################################
    ################ ADC Interface Calibration ##############
    ############################################################

    # Se ponen ambos ADC en modo de prueba
```

```
    if opts.prog_fpga:
        print 'Calibrating the time delay at the adc interface...'
      #opt0,    glitches0    =    adc.calibrate_mmcm_phase(fpga,    0,
['snapshot0',])
      opt1, glitches1 = adc.calibrate_mmcm_phase(fpga, 1, ['snapshot1',])
        time.sleep(0.5)

    ############################################################
    ################ OGP Calibration ##############
    ############################################################

    # Offset, gain and phase registers are cleaned up
    print 'Clearing OGP registers...'
    rww_tools.clear_ogp()
    time.sleep(2)

    # OGP current registers are printed on screen
    print 'OGP registers:  '
    rww_tools.get_ogp()

    print 'Doing OGP calibration...'
    #OGP calibration parameters
    rpt = 30                    # Number of repetitions to get the final
values
    snap_name='snapshot1'       # Snapshot block name (in the model)
    samp_freq=3600.0            # Sample rate of the ADC
    FNAME = 'snapshot_adc1_raw.dat'   #  Name  of  the  file  to  write
snapshot raw data
    avg_pwr_sinad = 0
    fr = opts.testfreq
    donot_clear = False

    for i in range(rpt):
      snap=adc.get_snapshot(fpga,       snap_name,       man_trig=True,
wait_period=2)
      np.savetxt(FNAME, snap,fmt='%d')
      ogp, pwr_sinad = fit_cores.fit_snap(fr, samp_freq, FNAME,\
                 clear_avgs = i == 0 and not donot_clear, prnt = i ==
rpt-1)
      avg_pwr_sinad += pwr_sinad

    sinad = avg_pwr_sinad/rpt

    ogp = ogp[3:]

    np.savetxt('ogp',ogp,fmt='%8.4f')

    print 'OGP registers:  '
    rww_tools.get_ogp()

    print 'Setting ogp'
    t = np.genfromtxt('ogp')

    rww_tools.set_offs(t[0], t[3], t[6], t[9])
```

```
    rww_tools.set_gains(t[1], t[4], t[7], t[10])
    rww_tools.set_phase(t[2], t[5], t[8], t[11])

    print 'OGP registers ADC 1:
    rww_tools.get_ogp()

    ##########################################################
    ############### INL Calibration ##############
    ##########################################################

    textname = 'snapshot_adc1_raw.dat.res'

    print 'INL corrections ... '
    rww_tools.clear_inl()
    rww_tools.get_inl()
    fit_cores.fit_inl(textname)

    rww_tools.set_inl('inl.meas')

    print 'INL registers: '
    rww_tools.get_inl()

    print 'done'

    exit()
```

Saturation_control.py

```
# Script written by Raul Sapunar (rasapunar@gmail.com)
# Is used to obtain the ADC saturation (input level) and plot the sampling
(snapshot)
# It works with the adc5g_saturation_status_2014_Aug_12_1651.bof
# You need to have KATCP, CORR and NUMPY installed.

import      corr,time,struct,sys,logging,pylab,matplotlib,math,Gnuplot,
Gnuplot.funcutils,array
from math import *
import adc5g
import numpy

bitstream = 'adc5g_saturation_snapshot_2014_Aug_08_1317.bof'
katcp_port=7147

def exit_fail():
    print 'FAILURE DETECTED. Log entries:\n',lh.printMessages()
    try:
        fpga.stop()
    except: pass
    raise
    exit()

def exit_clean():
```

```
    try:
        fpga.stop()
    except: pass
    exit()

data = []

def get_data():
    #get the data...
    data = numpy.array(adc5g.get_snapshot(fpga, "snapshot1"))
    return data

def continuous_plot(fpga):
        ok=1
        sat_value=0
        sat_porcentaje=0
        counter = 0
      n2 = 0
        bw=trunc(fpga.est_brd_clk())*8
      fpga.write_int('reset_var',75497471)

      g1.clear()
        g1.title('ADC1 Snapshot of a low frequency (10 MHz)'+bitstream+'
| Max frequency = '+str(bw)+' MHz')
        g1.xlabel('Sample number')
        g1.ylabel('ADC output')
        g1('set style data linespoints')
      g1('set yrange [-132:132]')
      g1('set xrange [0:500]')
      g1('set ytics 5')
      g1('set xtics 32')
      g1('set grid y')
      g1('set grid x')

        while ok==1 :
           samples = get_data()
           g1.plot(samples)
                sat_value = fpga.read_uint('max_value')
                sat_porcentaje = (sat_value-131)*100
                sat_porcentaje = 1.*sat_porcentaje/124
                n2 = "%.2f" % sat_porcentaje
                g1.title('ADC1  Snapshot  of  a  low  frequency  (10
MHz)'+bitstream+' | Max frequency = '+str(bw)+' MHz / max_sat = '+
str(sat_value)+' / '+n2+' %')
           time.sleep(0.3)

################   START OF MAIN   ################

if __name__ == '__main__':
    from optparse import OptionParser

    p = OptionParser()
    p.set_usage('spectrometer.py <ROACH_HOSTNAME_or_IP> [options]')
    p.set_description(__doc__)
```

```
    p.add_option('-s', '--skip', dest='skip', action='store_true',
        help='Skip reprogramming the FPGA and configuring EQ.')
    p.add_option('-b', '--bof', dest='boffile',type='str', default='',
        help='Specify the bof file to load')
    opts, args = p.parse_args(sys.argv[1:])

    if args==[]:
        print 'Please specify a ROACH board. Run with the -h flag to see
all options.\nExiting.'
        exit()
    else:
        roach = args[0]
    if opts.boffile != '':
        bitstream = opts.boffile

try:
    loggers = []
    lh=corr.log_handlers.DebugLogHandler()
    logger = logging.getLogger(roach)
    logger.addHandler(lh)
    logger.setLevel(10)

    print('Connecting to server %s on port %i... '%(roach,katcp_port)),
    fpga     =     corr.katcp_wrapper.FpgaClient(roach,     katcp_port,
timeout=10,logger=logger)
    time.sleep(1)

    if fpga.is_connected():
        print 'ok\n'
    else:
        print    'ERROR    connecting    to    server    %s    on    port
%i.\n'%(roach,katcp_port)
        exit_fail()

    print '------------------------'
    print 'Programming FPGA with %s...' %bitstream,
    if not opts.skip:
        fpga.progdev(bitstream)
        print 'done'
    else:
        print 'Skipped.'

    print 'waiting 3 seconds...'
    time.sleep(3)

    print 'Calibrating the time delay at the adc interface...'
    #opt0, glitches0 = adc.calibrate_mmcm_phase(fpga, 0, ['snapshot0',])
    opt1,     glitches1     =     adc5g.calibrate_mmcm_phase(fpga,     1,
['snapshot1',])

    #set up the figure with a subplot to be plotted
    g1 = Gnuplot.Gnuplot(debug=1)
    continuous_plot(fpga)
    print 'Plot started.'
```

```
except KeyboardInterrupt:
    exit_clean()
except:
    exit_fail()

exit_clean()
```

ADC_performance.py

```
# Script written by Raul Sapunar (rasapunar@gmail.com)
#  Is used to perform the performance tests of the ADC ASIAA
5GSPs(Frequency Response and SDFDR)
# It works with the adc5g_snapshot_zdok1_2014_Aug_06_2400.bof
# You need to have KATCP, CORR and NUMPY installed.
#  Some functions and libraries used in the code are from a CASPER
repository: https://github.com/sma-wideband/mlib_devel

import numpy as np
import adc5g as adc
import corr
import time
import sys
import struct
import pylab
import fit_cores
import rww_tools
import scipy
import telnetlib
import math
import matplotlib.pyplot as plt
from matplotlib import mlab

# Parameters
numpoints=16384
samp_freq = 4800.0
snap_name = "snapshot1"

################   START OF MAIN   ################

if __name__ == '__main__':
    from optparse import OptionParser

    p = OptionParser()
    p.set_description(__doc__)
    p.add_option('-v',    '--verbosity',    dest='verbosity',type='int',
default=0,
        help='Verbosity level. Default: 0')
    p.add_option('-r',        '--roach',        dest='roach',type='str',
default='172.17.89.155',
        help='ROACH IP address or hostname. Default: 172.17.89.155')
```

```
    p.add_option('-b',      '--boffile',      dest='boffile',type='str',
default='adc5g_snapshot_zdok1_2014_Aug_06_2400.bof',
        help='Boffile to program. Default: ami_fx_sbl_wide.bof')
    p.add_option('-f',   '--testfreq',   dest='testfreq',   type='float',
default=10.0,
        help='sine wave test frequency input in MHz. Default = 10')
    p.add_option('-z', '--zdok', dest='zdok', type='int', default=1,
        help='ZDOK where the ADC is connected. Default = 1')

    opts, args = p.parse_args(sys.argv[1:])

    # Connect to ROACH 2
    print 'Connecting to %s'%opts.roach
    fpga = corr.katcp_wrapper.FpgaClient(opts.roach,7147)
    time.sleep(0.2)
    print 'ROACH is connected?', fpga.is_connected()

    # Parameters of rww_tools.py
    rww_tools.roach2 = fpga
    rww_tools.snap_name=snap_name      # nombre del snapshot del ADC
    rww_tools.samp_freq=samp_freq           # frecuencia de muestreo del
adc en interleave mode
    FNAME = 'snapshot_adc0_raw.dat'
    rww_tools.zdok = opts.zdok         #  se  setea  ADC  que  se  quiere
calibrar

    # Program FPGA with the boffile
    if opts.prog_fpga:
        print 'Programming ROACH with boffile %s'%opts.boffile
        fpga.progdev(opts.boffile)
        time.sleep(0.5)

    # Estimation of the clock speed of the ADC
    print 'Estimating clock speed...'
    clk_est = fpga.est_brd_clk()
    print 'Clock speed is %d MHz'%clk_est
    if opts.clockrate is None:
        clkrate = clk_est*16     # Debe dar igual a la freq de muestreo
del ADC en interleave mode (5gs)
    else:
        clkrate = opts.clockrate

    print 'Calibrating the time delay at the adc interface...'
    #opt0, glitches0 = adc.calibrate_mmcm_phase(fpga, 0, ['snapshot0',])
    opt1, glitches1 = adc.calibrate_mmcm_phase(fpga, 1, ['snapshot1',])
    time.sleep(0.5)

    ##################################################################
    ################ Frequency Response and SFDR #####################
    ################ from 100 MHz to 2400 MHz    #####################
    ##################################################################

    start = 100
    end = 2400
```

```
    delta = 50
    repeat = 10
    powerlevel = 7

    #For controlling signal generator via Ethernet
    agi = telnetlib.Telnet("172.17.89.54",5025)
    agi.write("output on\r\n")
    agi.write("power -0.1dbm\r\n")

    frfile = open('freqResponse.dat', 'a')
    frfile2 = open('sfdr.dat', 'a')
    f = samp_freq / numpoints
    nstart = int(0.5+start/f)
    nend = int(0.5+end/f)
    nstep = int(0.5+delta/f)
    freq_array = []
    maxpower_array = []
    fi = 0

    for n in range(nstart, nend, nstep):
      freq = f*n
      agi.write("freq "+ str(freq) +"mhz\r\n")
      time.sleep(0.5)

      for i in range(repeat):
        power, freqs = adc.get_psd(fpga, snap_name, samp_freq*1e6, 8,
numpoints)
        if i == 0:
          sp = power
        else:
          sp += power
        sp /= repeat

      power=np.log10(sp)
      power = 10*power
      peakpower=max(power)

      res = heapq.nlargest(10, power)
      sfdr = (res[0] - res[3])

      freq_array.append(freq)
      maxpower_array.append(peakpower)
      #print freq,peakpower
      output="%f %f\n" % (freq,peakpower)
      output2="%f %f\n" % (freq,sfdr)

      frfile.write(output)
      frfile2.write(output2)
    frfile.close()
    frfile2.close()
    agi.write("output off\r\n")

    print 'done'
    exit()
```

Roach2_spec_32kch_snap.py

```
# Script written by Raul Sapunar (rasapunar@gmail.com), based on script
of CASPER Tut 3
# This script calibrate the ADC-FPGA interface delay, configure a 32K
wideband spectrometer
# and plot the received data.
# It works with the spec1_r2_1800_16_snap_2014_Oct_09_1412.bof
# You need to have KATCP and CORR Installed.

import
corr,time,numpy,struct,sys,logging,pylab,matplotlib,math,Gnuplot,
Gnuplot.funcutils,array
import adc5g
from math import *

bitstream = 'spec1_roach2_1200_2014_Jul_07_1524.bof'
katcp_port=7147

def exit_fail():
    print 'FAILURE DETECTED. Log entries:\n',lh.printMessages()
    try:
        fpga.stop()
    except: pass
    raise
    exit()

def exit_clean():
    try:
        fpga.stop()
    except: pass
    exit()

def get_data():
    #Get snapshot data
    snap1 = []
    snap1 = numpy.array(adc5g.get_snapshot(fpga, "snapshot1"))

    #get the data from de RAM
    acc_n = fpga.read_uint('acc_cnt')

    length1 = 4096
    bytewidth = 8


a_0=struct.unpack('>4096Q',fpga.read('dout0_0',length1*bytewidth,0))

a_1=struct.unpack('>4096Q',fpga.read('dout0_1',length1*bytewidth,0))

a_2=struct.unpack('>4096Q',fpga.read('dout0_2',length1*bytewidth,0))
```

```
a_3=struct.unpack('>4096Q',fpga.read('dout0_3',length1*bytewidth,0))

a_4=struct.unpack('>4096Q',fpga.read('dout0_4',length1*bytewidth,0))

a_5=struct.unpack('>4096Q',fpga.read('dout0_5',length1*bytewidth,0))

a_6=struct.unpack('>4096Q',fpga.read('dout0_6',length1*bytewidth,0))

a_7=struct.unpack('>4096Q',fpga.read('dout0_7',length1*bytewidth,0))

    interleave_a=[]
    interleave_log=[]

    shift = 18

    for i in range(4096):

interleave_a.append(float((a_0[i]/((2**shift)*opts.acc_len))+1))

interleave_a.append(float((a_1[i]/((2**shift)*opts.acc_len))+1))
     interleave_a.append(float((a_2[i]/((2**shift)*opts.acc_len))+1))
     interleave_a.append(float((a_3[i]/((2**shift)*opts.acc_len))+1))
     interleave_a.append(float((a_4[i]/((2**shift)*opts.acc_len))+1))
     interleave_a.append(float((a_5[i]/((2**shift)*opts.acc_len))+1))
     interleave_a.append(float((a_6[i]/((2**shift)*opts.acc_len))+1))
     interleave_a.append(float((a_7[i]/((2**shift)*opts.acc_len))+1))

    for k in range(8*4096):
        interleave_log.append(10*log10(interleave_a[k]))
    return acc_n, interleave_a, interleave_log, snap1

def continuous_plot(fpga):
        ok=1
        bw=trunc(fpga.est_brd_clk())*8

     g1.clear()
        g1.title('ADC1  ROACH  2  spectrum  using  '+bitstream+'  |  Max
frequency = '+str(bw)+' MHz')
        g1.xlabel('Channel #')
        g1.ylabel('Power AU (dB)')
        g1('set style data linespoints')
     g1('set yrange [-20:120]')
     g1('set xrange [-50:33000]')
     g1('set ytics 5')
     g1('set xtics 2048')
     g1('set grid y')
     g1('set grid x')

        while ok==1 :
           acc_n, interleave_a, interleave_log, snap1 = get_data()
                g1.plot(interleave_log)
           time.sleep(0.3)
```

```
################   START OF MAIN   ################

if __name__ == '__main__':
    from optparse import OptionParser
    p = OptionParser()
    p.set_usage('spectrometer.py <ROACH_HOSTNAME_or_IP> [options]')
    p.set_description(__doc__)
    p.add_option('-l',           '--acc_len',           dest='acc_len',
type='int',default=2*(2**28)/2048,
        help='Set the number of vectors to accumulate between dumps.
default is 2*(2^28)/2048, or just under 2 seconds.')
    p.add_option('-g',              '--gain',              dest='gain',
type='int',default=0x00001000,
        help='Set the digital gain (6bit quantisation scalar). Default
is 0xffffffff (max), good for wideband noise. Set lower for CW tones.')
    p.add_option('-s', '--skip', dest='skip', action='store_true',
        help='Skip reprogramming the FPGA and configuring EQ.')
    p.add_option('-b', '--bof', dest='boffile',type='str', default='',
        help='Specify the bof file to load')
    opts, args = p.parse_args(sys.argv[1:])

    if args==[]:
        print 'Please specify a ROACH board. Run with the -h flag to see
all options.\nExiting.'
        exit()
    else:
        roach = args[0]
    if opts.boffile != '':
        bitstream = opts.boffile

try:
    loggers = []
    lh=corr.log_handlers.DebugLogHandler()
    logger = logging.getLogger(roach)
    logger.addHandler(lh)
    logger.setLevel(10)

    print('Connecting to server %s on port %i... '%(roach,katcp_port)),
    fpga     =     corr.katcp_wrapper.FpgaClient(roach,     katcp_port,
timeout=10,logger=logger)
    time.sleep(1)

    if fpga.is_connected():
        print 'ok\n'
    else:
        print    'ERROR    connecting    to    server    %s    on    port
%i.\n'%(roach,katcp_port)
        exit_fail()

    print 'Programming FPGA with %s...' %bitstream,
    if not opts.skip:
        fpga.progdev(bitstream)
        print 'done'
    else:
```

```
        print 'Skipped.'

    print 'Calibrating the time delay at the adc interface...'
    #opt0, glitches0 = adc.calibrate_mmcm_phase(fpga, 0, ['snapshot0',])
    opt1,    glitches1    =    adc5g.calibrate_mmcm_phase(fpga,    1,
['snapshot1',])
    time.sleep(0.5)

    print 'Configuring accumulation period...',
    fpga.write_int('acc_len',opts.acc_len)
    print 'done'

    print 'Resetting counters...',
    fpga.write_int('cnt_rst',1)
    fpga.write_int('cnt_rst',0)
    print 'done'

    print 'Setting digital gain of all channels to %i...'%opts.gain,
    if not opts.skip:
        fpga.write_int('gain',opts.gain) #write the same gain for all
inputs, all channels
        print 'done'
    else:
        print 'Skipped.'

    #set up the figure with a subplot to be plotted
    g1 = Gnuplot.Gnuplot(debug=1)
    continuous_plot(fpga)
    print 'Plot started.'

except KeyboardInterrupt:
    exit_clean()
except:
    exit_fail()

exit_clean()
```

APÉNDICE B

RESULTADOS SPECTRUM ANALYZER

Se presentan los espectros obtenidos con un Agilent CXA Signal Analizer N9000A de las señales utilizadas para caracterizar los espectrómetros, es decir de sinusoides de 450 MHz, 659,18 MHz, 1350 MHz y 1648 MHz. No se agregó un piso de ruido para esta prueba ya que el objetivo es únicamente detectar los espurios reales provenientes del generador de señales utilizado.

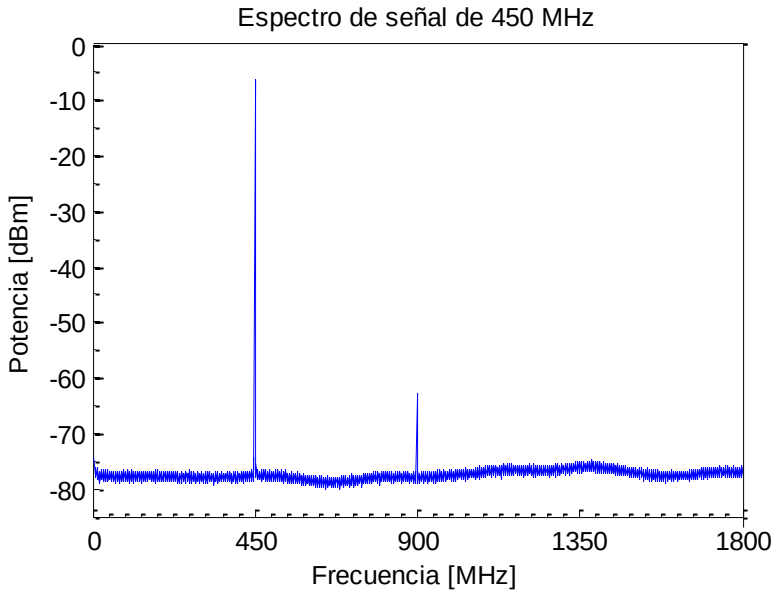


Figura B.1: Espectro de señal de 450 MHz usando Spectrum Analyzer

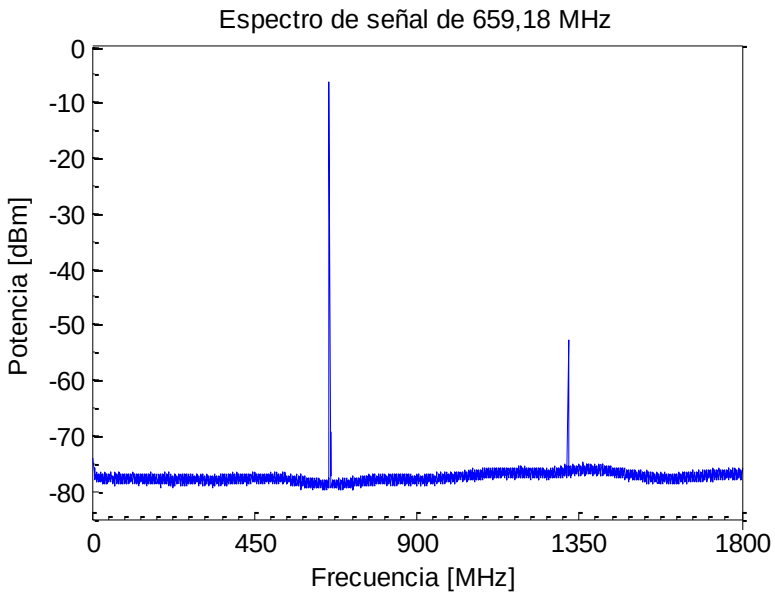


Figura B.2: Espectro de señal de 659,18 MHz usando Spectrum Analyzer

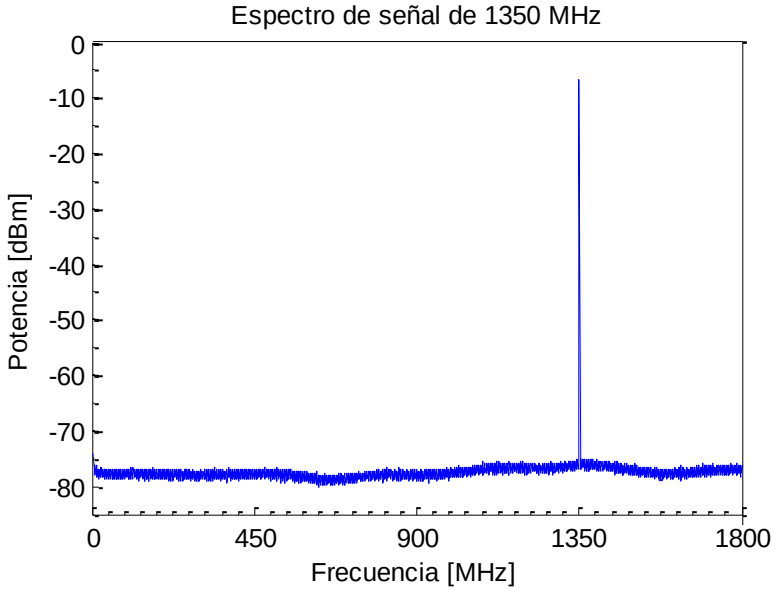


Figura B.3: Espectro de señal de 1350 MHz usando Spectrum Analyzer

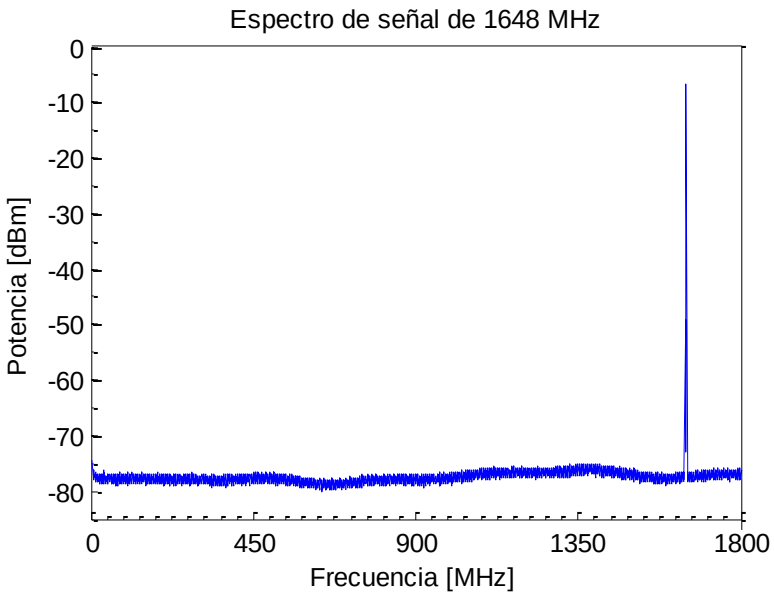


Figura B.4: Espectro de señal de 1648 MHz usando Spectrum Analyzer

APÉNDICE C

CORNER TURNER SPECTROMETER

Este espectrómetro utiliza el algoritmo Cooley-Tukey FFT, el cual expresa la DFT de largo $N = N_1 N_2$ en términos de dos DFT más pequeñas, de tamaños N_1 y N_2. Para realizar esto el algoritmo sigue los siguientes pasos [11]:

1. Almacenar las muestras temporales provenientes de los ADC en una matriz de $N_2 \times N_1$, donde las filas correspondan a datos continuos en el tiempo.
2. Computar una DFT de N_2 puntos a cada columna de la matriz, generando una matriz de las mismas dimensiones en el dominio de la frecuencia.
3. Multiplicar cada uno de los *N* datos espectrales obtenidos por un factor, el cual es único para cada dato. Estos son conocidos como "*twiddle factors"*
4. Computar una DFT de N_1 puntos para cada fila de la matriz ya ponderada por los factores, obteniendo una matriz de las mismas dimensiones
5. Se leen los datos espectrales por columna para obtener los *N* valores espectrales equivalentes a haber utilizado una DFT compleja de *N* puntos.

A modo general, este algoritmo factoriza los índices *k* y n de la DFT compleja ecuación (2. 12) como $k = N_2 k_1 + k_2$ y $n = N_1 n_2 + n_1$. Así, la DFT compleja queda expresada de la siguiente forma:

$$X(N_2 k_1 + k_2) = \sum_{n_1=0}^{N_1-1} \left[e^{\frac{-j2\pi n_1 k_2}{N}} \right] \left(\sum_{n_2=0}^{N_2-1} x(N_1 n_2 + n_1)\, e^{\frac{-j2\pi n_2 k_2}{N_2}} \right) e^{\frac{-j2\pi n_1 k_2}{N_1}} \qquad \text{(C. 1)}$$

Para $0 \leq k_1 \leq N_1 - 1$ y $0 \leq k_2 \leq N_2 - 1$

El primer y último término exponencial que aparecen en la ecuación (C. 1) corresponden a los "*twiddle factors".*

En la Figura C.1 se presenta un diagrama que resume el algoritmo explicado anteriormente.

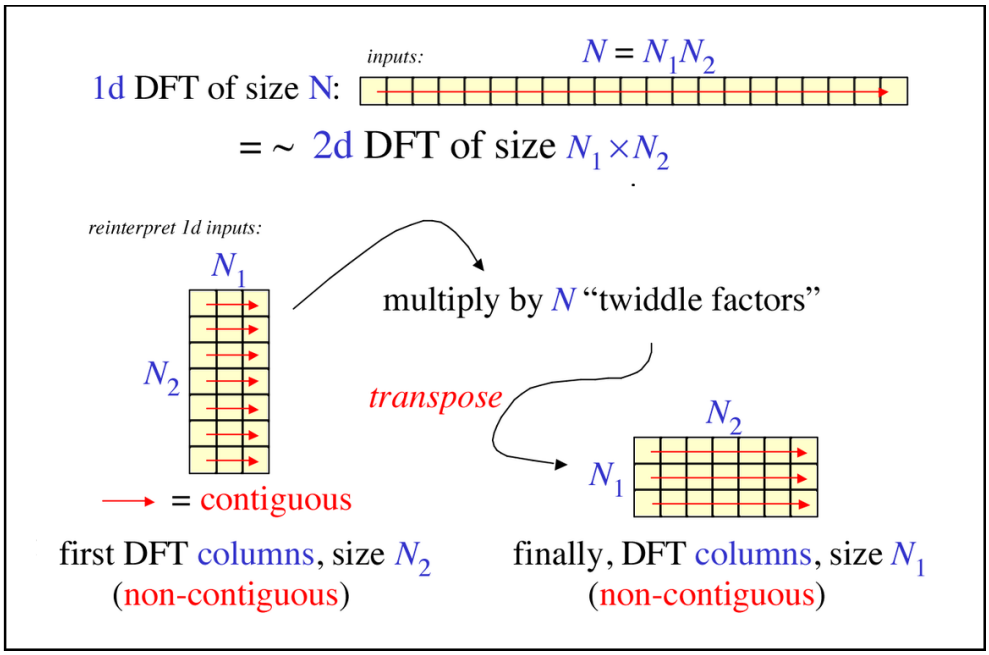


Figura C.1: Algoritmo Cooley –Turkey (primer método)[27]

El mismo algoritmo para calcular la FFT puede ser implementado de una segunda forma, en la que las muestras temporales se ordenan en las columnas de la matriz de $N_2 \times N_1$, para luego calcular N_1 DFTs de N_1 puntos (sobre cada fila). Posteriormente se multiplica cada valor obtenido por cada *twiddle factor*, para finalmente computar DFTs de N_2 puntos de cada columna. Para obtener los N valores espectrales equivalentes a haber utilizado una DFT compleja de N puntos se leen los valores de la última matriz obtenida de fila en fila. En la Figura C.2 a continuación, se muestra un diagrama de esta versión del algoritmo.

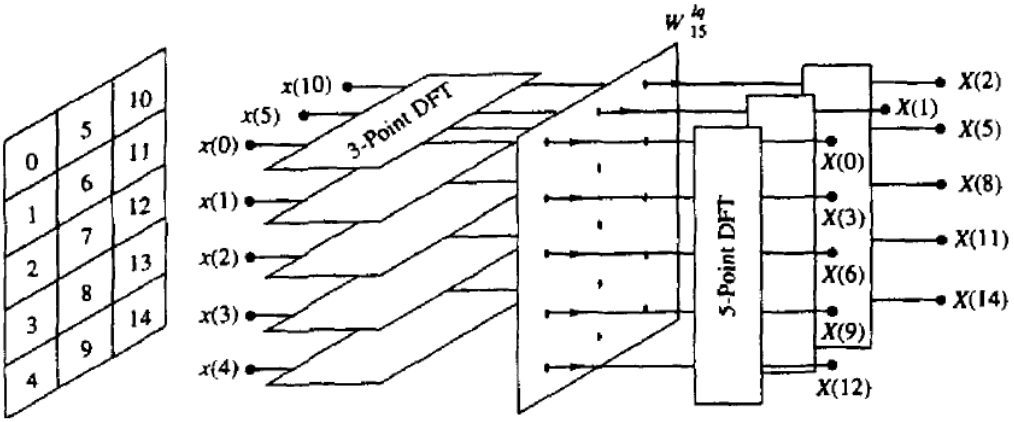


Figura C.2: Algoritmo Cooley-Turkey (segundo método)[28]

La matriz intermedia entre ambas DFT se conoce como "Corner Turner", donde el reordenamiento de los datos puede realizarse leyendo y escribiendo datos sobre la misma matriz guardada en una memoria.

[27] Figura extraída de http://en.wikipedia.org/wiki/Cooley%E2%80%93Tukey_FFT_algorithm
[28] Figura extraída de Proakis [11]

Como este algoritmo usa DFTs de menor tamaño, es posible conservar el algoritmo FFT utilizado en los espectrómetros diseñados en ROACH 2 para calcular dichas transformadas, es decir el algoritmo Radix-2 con *Pipeline.* Esto implica que los valores N, N_1 y N_2 deben ser potencias de 2.

Para implementar este algoritmo FFT en ROACH 2 se recomienda usar las memorias QDR para la primera matriz de datos temporales y para la matriz Corner Turner, ya que dichas memorias son de escritura y lectura rápida. Para almacenar los N datos espectrales finales (*buffer)*, se recomienda utilizar la memoria DRAM externa. De esta forma las BRAM del FPGA se utilizarán únicamente para guardar datos internos dentro de los bloques del diseño del espectrómetro, tales como el filtro FIR y la FFT.