Trabajo Fin de Grado

Ingeniería Electrónica, Robótica y Mecatrónica

Mención en Instrumentación Electrónica y Control

Sobre el Uso de Técnicas Chopper para la Reducción del Ruido Flicker en Amplificadores para la Captación de Señales Neuronales

Autor: Norberto Pérez Prieto

Tutor: Ángel Rodríguez Vázquez

Dep. de Electrónica y Electromagnetismo

Área de Electrónica

Escuela Técnica Superior de Ingeniería

Sevilla, 2016

Trabajo Fin de Grado
Ingeniería Electrónica, Robótica y Mecatrónica
Mención en Instrumentación Electrónica y Control

Sobre el Uso de Técnicas Chopper para la Reducción del Ruido Flicker en Amplificadores para la Captación de Señales Neuronales

Autor:
Norberto Pérez Prieto

Tutor:
Ángel Rodríguez Vázquez
Catedrático de Universidad

Dep. de Electrónica y Electromagnetismo
Área de Electrónica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2016

Trabajo Fin de Grado: Sobre el Uso de Técnicas Chopper para la Reducción del Ruido Flicker en Amplificadores para la Captación de Señales Neuronales

Autor: Norberto Pérez Prieto

Tutor: Ángel Rodríguez Vázquez

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

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2016

El Secretario del Tribunal

A mi familia y amigos, por creer en mí y apoyarme desde el principio de esta aventura…

Agradecimientos

A mi tutor, Ángel Rodríguez Vázquez, no solo por darme la oportunidad de embarcar en este reto mutuo, sino por todo el apoyo, orientación y confianza aportados durante el desarrollo de este proyecto, especialmente en los momentos más difíciles del mismo.

A José Luis Valtierra Sánchez de la Vega y Rocío del Río Fernández, por prestar su desinteresada ayuda siempre que fue necesario.

A todos mis compañeros del Instituto de Microelectrónica de Sevilla y de la facultad, por hacer el día a día de este proyecto mucho más ameno.

A mi Rayo, por hacer que cada día empezara de una forma distinta y divertida y aportarme muchos momentos para recordar.

A todos los que en los momentos difíciles estuvieron ahí y me tendieron la mano cuando la necesité: Yennifer, Alberto, Germán, Andrés, Valentín, Antonio… Y en especial a mi familia.

Y por supuesto a mis padres y María, por ser mis pilares en todo momento y no dejar que cayera nunca.

Norberto Pérez Prieto

Sevilla, 2016

Resumen

La captación de señales neuronales mediante electrodos conectados a circuitos micro-electrónicos es necesaria para aplicaciones clínicas y para el control de prótesis senso-motoras, entre otras muchas aplicaciones bio-médicas. En todas estas aplicaciones, la preservación de la información contenida en las imágenes captadas depende críticamente de las prestaciones de los amplificadores empleados en la cabecera de la cadena de procesamiento electrónica. El problema es que se trata de señales muy débiles (rango de μV) y de baja frecuencia (rango de sub-Hz), lo que implica una enorme influencia del ruido flicker. Al margen de esta influencia, el diseño de estos amplificadores, y de las cadenas de procesamiento completas, está condicionado por restricciones severas de área y consumo de potencia.

En el Instituto de Microelectrónica de Sevilla está activa una línea de investigación sobre el diseño de interfaces de señal-mixta para captación de señales neuronales. Se han concebido, prototipado en forma de chips y validado mediante medidas "in-vitro" e "in-vivo" chips con 64 canales, con calibración "on-chip" y compresión de la señal "on-chip", con captación de energía mediante enlaces inductivos. Estos circuitos emplean amplificadores seleccionados mediante técnicas de optimización para conseguir mínimo ruido con mínimo consumo de potencia. Sin embargo, no incluyen técnicas específicas para la reducción del ruido flicker. Además, estudios posteriores han permitido vislumbrar la posibilidad de mejorar las topologías de amplificadores, en particular usando la topología denominada *active-feedback time constant enhanced neural amplifier,* que se presenta en el Capítulo1 de esta Memoria.

Este trabajo Fin de Grado se propone con el objetivo de desarrollar modelos y técnicas para reducción del ruido flicker en amplificadores neuronales, con una doble perspectiva:

- Modelar dicho ruido en este tipo de amplificadores con vistas a la optimización del diseño de los mismos.
- Incorporar técnicas de modulación Chopper en los amplificadores neuronales y evaluar su impacto sobre las prestaciones de los amplificadores. En particular, estudiar, a nivel de modelos eléctricos, cómo afecta la aplicación de la técnica de Chopper en amplificadores del tipo *active-feedback time constant enhanced neural amplifier*.
- Explorar la posibilidad de generar ruido flicker mediante circuitos simples, adecuados para ser embebidos "on-chip" en sistemas de captación de señales neuronales, con los propósitos, no explorados en este trabajo, de auto-testado y calibración.

Los modelos y técnicas propuestas nos han permitido reducir hasta 40dB la potencia del ruido en el amplificador para frecuencias inferiores a 1 Hz, lo cual nos permite constatar la validez de los resultados. De hecho, sobre la base de estos resultados, se está trabajando en la actualidad para diseñar y prototipar un chip que integra las soluciones propuestas en este trabajo. Respecto a la generación de ruido "on-chip" se han propuesto combinaciones de mapas discretos que pueden ser parametrizados para obtener densidades espectrales de potencia con distribución frecuencia propia de distintos tipos de ruido pertinentes para los objetivos del trabajo.

Abstract

The uptake of neural signals through electrodes attached to micro-electronic circuits is needed for clinical applications and control sensorimotor prostheses, among many other bio-medical applications. In all these applications, the preservation of the information contained in the captured images depends critically on the performance of the amplifiers used in the header of the electronic processing chain. The problem is that there are very weak (μV range) and low frequency (sub - Hz range) signals, which implies a huge influence of flicker noise. Apart from this influence, the design of these amplifiers, and complete processing chain, is conditioned by severe restrictions of area and power consumption.

At the Institute of Microelectronics of Seville there is an active research on the interface design-mixed signal to seize neural signals. They are designed, prototyping in the form of chips and validated using measures "in-vitro" and "in-vivo" chips with 64 channels, calibrated on-chip and signal compression on-chip with capture energy by inductive links. These circuits employ amplifiers selected by optimization techniques to achieve minimal noise with minimal power consumption. However, they do not include specific techniques to reduce the flicker noise. In addition, further studies have allowed to glimpse the possibility of improving this kind of amplifiers, particularly using the topology called *active-feedback time constant neural enhanced amplifier*, which is presented in Chapter 1 of this Report.

This work is proposed with the aim of developing models and techniques for reducing flicker noise in neural amplifiers with a dual perspective:

- Modeling such noise in this type of amplifiers with the purpose of optimizing their design.
- Incorporate Chopper modulation techniques in neural amplifiers and assess their impact on the performance of amplifiers. In particular, to study at electric models level, how the application of the technique Chopper affects amplifiers of the *active-feedback amplifier time constant neural enhanced* kind.
- Explore the possibility to generate flicker noise by simple circuits, suitables for embedded on-chip systems for capturing neural signals, for purposes not explored in this work, as self-testing and calibration.

The proposed models and techniques have allowed us to reduce up to 40dB noise power in the amplifier for less than 1 Hz frequencies, which let us verify the validity of the results. In fact, there is currently some work based on these results to design and prototype a chip that integrates the solutions proposed in this project. Regarding noise generation on-chip, it has been proposed combinations of discrete maps that can be parameterized to obtain spectral power densities with natural frequency distribution of different types of noise, relevant to the objectives of the work.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1 INTRODUCCIÓN

Los amplificadores usados para captación de señales neuronales deben manejar señales que van desde los potenciales de campo locales (LFP) hasta los potenciales de acción (AP). Estas ondas cerebrales, por lo general, se caracterizan por ser de muy baja amplitud y frecuencia; excepto por los potenciales de acción que llegan a los 7kHz, el resto no pasan de los 100Hz. Es por ello que el ruido Flicker tiene un impacto muy importante sobre la operación de los amplificadores neuronales. Dado que el ruido Flicker está, en términos cualitativos y entre otros factores, asociado a la falta de homogeneidad de los materiales, una forma de corregirlo es aumentar las dimensiones de los dispositivos. Sin embargo, esta estrategia compromete en gran medida los estrictos requerimientos de área y consumo de potencia con los que deben diseñarse los amplificadores neuronales. Por una parte, el área debe ser mínima porque las aplicaciones prácticas requieren captar conjuntos, tan grandes como sea posible de señales; esto es, más que la señal de una neurona aislada, interesa captar las señales de un conjunto de neuronas, tanto las variaciones espaciales como las variaciones temporales. Para esto se usan "arrays" de electrodos y se busca que el espaciado entre ellos ("pitch") sea el menor posible. Por otra parte, el consumo de potencia debe ser mínimo para evitar calentamientos de los tejidos y aliviar la carga de la circuitería de gestión de la energía. Y tamaños grandes significa condensadores parásitos grandes y, por lo tanto, mayor consumo de potencia.

Una técnica usada tradicionalmente para reducir el impacto del ruido Flicker y el offset DC de los circuitos amplificadores es la modulación Chopper. En este Trabajo Fin de Grado se aborda la viabilidad de esta técnica en amplificadores neuronales, con énfasis particular en la topología denominada *active-feedback time constant neural enhanced amplifier*. Dada las limitaciones temporales propias de un TFG, el análisis y la valoración se limitan al nivel de modelado, aunque los resultados obtenidos se pretende que sirvan de base para el desarrollo de pruebas de concepto en forma de circuitos integrados.

Junto con el modelado y el análisis, se desarrollan técnicas para la generación de ruido usando mapas discretos sintetizables mediante circuitos de señal-mixta que usan solo comparadores como elementos no-lineales y que, por tanto, pueden ser fácilmente embebidos en un chip multi-canal para las funciones de auto-verificación y calibración.

1.1 Neural Recording: concepto y arquitectura típica de canal

Neural Recording ha sido uno de los campos de la neuroingeniería más estudiado en los últimos años debido a su efectividad a la hora de tratar enfermedades neuronales como la epilepsia, el Parkinson o el Alzheimer [1]. Además, el nacimiento y desarrollo de interfaces capaces de controlar manipuladores robóticos a través de la información recopilada y procesada proveniente del cerebro, conocidas como Interfaces Cerebro-Máquina (BMIs) [2] (figura 1-1) han supuesto también un aliciente para la evolución las técnicas de Neural Recording.

FIGURA 1-1. INTERFAZ CEREBRO-MAQUINA CON BRAZO MANIPULADOR [3].

Las técnicas de Neural Recording se clasifican en función de su nivel de invasión en el cerebro [2], siendo la menos invasiva el encefalograma (EEG), que recoge la actividad de millones de neuronas mediante la colocación de un grupo de electrodos en la superficie del cráneo.

Ascendiendo en esta escala, se encuentran las electrocortigrafías (ECoG), donde los electrodos se colocan directamente en la superficie del cerebro. Como escalón más alto de invasión, están los micro-electrodos colocados en el cerebro que son capaces de capturar la información del potencial de acción de una solo neurona [1].

Los circuitos con los que se trabajará en este proyecto tendrán este nivel de invasión, ya que actualmente se están realizando muchos avances gracias a ellos en múltiples áreas, como la detección de la epilepsia [4].

Las características que deben cumplir las arquitecturas implicadas en Neural Recording son las siguientes [1]:

- Biocompatibilidad de los materiales del sensor. Normalmente la circuitería es cubierta con silicona.
- Miniaturización de los dispositivos sin perder la alta resolución de los mismos.
- Comunicación inalámbrica para evitar el uso de cables que aumentan el riesgo de infección.
- Mínimo consumo para evitar el uso de grandes baterías y el reducir el riesgo de trabajar a altas temperaturas.
- Buenas técnicas de compresión de datos para reducir la cantidad de datos transmitidos.

Como ya se mencionó anteriormente, el Neural Recording engloba desde la captura del estímulo eléctrico hasta el envío de la señal digital preparada para su procesamiento. Por ello, la arquitectura típica de canal que se usa es la que muestra la figura 1-2. Hay que tener en cuenta que esta arquitectura es para captar la información de una a cuatro neuronas. Por lo tanto, para captar la información proveniente de un grupo de neuronas correspondiente a una región cerebral concreta se debe emplear una arquitectura multicanal.

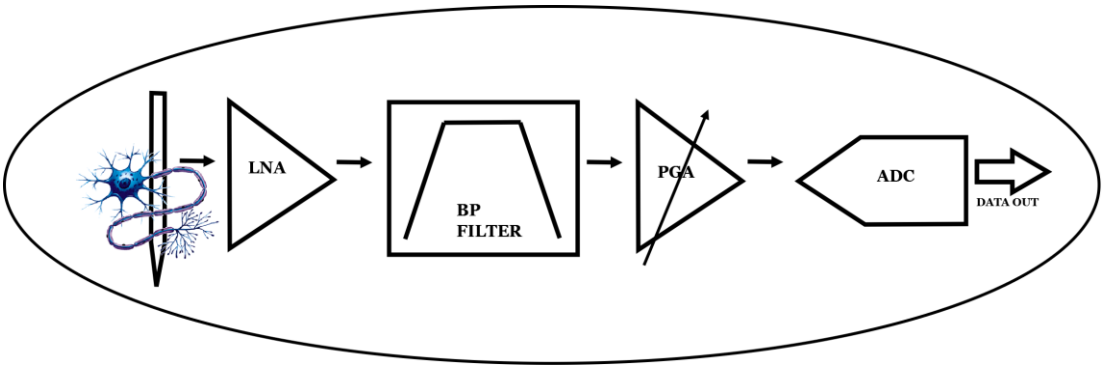


FIGURA 1-2. ARQUITECTURA TÍPICA DE CANAL EMPLEADA EN NEURAL RECORDING.

Los componentes del canal, excepto por el amplificador de bajo ruido (LNA) al que se le dedicará una atención especial en el siguiente apartado, son descritos [1] brevemente a continuación:

- Filtro Paso Banda (BP FILTER): Debido a la interfaz de los electrodos, se producen grandes tensiones dc de offsets. Además, las señales implicadas en Neural Recording, como se mostrará en el siguiente apartado, van de 1 Hz a 7kHz, por lo que todas las frecuencias que no estén dentro de ese ancho de banda pueden provocar errores en la arquitectura. Por lo general, el filtro paso banda se incluye en la topología del propio amplificador como se verá posteriormente.
- Amplificador de ganancia programable (PGA): La amplitud de la señal de entrada, aunque se encuentre acotada en unos márgenes, puede variar bastante con el paso del tiempo debido a la degradación de los electrodos, la actividad neuronal, etc. Por ello, será necesario una etapa amplificadora cuya ganancia vaya variando según la señal amplitud de la señal de entrada. De esta forma, su salida, que corresponde con la entrada del convertidor ADC, se mantendrá dentro de unos márgenes constantes impuestos por el propio convertidor.
- Convertidor Analógico-Digital (ADC): Los datos acabarán siendo procesados por algún tipo de CPU, por lo que los datos de salida de esta arquitectura deberán ser digitales. Para ello se precisa de un ADC con una resolución típica de entre 8 y 10 bit y una frecuencia de muestreo de 30kS/s.

Todo esto que se ha mencionado constituirá una arquitectura "on-chip" con dos restricciones fundamentales: el tamaño y el consumo.

1.2 Señales típicas implicadas en Neural Recording. Amplificadores de interface: características y topología

1.2.1 Señales implicadas en Neural Recording

Como se ha visto ya, el Neural Recording se encarga de captar la información de las señales del sistema nervioso, más concretamente del cerebro. Por ello, las señales implicadas en esta técnica son las propias señales cerebrales, que pueden ser divididas en dos grandes grupos: los potenciales de campo locales (LFP), que a partir de ahora también se referirá a ellos simplemente por ondas cerebrales, y los potenciales de acción (AP).

Los potenciales de campo locales son señales similares a una señal sinusoidal. Se caracterizan por su muy baja amplitud [2] que suele rondar entre 100µ-1mV y por tener una frecuencia de oscilación de entre 1-100 Hz. Éstos, a su vez, se pueden subdividir en distintos tipos de ondas según su frecuencia [5]:

- Delta (0.5-3 Hz): Representan el inconsciente y son creadas fundamentalmente cuando se está en un estado de sueño profundo. También representan la intuición y la curiosidad.
- Theta (4-7 Hz): Representan el subconsciente y son las más importantes durante la fase REM, estados de meditación y creativos.
- Alpha (8-14 Hz): Se presentan principalmente en estados de relax y de vigilia.
- Beta (15-38 Hz): Representan la consciencia despierta además de la conciencia lógica y el pensamiento analítico.
- Gamma (38-100Hz): Son las más importantes en estados de excitación y situaciones trascendentales.

Por otro lado, se encuentran los potenciales de acción (figura 1-3). Éstas son ondas con una amplitud más alta a los LFP [7], en las que pasan de un estado de reposo situado a -70mV a un estado de acción de unos 40mV. Su frecuencia también es más alta y va desde los 200 Hz a los 7kHz, por lo que presentan un ancho de banda mucho mayor al de los potenciales de campo locales.

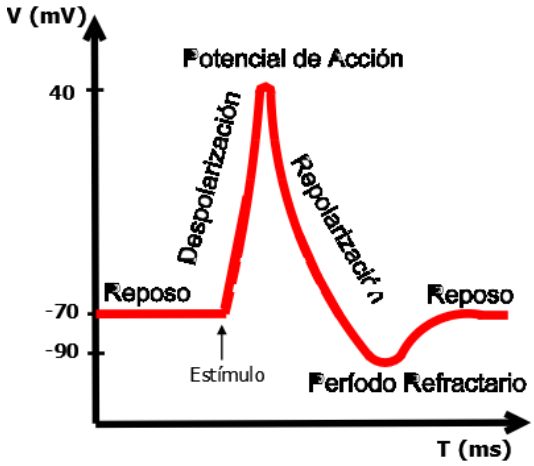


FIGURA 1-3. POTENCIAL DE ACCIÓN NEURONAL.

1.2.2 Características amplificadores de interface

Viendo las características de todas estas señales cerebrales, se pueden establecer los requisitos principales que tendrán que tener los circuitos encargados de amplificar esas señales procedentes del cerebro.

Estos circuitos se conocen por el nombre de amplificadores de interface. Los requisitos de estos amplificadores son [6], principalmente, un consumo bajo, un bajo ruido y un tamaño reducido.

El bajo consumo es una necesidad por el hecho de que estos amplificadores se van a encontrar situados en el cerebro y, como se dijo anteriormente, no se pueden permitir altas temperaturas ni grandes baterías.

El bajo ruido es debido a que se está trabajando con señales de muy baja amplitud, por lo que, si el amplificador posee un ruido equivalente a la entrada elevado, estas perturbaciones van a ejercer una influencia tal sobre la señal de salida que van a suponer la pérdida de la información sobre la señal de entrada original que se pretendía amplificar.

Por último, el tamaño que ocupan debe ser el menor posible ya que se van a encontrar en áreas del cerebro muy concretas y de alta sensibilidad, por lo que cuanto menor sea el tamaño del dispositivo, menos daños se podrían causar en el individuo.

Además de esto, por la frecuencia a la que se presentan las señales que se pretenden analizar, este

amplificador deberá tener una etapa paso banda para eliminar las frecuencias que no interesan tratar y que solo aportarán ruido al sistema.

1.2.3 Topología amplificadores de interface

Existen muchos tipos de topologías distintas para implementar amplificadores de bajo ruido en aplicaciones de Neural Recording. No obstante, en la última década principalmente, una se ha impuesto a las demás en la elección de los diseñadores [8] [9] [10] [11]. Esta topología se conoce como *capacitive feedback network* [8] (figura 1-4). Básicamente, consiste en un amplificador operacional de transconductancia (OTA) con un condensador a la entrada, C_i, realimentado a través de un condensador, C_f, y de una "resistencia", R_f, en paralelo y un condensador a la salida del circuito, C_l. A partir de esta arquitectura, nacen muchas otras con arquitecturas inspiradas en ésta [11] [12].

Se ha utilizado la palabra resistencia entre comillas debido a que realmente se trata de una pseudoresistencia formada de la colocación de varios dispositivos MOS colocados en serie y alimentados mediante una fuente de tensión variable. De esta forma, se le confiere la propiedad a esta pseudoresistencia de ser sintonizable, ya que variando la tensión de alimentación de los MOS colocados en serie, se varía su resistencia.

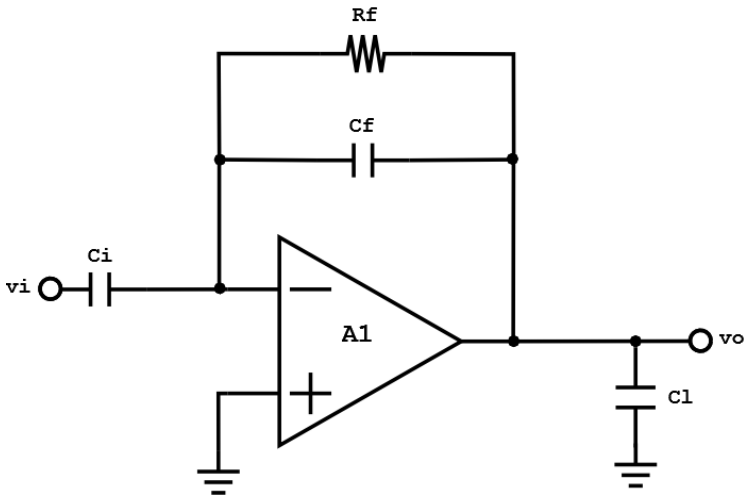


FIGURA 1-4. CAPACITIVE FEEDBACK NETWORK AFE.

Aunque esta topología no será usada durante el transcurso del proyecto, resulta interesante comprender cómo se consiguen la ganancia y el ancho de banda deseados mediante los parámetros del circuito, ya que para la topología que se tratará durante el trabajo se obtendrán estos valores de forma similar a la que se verá a continuación.

Los valores que resultarán de interés serán, por tanto, la ganancia en el ancho de banda medio (1-1) , la frecuencia del polo paso bajo (1-2), y la frecuencia del polo paso alto (1-3).

$$A_M = \frac{C_i}{C_f} \tag{1-1}$$

$$f_{lp} = \frac{gm1}{A_M \cdot C_L} \tag{1-2}$$

$$f_{hp} = \frac{1}{R_f \cdot C_f} \tag{1-3}$$

Como se muestra en (1-3), la frecuencia del polo paso alto va a ser sintonizable, ya que como se vio anteriormente, el valor de la pseudoresistencia era configurable mediante una fuente de tensión variable.

Una vez se ha visto esto, se puede hacer una pequeña introducción a la topología del amplificador que se tratará durante el trabajo. Ésta recibe el nombre de *active-feedback time constant neural amplifier* [9] (figura 1-5) y aunque se le dedicará un capítulo concreto del trabajo, se va realizar una pequeña introducción a ella.

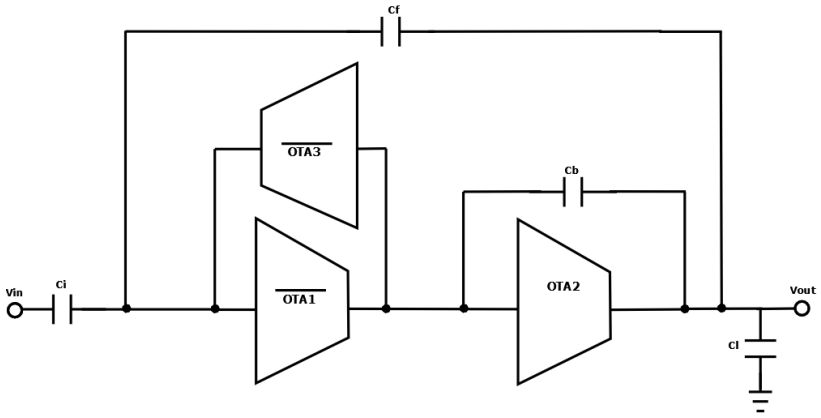


FIGURA 1-5. ESQUEMA SINGLE-ENDED ACTIVE-FEEDBACK TIME CONSTANT NEURAL ENHANCED AMPLIFIER.

La ganancia en el ancho de banda medio de este amplificador viene dado de la misma forma que para el *capacitive feedback network* (1-1). Sin embargo, son remarcables dos puntos de esta topología: el polo del filtro paso bajo se consigue mediante la utilización de otro amplificador operacional de transconductancia (OTA 3) que es configurable mediante una fuente de corriente [9]. Por otra parte, el polo de baja frecuencia se consigue principalmente por la realimentación con el condensador C_B del segundo amplificador de transconductancia (OTA 2). De esta forma, se consiguen resultados precisos y efecientes [9], aunque ya se verá todo esto con mayor profundidad durante el cuarto capítulo.

1.3 Ruido Flicker en amplificadores de interface

El ruido Flicker o ruido rosa, es uno de los ruidos de los que más se desconoce de todos los existentes. La causa de formación de este ruido no está determinada totalmente. En transistores MOS, por lo general, se asocia a los centros de recombinación [14]. Sin, embargo, algunos autores abogan por diferentes causas de formación de este ruido para transistores nMOS y para transistores pMOS [13]. La densidad espectral de potencia que presenta el ruido Flicker es inversamente proporcional a la frecuencia y es lo que caracteriza a este tipo de ruido. Por esta propiedad también se le conoce como ruido $\frac{1}{\mathrm{f}}$. Su función de densidad espectral de potencia puede ser definida (1-4) como:

$$S_{vFlicker}(f) = \frac{K}{f} \tag{1-4}$$

Donde K es una constante empírica que depende de cómo ha sido formado el ruido [14].

Para transistores MOS, la función de densidad espectral de potencia de este ruido vendrá dada por una fuente de corriente en el drenador que puede ser modelada por la siguiente ecuación (1-5):

$$I_{nFlicker}{}^2(f) = \frac{K \cdot gm^2}{W \cdot L \cdot C_{ox} \cdot f} \tag{1-5}$$

Donde gm es la transconductancia del transistor, W la anchura, L la longitud y C_{ox} la capacidad del óxido por unidad de área.

Analizando esto, es fácil deducir que este ruido ejercerá una importante influencia en circuitos que trabajen con señales de entrada de baja frecuencia, como LFP empleados en Neural Recording. En la figura 1-6 se muestra cómo afecta este ruido a una onda beta cerebral, mientras que en la figura 1-7 se muestra cómo afecta a un potencial de acción. Cómo se han generado las ondas para estas simulaciones se explicará durante el capítulo tercero.

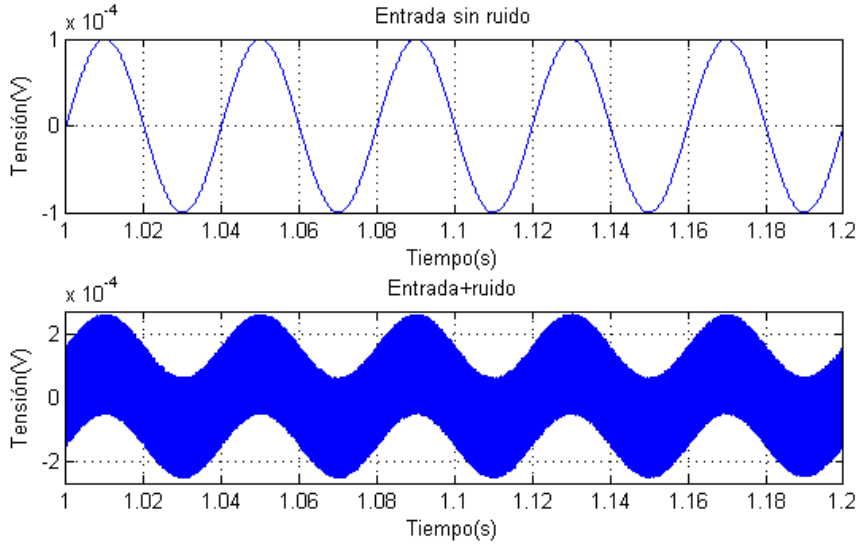


FIGURA 1-6. LFP SIN RUIDO Y LFP CON RUIDO FLICKER. LA SEÑAL SE VE PERJUDICADA DE MANERA CONSIDERABLE POR EL RUIDO.

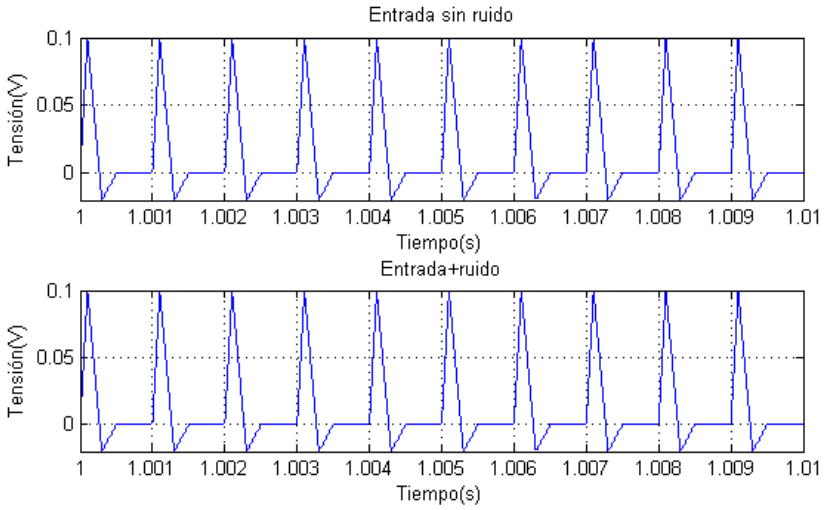


FIGURA 1-7. AP SIN RUIDO Y AP AFECTADO POR RUIDO FLICKER. NO SE ENCUENTRAN VARIACIONES EN LA SEÑAL.

Apreciando los resultados mostrados en la figura, se deduce que los potenciales de acción son señales a las que no es necesario tratar para reducir el ruido Flicker ya que éste no ejerce gran influencia

sobre ellas. Esto es debido a que los potenciales de acción tienen una frecuencia bastante más alta (hasta 7kHz) que la frecuencia a la que es dominante el ruido Flicker. Además de esto, la magnitud de este tipo de potenciales es considerable (del orden de decenas de mV) por lo que el ruido no va a ser tan apreciable como en el caso de los LFP. Por ello, la utilización de técnicas para reducir el ruido Flicker con este tipo de señales puede llegar a ser contraproducente. Todo esto se tratará en el capítulo tercero de una forma más extensa.

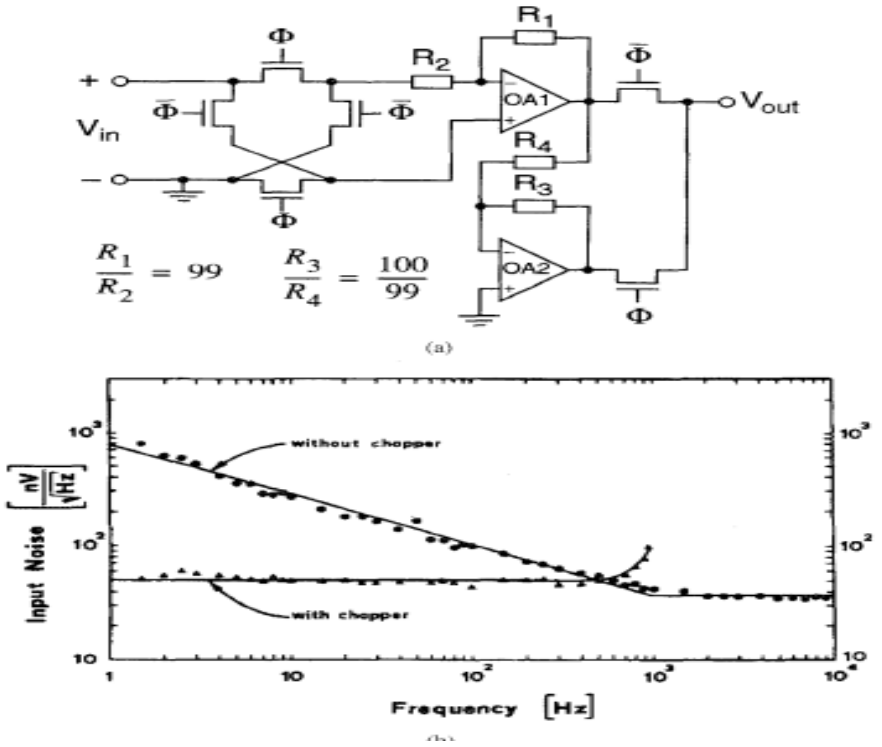


FIGURA 1-8. AMPLIFICADOR CHOPPER EXPERIMENTAL. A) ESQUEMÁTICO. B) PSD DEL RUIDO MEDIDO [15].

Para reducir/eliminar este ruido, existen diversas técnicas [15] como pueden ser el Auto-Zero o el Chopper. En la figura 1-8 se muestra la diferencia de densidad espectral de potencia para un circuito con una técnica de eliminación de ruido Flicker (Chopper) y sin ella [15]. De esta forma, se corrobora la eficacia de ciertos métodos para el fin propuesto y la necesidad de estudiar estas técnicas en profundidad para seguir desarrollando mejores estrategias para la eliminación de este tipo de perturbaciones.

1.4 Objetivos del trabajo

Analizados los problemas que se plantean debidos al ruido Flicker en amplificadores de Neural Recording, es vital encontrar técnicas que lo reduzcan. Para este trabajo en concreto se hará un estudio sobre la técnica de Chopper, y se aplicará, en primer lugar, para un ruido Flicker genérico y un amplificador ideal (ancho de banda infinito) y posteriormente se concretará para el ruido Flicker generado por el amplificador descrito en [9], introduciendo en el esquema un modelo del amplificador donde se tendrán en cuenta las no idealidades del mismo.

Por tanto, el objetivo principal del trabajo será el de analizar cómo funciona la técnica de Chopper y qué cantidad de ruido es capaz de reducir, para, finalmente, aplicarla al amplificador citado. Con esto se pretende hacer una primera aproximación de cómo afectara esta técnica al *active-feedback time*

constant neural enhanced amplifier y dependiendo de los resultados, continuar en una posible línea de investigación haciendo un diseño a más bajo nivel de las técnicas aplicadas sobre este amplificador.

Un sub-objetivo del trabajo es el de ser capaces de generar mediante diferentes técnicas un ruido determinado, de tal forma que tenga una posible implementación electrónica.

Se deduce, pues, que el trabajo tendrá tanto contenidos teóricos (descripción de los tipos de ruido, modelos de mapas discretos para generar ruidos, técnica de Chopper, no idealidades de un amplificador, tratamiento de la señal, etc.) como prácticos (la implementación de esto en varios entornos de programación/diseño) de gran relevancia en la actualidad del Neural Recording.

En cuanto a formación se refiere, este proyecto tiene varios puntos importantes a destacar:

- Búsqueda y análisis de información en artículos de divulgación científica de alta complejidad.
- Familiarización con el flujo de diseño de circuitos, partiendo de un nivel de abstracción alto.
- Implementación de diseños en el entorno de MatLab-Simulink.
- Implementación de diseños en el entorno de Cadence.
- Primeros pasos en el trabajo de circuitos de señal mixta en Cadence.
- Recopilación de conocimientos adquiridos durante el Grado.
- Capacidad de síntesis y análisis de resultados.
- Aprendizaje al trabajo conjunto con otros diseñadores.

1.4.1 Metodología

La metodología empleada en este trabajo será la del flujo típico de diseño de circuitos integrados. Más concretamente este proyecto englobará lo relacionado con un punto de abstracción alto, desde afianzar los conceptos, definir técnicas y objetivos hasta implementaciones de alto nivel sabiendo cómo deben ser los resultados obtenidos. De esta forma, si los resultados son positivos, será mucho más sencillo seguir descendiendo en la escala de abstracción para futuros trabajos/líneas de investigación. La implementación se realizará en MatLab-Simulink y en Cadence.

Los criterios para evaluar el rendimiento se centrarán en gráficas temporales y frecuenciales y cálculo del ruido integrado en banda.

2 GENERACIÓN DE RUIDO

Como se comentó en el apartado anterior, este trabajo se centrará en disminuir el ruido Flicker en circuitos de Neural Recording, los cuales sufren un gran impacto debido a este tipo de perturbación.

Para ello, es de una importancia fundamental en el desarrollo del proceso entender cuáles son las principales distribuciones espectrales de ruido que afectarán a nuestro circuito, así como algunas de las técnicas empleadas para poder generar estas distribuciones de forma controlada.

Esto último es de gran interés desde el punto de vista práctico. Conseguir técnicas que permitan la implementación electrónica de circuitos capaces de generar una distribución de ruido según las necesidades del diseñador de arquitecturas de Neural Recording (aplicable también a otros campos), supone poder añadirlas como sistemas embebidos y ver cómo se comporta el circuito físico al añadir estas perturbaciones.

En este caso concreto, ha sido proporcionado de forma directa, por parte del diseñador del circuito, la densidad espectral de ruido obtenida empíricamente del amplificador *active-feedback time constant neural enhanced amplifier*. Por lo tanto, el objetivo final de este apartado será el de conseguir generar mediante una conformación de ruidos, utilizando diversas técnicas de generación, una distribución espectral que se aproxime cabalmente a la de este amplificador. Con esta distribución generada se trabajará en apartados posteriores para ver cómo influyen diversas técnicas en su reducción.

2.1 Distribución espectral típica de la potencia de ruido MOS y ruido equivalente a la entrada

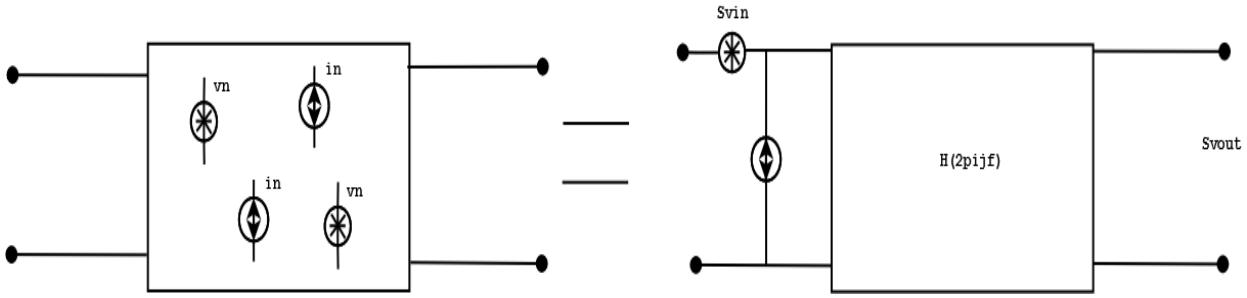
Para analizar cualquier tipo de señal, se pueden recurrir a dos tipos de análisis: temporal y frecuencial.

Durante este trabajo se van a realizar diferentes técnicas frecuenciales (modulación principalmente) para la reducción del ruido, por lo que va a ser de vital importancia conocer la distribución espectral de potencia típica de ruido en los amplificadores que se van a tratar. Esta densidad espectral de potencia (PSD, "Power Spectral Density") será dada como el valor medio de la potencia normalizada de ruido en un ancho de banda de 1 Hz y su unidad será V^2/Hz [14].

En primer lugar, se debe elegir un modelo para expresar el ruido en el circuito. En este caso, se va a utilizar un modelo de fuente de ruido equivalente a la entrada del circuito, es decir, fijando la entrada del circuito a cero, se crea un modelo de pequeña señal del mismo dónde se colocan las fuentes de ruido asociadas a los dispositivos básicos del circuito.

Posteriormente, se calcula la contribución de cada uno de ellos a la tensión de salida y por último se coloca la suma de todos ellos (en caso de que sean incorrelados entre ellos, ya que si existe correlación entre las fuentes se debe tener en cuenta también un cierto coeficiente de correlación) como una fuente de tensión y/o intensidad a la entrada del circuito.

De esta forma el circuito se puede modelar como una función de transferencia sin ruido como se puede apreciar en la figura 2-1, lo que facilita las operaciones a la hora de trabajar con los diseños. El



ruido a la salida será, por tanto:

FIGURA 2-1. CIRCUITO CON RUIDO INTEGRADO Y SU EQUIVALENTE CON RUIDO EQUIVALENTE A LA ENTRADA. EL RUIDO A LA SALIDA SERÁ EL RUIDO A LA ENTRADA MULTIPLICADO POR LA FUNCIÓN DE TRANSFERENCIA DEL SISTEMA AL CUADRADO (2-1).

$$S_{vout}(f) = S_{vin}(f) \cdot \left|H(j2\pi f)\right|^2 \tag{2-1}$$

Ahora es preciso conocer los tipos de ruido que interfieren en amplificadores basados en tecnologías MOS. Estos son principalmente: Flicker o ruido rosa, que ya se caracterizó en el primer capítulo, y ruido blanco [14].

El ruido blanco es principalmente debido al ruido térmico, el cual es provocado por el movimiento aleatorio de las cargas en un conductor consecuencia de una excitación térmica. La característica principal de este tipo de ruido es que su densidad espectral de potencia es plana, es decir, que será constante para toda frecuencia. La fuente equivalente de corriente debida a este ruido vendrá dada por [14]:

$$I^2_{nWhiteNoise}(f) = 4 \cdot k \cdot T \cdot gd0 \cdot \gamma \tag{2-2}$$

Donde k es la constante de Boltzmann ($1.38 \cdot 10^{-23}$ J/K), T es la temperatura absoluta en grados Kelvin, *gd0* la conductancia del canal cuando $V_{DS} = 0$ (que puede demostrarse que coincide con la transconductancia del MOS en saturación) y γ un parámetro que depende de la polarización y del canal del dispositivo, siendo 1 si $V_{DS} = 0$; y 2/3 cuando el dispositivo es de canal largo y se encuentra en saturación [14].

Atendiendo a las ecuaciones (1-5) y (2-2) se deduce que el ruido Flicker será predominante a frecuencias bajas, hasta una cierta frecuencia, denominada frecuencia de esquina ("corner"), *fc*, dónde el ruido blanco será mayor. Esta frecuencia de esquina es de gran importancia debido a que nos limita la franja frecuencial en la que se debe tener especial consideración con el ruido Flicker y, como se verá en apartados consecutivos, será uno de los parámetros a tener en cuenta a la hora de realizar técnicas para reducir el ruido. Esta frecuencia de esquina podría calcularse como:

$$I^2_{nFlicker}(fc) = I^2_{nWhiteNoise}(fc) \tag{2-3}$$

Para calcular la fuente de corriente equivalente a la entrada total, basta con sumar las dos fuentes equivalentes antes mencionadas:

$$I^2_{nTotal}(f) = I^2_{nFlicker}(f) + I^2_{nWhiteNoise}(f) \tag{2-4}$$

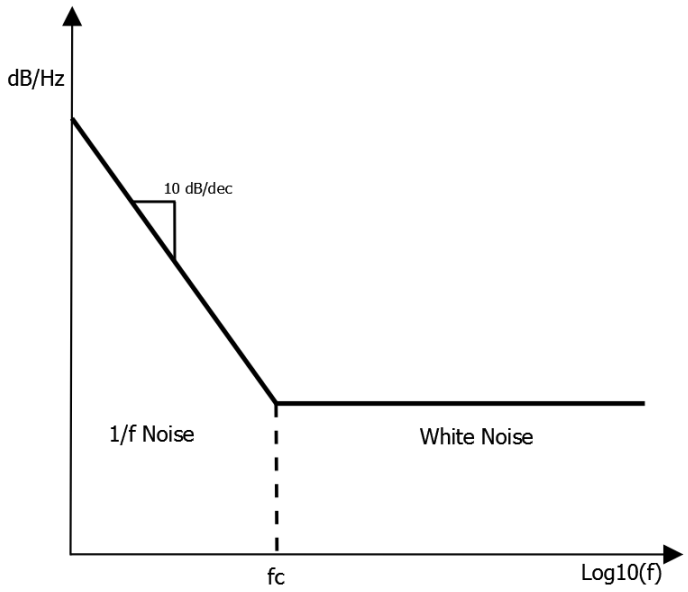


FIGURA 2-2. DENSIDAD ESPECTRAL DE POTENCIA TÍPICA MOS. A PARTIR DE LA FRECUENCIA DE ESQUINA LA DENSIDAD ESPECTRAL DE RUIDO BLANCO ES MAYOR QUE LA DEL FLICKER, TAL Y COMO SE PREDIJO ANTERIORMENTE.

La densidad espectral de potencial típica quedará, por consiguiente, como se muestra en la figura 2-2. Con ésta definida, ahora se precisa encontrar unos métodos para poder generar un ruido con esta distribución característica.

2.2 Modelos para la generación de ruido "on-chip" mediante mapas discretos

La importancia de encontrar una buena técnica para poder generar una distribución de ruido concreto, no solo reside en la necesidad de poder implementarlo "computacionalmente", sino que también existe el requisito de poder realizar un circuito específico que genere estos ruidos, como ya se dijo durante la introducción de este capítulo.

Las arquitecturas de Neural Recording se presentan, por lo general, en sistemas "on-chip", por lo que si se quisiera ver cómo se comporta una determinada arquitectura de este tipo ante una señal de ruido, lo ideal sería disponer de un sistema embebido capaz de generar una señal de ruido concreta. Por ello, es muy importante encontrar implementaciones en circuitos capaces de generar un ruido concreto. De esta forma se podría realizar un testeo real y eficaz del dispositivo.

Desde hace ya muchos años, se ha demostrado que mediante caos determinista se llega a diseñar fuentes de ruido [16] [17]. Los sistemas caóticos dinámicos tienen dos propiedades que los hacen especialmente interesantes [18]:

- Se describen por ecuaciones matemáticas, lo que permite una práctica implementación en circuitos.
- Son muy sensibles a sus condiciones iniciales, lo que electrónicamente los hacen impredecibles.

Los mapas discretos [19] constituyen el modelo matemático más sencillo para exhibir el caos y presentan la probabilidad de ser llevados al dominio electrónico mediante circuitos con capacidades conmutadas, por lo que serán estudio de este trabajo como fuentes de ruido.

Volviendo a lo que se habló en el apartado anterior, interesa generar dos ruidos: uno con una distribución Flicker y otro con una distribución de ruido blanco, por lo que se tendrá que recurrir a dos

mapas discretos diferentes.

2.2.1 Mapa discreto con mecanismo *saltador* para generación de ruido Flicker

En primer lugar, se debe generar el ruido Flicker. Éste va a ser el más importante a la hora de realizar análisis posteriores, por lo que es necesario que se genere de la forma más fiable posible.

El mapa discreto con mecanismo de transiciones *saltador* entre estados caóticos, también conocido como mapa zig-zag [20], mostrado en la figura 2-3, presenta una distribución espectral de potencia que se puede asimilar con la de una distribución $\frac{1}{f^{\alpha}}$ por lo que resulta un buen método para generar el ruido Flicker [18].

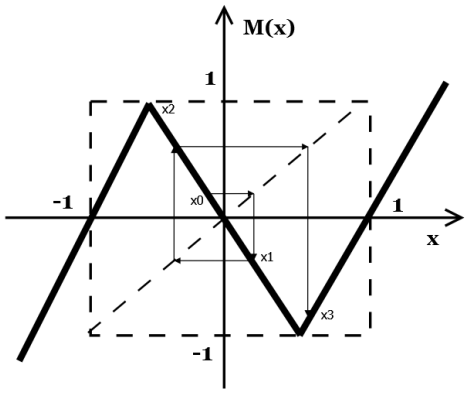


FIGURA 2-3. MAPA DISCRETO CON MECANISMO SALTADOR ENTRE ESTADOS CAÓTICOS.

Si se elige un parámetro *ml* como la pendiente positiva de la figura 2-3., *me* como la pendiente negativa y *B* como la distancia en el eje de abscisas entre el origen de coordenadas y el cambio de pendiente, definida como *1/ml*, se obtiene una definición matemática del mapa como:

$$x_{n+1} = M\left(x_n\right)$$
$$M\left(x_n\right) = \begin{cases} me \cdot x_n & -1 < x_n < -B \\ ml \cdot x_n & -B < x_n < B \\ me \cdot x_n & B < x_n < 1 \end{cases} \quad (2\text{-}5)$$

Se puede demostrar [18] que para que los puntos de $M(x_n)$ del primer cuadrante después de varias iteraciones alcanzarán el tercero y viceversa, se debe cumplir que $M(meB) < 0$.

Una vez se obtiene la función matemática para este mapa discreto, será posible su implementación electrónica y computacional como se mostrará en los siguientes apartados.

2.2.1.1 Implementación del mapa discreto con mecanismo *saltador* para generación de ruido Flicker en MatLab-Simulink.

Esta implementación computacional se realizará mediante MatLab-Simulink: Mediante Simulink se hará un modelo, Flicker.slx, que se puede ver en la figura 2-4, para generar este mapa discreto; y mediante el uso de MatLab se utilizarán una serie de "scripts" para la representación frecuencial del ruido generado.

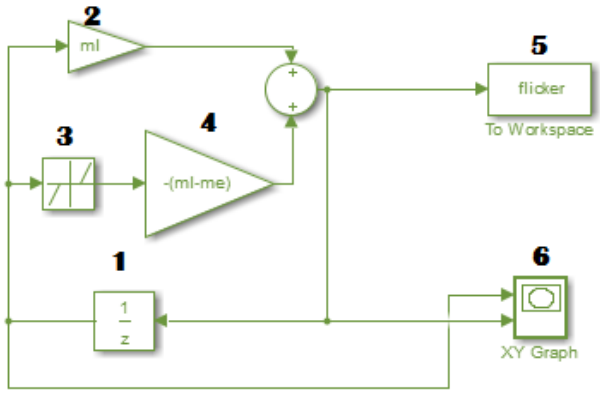


FIGURA 2-4. IMPLEMENTACIÓN DEL MAPA DISCRETO SALTADOR EN SIMULINK.

Llegados a este punto se debe hacer un inciso importante en cuanto a nomenclatura para el resto del trabajo. Para hacer referencia a parámetros que se utilizan en las ecuaciones matemáticas y/o en los "scripts", se utilizará la *cursiva*. Sin embargo, y debido a que en los "scripts", en Simulink y en Cadence no se puede utilizar la *cursiva*, en estos entornos los parámetros serán nombrados de forma normal. Con este inciso se pretende evitar las confusiones a la hora del entendimiento del conjunto del trabajo. Un ejemplo concreto sería: en (**2**) se hace referencia al parámetro *ml* que se definió anteriormente, no obstante, al no poder utilizar *cursiva*, en Simulink se le llama simplemente como ml.

En el primer punto (**1**), se introduce un Delay para poder separar los valores x_n y x_{n+1}. Posteriormente, se añade la ganancia *ml* (**2**) que será la pendiente positiva, y *–(ml-me)* (**4**) para que cuando $x_n > |B|$ la pendiente sea *me*: esto se consigue mediante una zona muerta (**3**) entre los valores *–B* y *B*. Por último, se introduce un XY Graph (**6**) para visualizar el mapa discreto generado, figura 2-5, y un ToWorkSpace (**5**) para poder usarlo con los "scripts" que se explican en el Anexo A.

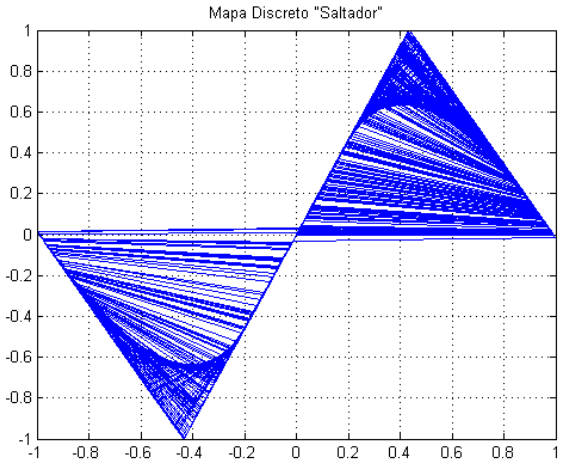


FIGURA 2-5. MAPA DISCRETO SALTADOR GENERADO EN SIMULINK PARA ML=2.3, ME=-1.8, TS= 1E-3S. RESULTA UNA BUENA APROXIMACIÓN A LA FUNCIÓN DE LA FIGURA 2-3.

Uno de los parámetros fundamentales, ya que se va a trabajar en tiempo discreto, es el tiempo de muestreo del mapa discreto (*Ts*). El empleo de un tiempo discreto u otro para este mapa va a provocar

la generación de distribuciones de ruido con diferentes características. En primer lugar, espectralmente, se puede predecir que la elección de un valor u otro para *Ts* va a suponer la presencia de "aliasing" a las frecuencias *n·1/Ts*. También se verá que influye en otras características del ruido como por ejemplo la pendiente del Flicker.

En apartados posteriores se va a realizar una semi-parametrización de todos estos parámetros para caracterizar las distribuciones de ruido. Sin embargo, es de especial interés comprobar cómo va influir en el tipo de señales que se están tratando que la frecuencia de muestreo de estos mapas (*1/Ts)* sea mayor o menor a la frecuencia máxima de estas señales (7 kHz). Para ello, se usará una señal sinusoidal de 7kHz frecuencia y 1mV de amplitud; *ml*=2.3 y *me*=-1.8; y *Ts*=2e-4s para la simulación a frecuencias menores (5kHz) y *Ts= 1e-4s* para la de frecuencias mayores (10kHz). Al ruido se le ha puesto una ganancia a la salida de 1/100000 para que no sea mayor a la señal de entrada. Los resultados pueden verse en las figuras 2-6 y 2-7. Para realizar esta comparación, se ha utilizado el "script" Flicker_neuralm.m y el archivo de Simulink Flicker_neural.slx.

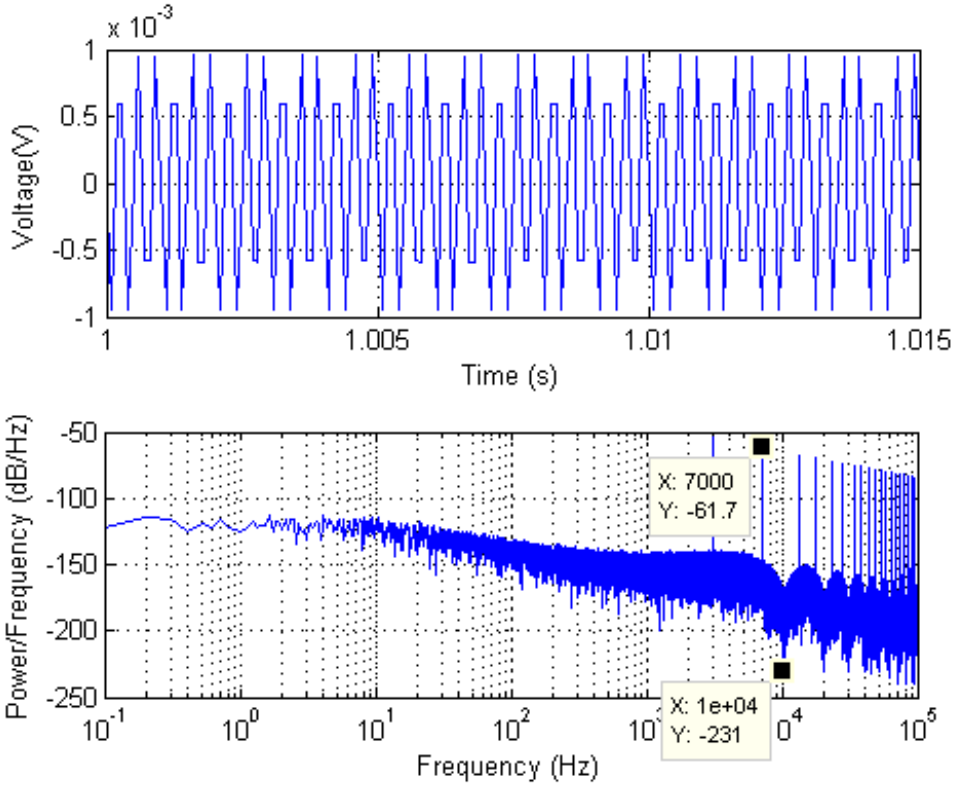


FIGURA 2-6. SEÑAL DE ENTRADA DE 7 KHZ Y 1MV A LA QUE SE LE AÑADE RUIDO POR MAPAS DISCRETOS CON TS=1E-4S. SE PRODUCE "ALISASING" A LOS 10KHZ Y SUS SIGUIENTES MÚLTIPLOS DEBIDO AL VALOR DE TS. LA SEÑAL DE ENTRADA SE VE BASTANTE AFECTADA POR EL RUIDO.

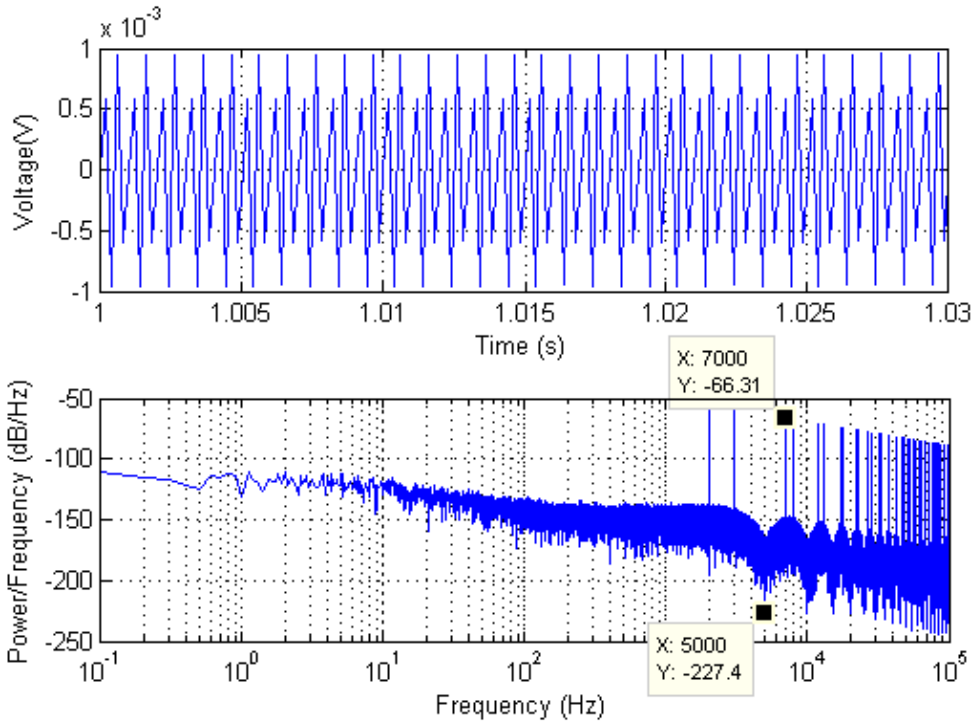


FIGURA 2-7. SEÑAL DE ENTRADA DE 7 KHZ Y 1MV A LA QUE SE LE AÑADE RUIDO POR MAPAS DISCRETOS CON TS=2E-4S. SE PRODUCE "ALISASING" A LOS 5KHZ Y SUS SIGUIENTES MÚLTIPLOS DEBIDO AL VALOR DE TS. LA SEÑAL DE ENTRADA SE VE MÁS AFECTADA POR EL RUIDO QUE PARA TS=1E-4.

Se puede concluir observando ambas figuras (2-6 y 2-7), que para frecuencias de muestreo del mapa menores que la frecuencia fundamental de la señal de entrada, el ruido generado afecta más a la señal de entrada que para frecuencias de muestreo mayores. Esto es debido, principalmente, al encontrarse la frecuencia fundamental de la señal de entrada después de haberse producido el "aliasing", que provoca una atenuación en la señal de entrada (en las figuras se aprecia como pasa en 7kHz de -61.7db/Hz para 10kHz a -66.31dB/Hz para 5kHz de frecuencia de muestreo) haciendo que ésta sea más perturbada para un ruido similar.

Para representar los datos en el dominio de la frecuencia correspondientes a estos mapas discretos, se usa Flicker_Code.m cuyo código se muestra a continuación. Hay que decir, que durante el trabajo se recogerán tan solo aquellos códigos empleados más importantes, ya que el resto serán derivados de éstos.

```
clear all;
clear all;
close all;
clc;

Tm=1e-6; %Tiempo de muestreo
Fm=1/Tm; %Frecuencia de muestreo

%Parámetros del mapa discreto
ml=2.3;
me=-1.78;
B=1/ml;
Ts=1e-3;
```

```
%Abrimos simulink
find_system('Name','Flicker');
open_system('Flicker');



str1 = sprintf('Ts2=%i ml2= %f me2= %f',Ts2,ml2,me2);
%Esperamos que termine de ejecutarse
[t,y]=sim('Flicker');
%Usamos las funciones spectrum y pintapsd y representamos dos gráficas:
%Normal y una ampliada

[f X]=spectrum(flicker,Fm);
PSDPink = 1./f(2:end);
pintapsd(f,X);
title(str1);
hold on;
semilogx((f(2:end)),10*log10(PSDPink/100),'r','linewidth',2)
xlim([0 2/Ts]);
legend('Flicker Generado','Flicker Teórico 1/f');




pintapsd(f,X);
title(str1);
hold on;
semilogx((f(2:end)),10*log10(PSDPink/100),'r','linewidth',2)
xlim([0 100]);
legend('Flicker Generado','Flicker Teórico 1/f');
```

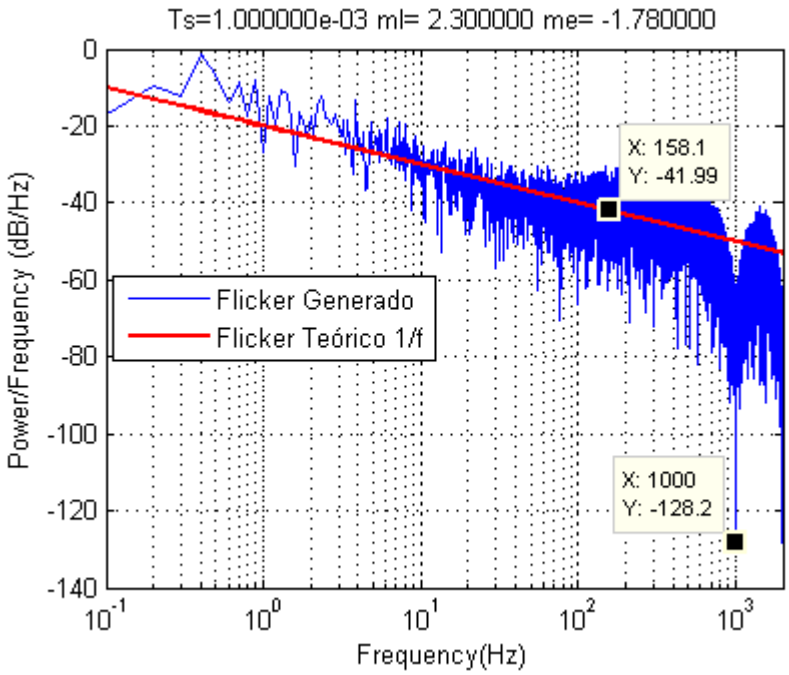


FIGURA 2-8. FLICKER SIMULADO MEDIANTE MAPAS DISCRETOS. SE APRECIA UNA BUENA APROXIMACIÓN A 1/F. LA FRECUENCIA DE ESQUINA ESTÁ EN TORNO A LOS 160HZ. A PARTIR DEL EFECTO DE "ALIASING" PRESENTA UN COMPORTAMIENTO ANÓMALO QUE NO REPRESENTA A NINGUNA DISTRIBUCIÓN DE RUIDO CONCRETA.

La observación de la figura 2-8 conlleva a deducir que este método resulta una buena aproximación para la generación de ruido Flicker ideal hasta frecuencias cercanas a la frecuencia de muestreo del mapa discreto, debido a que primero se ve transformado en un ruido blanco (a partir de la frecuencia de esquina) y posteriormente presenta un comportamiento anómalo debido al "aliasing".

Como conclusión, se ha podido comprobar cómo el mapa discreto *saltador* resulta un buen método para generar un ruido muy aproximado al ruido Flicker, cuyas características van a depender de los parámetros que se empleen en él. Además, como se verá en capítulos posteriores, se conocen métodos para su implementación en circuitos.

2.2.1.2 Parametrización empírica mapa discreto saltador

Debido a que se tratan de mapas discretos basados en funciones no lineales, a priori no se puede realizar ningún tipo de procedimiento matemático para predecir el comportamiento que va a presentar la densidad espectral de potencia de la señal generada. No obstante, lo que sí se puede hacer es realizar una serie de iteraciones para distintos valores de *Ts*, *ml* y *me* para ver las características del Flicker para cada uno de esos valores y establecer unos criterios básicos de comportamiento.

Los resultados obtenidos serán recogidos en la tabla 2-1, intentando que estos sean lo más concreto posibles y que den al lector una información lo suficientemente sólida para que pueda generar sus propias distribuciones de ruido.

El código empleado queda recogido en el "script" Flicker_Code_Par.m y es el que se muestra a continuación:

```
Tm=1e-6; %Tiempo de muestreo
Fm=1/Tm; %Frecuencia de muestreo



%Parámetros del mapa discreto
ml=2.3;
me=-1.8;
Ts=1e-3;

%Abrimos simulink
find_system('Name','Flicker');


open_system('Flicker');


for ml=1.5:0.5:3
    for me=-3:0.5:-1.5
        B2=1/ml;


str1 = sprintf('Ts=%i ml= %f me= %f',Ts,ml,me);
```

```
[t,y]=sim('Flicker');


%EN CADA BUCLE SE EJECUTA Y NOS MUESTRA UNA GRÁFICA DE COMPORTAMIENTO
[f X]=spectrum(flicker,Fm);

PSDPink = 1./f(2:end);

pintapsd(f,X);
title(str1);
hold on;
semilogx((f(2:end)),10*log10(PSDPink/100),'r','linewidth',2)
xlim([0.1 2/Ts]);


end


end
```

TABLA 2-1. RESULTADOS EXPERIMENTALES MAPA DISCRETO SALTADOR PARA DISTINTAS ITERACIONES. FC REPRESENTA LA FRECUENCIA DE ESQUINA DEL RUIDO GENERADO, MDB1 REPRESENTA LA MAGNITUD DE LA SEÑAL EN 0.1HZ Y MDBC LA MAGNITUD EN LA FRECUENCIA DE ESQUINA. EL OBJETIVO DE ESTOS PARÁMETROS ES QUE SE PUEDA DEDUCIR CÓMO ES LA PENDIENTE DEL FLICKER GENERADO. EXISTEN RESULTADOS RELEVANTES PARA PARÁMETROS INTERMEDIOS ENTRE CON LOS QUE SE OBTIENEN RESULTADOS RELEVANTES Y NO RELEVANTES.

	Ts=1e-3s	**Ts=1e-3s**	**Ts=1e-3s**	**Ts=1e-3s**	**Ts=1e-4s**	**Ts=1e-4s**	**Ts=1e-4s**	**Ts=1e-4s**
	me=-3	**me=-2.5**	**me=-2**	**me=-1.5**	**me=-3**	**me=-2.5**	**me=-2**	**me=-1.5**
ml=3	No resultados relevantes.	***fc*=1Khz *mdb0*= -30dB/Hz *mdbc*= - 45dB/Hz**	No resultados relevantes.	No resultados relevantes.	No resultados relevantes.	***fc*=10Khz *mdb0*= -40dB/Hz *mdbc*= -60dB/Hz**	***fc*=10Khz mdb0= -40dB/Hz *mdbc*= -60dB/Hz**	No resultados relevantes.
ml=2.5	***fc*=200Hz *mdb0*=- -30 dB/Hz *mdbc*= -40dB/Hz**	***fc*=800Hz *mdb0*= -30dB/Hz *mdbc*= -45dB/Hz**	***fc*=300Hz *mdb0*= -24 dB/Hz *mdbc*= -45dB/Hz**	No resultados relevantes.	No resultados relevantes.	No resultados relevantes.	***fc*=8Khz *mdb0*= -34dB/Hz *mdbc*= -60dB/Hz**	No resultados relevantes.
ml=2	No resultados relevantes.	No resultados relevantes.	No resultados relevantes.	No resultados relevantes.	No resultados relevantes.	No resultados relevantes.	No resultados relevantes.	No resultados relevantes.
ml=1.5	No resultados relevantes.	No resultados relevantes.	No resultados relevantes.	No resultados relevantes.	***fc*=6KHz *mdb0*= -42dB/Hz *mdbc*= -60dB/Hz**	No resultados relevantes.	No resultados relevantes.	No resultados relevantes.

Con los resultados de la tabla se puede llegar a una serie de conclusiones en cuanto a los parámetros del mapa discreto para la generación del Flicker:

- *Ts*: Si *me* está entre -1.7 y -2.5 aproximadamente: *Ts* es el principal factor que define la frecuencia de esquina del ruido Flicker. Cuanto menor sea este parámetro, mayor será la frecuencia de esquina, ya que depende del "aliasing". Para *Ts*=1e-3s en combinación con el

resto de parámetros se obtienen los resultados más interesantes desde el punto de vista práctico, ya que se obtienen pendientes más similares a 1/f para frecuencias bajas. Por lo general, para *Ts* ≤1e-5s los resultados se aproximan más a un ruido blanco que a un ruido Flicker.

- *ml*: Es el principal causante de que un resultado sea o no relevante: si se encuentra entre 3 y 2.3 el resultado lo será. Por lo general, para un mismo valor de *me*, cuanto más cercano esté *ml* a 2.3, más suave será la pendiente del Flicker y menor será su frecuencia de esquina.
- *me:* A priori es el parámetro más aleatorio, pero cabe destacar dos aspectos: sus valores óptimos se encuentran cercanos al -2, reduciendo la frecuencia de esquina si nos acercamos al -3; y son responsables de la pendiente a muy baja frecuencia. Las distribuciones de Flicker más interesantes (figura 2-8, figura 2-9) se encuentran en valores cercanos a -1.78 para *ml*=2.3 y *Ts*=1e-3; y -1.7 para *ml*=2.5 y *Ts*=1-4. Estos son los parámetros que se han considerado óptimos para los distintos valores de Ts.

Ya se tiene una semi-parametrización más o menos exacta de los parámetros, por lo que se puede implementar en el dominio electrónico.

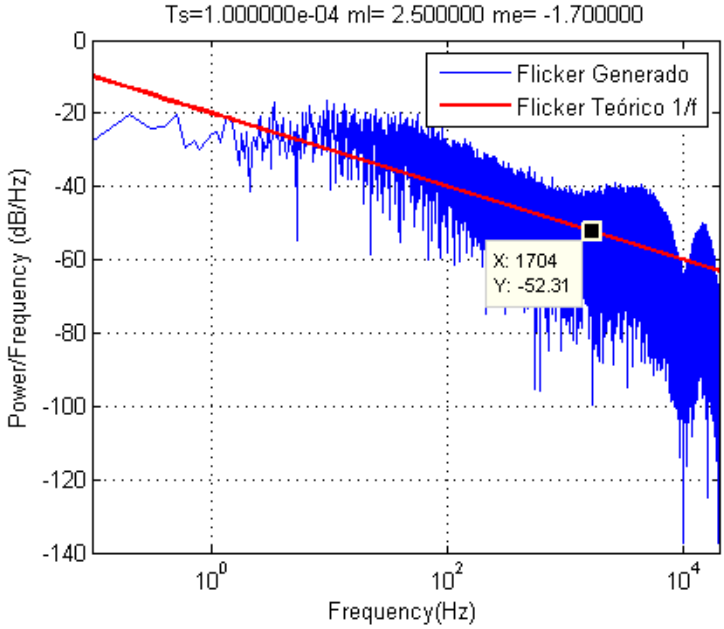


FIGURA 2-9. RUIDO FLICKER MEDIANTE MAPAS DISCRETOS PARA TS=1E-4S. LA FRECUENCIA DE ESQUINA SE SITÚA EN TORNO A LOS 2KHZ.

2.2.1.3 Implementación circuital del mapa discreto con mecanismo *saltador* para generación de ruido Flicker

Se ha mostrado [18] que los mapas discretos pueden ser llevados al dominio electrónico mediante el uso de circuitos basados en capacidades conmutadas. El circuito mostrado a continuación (figura 2-10) es una implementación en capacidades conmutadas de (2-5)

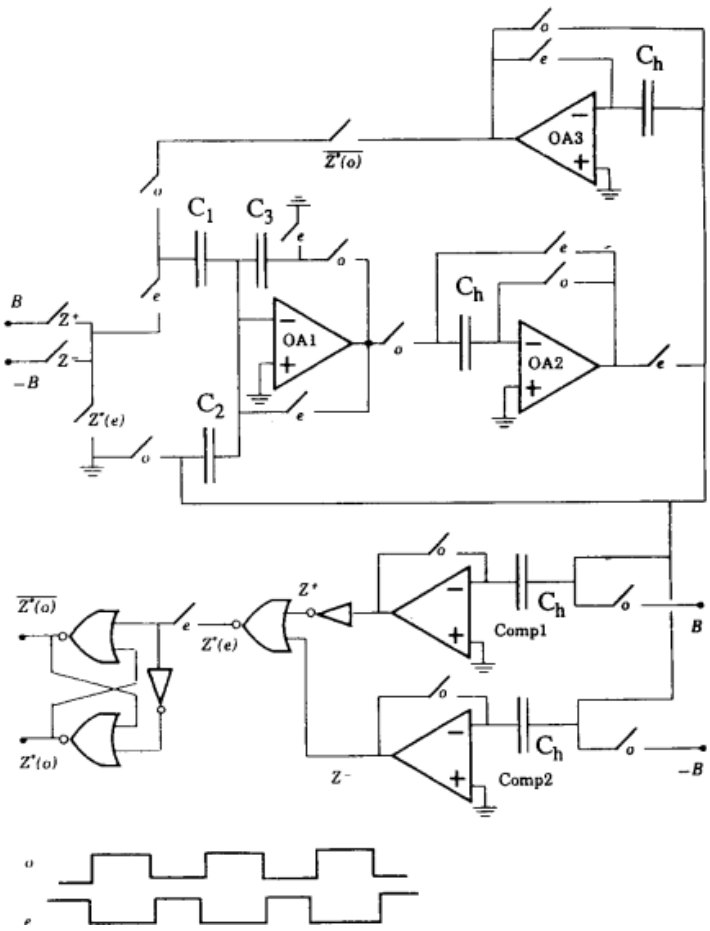


FIGURA 2-10. ESQUEMÁTICO CIRCUITO CAPACIDADES CONMUTADAS PARA EL MAPA SALTADOR [18].

El Comp1 y Comp2 codifican las partes del mapa no lineal de la figura 2-3 de la siguiente forma:

- Para la zona central del mapa → $Z^+ = Z^- = 0,\ Z^* = 1$
- Para la zona de la izquierda del mapa → $Z^+ = Z^* = 0,\ Z^- = 1$
- Para la zona de la derecha del mapa → $Z^- = Z^* = 0,\ Z^+ = 1$

Consiguiendo el control de Z mediante conmutadores, se logra alcanzar la no linealidad ya descrita. Con el Op-amp OA1 se realiza una operación de suma ponderada, mientras que con OA2 y OA3 se proporciona al sistema el retraso requerido para el bucle de iteración. Para modelar los valores de *ml* y se *me* usan las capacidades C_1, C_2 y C_3:

$$me = \frac{C_2 - C_1}{C_3}; ml = \frac{C_2}{C_3} \tag{2-6}$$

Los resultados experimentales [18] del mismo para diversas configuraciones paramétricas son los que pueden observarse en la figura 2-9.

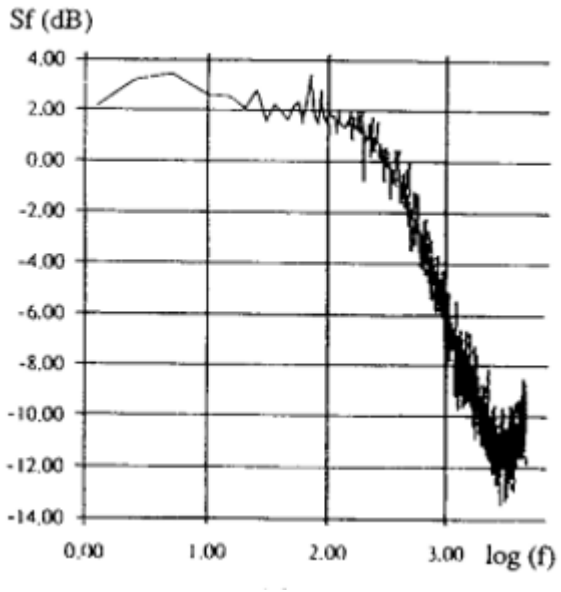


FIGURA 2-11. RESULTADOS EXPERIMENTALES MAPA SALTADOR PARA ML=2.7, ME=-1.8 [18]. LA PENDIENTE OBTENIDA RESULTA MUY SIMILAR A LA DE UN RUIDO FLICKER REAL.

Se puede apreciar que los resultados obtenidos son una muy buena forma de generar el ruido Flicker en circuitos, sin tener que recurrir a técnicas cómo las basadas en filtros para ruido blanco, lo que será muy útil en la práctica para los sistemas embebidos ya mencionados.

2.2.2 Mapa discreto Bernoulli para generación de ruido blanco

Por otra parte, se debe conseguir un ruido blanco que se superponga al ruido Flicker en frecuencias superiores a la frecuencia de esquina.

Para poder generar este ruido blanco se va usar un mapa discreto de Bernoulli (figura 2-12), que como ya se ha demostrado antes en numerosas ocasiones [21] [22], es un tipo de sistema dinámico no lineal capaz de producir una serie de números aleatorios que espectralmente presentan una distribución de potencia característica similar a la del ruido blanco. Este tipo de mapas fue de los primeros en emplearse para la generación de ruido blanco en circuitos.

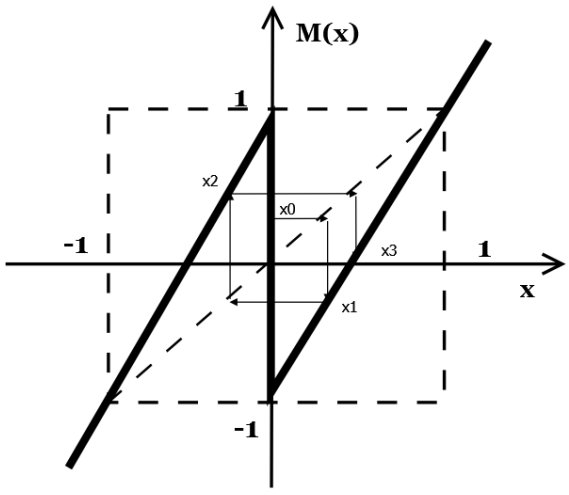


FIGURA 2-12. IMPLEMENTACIÓN PRÁCTICA DEL MAPA DE BERNOULLI.

El primer paso a tomar para su realización, es hacer un modelo matemático que cumpla lo establecido en la figura 2-12. Para ello, solo se dispondrá de un parámetro variable, que será el estado inicial, que como ya se vio en el apartado segundo de este capítulo cuando se introdujeron los mapas discretos "on chip", los números aleatorios generados dependerán mucho del parámetro que se tome inicial. La función matemática para la forma en que se implementará este mapa es la siguiente:

$$x_{n+1} = M\left(x_n\right)$$
$$M\left(x_n\right) = \begin{cases} mb \cdot x_n + 1, & -1 < x < 0 \\ mb \cdot x_n - 1, & 0 \le x < 1 \end{cases} \tag{2-7}$$

Se debe aclarar que si se pretende que el mapa de Bernoulli se encuentre en el cuadrado (-1,-1; -1,1; 1,-1; 1.1), *mb* debe ser menor o igual a 2. Sin embargo, no se tomará el 2 como valor para *mb* ya que volvería la función par y se obtendrían la mitad de valores de puntos aleatorios que si se tomara otro valor cercano a 2 pero distinto de éste. Debido a que los resultados obtenidos son muy similares para todos esos valores, se impondrá *mb*=1.83. En este capítulo no hará falta, por tanto, parametrización.

2.2.2.1 Implementación computacional del mapa de Bernoulli para generación de ruido blanco

El modelo en Simulink recibe el nombre de WNG.slx (figura 2-13) mientras que el archivo de MatLab empleado es WN_Code.m.

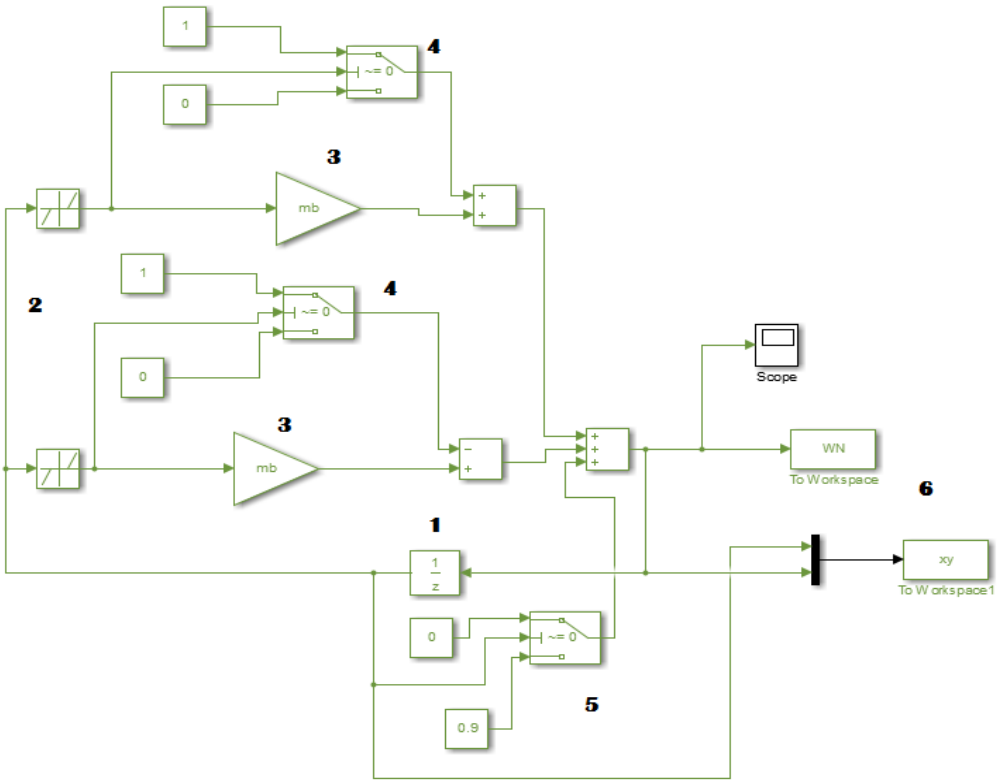


FIGURA 2-13. IMPLEMENTACIÓN SIMULINK MAPA BERNOULLI.

Para este modelo, se continua empleando un Delay (**1**) para separar el estado x_n del x_{n+1}. Posteriormente se colocan dos Dead Zone (**2**) para diferenciar los intervalos de trabajo de la función,

es decir, de -1 a 0 y de 1 a 0. Una vez la señal toma un camino u otro se multiplica por la pendiente *mb* (**3**) y se le suma o resta 1 (**4**) según la zona en la que esté. El Switch en (**4**) se coloca para el momento en que se encuentre en la zona muerta, no se sume o reste el 1 a la salida del sistema. Es importante añadir un Switch (**5**) por si en algún momento la señal se hace 0, ésta no genere conflictos y se cambie por otro parámetro. Por último, se utilizan To Workspace (**6**) para representar los resultados.

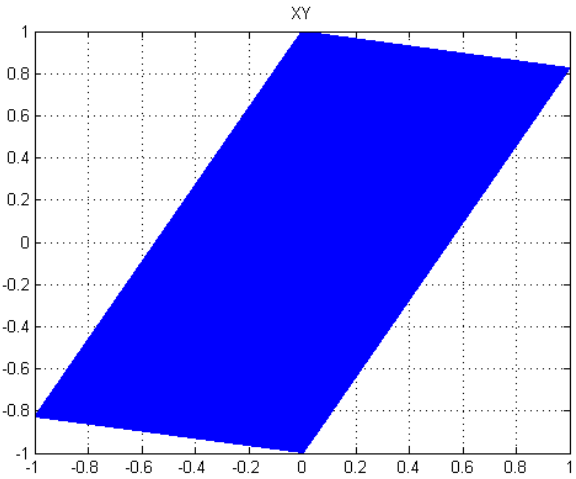


FIGURA 2-14. REPRESENTACIÓN EN TRAZO CONTINUO DEL MAPA BERNOULLI GENERADO EN SIMULINK. SE CUMPLE LO IMPUESTO EN LA ECUACIÓN (2-7)

El "script" utilizado para este apartado es el siguiente:

```
Tm=1e-6; %Tiempo de muestreo
Fm=1/Tm; %Frecuencia de muestreo

%Parámetros del mapa discreto
mb=1.83;

%Abrimos simulink
find_system('Name','WNG');
open_system('WNG');

%Esperamos que termine de ejecutarse
[t,y]=sim('WNG');
%Usamos las funciones spectrum y pintapsd y representamos dos gráficas:
%Normal y una ampliada

[f X]=spectrum(WN,Fm);
PSDWhite = 1./f(2:end).^0;
pintapsd(f,X);
title('PSD Ruido Blanco');
```

```
hold on;
plot(log10(f(2:end)),10*log10(PSDWhite/3000000),'r','linewidth',2)
xlim([0 5]);
legend('Ruido Blanco','Ruido blanco Teórico');
figure(); %Representamos XY
plot(xy(:,1),xy(:,2));
grid;
title('XY');
clc;
```

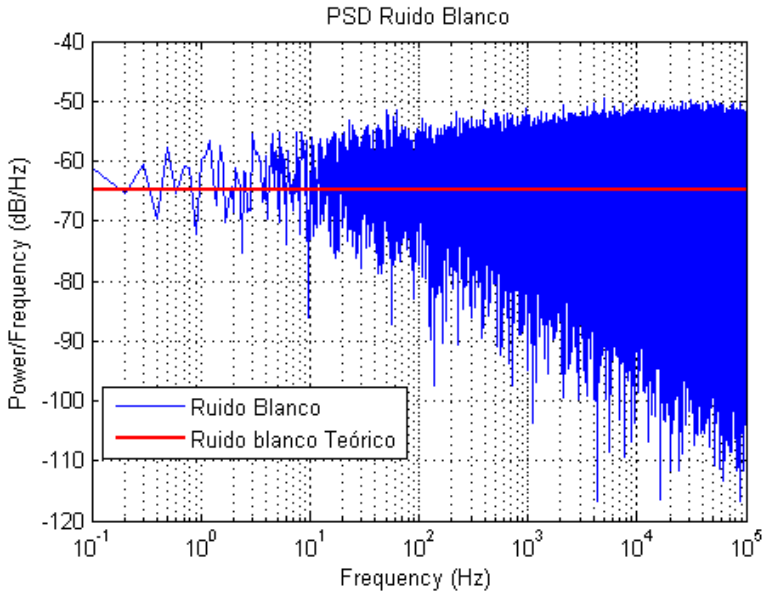


FIGURA 2-15. RUIDO BLANCO GENERADO MEDIANTE LA IMPLEMENTACIÓN DEL MAPA DE BERNOULLI PARA MB=1.83. ES UNA BUENA APROXIMACIÓN PARA EL RUIDO BLANCO, AUNQUE AL AUMENTAR LA FRECUENCIA, AUMENTA EL ANCHO DE LA DENSIDAD ESPECTRAL DE POTENCIA, POR LO QUE EN APARTADOS POSTERIORES HABRÁ QUE APLICAR CIERTO FACTOR DE ESCALA PARA CORREGIR ESTO.

2.2.2.2 Implementación circuital del mapa de Bernoulli para generación de ruido blanco

Ya se explicó durante el capítulo de la implementación circuital del Flicker que los circuitos integrados para la realización de los mapas discretos con el fin de generar ruidos se realizaban usando capacidades conmutadas. En este caso la función a implementar va a ser la siguiente (2-8)[22], que es equivalente a la (2-7) en los intervalos definidos anteriormente y para el valor de B (no se debe confundir con la B utilizada para el mapa discreto *saltador*) que se calculará a continuación:

$$x_{n+1} = B \cdot x_n - mb \cdot \mathrm{sgn}(x_n); \tag{2-8}$$

El circuito empleado para implementar (2-8)[22] es el utilizado en la figura 2-16. En este caso el mapa de Bernoulli aparte del parámetro *mb* que se mostró anteriormente, también tiene un parámetro B que define junto a *mb* los límites del cuadrado en los que se desarrolla del mapa estudiado. Para el mapa de Bernoulli que se ha empleado, el cuadrado estaba comprendido entre (-1,-1), (-1,1), (1,-1) y (1,1). Por lo tanto, y teniendo en cuenta que *mb*=1.83 [22]:

$$-1=\frac{mb}{1-B};1=\frac{mb}{B-1}\rightarrow B=2.83 \tag{2-9}$$

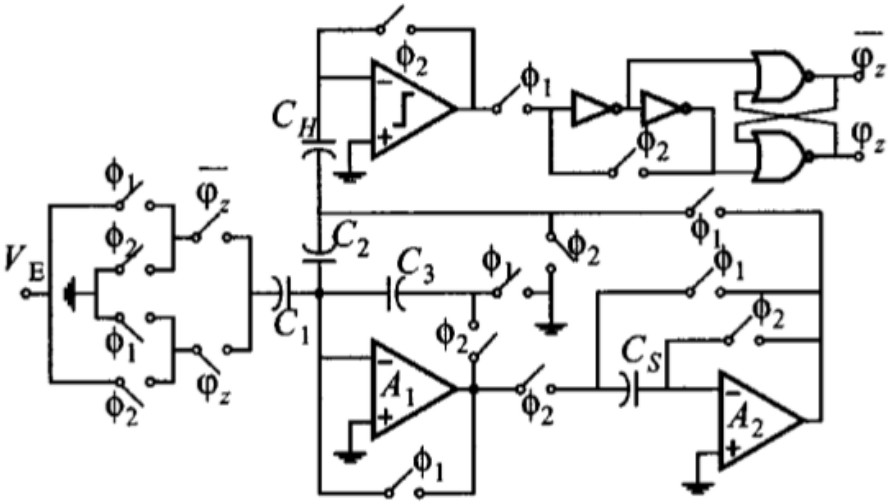


FIGURA 2-16. ESQUEMÁTICO CAPACIDADES CONMUTADAS PARA MAPA DE BERNOULLI [22].

El Op-amp "A1" y las capacidades adyacentes se encargan de realizar la suma ponderada y de introducir medio ciclo de retraso mientras que el Op-amp "A2" se encarga el otro medio ciclo de retraso restante. *B* y *mb* vienen dadas por [22]:

$$B=\frac{C2}{C3};\quad mb=\frac{C1}{C3}\cdot V_E \tag{2-10}$$

Dependiendo del valor de φ_z [22], que es controlada por el comparador dinámico, el valor V_E es sumado o restado a la entrada el Op-amp "A1". Esto permite la realización de la no linealidad.

El comparador citado anteriormente consiste en amplificador con cancelación de offset, seguido por un amplificador con sentido regenerativo y un "latch" basado en una puerta NOR [21]. Los resultados medidos experimentalmente para *B*=1.906 [22] son los mostrados en la siguiente figura (figura 2-17):

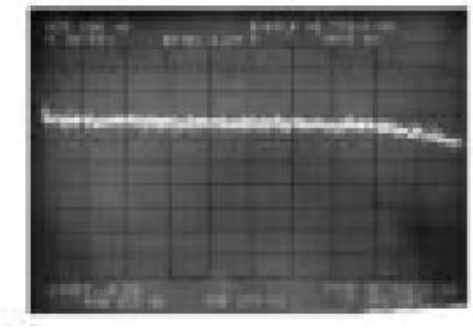


FIGURA 2-17. PSD EXPERIMENTAL RUIDO BLANCO MEDIANTE MAPA BERNOULLI [22]. LOS RESULTADOS EXPERIMENTALES SATISFACEN LAS CARACTERÍSTICAS PRINCIPALES DEL RUIDO BLANCO, POR LO QUE EL MÉTODO RESULTA UNA BUENA APROXIMACIÓN.

2.3 Generación de ruido mediante funciones

En este apartado se va a realizar la generación de ruido mediante funciones que incorporan tanto MatLab-Simulink como Cadence. El objetivo va a estar en la generación de ruido Flicker ya que la generación del ruido blanco es arbitraria en ambos entornos debido a que ambos disponen de varias funciones [23] [24] para la generación de un ruido blanco.

2.3.1 Generación de ruido mediante MatLab-Simulink

Una forma que tiene MatLab sencilla y más o menos precisa para generar un ruido coloreado es dsp.ColoredNoise(*x*,*y*) [23]. Esta función selecciona el ruido a generar mediante $\frac{1}{|f^x|}$, pudiendo generar hasta cinco tipos de ruidos para los diferentes valores de *x*: Para *x*=-2 el ruido será violeta; para *x*=-1 azul (estos dos primeros se descartan en el análisis ya que son irrelevantes para el objetivo del trabajo); para x=0 el ruido es blanco (figura 2-18); para *x*=1 el ruido es rosa, o Flicker, (figura 2-19) y finalmente para *x*=2 el tipo de ruido será marrón (figura 2-20). Por otra parte, con el parámetro *y* se indica el número de puntos que va tener la señal generada. En este caso se ha usado *y*=1/1e-7, ya que ése es el periodo de muestreo usado en Simulink y hará falta posteriormente implementar la función en Simulink.

Para que éste comando produzca el conjunto de números, se debe usar step(dsp.ColoredNoise(*x*,*y*)).

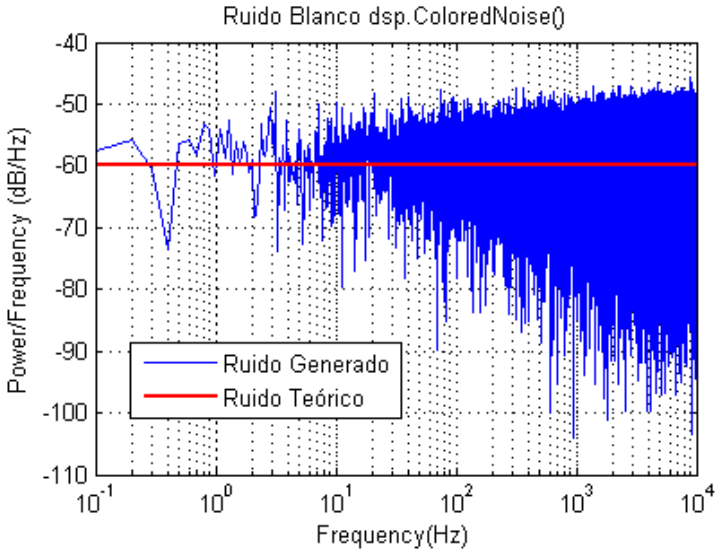


FIGURA 2-18. RUIDO BLANCO GENERADO MEDIANTE DSP.COLOREDNOISE(). PARA APARTADOS POSTERIORES, TAMBIÉN HABRÁ QUE AÑADIR UN FACTOR DE ESCALA PARA ESTE TIPO DE RUIDO GENERADO DE ESTA FORMA.

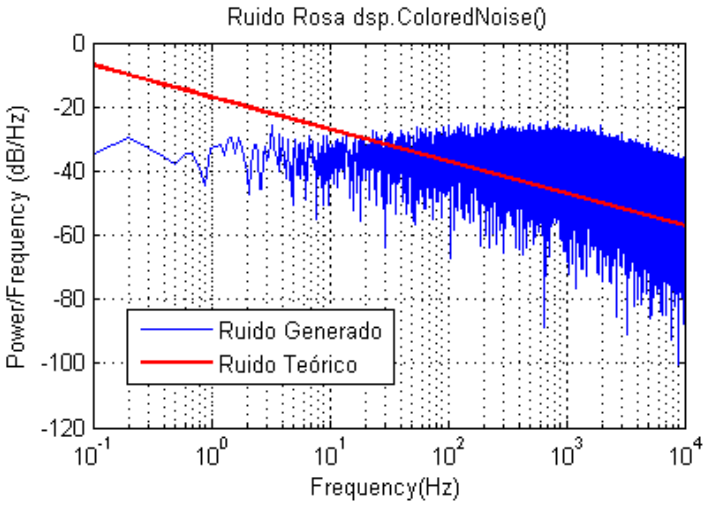


FIGURA 2-19. RUIDO ROSA GENERADO MEDIANTE DSP.COLOREDNOISE().

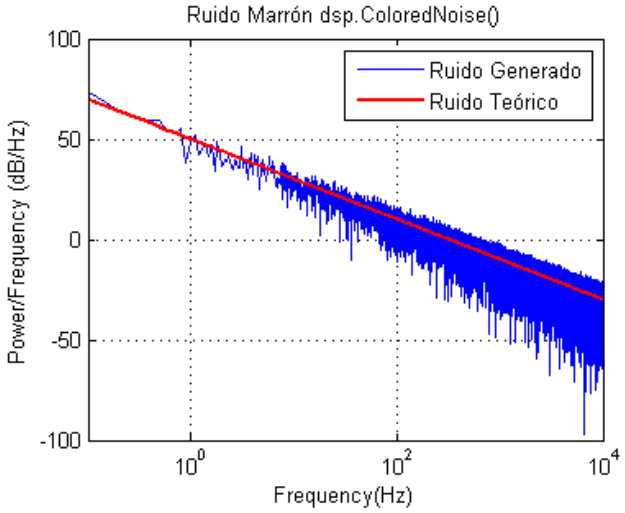


FIGURA 2-20. RUIDO MARRÓN GENERADO MEDIANTE DSP.COLOREDNOISE().

Para poder implementar este comando en Simulink, lo primero a realizar es crear un vector que representaría el tiempo de la simulación, es decir, si por ejemplo la simulación tuviera un periodo de muestreo de 1e-6s y durara un 1s, este vector se crearía de la siguiente forma:

t = [0:1e-6:1]'

El siguiente paso a seguir es formar una matriz con dos columnas cuya primera columna sea el vector que acabamos de crear, y su segunda columna sea el vector que contiene los puntos generados por el comando dsp.ColoredNoise. Una vez se tenga esta matriz, la implementación en Simulink es tan sencilla como introducirla en un bloque From Workspace, de manera que Simulink irá leyendo un valor de ruido cada tiempo de muestreo.

A la vista de los resultados, se puede llegar a la conclusión que este comando de MatLab permite generar un ruido determinado de manera sencilla, ideal y sin un coste computacional tan elevado como el de los mapas discretos. Para conformar el ruido deseado, bastaría con sumar varios tipos de ruido multiplicándolos por una ganancia concreta y así establecer la frecuencia de esquina del sistema.

No obstante, esta función tiene dos grandes desventajas:

- Esta función genera una secuencia de números aleatorios distintos cada vez que se ejecuta (mediante mapas discretos siempre generábamos la misma secuencia). Esto se va a traducir que las PSD para cada ejecución van a ser parecidas, pero no iguales, lo que va a influir de manera importante cuando se desee conformar una distribución concreta y precisa de ruido. En el caso práctico de los mapas discretos ocurriría lo mismo, sin embargo, en los simulados siempre se tiene el mismo valor inicial y por consiguiente la misma distribución de ruido.
- A priori sería de gran dificultad su implementación electrónica, ya que su algoritmo de generación de ruido es complejo y se trata de un modelo autorregresivo [23].

2.3.2 Generación de ruido mediante Cadence

Aunque se lleve a cabo la gran parte del proyecto mediante MatLab-Simulink, el diseño de circuitos a bajo nivel se realiza mediante software específico para esto. En concreto, se va a utilizar Cadence y aunque no se llegue durante este trabajo a los niveles de abstracción más bajos, es importante desde el punto de vista de futuras investigaciones y una posible continuación del trabajo, la implementación de los "scripts" usados en MatLab-Simulink en este entorno.

La forma más sencilla de introducir un ruido determinado en un circuito ideal en Cadence [24] es mediante un Port (el cual se encuentra en la librería analogLib). Este componente consiste en una fuente de tensión que lleva una resistencia en serie y permite al usuario seleccionar el tipo de señal que se quiere generar (dc, sine, pulse…). Además, y este es el punto interesante, permite añadir ruido a la misma de dos formas diferentes: introduciendo un archivo de ruido donde se indican las frecuencias y el valor de los puntos de ruido, como se verá en el apartado cuatro de este capítulo; o introduciendo un conjunto de pares de puntos potencia-frecuencia de ruido (figura 2-21).

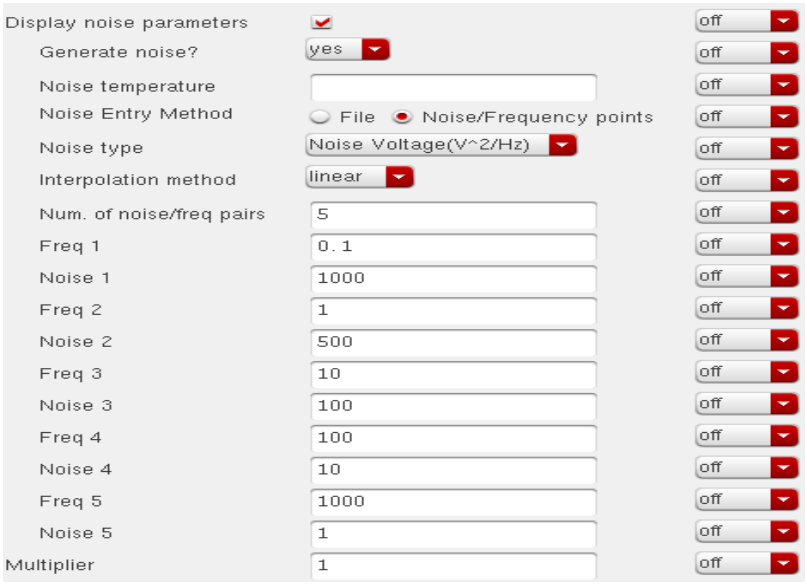


FIGURA 2-21. PARÁMETROS DE RUIDO EN PORT POR PARES DE PUNTOS. SE INTRODUCEN UN CONJUNTO DE VALORES DE DENSIDAD ESPECTRAL DE POTENCIA DE RUIDO (V^2/Hz) Y LAS FRECUENCIAS PARA LAS QUE SE PRODUCEN ESOS VALORES. SE DEBE ELEGIR TAMBIÉN EL TIPO DE INTERPOLACIÓN QUE REALIZARÁ CADECEN PARA CALCULAR EL RESTO DE PUNTOS (LINEAL O LOGARÍTMICA).

Este apartado se va a centrar en cómo se genera el ruido en Cadence y con qué tipo de análisis se puede representar, tanto en transitorio como en frecuencial. Por lo tanto, el ruido generado no va a ser necesario que sea un ruido Flicker "preciso". Esto va a ser así ya que, como se verá en el próximo

apartado, se ha proporcionado el conjunto de puntos de ruido del amplificador que vamos a tratar, por lo que la importancia reside en saber cómo tratar ese conjunto de ruido para añadirlo a cualquier circuito que realicemos en Cadence.

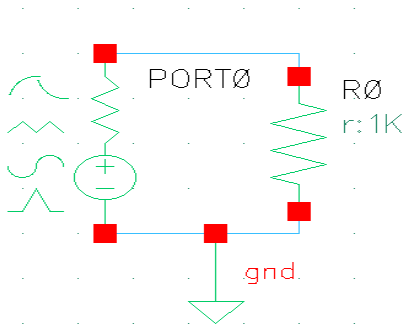


FIGURA 2-22. ESQUEMÁTICO PRUEBA GENERACIÓN RUIDO CADENCE.

Para analizar el ruido generado con los pares de puntos de ruido-frecuencia, se van a realizar dos tipos de análisis con ADE L sobre el circuito de la figura 2-22: Un análisis de ruido que muestra la representación espectral en pequeña señal del ruido a la entrada y la salida (figura 2-23) y un análisis transitorio dónde se marcará la opción para habilitar el ruido y se elegirán las frecuencias máximas y mínimas de ruido, así como las contribuciones de ruido de los distintos elementos que forman el diseño (figura 2-25).

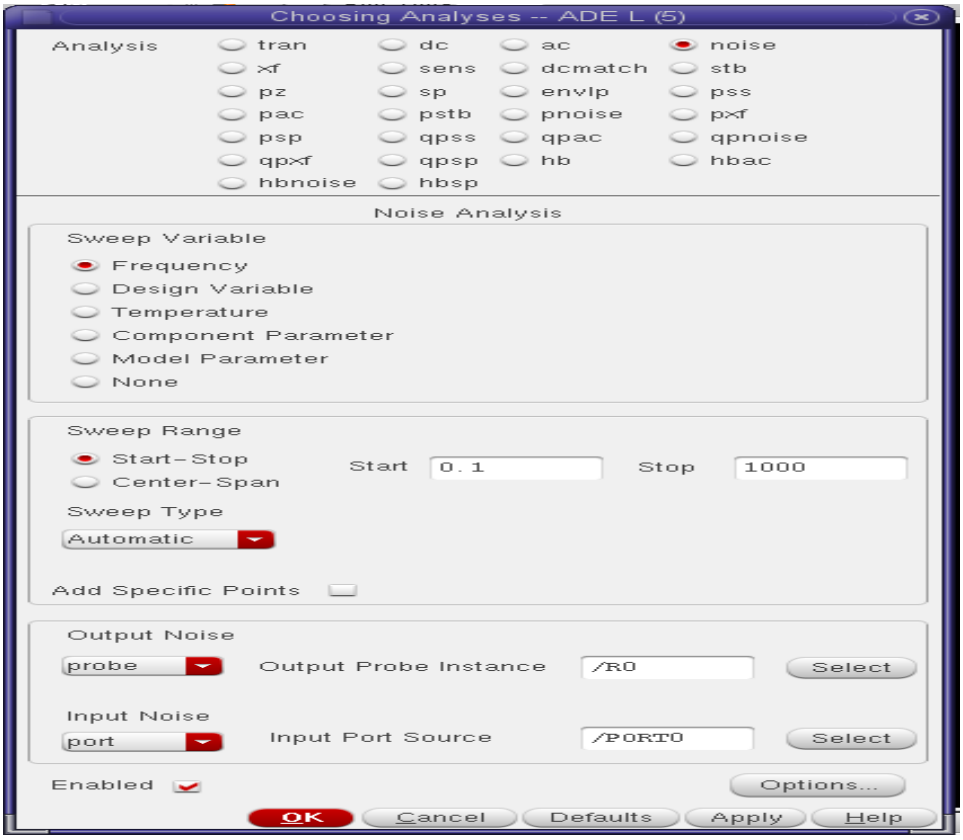


FIGURA 2-23. ANÁLISIS DE RUIDO CON ADE L CADENCE.

La representación de los puntos se realiza en ADE L en Result>Direct Plot> Squared Noise Input, para representar la potencia de ruido equivalente a la entrada (figura 2-24). Es importante tener en cuenta que este análisis es de pequeña señal, por lo que para las técnicas que se van a usar de eliminación de ruido no va a ser muy útil. Sin embargo, es bueno tener en cuenta que también es realizable este tipo de análisis.

Se puede apreciar en la figura 2-24 que con cinco pares de puntos se consigue formar un ruido aproximado al Flicker. No obstante, harían falta más puntos y más precisión en su cálculo para conseguir una aproximación mejor, pero como se dijo anteriormente, este apartado es un primer paso a la representación de ruido en Cadence y no se va a generar un ruido preciso.

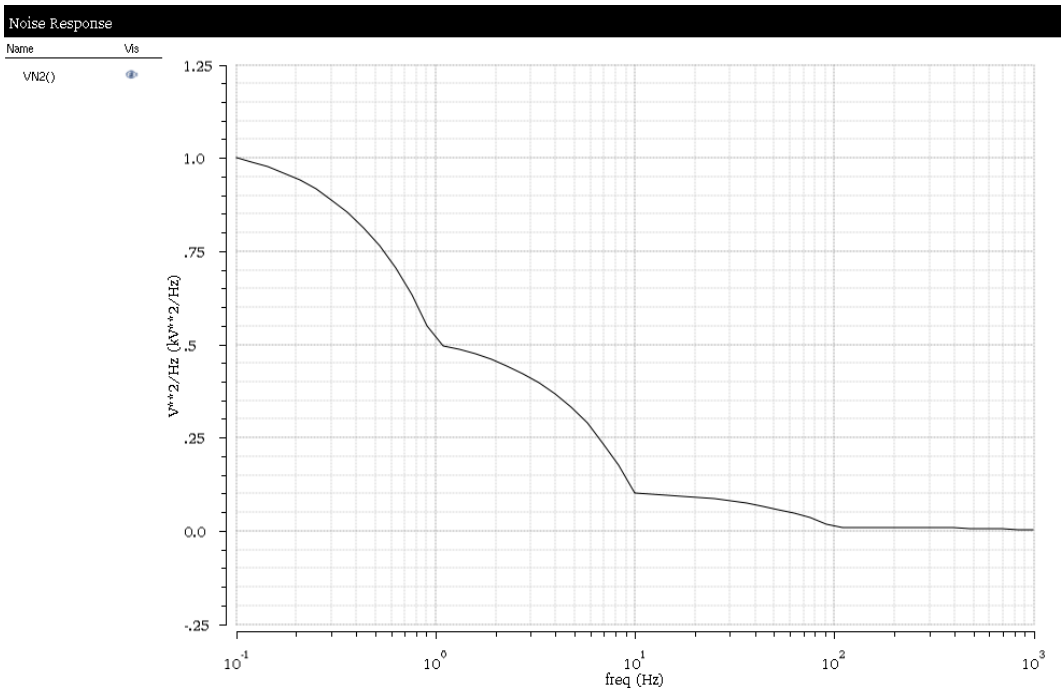


FIGURA 2-24. PSD PEQUEÑA SEÑAL RUIDO GENERADO POR PARES DE PUNTOS POTENCIA-FRECUENCIA.

Por otro lado, se ha realizado un análisis transitorio (figura 2-25) como ya se comentó, para ver cómo se comporta el ruido en gran señal, tanto en el dominio del tiempo, como en el de la frecuencia.

Para obtener esta última representación, se debe recurrir a la herramienta Calculator de ADE L. Dentro de ésta, se selecciona la señal que interesa y usando la función PSD se consigue una buena representación de la densidad espectral de potencia de la señal empleada (figura 2-26).

Sin embargo, para conseguir esta representación se deben, primero, configurar los parámetros siguientes:

- From/To se debe elegir el intervalo de tiempo donde empieza y termina el análisis espectral.
- Number of Samples define el número de puntos que va a tomar la transformada de Fourier, por lo que también define en parte la frecuencia máxima.
- Window Type define el tipo de ventana a usar y Window size el tamaño de la misma.
- Coherent Gain factor va de 0 a 1 y es la escala de nuestra señal.

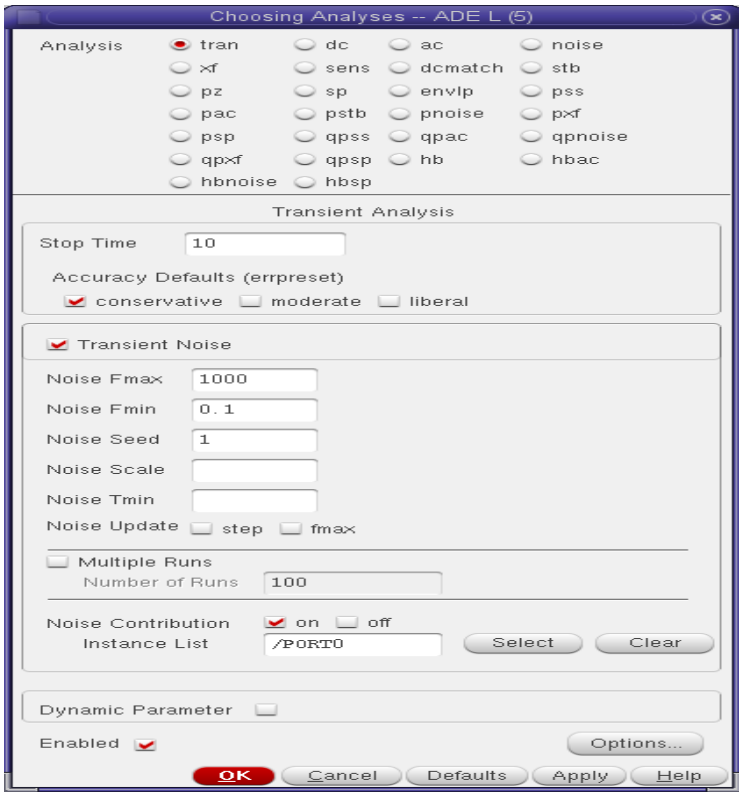


FIGURA 2-25. ANÁLISIS TRANSITORIO CON RUIDO CON ADE L CADENCE.

En la figura 2-26 se puede intuir el comportamiento como Flicker en frecuencia de esta señal de ruido que se ha generado. También se debe concluir que es necesario hacer análisis frecuenciales más precisos cuando se vayan a tratar señales de más importancia, ya que este caso era tan solo para mostrar cómo se hace en este entorno el análisis espectral.

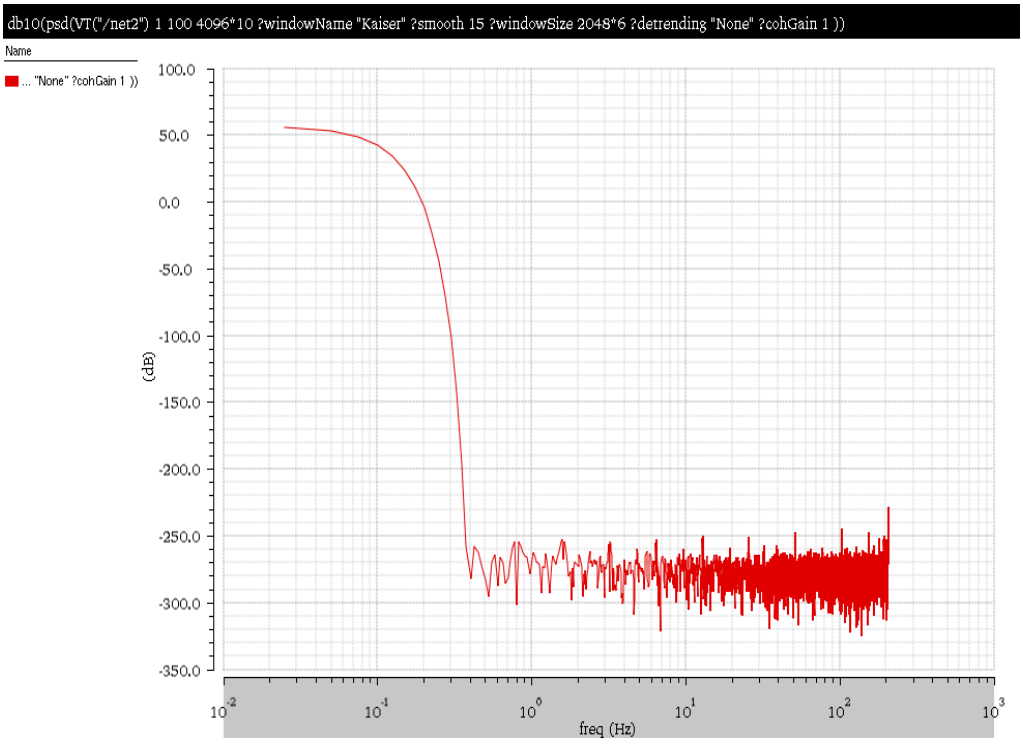


FIGURA 2-26. PSD RUIDO FLICKER PARES DE PUNTOS POTENCIA-FRECUENCIA CADENCE.

2.4 Ajuste de ruido equivalente a la entrada del Active-feedback time constant neural enhanced amplifier

Durante el transcurso del capítulo segundo, se han introducido diversas formas de generar distintas distribuciones de ruido, especialmente la de ruido Flicker. En este último apartado, se van a utilizar esas metodologías para aproximar el ruido generado al que ruido equivalente a la entrada del *active-feedback time constant neural enhanced amplifier*. Este ruido equivalente a la entrada ha sido proporcionado por el diseñador del circuito, en forma de archivo de pares de puntos en los cuales se da la densidad espectral de potencia del ruido equivalente a la entrada (V^2/Hz) y la frecuencia a la que se produce ese valor (Hz).

El citado archivo de ruido ha sido producido por Cadence de uno de los análisis de ruido realizados sobre el diseño y exportado en formato .txt.

El apartado, por consiguiente, se va a dividir en dos subbloques: uno donde se trabajará con MatLab-Simulink y otro con Cadence.

2.4.1 Ajuste del ruido equivalente a la entrada del Active-feedback time constant neural enhanced amplifier mediante MatLab-Simulink

Lo primero a realizar es introducir los valores del ruido equivalente a la entrada del amplificador. Para ello creamos la siguiente matriz en MatLab, siendo la primera columna las frecuencias y la segunda los valores de la densidad espectral de potencia:

```
Pot_Flicker_JoseLuis= [
0.1318256738556408, 3.385965273595448e-10
0.2290867652767775, 1.123403584264365e-10
0.3981071705534976, 3.738254414585218e-11
0.6918309709189371, 1.249104177576001e-11
1.202264434617414,  4.199739225569767e-12
2.089296130854041,  1.426123264030147e-12
3.630780547701017,  4.935624203575941e-13
6.30957344480194,   1.775074989005311e-13
10.96478196143186,  6.801520591009715e-14
19.05460717963249,  2.844089499178507e-14
33.11311214825915,  1.326688534832312e-14
57.54399373371577,  7.006419703965821e-15
100.0000000000001,  4.203143755847935e-15
173.7800828749378,  2.845241811719534e-15
301.9951720402021,  2.136077669290098e-15
524.8074602497734,  1.739299350525371e-15
912.0108393559111,  1.519971705634813e-15
1584.893192461116,  1.393783361130265e-15
2754.228703338171,  1.31997142822247e-15
4786.300923226392,  1.276655527898097e-15
8317.637711026726,  1.251877832540469e-15
14454.3977074593,   1.240053609623719e-15
```

```
25118.86431509585,   1.242030057749891e-15
43651.58322401669,   1.271309245151256e-15
75857.75750291854,   1.377792337131149e-15
131825.673855641,    1.723653753056017e-15
229086.7652767778,   2.821872852187909e-15
398107.1705534982,   6.237093148284751e-15
691830.9709189382,   1.499038875908159e-14
1202264.434617416,   2.666175464718815e-14
2089296.130854045,   2.711467128285201e-14
3630780.547701024,   1.621294381642545e-14
6309573.444801951,   7.040109042257431e-15
10964781.96143188,   2.754975943860542e-15
19054607.17963253,   1.152753387306228e-15
33113112.14825921,   5.991689388626329e-16
57543993.73371588,   4.114026476835569e-16]
```

Después de introducirla, lo ideal sería realizar una representación gráfica (figura 2-27) de la misma para analizar si los tipos de ruido que conforman la distribución son tal y como se predijo en el primer apartado del trabajo.

Se ha utilizado el siguiente código para la representación del mismo:

```
figure();
semilogx((Pot_Flicker_JoseLuis(:,1)),10*log10(Pot_Flicker_JoseLuis(:,2)),'ro-
','linewidth',2);
grid;
title('Ruido Equivalente a la Entrada A-FTCENA');
xlim([0.15 30000]);
xlabel('Frequency (Hz)')
ylabel('Power/Frequency (dB/Hz)')
```

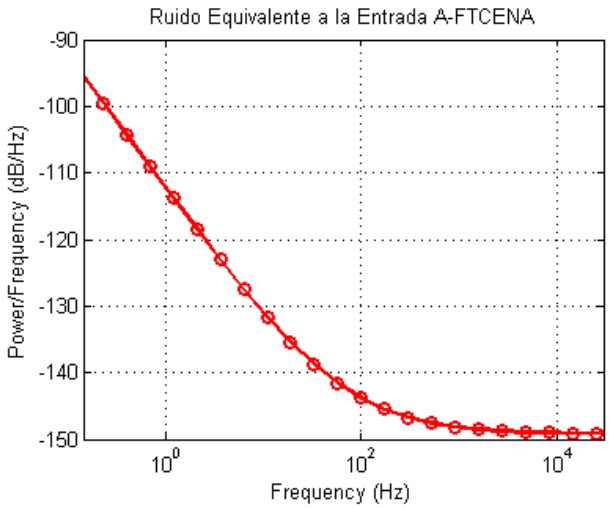


FIGURA 2-27. REPRESENTACIÓN EN MATLAB DEL RUIDO EQUIVALENTE A LA ENTRADA DEL ACTIVE-FEEDBACK TIME CONSTANT NEURAL ENHANCED AMPLIFIER. SE REPRESENTAN DE 0.15HZ A 30KHZ DEBIDO A QUE A PARTIR DE ESTA FRECUENCIA, EL VALOR DE LA DENSIDAD ESPECTRAL DE POTENCIA SE MANTIENE PRÁCTICAMENTE CONSTANTE.

Se pueden apreciar en las figuras dos aspectos muy importantes a tener en cuenta:

- La frecuencia de esquina se encuentra aproximadamente en 500 Hz (momento en el que deja de caer a 10dB por década). Esto va a tener que tenerse en cuenta a la hora de generar un ruido que se aproxime a éste y a la hora de emplear las distintas técnicas para la eliminación del ruido a bajas frecuencias.
- El ruido Flicker a bajas frecuencias tiene una pendiente muy elevada, más aproximada a $1/f^2$ que a $1/f$. Por ello se van a tener que realizar varios procedimientos alternativos para poder ajustar un ruido que se le aproxime.

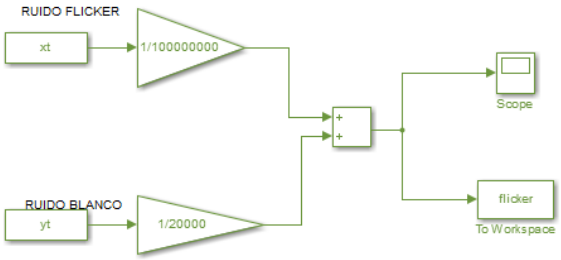


FIGURA 2-28. MODELO SIMULINK GENERACIÓN RUIDO ACTIVE-FEEDBACK TIME CONSTANT NEURAL ENHANCED AMPLIFIER MEDIANTE DSP.COLOREDNOISE().

La metodología primera que se va a emplear consiste en generar el ruido mediante dsp.ColoredNoise(*x*,*y*). Para ello, como se muestra en la figura 2-28, lo que se hará es utilizar dos funciones dsp.ColoredNoise diferentes: una para generar un ruido blanco y otro para generar un ruido marrón, ya que como se vio anteriormente, el ruido del amplificador se ajustaba más a $1/f^2$ que a un ruido rosa convencional. Por tanto, se tendrá:

- bn = dsp.ColoredNoise(2,1e7+1); → Ruido Marrón.
- wn = dsp.ColoredNoise(0,1e7+1); → Ruido Blanco.

Se multiplica en Simulink por una constante cada uno de los ruidos para fijar la frecuencia de esquina y la magnitud inicial de los ruidos. También se multiplica por un cierto factor de escala, en este caso 0.5 para conseguir que la aproximación posterior por el ruido integrado en banda sea la mejor posible, ya que como se verá este cálculo es realmente lo que dicta la validez de la aproximación.

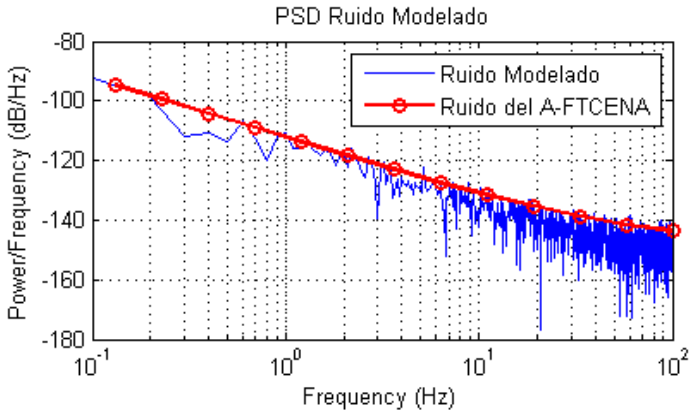


FIGURA 2-29. COMPARACIÓN RUIDO GENERADO DSP.COLOREDNOISE() CON RUIDO EQUIVALENTE A LA ENTRADA 100 HZ.

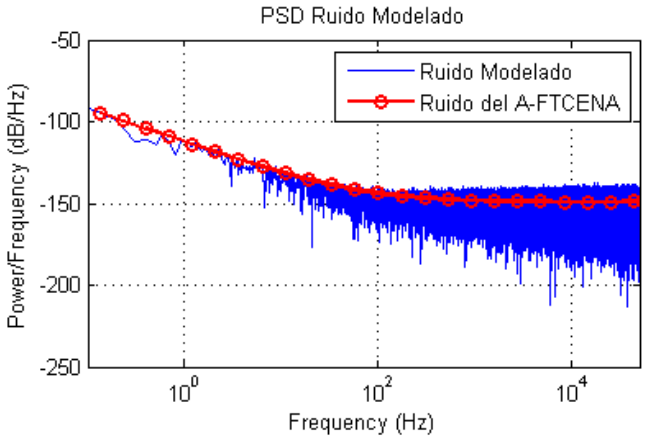


FIGURA 2-30. COMPARACIÓN RUIDO GENERADO DSP.COLOREDNOISE() CON RUIDO EQUIVALENTE A LA ENTRADA 30 KHZ.

Aunque resulte el ruido generado, a primera vista, una buena aproximación, no se debe olvidar, como ya se comentó cuando se introdujo esta función, que se generan distintas PSDs cada vez que el programa se ejecuta lo que puede hacer que los resultados, aunque sean por poco, vayan cambiando.

Para comprobar cuantitativamente el parecido de ambas distribuciones, lo ideal sería someter a ambas a un análisis mediante la integración en banda. Si en este análisis se consiguen resultados parejos para varias iteraciones (ya que como se acaba de decir las PSDs generadas van variando), se podrá confirmar que se trata de una aproximación correcta y se podrá usar en apartados posteriores.

Para llevar a cabo la integración en banda (figura 2-31) se han utilizado los códigos citados en Anexo B.

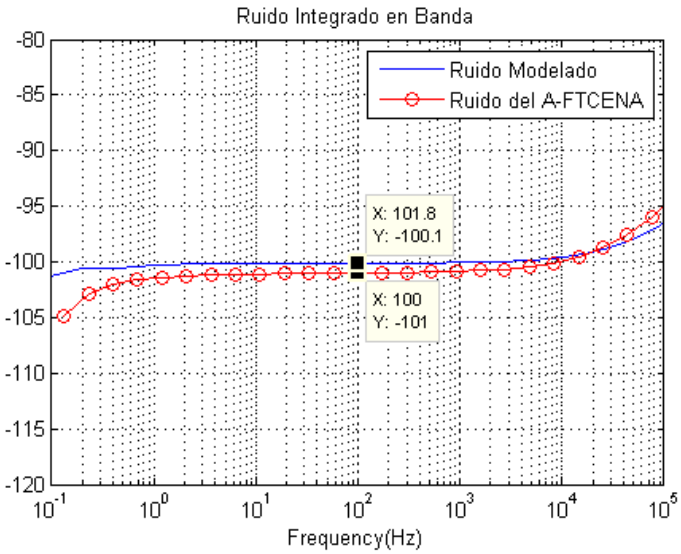


FIGURA 2-31. COMPARACIÓN RUIDO INTEGRADO EN BANDA DSP.COLOREDNOISE() CON RUIDO EQUIVALENTE A LA ENTRADA. PARA FRECUENCIAS MEDIAS LA DIFERENCIA ES DE 1DB, POR LO QUE RESULTA UNA BUENA APROXIMACIÓN.

El cálculo que se realiza en Script_RuidoIntegradoEnBanda_JL.m da que, para una frecuencia aproximada de 100 Hz, el ruido integrado en banda del ruido generado por la función dsp.ColoredNoise vale -100.085 dB, mientras que el ruido generado por el amplificador estudiado tiene un valor de -101.07 dB.

Con esto, se puede concluir que con esta función para generar ruido se consigue una muy buena aproximación al ruido que se desea modelar. No obstante, hay que remarcar de nuevo, que cada vez que se usa la función, se generará un nuevo conjunto de puntos y por tanto los resultados, aunque en un espacio pequeño, van variando, lo que resulta perjudicial para el trabajo.

Otra forma de generar un ruido en MatLab es la de los mapas discretos. Debido a la elevada pendiente a bajas frecuencias de la distribución espectral de ruido del amplificador estudiado, la conformación del ruido se va a tener que conformar mediante tres mapas discretos diferentes (figura 2-32):

- Un mapa discreto *saltador* para generar Flicker a muy bajas frecuencias (0.2 a 10 Hz) con *ml*=2.3, *me*=-1.8, y *Ts*=1s.
- Un mapa discreto *saltador* para generar Flicker a bajas frecuencias (10 a 1 kHz) con ml=2.4, me=-1.8, y Ts=1e-3s.
- Un mapa de Bernoulli para generar ruido blanco a partir de la frecuencia de esquina (1kHz) con mb=1.83.

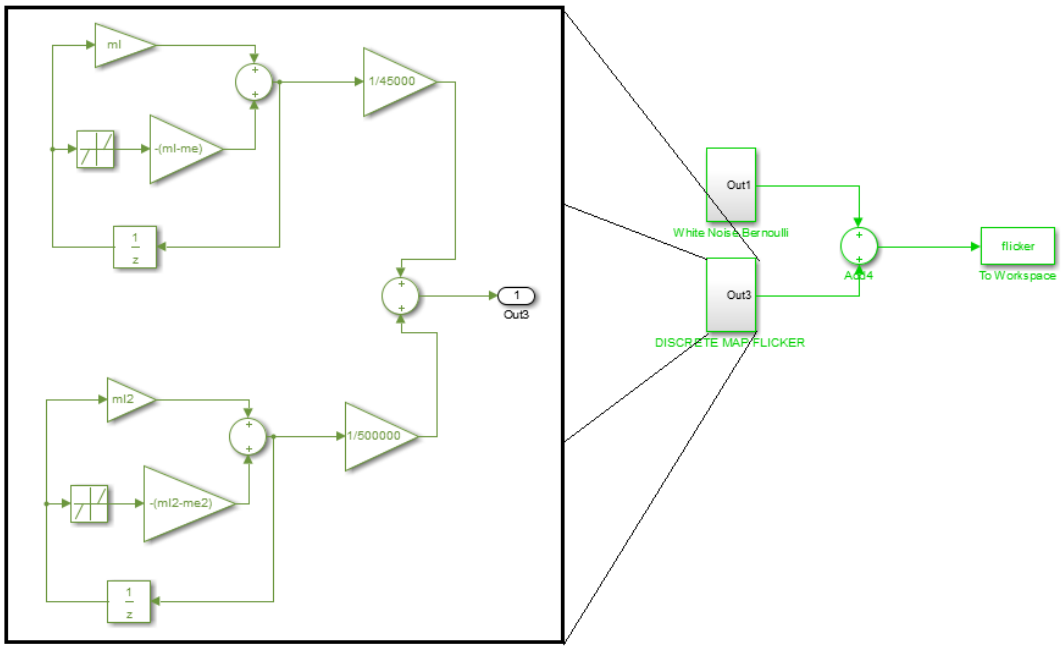


FIGURA 2-32. MODELO SIMULINK DE GENERACIÓN DEL RUIDO DEL ACTIVE-FEEDBACK TIME CONSTANT NEURAL ENHANCED AMPLIFIER MEDIANTE MAPAS DISCRETOS.

Como en el caso anterior, la señal se debe multiplicar por cierta ganancia para conseguir la frecuencia de esquina y las magnitudes deseadas, así como por un factor de escala (0.55) para el ruido integrado en banda. Los resultados pueden apreciarse en las figuras 2-33 y 2-34.

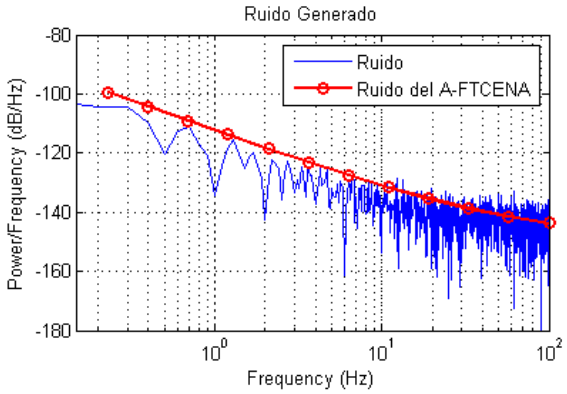


FIGURA 2-33. COMPARACIÓN RUIDO GENERADO MEDIANTE MAPAS DISCRETOS CON RUIDO EQUIVALENTE A LA ENTRADA 100 HZ.

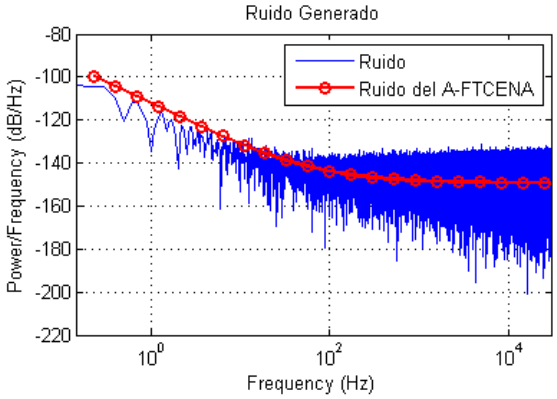


FIGURA 2-34. COMPARACIÓN RUIDO GENERADO MEDIANTE MAPAS DISCRETOS CON RUIDO EQUIVALENTE A LA ENTRADA 30KHZ.

A primera vista se puede deducir que este método puede resultar una buena aproximación para la generación del ruido deseado. Sin embargo, se realizará un análisis del ruido integrado en banda (figura 2-35) para comprobar si se está en lo cierto.

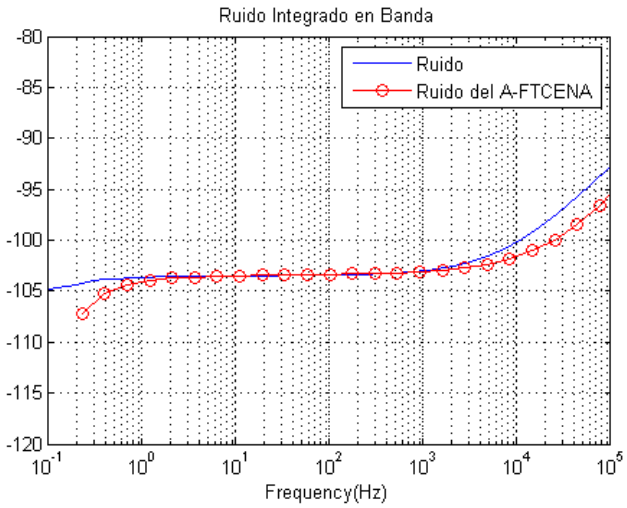


FIGURA 2-35. COMPARACIÓN RUIDO INTEGRADO EN BANDA MEDIANTE MAPAS DISCRETOS CON RUIDO EQUIVALENTE A LA ENTRADA. PARA UNA FRECUENCIA APROXIMADA DE 1 KHZ, EL RUIDO INTEGRADO EN BANDA DE LOS MAPAS DISCRETOS ES -103.0016DB MIENTRAS QUE EL DEL AMPLIFICADOR ESTUDIADO ES -103.1127DB.

Como conclusión, los mapas discretos constituyen un muy buen método para generar un ruido de unas determinadas características, consiguiendo una gran fiabilidad en la aproximación, como se acaba de demostrar.

Sin embargo, el único punto negativo de estos mapas es, aunque se haya hecho una aproximación paramétrica con respecto a la formación de ruido, que sus parámetros aún presentan cierta componente de aleatoriedad, por lo que puede costar en ciertas ocasiones esfuerzos algo más grandes para generar una buena aproximación que con el método de la función dsp.ColoredNoise(). No obstante, los resultados son mucho más precisos, fiables y capaces de ser implementados.

2.4.2 Ajuste del ruido equivalente a la entrada del Active-feedback time constant neural enhanced amplifier mediante Cadence

Ya se habló en uno de los apartados del capítulo (2.3.2) de que en Cadence existía la opción de generar un ruido mediante un Port solo con introducirle el archivo de puntos de ruido (figura 2-36). Para esta simulación vamos a usar el mismo esquemático de la figura 2-22.

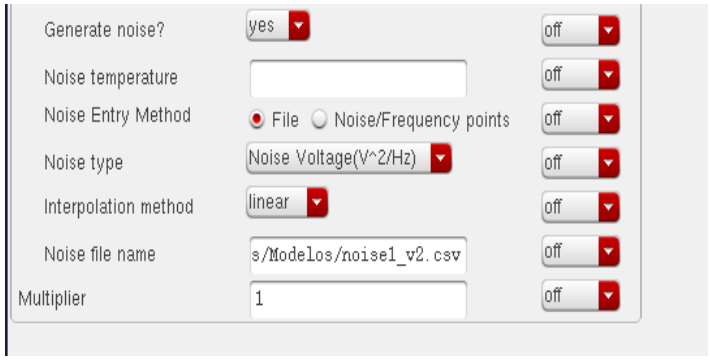


FIGURA 2-36. GENERACIÓN RUIDO CADENCE MEDIANTE ARCHIVO DE RUIDO.

Una vez tenemos nuestro ruido generado con el Port, se somete a los mismos análisis que en el apartado que se acaba de nombrar para comprobar que el ruido producido ha sido el correcto (figura 2-37).

Concluyendo este apartado es conveniente decir que esta forma de generar ruidos en Cadence es útil, fiable y muy productiva, ya que, si se van usar técnicas para eliminar un ruido, no es necesario que se introduzca el circuito que lo produce para poder trabajar con el ruido del mismo, lo que proporciona a los distintos usuarios un abanico de posibilidades para aislar el ruido de sus diseños y poder operar con ellos.

No obstante, lo complicado en este entorno, es la representación frecuencial ya que se deben elegir muy bien los parámetros para obtener una representación correcta.

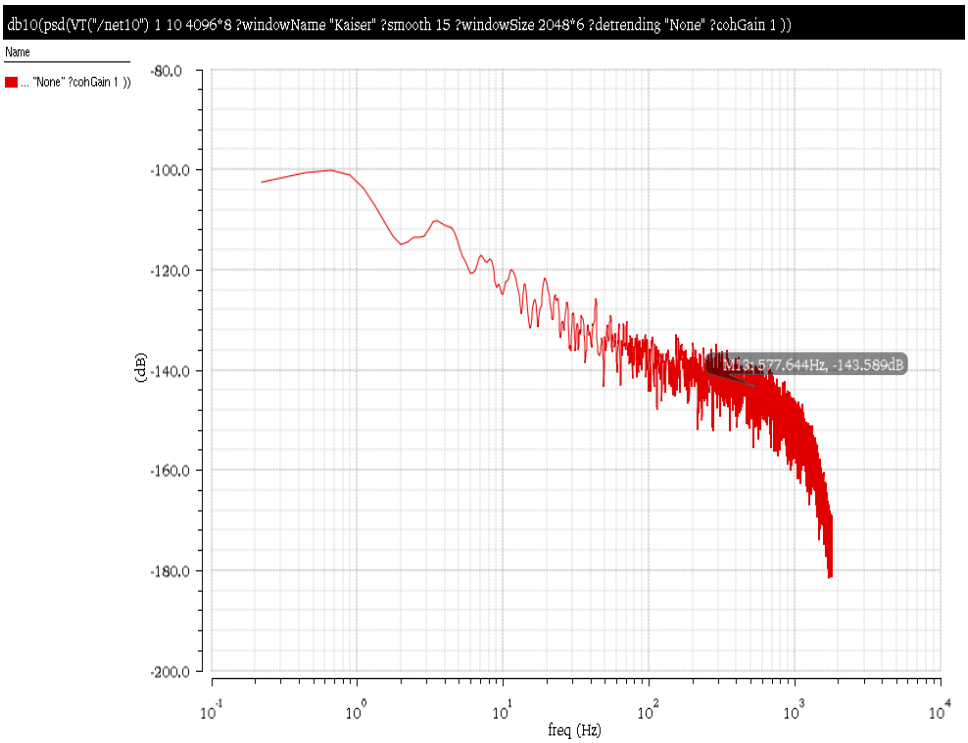


FIGURA 2-37. RUIDO FLICKER EQUIVALENTE ENTRADA EN CADENCE MEDIANTE ARCHIVO.

3 ARQUITECTURA BÁSICA CHOPPER

En primer lugar, antes de comenzar a buscar soluciones al problema que se plantea, es necesario acotar y comprender de forma precisa y completa todo lo que éste incumbe.

Por ello, en el apartado anterior se realizó un análisis en profundidad del problema y se comentaron diversas maneras para poder sintetizarlo en varios entornos de software distintos.

La literatura dota de diversas soluciones para paliar el ruido a baja frecuencia [15] para cualquier tipo de amplificador de alta ganancia, y para circuitos usados en Neural Recording en concreto [10] [6]. Para el trabajo particular que se realiza, se va a estudiar la técnica de Chopper para conseguir eliminar el ruido a baja frecuencia y offset.

3.1 Concepto y Arquitectura del Amplificador Chopper

El concepto de Chopper en Neural Recording es concebido como una técnica de eliminación de ruido a baja frecuencia y offset en los circuitos de amplificación. Este método se basa en el dominio frecuencial, ya que se trata de una técnica de modulación de la señal. Para ello, se precisa de la utilización de llaves ("switches"), como se verá a continuación. No se trata de un concepto nuevo, ya que se utilizaba hace alrededor de 70 años para fabricar ganancias dc de alta precisión con amplificadores ac-acoplados [15].

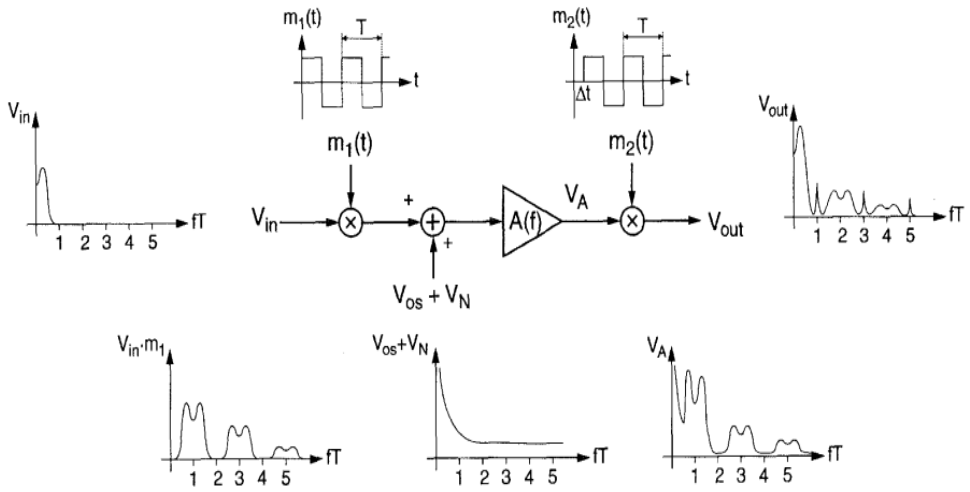


FIGURA 3-1. CONCEPTO DE MODULACIÓN CHOPPER Y REPRESENTACIÓN EN FRECUENCIA [15].

El concepto de la modulación Chopper (figura 3-1) ideal es el siguiente: la señal de entrada se modula mediante la señal m1(t) hasta la frecuencia de Chopper. Esta señal m1(t) es la señal de modulación Chopper y se trata de una onda cuadrada que va desde 1 hasta -1 con un periodo $T = 1/f_{Chopper}$(modulación por anchura de pulsos).

De esta forma la señal de entrada, que se llamará a partir de ahora Vin, en el dominio espectral debido al "aliasing", pasa a situarse principalmente en la frecuencia de Chopper y en múltiplos impares de ésta (impares debido a que la señal m1(t) es cuadrada).

Una vez modulada, a la entrada del amplificador, que se supondrá ideal (excepto por el ruido), la señal se presenta de la siguiente forma: se tiene Vin en armónicos de alta frecuencia, múltiplos impares de la frecuencia de Chopper, mientras que a baja frecuencia se le ha añadido el ruido del amplificador (Flicker).

A la salida del amplificador se obtiene esta misma señal, pero amplificada, que posteriormente será demodulada mediante una señal m2(t). Para el caso ideal m2(t)=m1(t).

Después de esta modulación, la señal de salida quedará: Vin se encontrará de nuevo en su posición inicial mientras que el ruido ocupará la posición que anteriormente ocupaba Vin, es decir, pasará a ocupar los armónicos impares múltiplos de la frecuencia de Chopper.

A esta señal de salida será necesario aplicarle una etapa para filtrar (filtro paso bajo) los armónicos impares múltiplos de la frecuencia de Chopper donde se encontrará situado el ruido. También se verá en el siguiente apartado que esta etapa es necesaria para filtrar otras frecuencias indeseadas.

Por lo tanto, como se mencionó al principio del capítulo, esta técnica será ideal para eliminar ruidos a bajas frecuencias (Flicker principalmente) y el offset del amplificador.

La arquitectura básica del amplificador Chopper es la mostrada en la figura 3-2 de forma simplificada. Se puede apreciar la presencia a la salida del filtro paso bajo que se comentó anteriormente. También es significativo ver que la modulación y demodulación se realiza mediante dos pares de llaves desfasadas entre sí la frecuencia de Chopper, es decir, cuando un par de llaves están abiertas, el otro par están cerradas y viceversa. De esta forma, se consigue una modulación cuadrada entre Vin y -Vin a la salida de las llaves. La implementación normal de estas llaves es mediante transistores MOS, que se colocarán en región saturada para simular la llave cerrada, o en corte para la llave abierta.

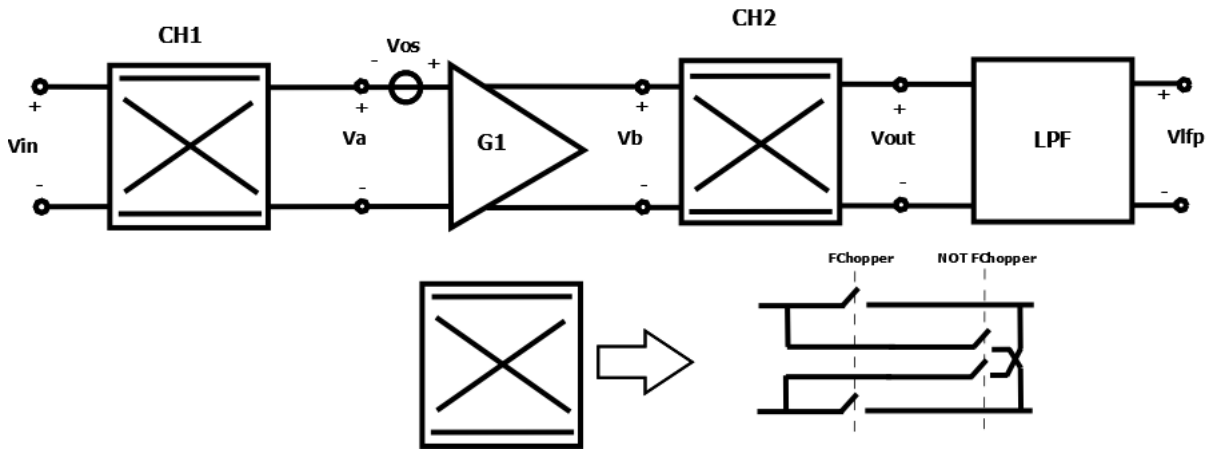


FIGURA 3-2. ARQUITECTURA BÁSICA AMPLIFICADOR CHOPPER. LA MODULACIÓN SE REALIZA CON PARES DE LLAVES DESFASADAS FCHOPPER.

Además de esta arquitectura básica, existen otras muchas topologías complejas de amplificadores Chopper para paliar las no idealidades que éstos presentan y que exigen un estudio más complejo de la técnica para analizar los pros y los contras que puede tener su implementación sobre el circuito con el que se quiere trabajar.

3.2 No idealidades y problemas asociados al amplificador Chopper

No se debe olvidar que todo lo visto anteriormente solo es válido para el caso ideal. En el caso real se tiene un factor limitante desde el punto de vista de diseño de estas arquitecturas [25]: los amplificadores tienen un ancho de banda finito. Para hacer un análisis de la forma más conceptual posible, se asumirá que el amplificador debido a este ancho de banda se comportará como un filtro paso bajo ideal, es decir, su ganancia va a valer A_o hasta la frecuencia de corte del amplificador, f_{corte}, a partir de la cual la ganancia va a pasar a ser nula.

Este hecho supone que la frecuencia de Chopper va a estar delimitada por el ancho de banda del

amplificador, ya que si se excede, al encontrarse Vin a la entrada del amplificador modulada a la frecuencia de Chopper y el ruido en la baja frecuencia, este último se amplificaría más que Vin, llegando Vin incluso a no amplificarse, que para las arquitecturas de Neural Recording donde se buscan amplificadores de alta ganancia, puede suponer un problema mayor que el de la propia presencia de ruido que se quería paliar. Además, el hecho del ancho de banda finito del amplificador conlleva otros dos problemas.

Por una parte, significa que el circuito de amplificación va a introducir un retraso ("delay") a la señal. Este efecto no es tan negativo si se tiene en cuenta, ya que se puede corregir haciendo que $\mathrm{m2(t)} = \mathrm{m1(t)} + \Delta t$; siendo Δt el retraso introducido por el amplificador, así se consigue que éste no influya en el sistema.

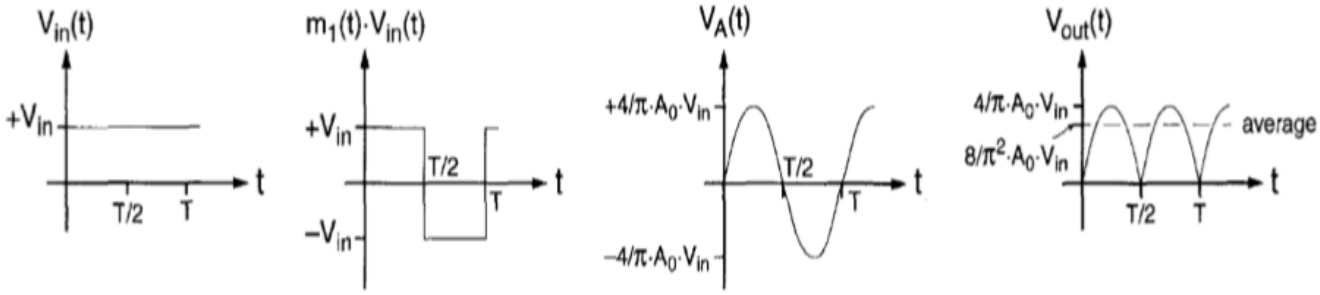


FIGURA 3-3. REPRESENTACIÓN TEMPORAL DE CHOPPER CON AMPLIFICADOR DE ANCHO DE BANDA FINITO [15]. LA ONDA DE SALIDA PRESENTA UNA ATENUACIÓN DE GANANCIA DEBIDO AL ANCHO DE BANDA DEL AMPLIFICADOR.

Por otra parte [15], si se supone que la señal de entrada del sistema Vin es una señal dc (caso extremo de señales de baja frecuencia), la señal de salida del amplificador, a la que se llamará $\mathrm{V_A}$, debido a este ancho de banda finito, pasa de ser la señal cuadrada que era a la entrada, a ser una señal sinusoidal con una frecuencia igual a la frecuencia de Chopper y con una amplitud $\frac{4}{\pi}(\mathrm{A_o} \cdot \mathrm{Vin})$. Después de ser demodulada por m2(t), la señal se convierte en una señal sinusoidal rectificada que incluye armónicos pares múltiplos de la frecuencia fundamental del Chopper. Por lo tanto, debe haber una etapa de filtro paso bajo posterior a esta para poder volver a tener la señal dc que teníamos a la entrada amplificada.

Tras esta etapa de filtrado, la señal dc tendrá un valor $\frac{8}{\pi^2}(\mathrm{A_o} \cdot \mathrm{Vin}) \cong 0.8 \cdot \mathrm{A_o} \cdot \mathrm{Vin}$, lo que supone una pérdida adicional de ganancia de la señal de salida del conjunto del sistema.

Además de estas no idealidades debidas al amplificador, existen otros problemas relacionados con esta técnica. Aunque no se entrará en mucho detalle, ya que los circuitos y diseños que se realizarán serán todos a un alto nivel de abstracción; por el hecho de que las llaves se implementan mediante transistores MOS, al encontrarse conmutando, sufrirán "spikes" debidos al efecto de la inyección de carga, que se verá en el circuito como un offset residual [15] [25].

Si se realiza un modelo del amplificador Chopper [25] (figura 3-4) donde las capacidades de C_1 a C_4 modelan este efecto de inyección de carga y R_1 y R_2 modelan las resistencias de los MOS y la resistencia de la fuente de entrada, se apreciará que si C_1 es idéntico a C_2 y C_3 es idéntico a C_4, es decir, si la estructura del amplificador es totalmente diferencial, no se producirá ningún tipo de offset residual por inyección de carga. Sin embargo, esto es difícil de conseguir por lo que se tendrá un offset residual, que operando se llega que es igual a [25]:

$$Vos_{res} = Vos_{res1} + Vos_{res2} = \frac{2 \cdot V_F \cdot F_{Chopper} \cdot (R_1 + R_2)(C_1 - C_2)(C_3 - C_4)}{G_1} \quad (3\text{-}1)$$

Siendo G_1 la transconductancia del amplificador, $F_{Chopper}$ la frecuencia de Chopper y V_F el voltaje conductor del reloj.

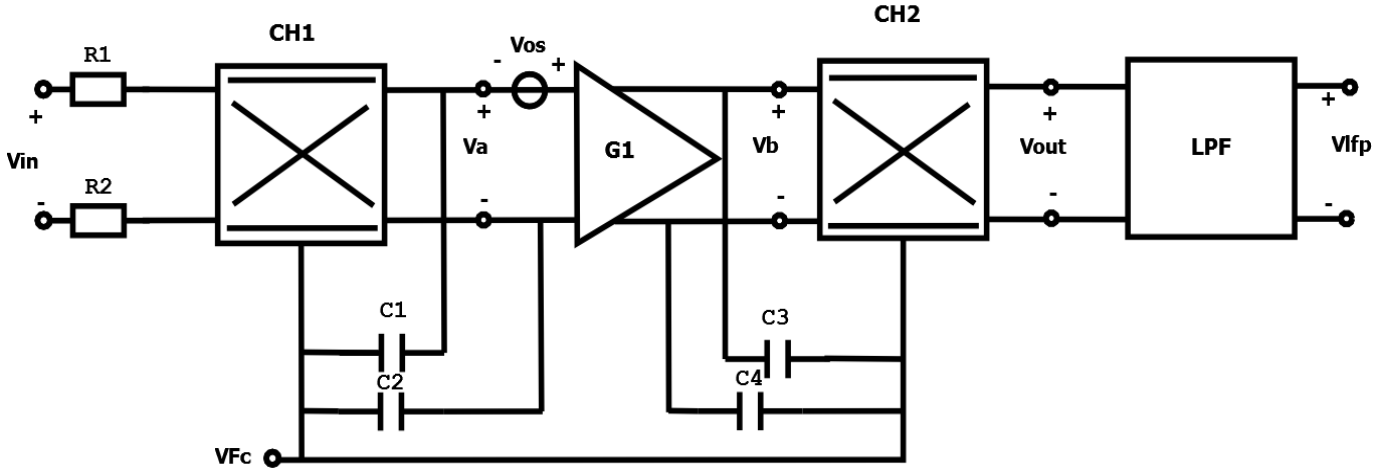


FIGURA 3-4. MODELO CHOPPER CON INYECCIÓN DE CARGA.

Por último, existe un problema debido a la presencia de offset en el amplificador. El offset, al igual que el ruido, no se elimina hasta la segunda modulación y posterior paso por el filtro paso bajo, lo que conlleva que éste sea amplificado antes de ser eliminado. Al amplificarlo junto con la señal Vin, puede suponer que esta señal de salida alcance la región de saturación del amplificador, lo que supondrá la pérdida de información. Este problema se puede solucionar con topologías avanzadas de amplificadores Chopper, como las estudiadas en el tercer capítulo de [25].

3.3 Requerimientos principales del amplificador Chopper

Cuando se realiza una técnica de Chopper, el principal parámetro a elegir es la frecuencia de modulación, es decir, la frecuencia de Chopper. Para seleccionarla, se deben tener en cuenta tres parámetros fundamentales:

- La frecuencia de corte del amplificador, ya que como se acaba de mostrar en el apartado anterior, si la frecuencia de Chopper es mayor que ésta se producirá una pérdida de ganancia en el sistema [15].
- La frecuencia de esquina del ruido Flicker. Esta frecuencia delimita hasta donde el ruido Flicker va ser mayor al ruido blanco, y por tanto va a influir de forma importante en el circuito. Si se elige una frecuencia de Chopper menor a esta frecuencia de esquina, a la salida se observará que no todo el ruido Flicker ha sido filtrado.
-
- La frecuencia de la señal de entrada. Para ello habrá que basarse en el teorema de Nyquist, es decir, si se necesita recuperar una señal en su totalidad después de muestrearla, la frecuencia a la que se muestrea debe ser al menos dos veces la frecuencia de la señal que se muestrea. Si la frecuencia de Chopper elegida es menor a dos veces la frecuencia de la señal de entrada, después de demodular la señal es imposible que se obtenga la misma señal de entrada amplificada en su totalidad.

Un resumen de lo expuesto puede ser:

$$f_{corte} > f_{chopper} > (f_c \ \& \ 2f_{entrada}) \quad (3\text{-}2)$$

Siendo “&” la representación del operador lógico and.

No obstante, existe un criterio más a la hora de elegir la frecuencia de Chopper.

Al estar trabajando con amplificadores para Neural Recording, éstos exigen ocupar áreas pequeñas. En el diseño micro-electrónico, uno de los principales inconvenientes que se pueden encontrar es que los condensadores, por lo general, ocupan un tamaño bastante más considerable que el resto de componentes. En el circuito Chopper, los condensadores principales se encuentran en el filtro paso bajo de la salida, por lo que cuanto menos selectivo y de menor orden sea el filtro, menor será el tamaño de las capacidades que habrá que colocar y menos complejo será su diseño. Para conseguir esto, cuanto más alejada esté la frecuencia de Chopper de la frecuencia de entrada, menos complejo y selectivo deberá ser el filtro para poder eliminar más fácilmente las componentes indeseadas, ya que la separación entre éstas y la señal de entrada amplificada y demodulada será mayor.

Sin embargo, debido a la frecuencia de corte del amplificador, esta distancia será finita y por lo tanto en el diseño del filtro en la práctica se establecerá un compromiso entre ruido a la salida y pérdida de ganancia en el filtro. Esto se verá un poco en apartados posteriores, pero no se indagará mucho ya que el diseño del filtro no es el objetivo de este trabajo.

3.4 Implementación del amplificador Chopper

En este apartado se va a implementar modelos básicos de alto nivel del amplificador Chopper, tanto en MatLab-Simulink como en Cadence. Hay que decir que la etapa amplificadora del circuito se considerará ideal (ancho de banda infinito y sin saturación) y por tanto se obviará (es decir, se tomará un amplificador de ganancia es unitaria). Se va a realizar de esta forma, debido a que se dedicará un capítulo a trabajar con un amplificador real donde se verán todas las no idealidades comentadas anteriormente, siendo el objetivo de este apartado el de mostrar cualitativamente el concepto estudiado y dejar preparados una serie de entornos para la posterior implementación del modelo de un amplificador real.

El ruido utilizado en ambos casos sigue siendo el ruido equivalente a la entrada del *active-feedback time constant neural enhanced amplifier*, generado en MatLab-Simulink mediante mapas discretos y en Cadence mediante su archivo de puntos y el componente Port perteneciente a analogLib.

3.4.1 Implementación del amplificador Chopper en MatLab-Simulink.

Para facilitar su explicación y comprensión, se dividirá este apartado en etapas. El “script” utilizado recibe el nombre de Script_Pink_Noise_JL.m y tan solo con ejecutarlo y elegir por teclado el tipo de señal de entrada, mostrará todas las gráficas y resultados que se expondrán a continuación. El modelo de Simulink empleado recibe el nombre de Chopper_JL.slx. El esquema en conjunto del que se irá detallando paso a paso es el siguiente:

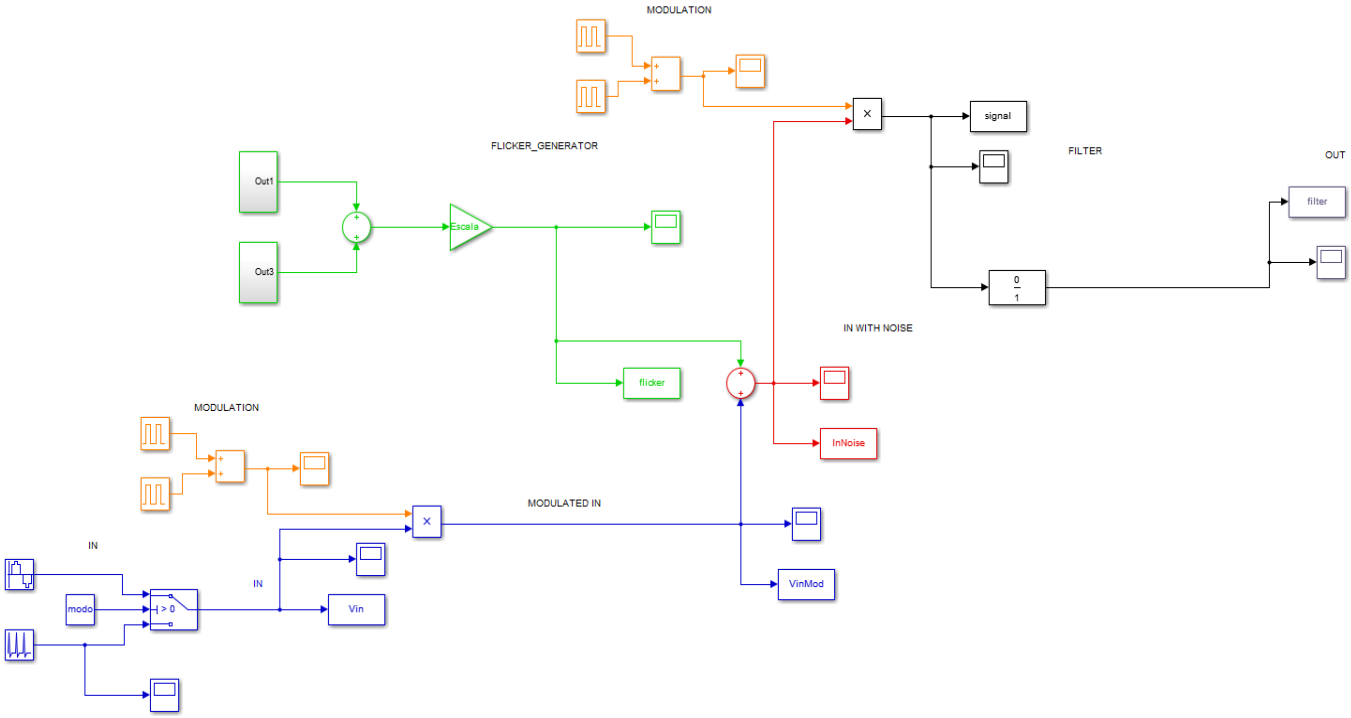


FIGURA 3-5. MODELO SIMULINK DEL CHOPPER IDEAL. CON LOS DIFERENTES COLORES SE REPRESENTAN LAS DIVERSAS ETAPAS DE ÉSTA TÉCNICA.

3.4.1.1 Señales de entrada MatLab-Simulink

Las señales de entradas que se utilizarán, simularán mediante MatLab-Simulink (figura 3-6) las señales implicadas en amplificadores de Neural Recording que ya se mostraron el capítulo primero del trabajo. Para ello, mediante un Switch se seleccionará el tipo de señal de entrada introduciendo un dato por teclado y pudiendo elegir entre potenciales acción y potenciales de campo locales.

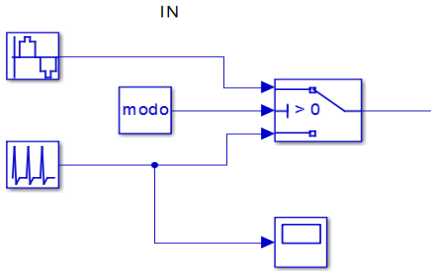


FIGURA 3-6. SIMULACIÓN SEÑALES ENTRADA SIMULINK (LFP Y AP).

Para el caso de los potenciales de campo locales se usa una señal sinusoidal de 0.1mV de amplitud y cuya frecuencia se ajusta según el tipo de onda cerebral que se elija mediante la introducción del tipo de onda por teclado, tal y como muestra el siguiente fragmento de código perteneciente al "script" principal:

```
%Frecuencias Onda
delta=1;
theta=5;
```

```
alpha=10;
beta=25;
gamma=40;
modo=input('Elija un modo (0 -> Potencial accion, 1 -> BrainWaves): ');
if modo ~=0
freq=input('Ingrese el tipo de onda cerebral: ');
filtro=(2*pi*200)/(s+2*pi*200); %El filtro de unas es distinto que el de otras
else
    filtro=(2*pi*5000)/(s+2*pi*5000);
    freq=1;
end
```

Se puede apreciar que se definirá un filtro u otro dependiendo si la entrada es una onda cerebral o un potencial de acción. Esto se debe a que el diseño del filtro no es un objetivo del trabajo y se deberían estudiar las características del mismo con un compromiso ganancia-ruido para poder filtrar todos los tipos de señales neuronales correctamente. Hay que tener en cuenta que los potenciales de acción son señales de frecuencias medias/altas, por lo que los efectos de la técnica de Chopper pueden resultar más perjudiciales que el efecto del propio ruido sobre éstas ondas (ya se mostraba como este ruido no afectaba prácticamente en nada a los potenciales de acción (figura 1-7).

No obstante, el *active-feedback time constant neural enhanced amplifier* que se estudiará está pensado para captar todo tipo de ondas cerebrales, por lo que los AP se tendrán en cuenta en el estudio del Chopper, aunque con un filtro distinto.

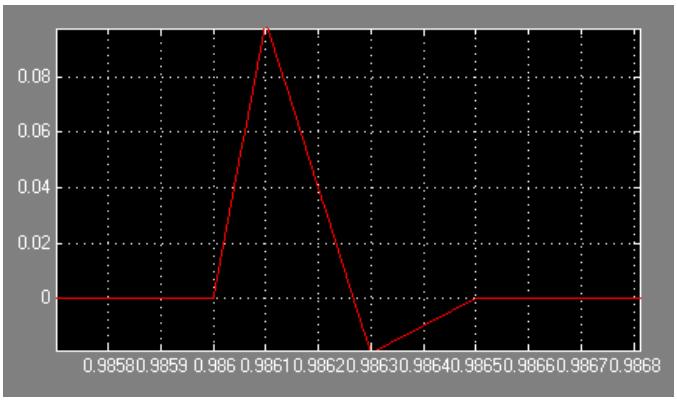


FIGURA 3-7. POTENCIAL DE ACCIÓN IMPLEMENTADO EN SIMULINK.

Estos potenciales de acción (figura 3-7) son implementados mediante Reapiting Sequence con una frecuencia de 1kHz. Un dato importante a tener en cuenta es que un potencial de acción puede interpretarse que tiene un offset de -70mV de los que pasa a los -40mV, baja a los -90mV y vuelve al estado de -70mV (debido a las transferencias de iones potasio y sodio en las neuronas, aunque no se entrará en términos biológicos). Sin embargo, al trabajar con MatLab-Simulink, la colocación de un offset supondrá que será multiplicado también cuando se opere con él, lo que dará errores respecto a lo que ocurriría eléctricamente en la realidad, por lo que se representarán los potenciales de acción, y todas las señales con offset del trabajo, mediante valores absolutos.

Para una mejor explicación de los bloques, se irá mostrando la señal tanto en el dominio de la frecuencia como del tiempo, al paso por cada uno de los bloques. Para la representación de la señal en el

dominio del tiempo se ha usado el "script" Script_Plot_Times_JL.m y para la representación en frecuencia el "script" Script_PSD_JL.m. La señal de entrada a usar será una onda cerebral beta de 25 Hz de frecuencia y 0.1mV de amplitud (figuras 3-8 y 3-9).

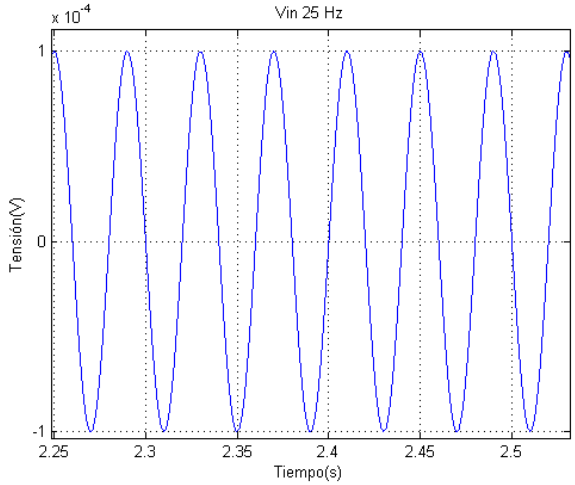


FIGURA 3-8. SEÑAL DE ENTRADA CHOPPER MATLAB DOMINIO TEMPORAL.

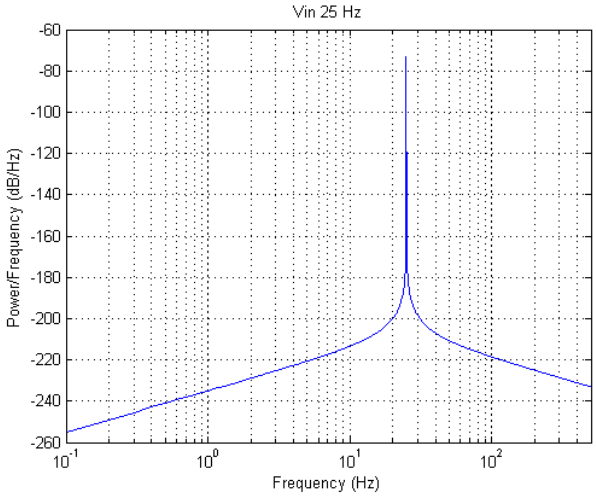


FIGURA 3-9. SEÑAL DE ENTRADA CHOPPER MATLAB DOMINIO FRECUENCIAL.

3.4.1.2 Bloque modulación MatLab-Simulink (I)

Para realizar la modulación Chopper se ha empleado dos bloques Pulse Generator (figura 3-10) cuyo periodo se define en el "script" principal y que será la inversa de la frecuencia de Chopper. Esta frecuencia se ha establecido, dado que no hay amplificador con ancho de banda finito, mediante el teorema de Nyquist y suponiendo que el potencial de acción tendrá frecuencia máxima de 5 kHz. Por lo tanto, la frecuencia de modulación empleada será 10kHz.

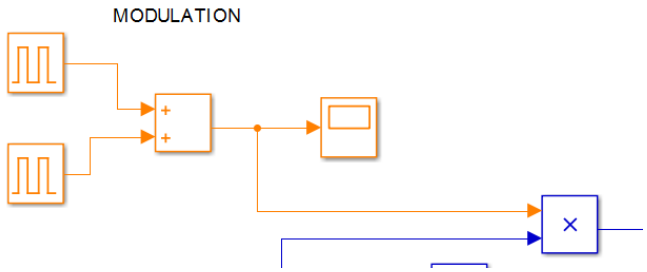


FIGURA 3-10. BLOQUE MODULACIÓN CHOPPER MATLAB.

Se puede apreciar en la figura 3-10 que para modular la señal lo que se acaba realizando es una multiplicación de ésta por los bloques de pulsos antes mencionados consiguiendo una señal temporal modulada y lo más importante: su desplazamiento en el dominio de la frecuencia a 10kHz, debido al "aliasing", y en menor parte a múltiplos impares de ésta (30kHz, 50kHz…) (figura 3-11).

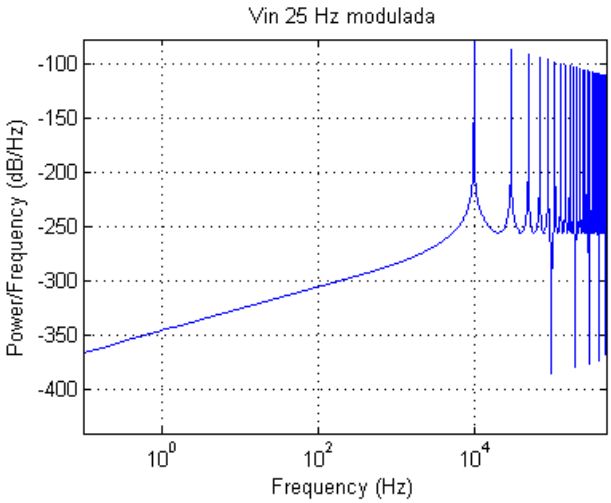


FIGURA 3-11. SEÑAL DE ENTRADA MODULADA CHOPPER MATLAB DOMINIO FRECUENCIAL.

3.4.1.3 Señal modulada+ruido MatLab-Simulink

En este apartado no se añadirán figuras sobre cómo se ha generado el ruido, ya que ha sido el mismo método que se usó en el apartado cuatro del capítulo segundo, es decir, mediante mapas discretos modelando un archivo de pares de puntos potencia-frecuencia de ruido. La única diferencia que existe es que, en este caso, la escala es multiplicada mediante un gain dentro de Simulink y no en el "script" de MatLab como se hacía anteriormente. El ruido resultante se la ha sumado mediante el uso del bloque Add a la señal de entrada modulada, obteniéndose la figura 3-12.

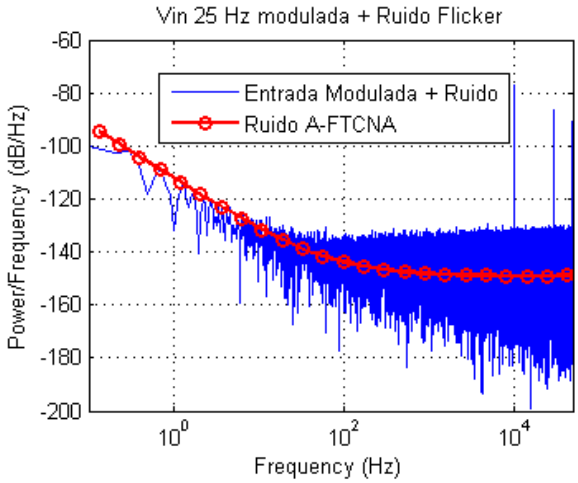


FIGURA 3-12. SEÑAL DE ENTRADA MODULADA + RUIDO CHOPPER MATLAB DOMINIO FRECUENCIAL.

3.4.1.4 Bloque modulación MatLab-Simulink (II)

Se utiliza el mismo bloque que en la figura 3-10 solo que ahora se multiplica por la señal modulada con el ruido añadido que simula la señal a la salida del amplificador. Se puede apreciar en la figura 3-14 como la señal de entrada vuelve a su frecuencia fundamental, para este caso 25 Hz, y como el ruido adopta la frecuencia de Chopper (10kHz) y armónicos impares posteriores. En la representación temporal, figura 3-13, se comprueba el efecto de estos armónicos de ruido en la señal demodulada.

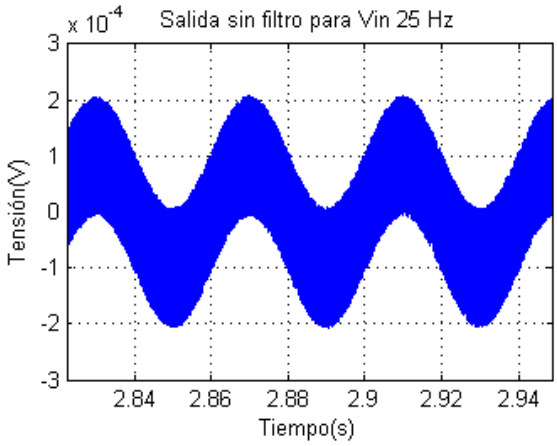


FIGURA 3-13. SEÑAL DEMODULADA SIN FILTRAR CHOPPER MATLAB DOMINIO TEMPORAL.

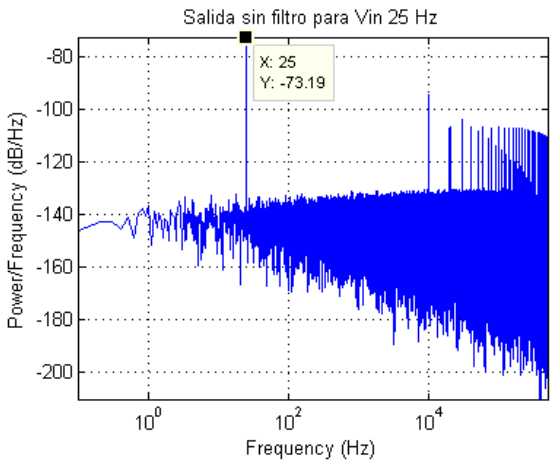


FIGURA 3-14. SEÑAL DEMODULADA SIN FILTRAR CHOPPER MATLAB DOMINIO FRECUENCIAL.

3.4.1.5 Bloque Filtro Paso Bajo MatLab-Simulink (II)

Al no ser el objetivo del trabajo el diseño del filtro, se realizará un filtro paso bajo de primer orden (3-3) que se pasará a tiempo discreto [23]. Se trabaja en tiempo discreto debido a que los mapas de generación de ruido son discretos y por tanto para que no haya errores en Simulink toda la simulación se hará de esta forma. La frecuencia de corte del filtro estará a 200 Hz si trabajamos con potenciales de campo locales y a 5kHz si es con potenciales de acción como se vio en el apartado 3.4.1.1 (donde también se explicó qué habría que hacer en lugar de realizar dos filtros).

$$H_{LP}(s) = \frac{w_c}{1 + w_c \cdot s} \tag{3-3}$$

El cálculo del polo en discreto [23] se realizará mediante la siguiente línea de código, y se implementará en Simulink mediante el bloque Transfer Function First Order.

```
s=tf('s'); %Hará falta para el filtro
…
…
filtro=(2*pi*200)/(s+2*pi*200);
…
…
%Filtro a discreto


filtrod=c2d(filtro,Tm);
pfiltro=pole(filtrod);
```

La señal de salida final será la representada en las figuras 3-15 y 3-16.

A primera vista, se puede apreciar que la modulación Chopper constituye una técnica eficaz para la eliminación del ruido Flicker. En la figura 3-17 se aprecia cuantitativamente cómo mejora la técnica de Chopper la señal, comparando la señal de entrada con el ruido añadido con la señal de salida del sistema. Sin embargo, estos análisis realizados han sido tan solo una primera aproximación y toma de contacto con el concepto de Chopper ya que en ningún momento se han tenido en cuenta las no idealidades (ancho de banda finito y saturación, principalmente) del amplificador, las cuales se tendrán en cuenta en el siguiente capítulo.

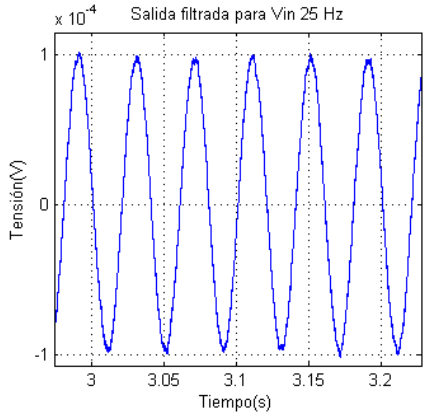


FIGURA 3-15. SEÑAL SALIDA FILTRADA CHOPPER MATLAB DOMINIO TEMPORAL.

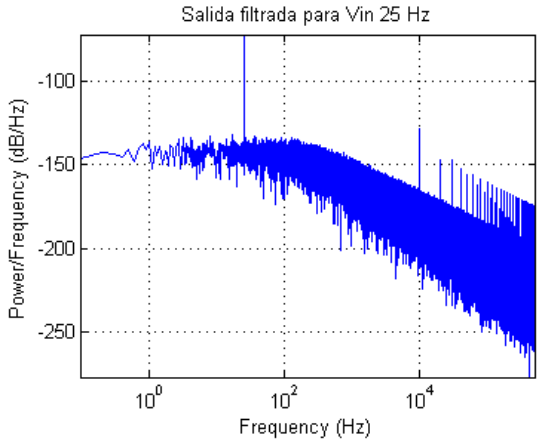


FIGURA 3-16. SEÑAL SALIDA FILTRADA CHOPPER MATLAB DOMINIO FRECUENCIAL.

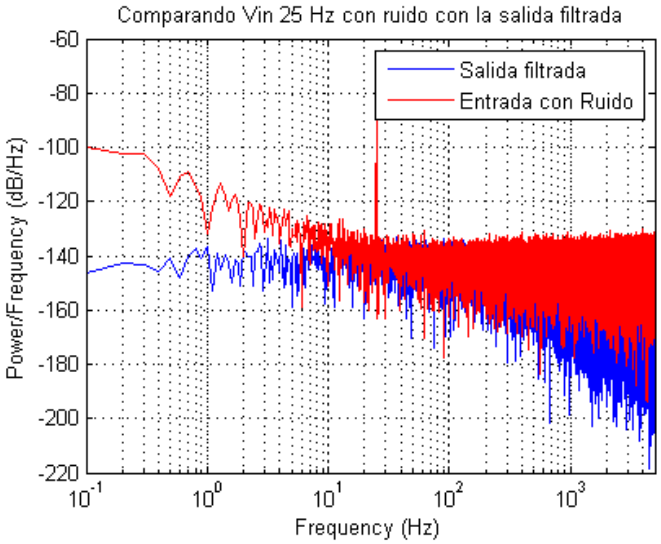


FIGURA 3-17. COMPARATIVA SEÑAL ENTRADA CON RUIDO Y SEÑAL SALIDA FILTRADA CHOPPER MATLAB DOMINIO FRECUENCIAL. REDUCCIÓN DE RUIDO DE UNOS 40DB/HZ PARA FRECUENCIAS MENORES A 1HZ.

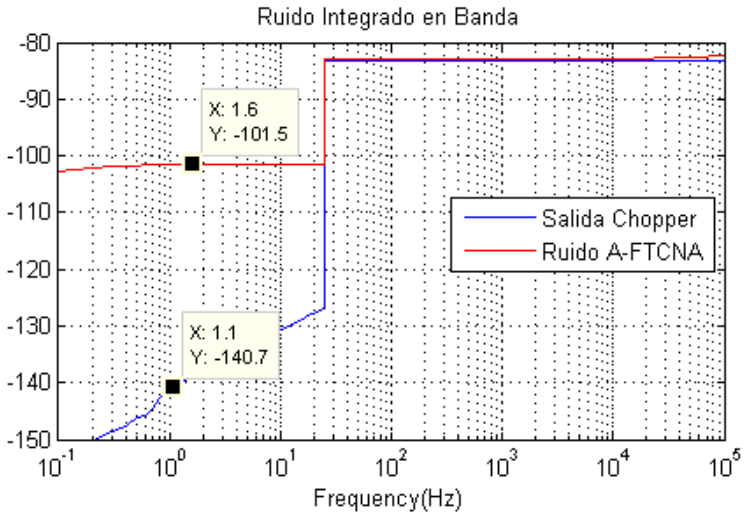


FIGURA 3-18. COMPARATIVA INTEGRACIÓN EN BANDA DE LA SEÑAL ENTRADA CON RUIDO Y SEÑAL SALIDA FILTRADA. PARA FRECUENCIAS MENORES A LA FRECUENCIA DE LA SEÑAL DE ENTRADA (25HZ) EL RUIDO SE REDUCE ENTORNO A 40DB.

Analizando estas dos últimas figuras, se puede apreciar cuantitativa y cualitativamente la eficacia de la técnica de Chopper en la eliminación de ruido a bajas frecuencias.

3.4.2 Implementación del amplificador Chopper en Cadence

Durante el transcurso de este apartado se van a realizar dos tipos de simulaciones: una de señal mixta empleando fuentes de tensión, el componente Port y bloques de Verilog-A de la librería ahdlLib [24] y otros que se diseñarán; y otra simulación totalmente analógica mediante llaves de la analogLib.

El objetivo de usar señal mixta es más educacional y destinado al aprendizaje que conceptual. Básicamente, se busca una familiarización con este tipo de diseños para en el futuro poder seguir haciendo análisis de circuitos de señal mixta, por no hablar de lo que supone el aprendizaje, aunque sea básico, de un nuevo lenguaje de diseño y de las herramientas que ofrece Cadence para su simulación. Además, así se asegura de que todo lo que se hará seguirá siendo ideal, ya que si se prescinde del amplificador real es para corroborar precisamente esto, y el introducir algún otro componente no ideal significaría que no serviría de nada lo anterior.

3.4.2.1 Implementación del amplificador Chopper en Cadence mediante señal mixta.

Para empezar este apartado, se va a explicar cómo se ha diseñado una de las componentes del circuito mediante Verilog-A, así de cómo y qué resultados se han obtenido de su simulación. De esta forma se mostrará de una manera clara el flujo de diseño en señal mixta empleado.

El componente a realizar será el filtro paso bajo. Este filtro se generará mediante la colocación de una resistencia en serie con un condensador (si se hiciera analógicamente se correría el riesgo de que la resistencia y/o el condensador introdujeran un efecto no deseado en el circuito). Primero se crea una vista de Verilog-A en Cadence, donde se escribirá el siguiente código:

```
// VerilogA for Ch_no_Amp, Low_Pass, veriloga

`include “constants.vams”
`include “discipline.h”

module Low_Pass(in, out, gnd) ;
  inout in,out,gnd;
```

```
 electrical in,out,gnd;

 parameter r=1k ;
 parameter c=1n ;

 analog begin
    // stage 1
    I(in,out) <+ V(in,out) / r;
    I(out,gnd) <+ ddt( V(out,gnd) * c );


 end
endmodule
```

Como se puede apreciar, lo primero a realizar es incluir las librerías necesarias. Posteriormente, se llamará al módulo que se desea crear y se le colocarán las entradas/salidas, las señales eléctricas (en este caso coinciden) y los parámetros que el usuario puede pasarle al código a través del esquemático. Con el comando analog begin iniciamos el flujo analógico que tendrá la señal definiendo la intensidad entre la entrada y la salida como V/R y la intensidad entre la salida y tierra como la derivada [24] de la tensión entre la salida y tierra por C.

Una vez el código acabado, se crea una vista symbol de él.

Ahora es necesario crear una celda diferente para realizar el testeo. Se crea una vista de esquemático (figura 3-19) y se coloca el circuito habitual para realizar un análisis AC. Terminado este esquemático se debe crear una vista config que se abrirá con Hierachy Editor (figura 3-19) donde se elegirá la jerarquía que deben seguir los análisis. Esto quiere decir que, si, por ejemplo, se tiene un símbolo que está formado a partir de un archivo verilog-a (como en el caso a tratar) se debe definir que se tenga en cuenta la vista de verilog-a antes que la del símbolo a la hora de realizar un análisis.

Se ha de decir que la figura 3-19 está editada para ver tanto esquemático como config en una misma imagen.

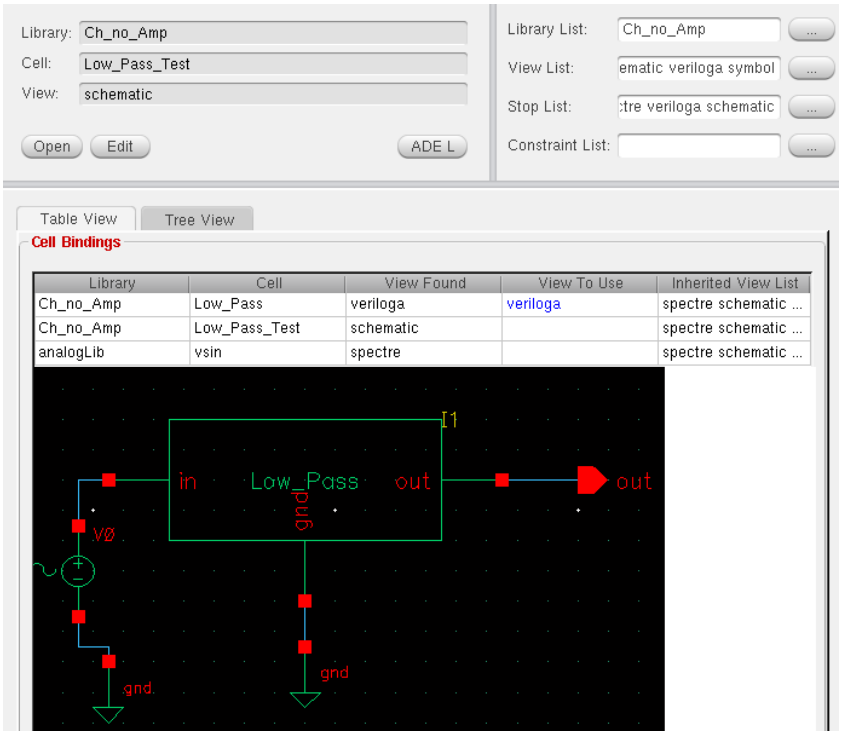


FIGURA 3-19. CONFIG Y ESQUEMÁTICO FILTRO PASO BAJO CADENCE. LA JERARQUÍA SERÁ EN EL ORDEN ESPECTRO, ESQUEMÁTICO, VERILOG-A Y SÍMBOLO.

Teniendo la celda de test finalizada, tan solo queda fijar los parámetros (*r*=1k, *c*=795.775n) para que la frecuencia de corte del filtro esté a 200 Hz. Hay que puntuar que no se han puesto unidades en los parámetros ya que no son parámetros eléctricos, sino numéricos. Por último, se realiza un análisis AC (figura 3-20) del filtro paso bajo con el que se puede concluir que el filtro diseñado es correcto ya que se obtiene la respuesta esperada.

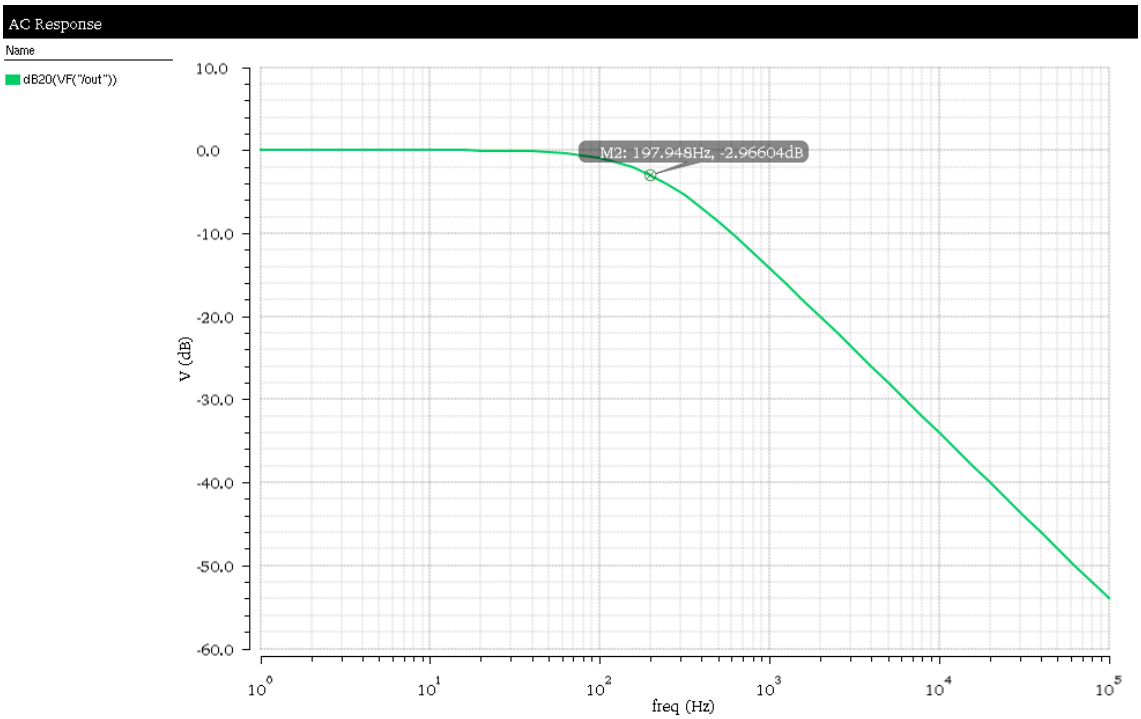


FIGURA 3-20. ANÁLISIS AC FILTRO PASO BAJO VERILOG-A CON FRECUENCIA DE CORTE DE 200HZ.

Partiendo desde este punto, ahora es más sencillo realizar el diseño del Chopper. Primero se realizará el esquemático del mismo (figura 3-21) donde se introducirán dos tipos de bloques: analógicos (analogLib) y digitales (ahdlLib y filtro creado para el trabajo). Los componentes serán los siguientes:

- Multiplier (ahdlLib): Bloque multiplicador cuya función en el circuito tratado será el de multiplicar señales para la modulación.
- Vpulse (analogLib): Con la señal de pulso entre 1 y -1 y un periodo de 1e-4 y 50% de duty cycle se realiza la modulación multiplicando estos pulsos por la señal a tratar.
- Port (analogLib): Para generar el ruido.
- Vsin (analogLib): Señal de entrada de 25Hz y 1mV.
- Adder (ahdLib): Bloque sumador que se usa para añadir el ruido a la señal principal.
- Low Pass: Filtro paso bajo diseñado para este trabajo y que se definió con anterioridad. Filtra las frecuencias indeseadas a la salida del circuito.

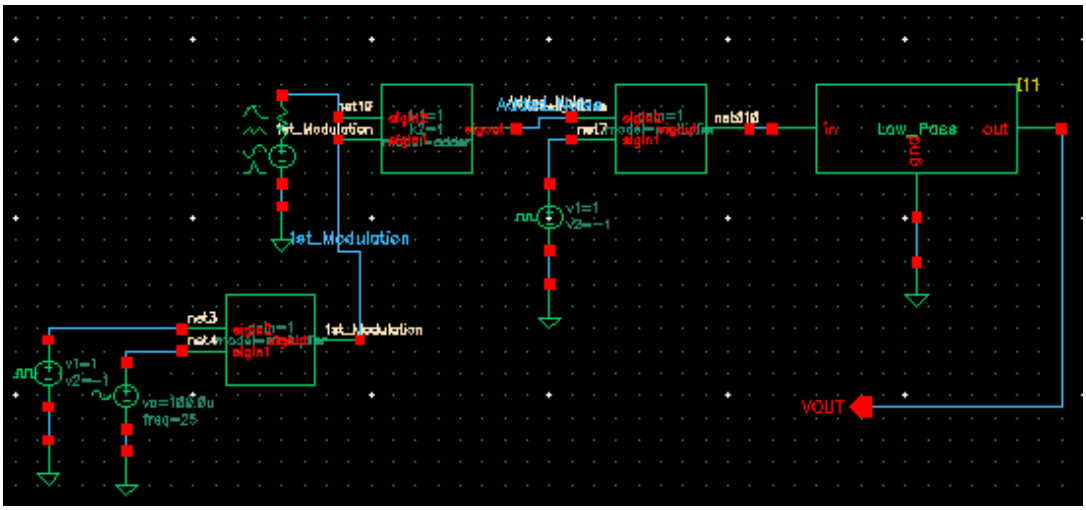


FIGURA 3-21. ESQUEMÁTICO CHOPPER SEÑAL MIXTA CADENCE.

Llegados a este punto, se debe realizar el análisis que se vio en el capítulo segundo, es decir, un análisis transitorio donde hay que señalar que se tenga en cuenta el ruido y posteriormente con Calculator utilizar la función PSD.

De la figura 3-22 a la figura 3-28 se representa todo el proceso de la técnica de Chopper que se describió anteriormente, por lo que no se va a repetir lo ya argumentado otra vez, ya que tan solo con las imágenes queda claro que el concepto se ha aplicado como es debido.

Es interesante mirar en cada figura la primera fila negra de texto que aparece, ya que en ella se puede observar el tipo de análisis PSD que se ha realizado para representarla, pues no todas las señales tienen las mismas frecuencias fundamentales lo que define en gran medida los parámetros del análisis a realizar.

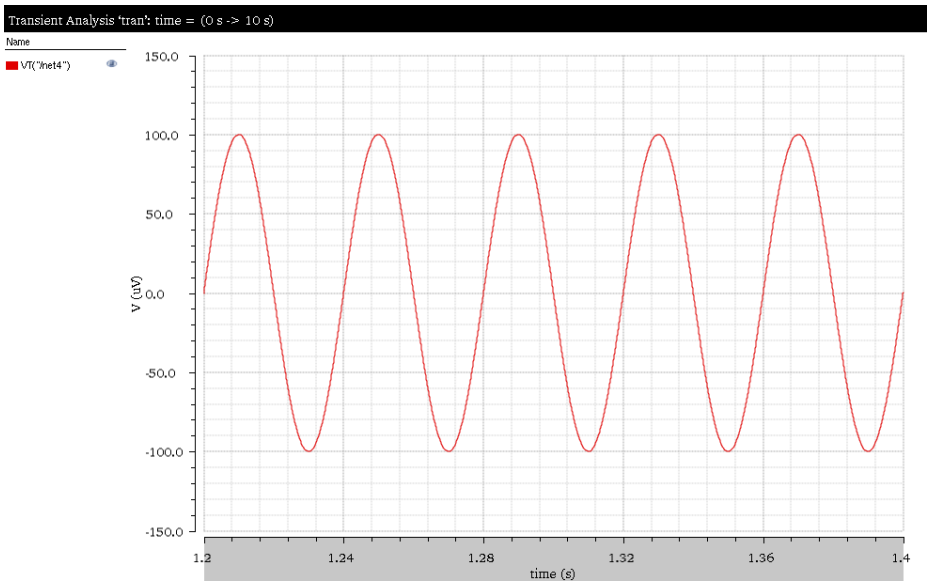


FIGURA 3-22. SEÑAL DE ENTRADA EN CADENCE DISEÑO MIXTO DOMINIO TEMPORAL.

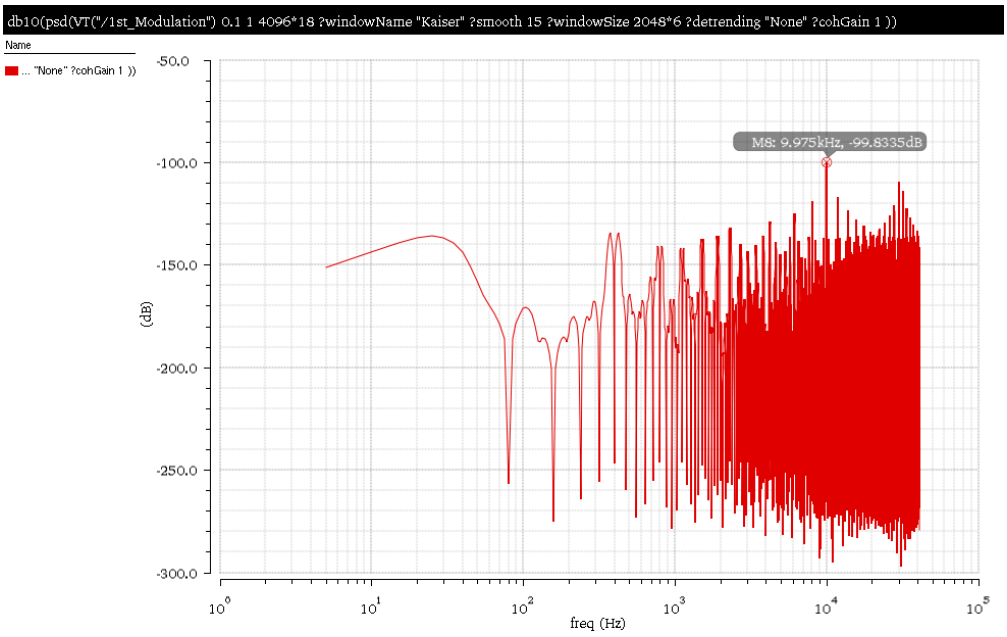


FIGURA 3-23. SEÑAL DE ENTRADA MODULADA 10KHZ EN CADENCE DISEÑO MIXTO DOMINIO FRECUENCIAL. DEBIDO AL "ALIASING" LA SEÑAL DE ENTRADA PASA A LOS 10KHZ.

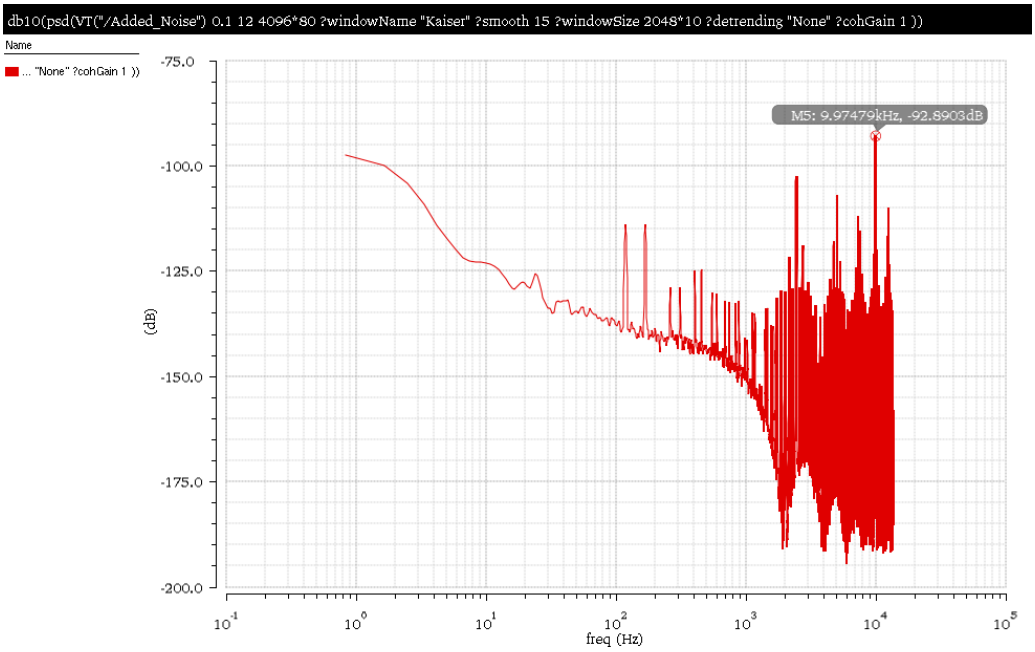


FIGURA 3-24. SEÑAL DE ENTRADA MODULADA 10KHZ + RUIDO EN CADENCE DISEÑO MIXTO DOMINIO FRECUENCIAL.

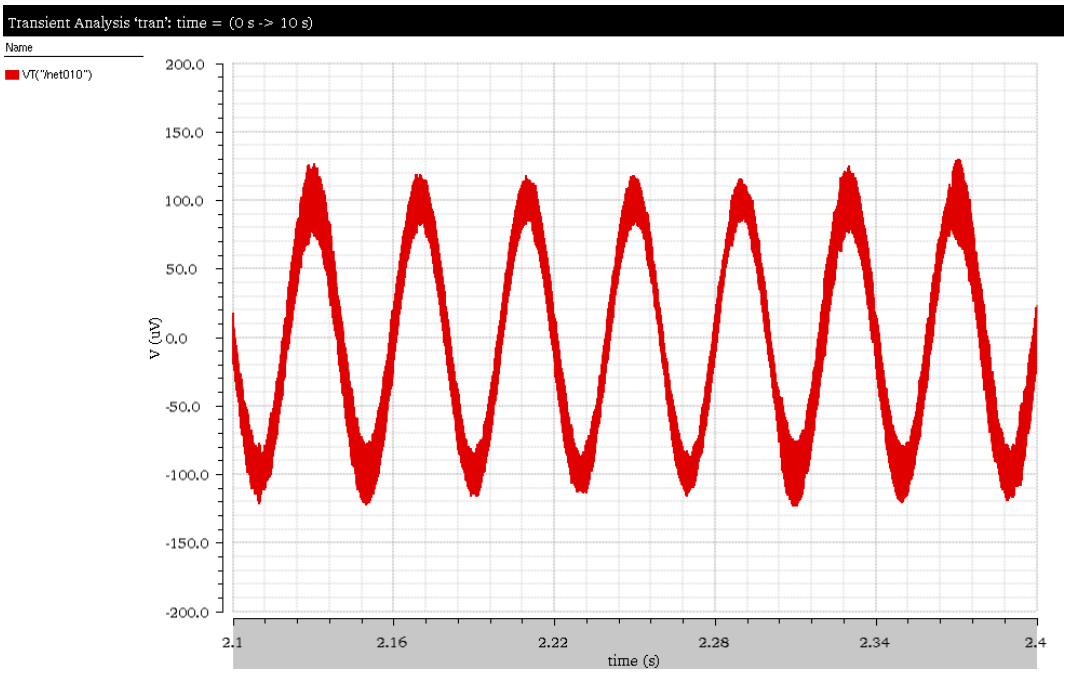


FIGURA 3-25. SEÑAL DE SALIDA SIN FILTRAR EN CADENCE DISEÑO MIXTO DOMINIO TEMPORAL. SEÑAL VISIBLEMENTE AFECTADA POR EL RUIDO.

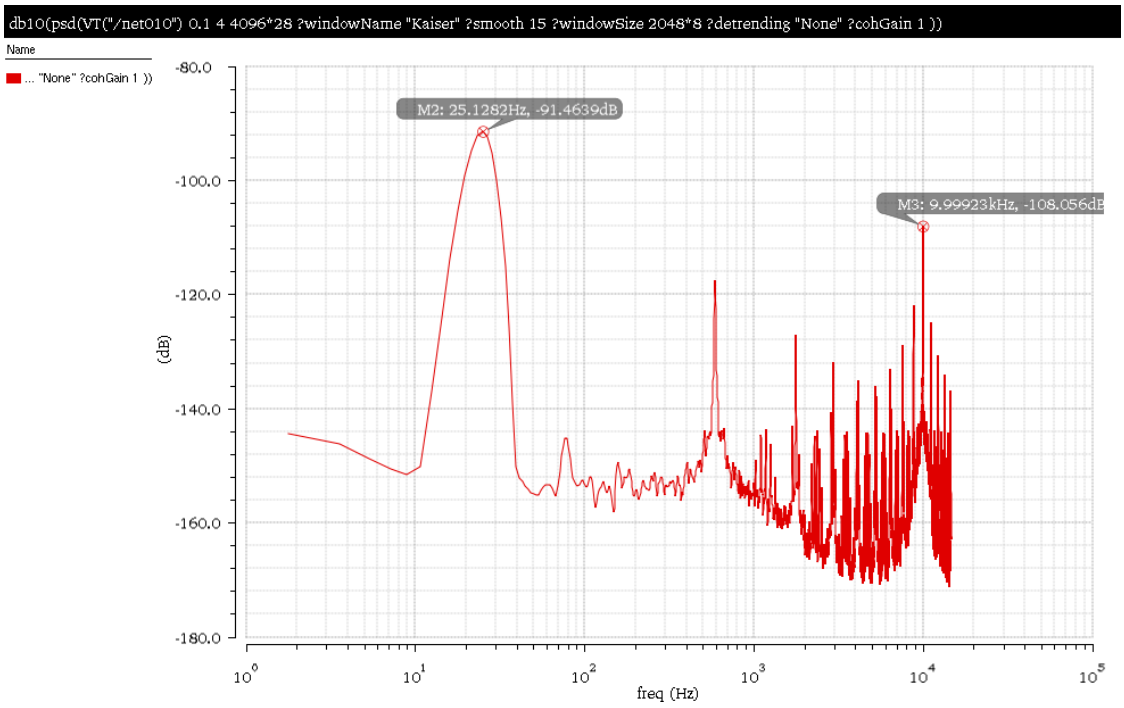


FIGURA 3-26. SEÑAL DE SALIDA SIN FILTRAR CADENCE DISEÑO MIXTO DOMINIO FRECUENCIAL.

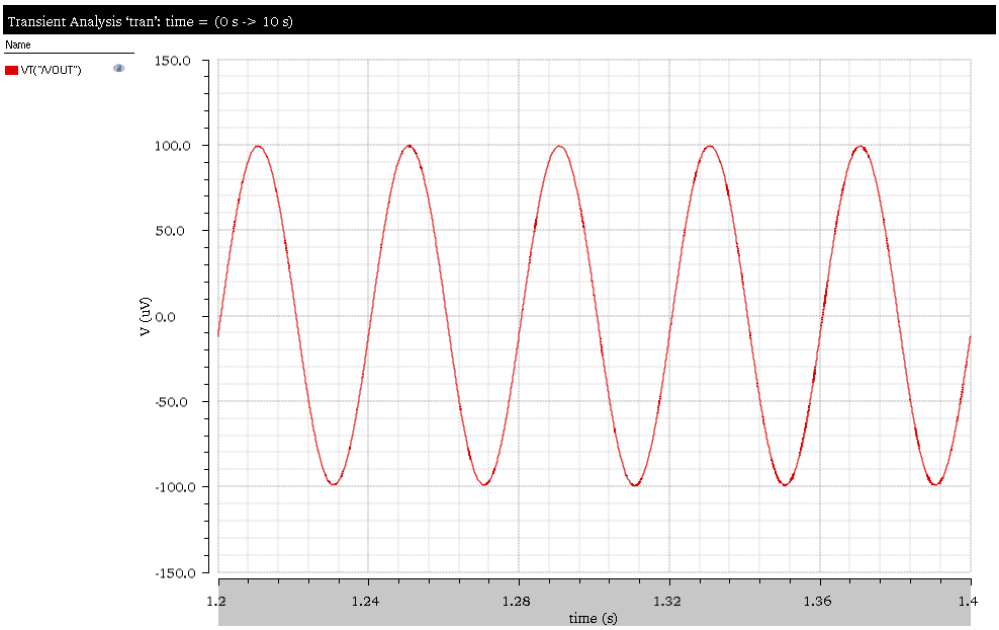


FIGURA 3-27. SEÑAL DE SALIDA FILTRADA CADENCE DISEÑO MIXTO DOMINIO TEMPORAL.

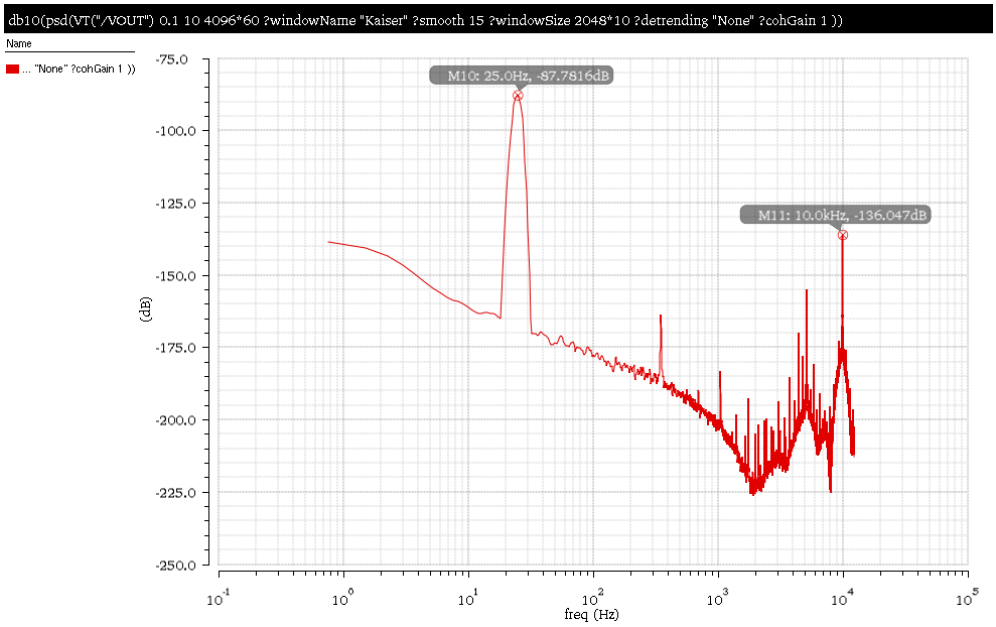


FIGURA 3-28. SEÑAL DE SALIDA FILTRADA CADENCE DISEÑO MIXTO DOMINIO FRECUENCIAL.

Observando detalladamente la figura 3-24 y la figura 3-28, se podría determinar entonces que, para una frecuencia aproximada de 1 Hz, se han reducido alrededor de 40dB/Hz el ruido de la señal, por lo que los resultados son bastante positivos.

Si se comparan estas gráficas con las correspondientes cuando se usó MatLab-Simulink, se puede apreciar que cualitativamente son iguales, y que por tanto el concepto ha quedado, aparte de claro, aplicado correctamente tanto en un entorno como en otro. Cuantitativamente se puede ver que, aunque no considerablemente, sí difieren los resultados. Esto es debido principalmente a que la forma de pasar la señal al dominio de la frecuencia, es decir la transformada de Fourier, no se realiza de igual manera (distintos números de puntos, tipo de ventana…). Además, las simulaciones no tienen el mismo tiempo

de muestreo y aunque se esté trabajando a nivel alto, los bloques usados no son exactamente iguales lo que puede inducir también discrepancias en los modelos, por no hablar del tipo de generación de ruido que también es diferente. No obstante, se reitera que se han conseguido satisfactoriamente los objetivos impuestos en este capítulo que era el de conceptualizar el amplificador Chopper ideal e implementarlo correctamente en varios entornos de programación/diseño.

Se habló al comienzo del apartado de que se realizó el mismo esquema de Chopper, pero con un modelo en Cadence analógico con llaves (figura 3-29). No obstante, introducir todos los resultados obtenidos supondría repetir las gráficas representadas con anterioridad, con una discrepancia mínima entre ellas, por lo que se va a aceptar que las figuras de la 3-22 a la 3-28 son válidas también para este circuito. De esta forma no se satura el trabajo con gráficas iguales.

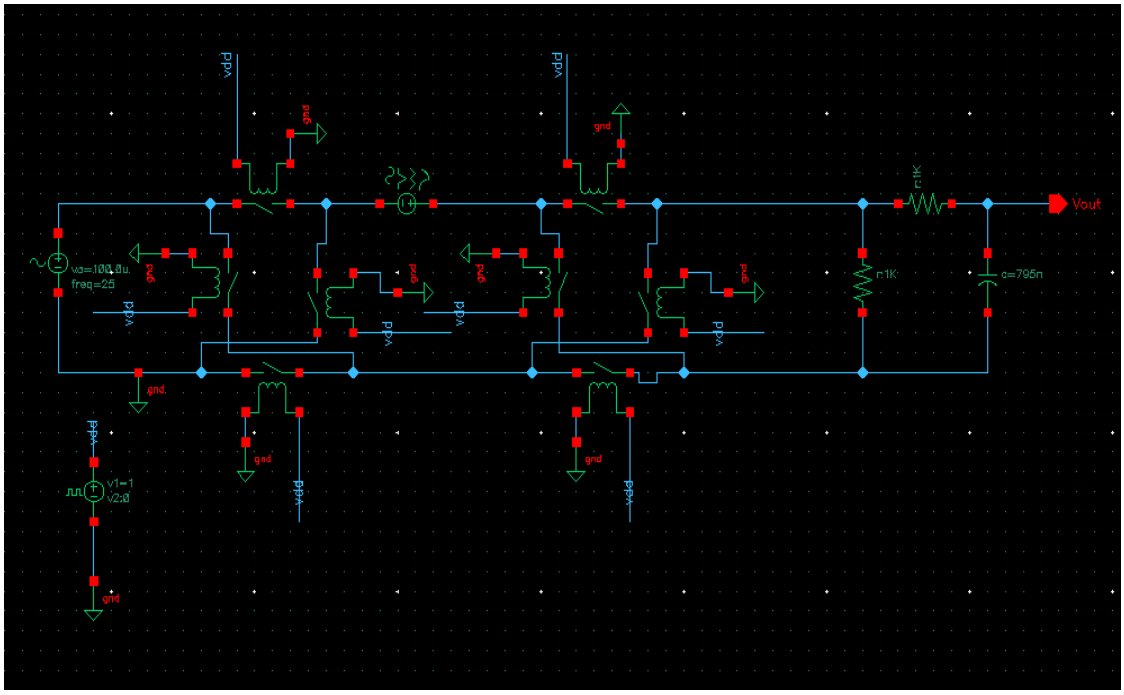


FIGURA 3-29. ESQUEMÁTICO CHOPPER "SWITCHES" CADENCE.

Comentar que se ha utilizado la fuente de pulso para abrir y cerrar las llaves. El esquema que se ha empleado de los interruptores es el básico que se mostró en la figura 2-2. También decir que en este caso se ha utilizado una fuente dc y no un Port para generar el ruido (la metodología es la misma) para evitar que la resistencia en serie de este componente pudiera alterar los resultados del análisis ya que se encuentra en serie también con el circuito analógico. El filtro se ha implementado de primer orden RC con frecuencia de corte de 200 Hz.

Este circuito servirá de primera toma de contacto para en proyectos futuros implementar el Chopper mediante transistores MOS, ya que el esquema será el mismo, pero sustituyendo las llaves por transistores, por lo que es de gran utilidad.

4 ARQUITECTURA BÁSICA CHOPPER APLICADA AL ACTIVE-FEEDBACK TIME CONSTANT NEURAL ENHANCED AMPLIFIER

En este capítulo se van a abordar los conceptos ampliamente trabajados en el capítulo anterior con la diferencia de que serán aplicados esta vez a un amplificador concreto, el cual presentará una serie de no idealidades, principalmente un ancho de banda finito, que como se verá en el transcurso del capítulo, supondrán restricciones a la hora de elegir los parámetros de modulación y del filtro.

En primer lugar, se comenzará el capítulo presentando el amplificador a trabajar, que ya se introdujo brevemente durante el capítulo primero, y modelándolo para posteriormente implementarlo en los entornos de trabajo que se tienen del capítulo anterior para, finalmente, analizar los resultados.

4.1 Presentación y modelado del Active-feedback time constant neural enhanced amplifier

El *active-feedback time constant neural enhanced amplifier* se trata de un amplificador actualmente aún en fase de diseño, cuyo propósito es exclusivo para Neural Recording y que tiene como objetivo amplificar todos los tipos de señales cerebrales, tanto potenciales de campo locales como potenciales de acción, mediante una arquitectura que asegure un compromiso entre bajo ruido, alta ganancia y bajo consumo [9] (ya se vio en el capítulo primero que éstas eran las máximas de los amplificadores para Neural Recording).

Antes de empezar a comentar el circuito en cuestión, se debe hacer mención de que se ha elegido este amplificador en concreto debido a que al tratarse de un amplificador ahora mismo en fase de diseño y de estar en contacto directo con el diseñador, gracias al Instituto de Microelectrónica de Sevilla, cualquier avance en la eliminación de ruido en baja frecuencia puede servir de gran ayuda. Incluso, puede ser el objeto de una investigación en paralelo tras la finalización de este trabajo si los resultados a alto nivel son satisfactorios.

El esquemático single-ended del amplificador se mostró ya en la figura 1-5 y se deben recalcar varios puntos que ya se introdujeron durante el capítulo primero:

- La ganancia del circuito viene dada por el cociente entre C_I y C_F.
- Se puede modelar como un filtro paso banda del que ya se definirán paramétricamente sus polos de alta y baja frecuencia.
- El polo de alta frecuencia viene impuesto principalmente por la realimentación que se realiza con el OTA3. Éste es sintonizable mediante una fuente de corriente.
- Los polos de baja frecuencia vienen impuestos principalmente por la realimentación C_B y las transconductancias de los OTA.

Una vez vista por encima una pequeña caracterización del circuito, se van a definir paramétricamente estos conceptos expuestos mediante las ecuaciones (4-5)(4-6)(4-7)[9]. Para facilitar el análisis, se tomarán las siguientes expresiones [9]:

$$\beta = -C_F \cdot gm1 + C_I \cdot gm2 - C_F \cdot gm2 - C_{p1} \cdot gm2 \tag{4-1}$$

$$\alpha = C_I \cdot C_F \cdot C_B + C_I \cdot C_B \cdot C_{p2} - C_I \cdot C_B \cdot C_{p3} - C_I \cdot C_{p2} \cdot C_{p3} + C_I \cdot C_B \cdot C_L + C_I \cdot C_{p2} \cdot C_L \tag{4-2}$$

$$A_2 = gm2 / gds2 \tag{4-3}$$

$$\Delta_{FB} = C_B^2 \cdot \beta^2 - 4 \cdot \alpha \cdot C_F \cdot gm1 \cdot gm2 \tag{4-4}$$

Siendo *gm1, gm2, gm3*, C_{p1}, C_{p2} y C_{p3} las transcodunctancias y capacidades parásitas correspondientes a cada OTA. Con estas simplificaciones, ya se pueden extraer paramétricamente alguna de las características más importantes del amplificador:

$$Gain_{MIDBAND} = \frac{C_I}{C_F} \tag{4-5}$$

$$HP_{POLE} = \frac{gm3}{C_F \cdot A_2} \tag{4-6}$$

$$LP_{POLES(2)} = \frac{-C_B \cdot \beta \pm \sqrt{\Delta_{FB}}}{2 \cdot \alpha} \tag{4-7}$$

Cuando se sustituye en las ecuaciones anteriores los datos con los parámetros de diseño del circuito [9] se obtienen los resultados que se muestran en la siguiente tabla (Tabla 4-1). Además, en esta tabla también se recogen otra serie de parámetros interesantes por tratarse de un amplificador destinado al Neural Recording.

Se puede apreciar, por las frecuencias que trata, lo ya mencionado al principio del capítulo: es un amplificador destinado a todos los tipos de ondas cerebrales, incluyendo los potenciales de acción (aunque éstos si están a su frecuencia máxima, es decir, a 7kHz, la ganancia será atenuada hasta los 45dB).

TABLA 4-1. RESULTADOS DE SIMULACIÓN DEL ACTIVE-FEDDBACK TIME CONSTANT ENHANCED NEURAL AMPLIFIER.

	Active-feedback time constant neural enhanced amplifier
Ganancia (dB)	**48**
Consumo (nW)	**454**
Frecuencia Paso Alto (Hz)	**0.129**
Frecuencia Paso Bajo (Hz)	**7k**
Ruido Integrado a la Entrada (µVrms)	**5.76** **(1-7k)Hz**
THD (%)	**0.5@1mVpp 1kHz**

Presentado y caracterizado el amplificador, el siguiente paso será modelarlo para obtener una función de transferencia característica que sea de fácil implementación para realizar simulaciones con ella. Para llegar a esta función característica del circuito, lo primero a realizar es el modelo en pequeña señal del mismo (figura 4-1) para así tener en cuenta todos los parámetros antes descritos (como capacidades parásitas y transconductancias) y que la función sea lo más aproximada al circuito posible. Llegados a este punto, se debe hacer un inciso conceptual muy importante.

Cuando se trabaja con un modelo de pequeña señal, los valores de los parámetros del mismo vienen definidos por el punto de polarización en el que se encuentra el sistema. En el caso que se trata, se va a aplicar una técnica de Chopper, lo que quiere decir que la señal de entrada del amplificador va a estar modulada y, lo más importante, oscilando entre Vin y –Vin. Esto va a propiciar que el estado de polarización del dispositivo pueda cambiar. En concreto, para el trabajo tratado, es posible que esto no ocurra debido a la poca magnitud de las señales de entrada (si no se tratan de potenciales de acción, no van más allá de 1mV). Sin embargo, es un punto posible, que, como diseñadores, se debe abordar y comprobar qué ocurre en la práctica (o al menos en un número suficiente de simulaciones). No obstante, esto conllevaría un trabajo bastante más largo y complejo, que se escapa de los límites de este trabajo. Sin embargo, hay que dejar claras este tipo de situaciones para cuando se siga profundizando en el tema no se cometan errores.

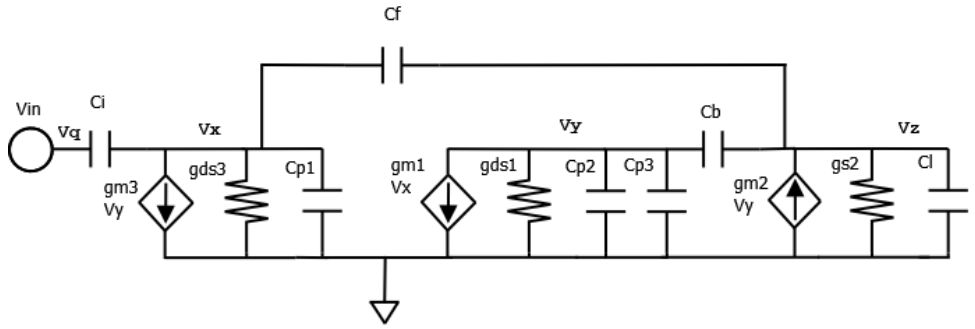


FIGURA 4-1. MODELO PEQUEÑA SEÑAL ACTIVE-FEEDBACK TIME CONSTANT NEURAL ENHANCED AMPLIFIER.

Con el modelo definido, se realizará un análisis nodal para sacar la función de transferencia del sistema. Para conseguir esta función de transferencia se ha utilizado el software de Wolfram Alpha llamado Mathematica. El archivo empleado basado en el procedimiento que se explicará a continuación es maxima_calc.nb. Esto se ha hecho así para aprovechar algunos "scripts" antiguos el diseñador del circuito ha proporcionado para facilitar la forma de operar.

La matriz nodal (para Vq, Vx, Vy y Vz respectivamente) que se obtiene del análisis del circuito es la siguiente:

$$\begin{matrix} C_I \cdot s & -C_I \cdot s & 0 & 0 \\ -C_I \cdot s & -gds3 + C_I \cdot s - C_F \cdot s + C_{p1} \cdot s & -gm3 & C_F \cdot s \\ 0 & -gm1 & gds1 - C_B \cdot s + C_{p2} \cdot s - C_{p3} \cdot s & C_B \cdot s \\ 0 & C_F \cdot s & gm2 + C_B \cdot s & gds2 - C_F \cdot s - C_B \cdot s + C_L \cdot s \end{matrix} \quad (4\text{-}8)$$

Para sacar la función de transferencia que interesa, se añade una columna final y una fila final a la matriz como la siguiente: (1,0,0,0,0) y la matriz pasa a ser de 5x5. Ahora, se crea un vector columna w = (0, 0, 0, 0, 1) que será la salida del sistema. De esta forma, al aplicar el comando LinearSolve de Mathematica para la matriz nodal y este vector, se resolverá la ecuación m*x=w, dando como salida de éste la matriz x (que debe ser 5x1). Cada fila de esta matriz representará la función de transferencia de cada nodo para obtener la salida indicada. En el caso que se trabaja, interesa la componente número cuatro de esta matriz, que será la función de transferencia deseada del sistema. El código usado de Mathematica es el siguiente:

```
ClearAll["Global`*"]
m = {{s*c1, -s*c1, 0, 0, 1}, {-s*c1, s*c1 + s*cx - gds3 - s*c2, -gm3,
    s*c2, 0}, {0, gm1, gds1 + s*cp2 - s*cb - s*cp3, s*cb, 0}, {0,
    s*c2, s*cb + gm2, -s*c2 + gds2 + s*cy - s*cb, 0}, {1, 0, 0, 0,
    0}}; (*gm1 debería ser menos, pero ya lo cambiaremos en los resultados
poniéndolo negativo*)
```

```
MatrixForm[m]

w = {{0}, {0}, {0}, {0}, {e}}

MatrixForm[LinearSolve[m, w]]
```

La función de transferencia obtenida es demasiado larga como para añadirla a este documento, sin embargo, posteriormente (4-9) (4-10) se pondrá la versión simplificada obtenida en MatLab cuando se sustituye el valor de cada parámetro.

Llegados a este punto, se tiene la función de transferencia característica del sistema y su modelo en pequeña señal, por lo que el siguiente apartado se basará en su implementación.

Antes de esto, es importante decir que los parámetros que se han proporcionado para el trabajo del amplificador son más recientes que los mostrados en [9]. Por lo tanto, habrá características, como por ejemplo el ancho de banda, que cambiarán entre lo expuesto en [9] y en lo que se obtendrá. Los parámetros utilizados serán los siguientes:

TABLA 4-2. PARÁMETROS PARA IMPLEMENTACIÓN ACTIVE-FEEDBACK TIME CONSTANT NEURAL ENHANCED AMPLIFIER.

Capacidades (F)	Transconductancias (Mhos)	Pseudoresistencias (Ω)
$C_I = 40p$	$gm1 = 11\mu$	$gds1 = 14n$
$C_F = 155f$	$gm2 = 670n$	$gds2 = 5n$
$C_{p1} = 97f$	$gm3 = 16p$	$gds3 = 8p$
$C_{p2} = 300p$		
$C_{p3} \simeq 0$		
$C_B = 0.68p$		

4.2 Implementación Active-feedback time constant neural enhanced amplifieren MatLab-Simulink

En este apartado, el esquema de implementación será básico, introduciendo primero los parámetros del amplificador y posteriormente la función de transferencia del mismo. Como se quiere trabajar en tiempo discreto (ya que los mapas de generación de ruido Flicker son discretos) se usará el comando c2d para pasar a discreto y tfdata (sys,Ts) para tener el numerador y denominador de esta función y poder implementarla en Simulink.

El código empleado se encuentra en el "script" Model_Elec.m. y se va a añadir a continuación al documento, pero con la línea de código que representa la función de transferencia paramétrica cortada (ya que como se dijo anteriormente, su longitud es tal que haría el documento mucho más engorroso):

```
Tm=1e-6;
Fm=1/Tm;


c1=40e-12;
c2=155e-15;
```

```
gds1=14e-9;
gds2=5e-9;
gds3=8e-12;
gm1=-11e-6; %Negativa porque en la función de tf no está considerado el -gm1 que se
muestra en la matriz
gm2=670e-9;
gm3=16e-12;
cp2=300e-15;
cy=4.6e-12;
cx=9.7e-16;
cb=0.68e-12;
cp3=0;

s=tf('s');
M_JLV=(s*(….. (línea cortada)
M_JLVd=c2d(M_JLV,Tm);
[num dem]=tfdata(M_JLVd);
set(cstprefs.tbxprefs,'FrequencyUnits','Hz'); %Hz unidades de Bode
Bode(M_JLVd);
grid;
```

En (4-9) y (4-10) se van a representar las funciones de transferencias (continua y discreta respectivamente), simplificadas con valores numéricos, obtenidas del sistema.

$$G(s)=\frac{-2.356\cdot10^{-36}s^3+2.993\cdot10^{-28}s^2-2.9848\cdot10^{-22}s}{7.542\cdot10^{-35}s^3+1.729\cdot10^{-29}s^2+1.146\cdot10^{-24}s+8.806\cdot10^{-25}} \tag{4-9}$$

$$Gd(z)=\frac{-0.03124z^3+5.442z^2-7.301z-1.891}{z^3-2.782z^2+2.577z-0.7952} \tag{4-10}$$

Como se aprecia en la última línea de código, se realiza una representación del Bode (figura 4-2) de la función de transferencia calculada (4-10) para comprobar que las especificaciones se aproximan a las mostradas en la Tabla 4-1, ya que esto es equivalente a realizarle un análisis AC al sistema real.

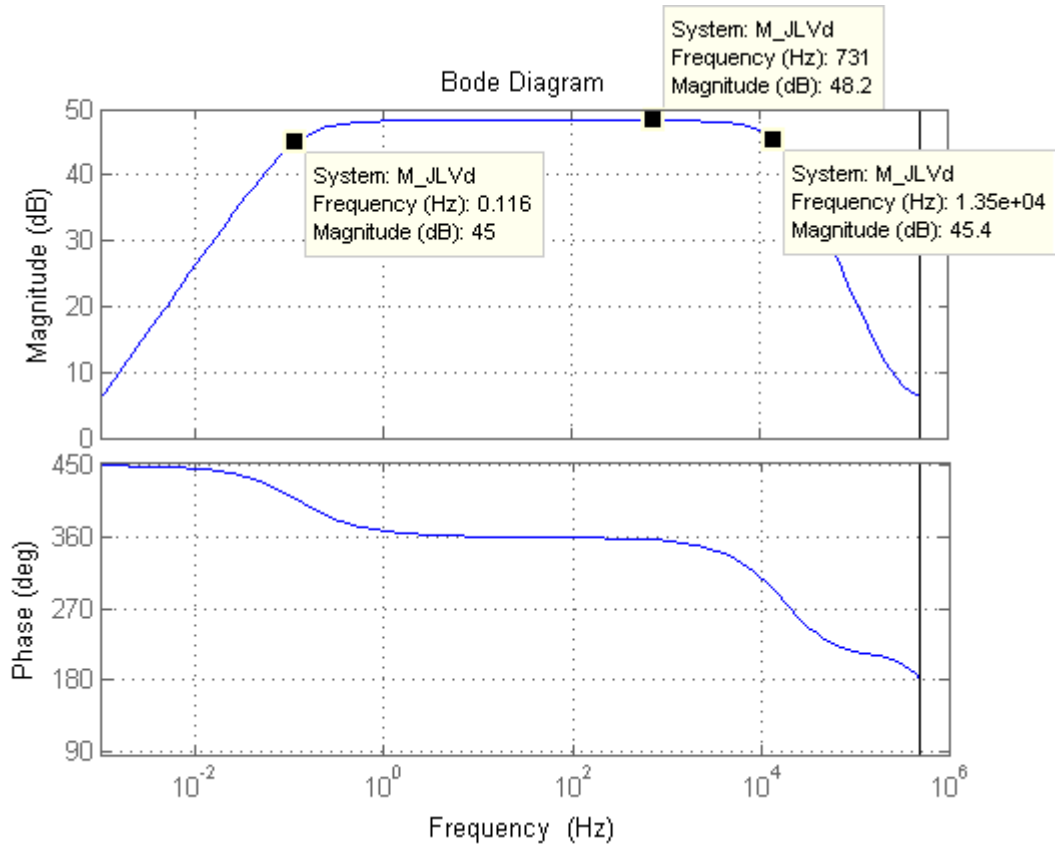


FIGURA 4-2. DIAGRAMA DE BODE FUNCIÓN DE TRANSFERENCIA DISCRETA MATLAB.

Se aprecia que la función de transferencia es correcta y que representa una muy buena aproximación para el circuito del amplificador. No obstante, y debido a que los parámetros usados difieren de los mostrados en [9], se tienen las siguientes diferencias:

- Ganancia en la banda media de 48.2 dB en lugar de 48 dB.
- Frecuencia de corte del paso alto en 0.116 Hz en lugar de 0.129 Hz.
- Frecuencia de corte del paso bajo en 13 kHz en lugar de 7 kHz.
- La pendiente del paso bajo no llega a -40dB cuando ha pasado una década. Esto puede ser debido a la presencia de las capacidades parasitas y al cambio de parámetros antes mencionado (no tenemos dos polos exactamente iguales para el paso bajo).

Se observa también que la fase es la correcta, ya que, aunque empiece en 450º, éstos representan 360+90 y se sabe que 360º=0º.

No obstante, se va a usar el archivo TF_JLV.slx para implementar Gd(z) en Simulink para comprobar que los resultados son correctos, no obstante, aunque el Bode sea el esperado, se deben hacer todas las comprobaciones pertinentes antes de proseguir con el trabajo para no ir acumulando errores.

Primero se introducirá una señal en el ancho de banda medio (1 kHz, figura 4-3), después otra a altas frecuencias (20 kHz, figura 4-4) y finalmente una a frecuencias bajas (0.1 Hz, figura 4-5).

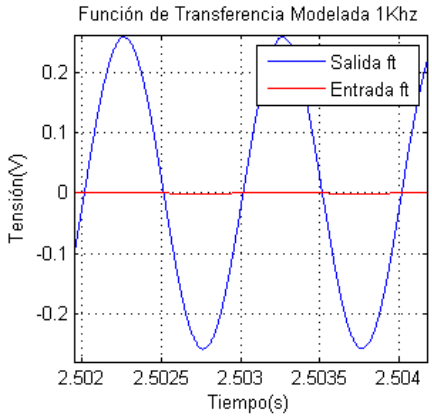


FIGURA 4-3. GD(Z) AMPLIFICADOR CON ENTRADA 1KHZ.

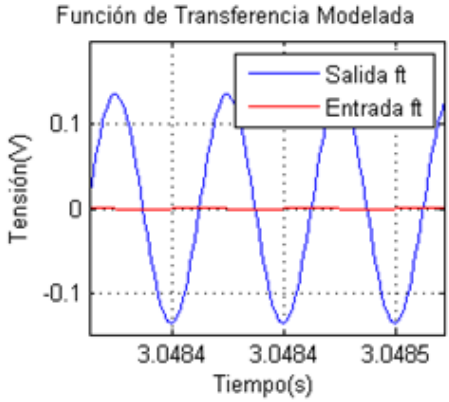


FIGURA 4-4. GD(Z) AMPLIFICADOR CON ENTRADA 20KHZ.

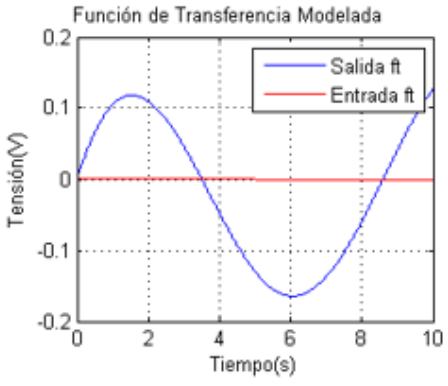


FIGURA 4-5. GD(Z) AMPLIFICADOR CON ENTRADA 0.1HZ.

En las figuras de la 4-4 a la 4-6 puede verse como la función de transferencia discreta se comporta tal y cómo debería hacerlo el amplificador con el que se trabaja, atenuando la ganancia para frecuencias por encima de 10 kHz y por debajo de 0.1 Hz.

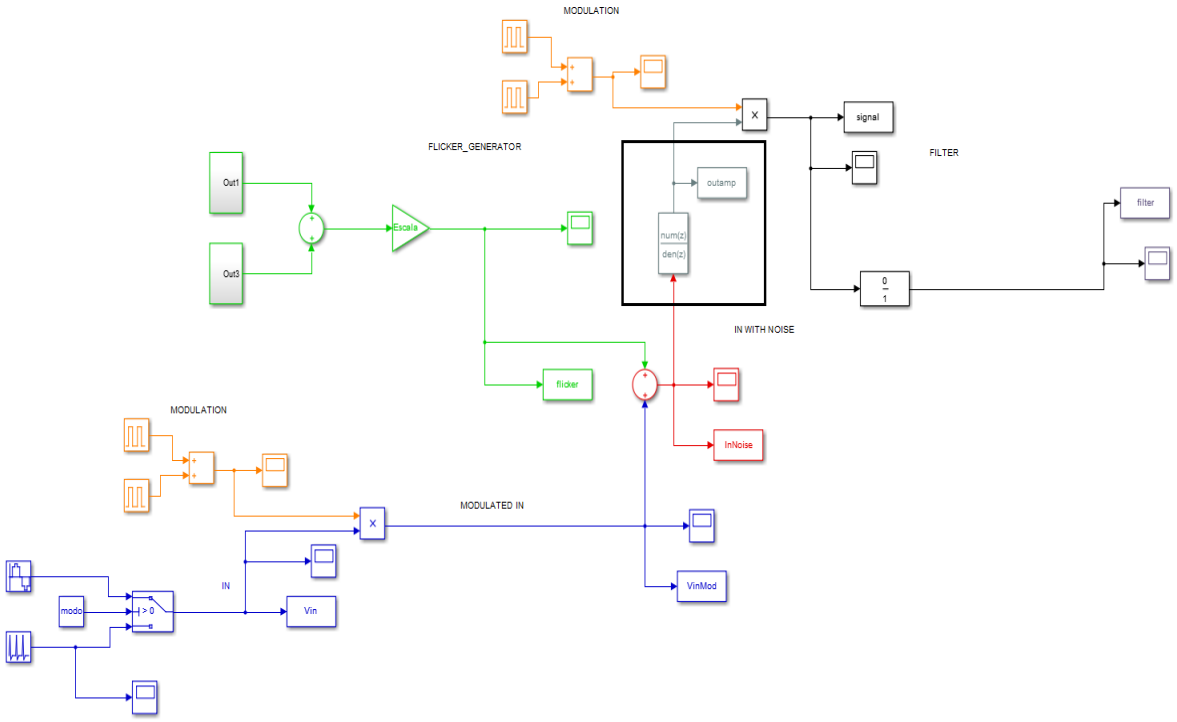


FIGURA 4-6. BLOQUE SIMULINK CHOPPER CON MODELO AMPLIFICADOR.

Una vez bien definida nuestra función de transferencia discreta, lo que se debe hacer es implementarla (figura 4-6) en el circuito Chopper del capítulo tercero. Los "scripts" utilizados van a ser los mismos que en el capítulo anterior con algunas discrepancias en la representación, pero nada remarcable como para ser expuesto. El archivo de Simulink empleado es Chopper_JL_Mod.slx.

Antes de hacer una simulación es primordial saber el por qué se realiza esa simulación. En este caso, introducir la función de transferencia real del amplificador no es simplemente para comprobar que amplifica la señal, sino que realmente se busca ver qué compromiso existe para el amplificador tratado entre dos parámetros que se vieron en el capítulo anterior: ganancia/ruido a la salida.

Estos dos parámetros van a estar definidos por la frecuencia de corte del amplificador y la frecuencia de Chopper principalmente. Si se aumenta la frecuencia de Chopper, el ruido quedará más separado frecuencialmente de la señal de entrada, por lo tanto, con un filtro menos selectivo se conseguirá menos ruido a la salida del circuito. Sin embargo, esto supondrá desplazar la señal de entrada a la entrada del amplificador a frecuencias cercanas o superiores a la frecuencia de corte del mismo, lo que supondrá una atenuación en la ganancia.

Por el contrario, elegir una frecuencia de Chopper menor supondrá tener el ruido más cercano a la señal de entrada, lo que se corresponde a más ruido a la salida o a la realización de un filtro más selectivo (que, o se le colocan bastantes pares de polos con un diseño "fino", o acabará reduciendo el ruido pero también atenuando considerablemente la señal). No obstante, en el paso por el amplificador no se atenuará la ganancia a la salida de éste.

4.2.1 Simulación Chopper con amplificador MatLab de LFP

Para poder realizar un análisis equitativo y valorar los resultados como es debido, se realizarán tres simulaciones para tres frecuencias de Chopper diferentes: una bastante inferior a la frecuencia de corte (1 kHz), otra cercana a la frecuencia de corte (11 kHz) y por último una superior a la frecuencia de corte (100kHz). La señal de entrada será una onda cerebral gamma (0.1Mv, 40Hz, -73.1dB/Hz en armónico

fundamental.). El filtro paso bajo será el mismo para todos (ganancia unitaria y frecuencia de corte de 200 Hz) para que no influya en los resultados. Para no saturar de gráficas el documento, se representarán las consideradas más características: salida del amplificador en el dominio de la frecuencia (figura 4-7), para comparar el valor de los armónicos donde se encuentra la señal de entrada modulada; salida del filtro temporal (figura 4-8) y frecuencial (figura 4-9) para apreciar la ganancia final de la señal y el ruido que presenta.

Por último, para la escala de ruido se tomará un valor más alto (1.1), ya que ahora mismo se está haciendo un análisis comparativo y aunque el ruido equivalente a la entrada en el circuito tratado no es despreciable, se observa que tampoco es de una potencia lo suficientemente elevada como para que en los resultados siguientes se puedan ver discrepancias suficientemente considerables en cuanto ruido.

Una vez realizados estos análisis para una potencia mayor de ruido que el real del circuito y haber elegido una frecuencia de Chopper más o menos óptima, como se narrará a continuación, se hará un análisis con el ruido original para apreciar cómo lo reduce la técnica de Chopper.

Para que la comparación quede de la forma más concisa y clara posible, se recogerán los datos más característicos en una tabla (Tabla 4-3).

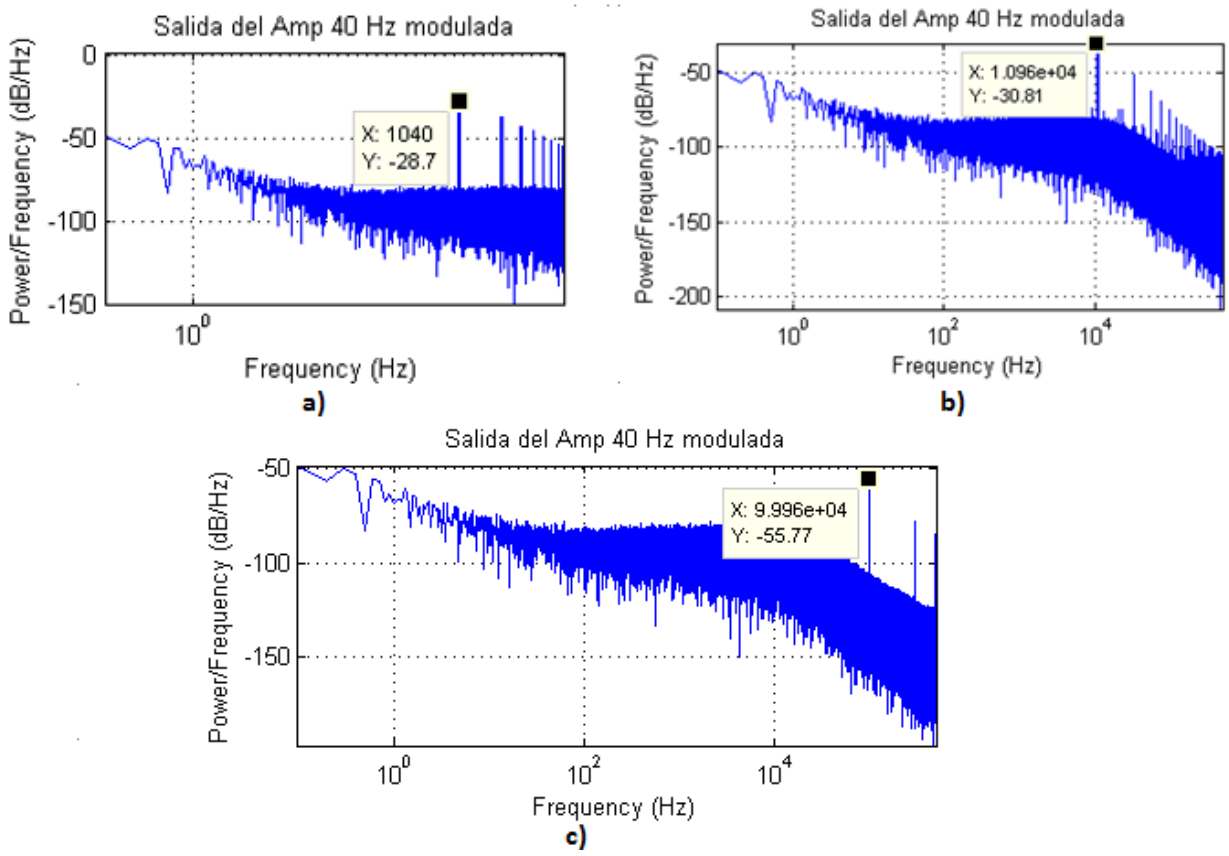


FIGURA 4-7. SALIDA AMPLIFICADOR DOMINIO FRECUENCIAL PARA A) 1 KHZ, B) 11KHZ Y C) 100KHZ. LA GANANCIA DEL AMPLIFICADOR SE VA ATENUANDO CONFORME SE AUMENTA LA FRECUENCIA DE CHOPPER SEGÚN INDICA EL DIAGRAMA DE BODE (FIGURA 4-3).

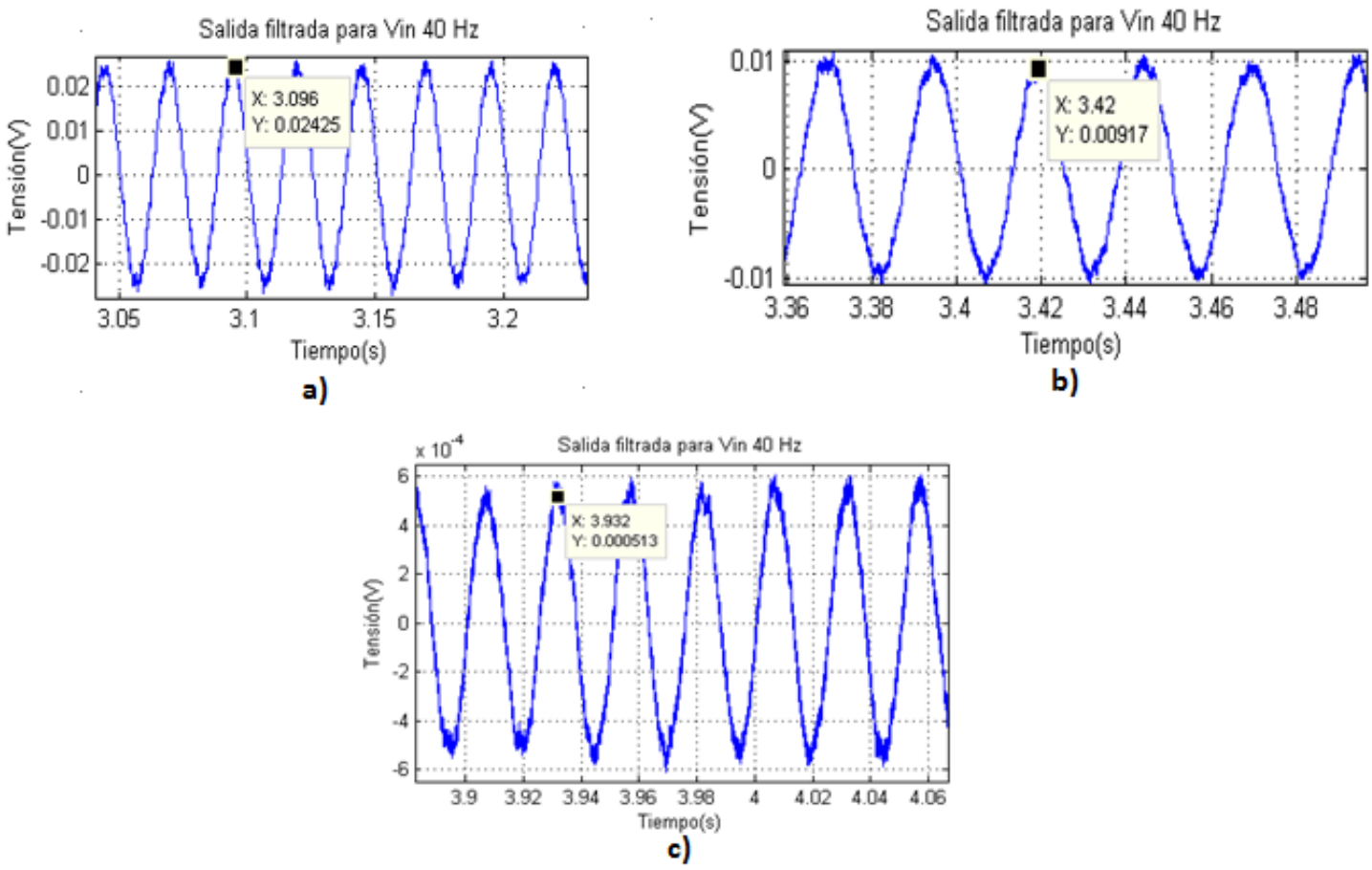


FIGURA 4-8. SALIDA CIRCUITO CHOPPER DOMINIO TEMPORAL PARA A) 1 KHZ, B) 11KHZ Y C) 100KHZ.

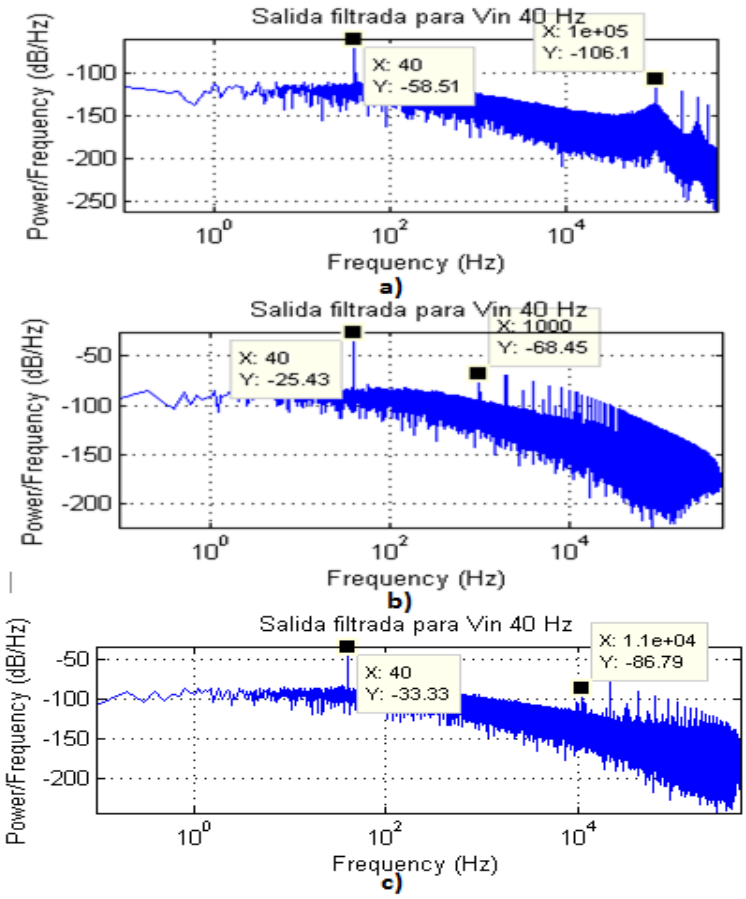


FIGURA 4-9. SALIDA CIRCUITO CHOPPER DOMINIO FRECUENCIAL PARA A) 100 KHZ, B) 1KHZ Y C) 11KHZ.

De las figuras 4-8 y 4-9 se deben comentar dos aspectos fundamentales: En cuanto al ruido, se aprecian en la figura 4-9 como éste disminuye su magnitud conforme las frecuencias de Chopper son mayores. Sin embargo, al ver las señales en la figura 4-8 parece que el ruido no se ha reducido tanto a mayor frecuencia. Esto es así porque la ganancia de la señal que se pretendía amplificar ha disminuido considerablemente al aumentar la frecuencia, por lo que el ruido es más apreciable en el dominio temporal.

TABLA 4-3. RESULTADOS SIMULACIÓN PARA DISTINTAS FRECUENCIAS DE CHOPPER.

	FChopper=1kHz	FChopper=11kHz	FChopper=100kHz
Ganancia salida amplificador (dB)	**48.2**	**47.2**	**21.2**
Ganancia salida circuito (dB)	**47.67**	**41**	**14**
Ruido a la salida del circuito (dB/Hz)	**-68.45 (1kHz)**	**-86.79 (11kHz)**	**-106.1 (100kHz)**
Ruido integrado en banda (1Hz-5kHz) (dB)	**-35.4194**	**-43.306**	**-68.6496**

Se comprueba experimentalmente lo explicado con anterioridad en los apartados teóricos sobre cómo influye la frecuencia de Chopper en la señal de salida. A la hora de elegir esta frecuencia se debe cavilar cuidadosamente sobre lo que se está trabajando, que para Neural Recording, se tienen que aplicar algunos puntos concretos del amplificador que se usa:

- Se trabaja con señales que no superan los 100 Hz de frecuencia y que su mayor valor de magnitud es 1mV. Es importante hacer un inciso: si se quisiera filtrar el ruido de los potenciales de acción, haría falta, aplicando el teorema de Nyquist, una frecuencia de Chopper mínima de 14 kHz. Sin embargo, ya se comentó que esta técnica no está relacionada con este tipo de modulación debido a que no se ven afectadas por el ruido Flicker en la gran parte de los casos debido a su elevada frecuencia y amplitud de señal (en comparación con las demás señales cerebrales).
- A la salida habrá un filtro que cuanto más separado esté el ruido de la frecuencia fundamental de la señal de entrada, menos exigente deberá de ser.
- En la siguiente etapa del canal de Neural Recording habrá un comparador, por lo que se busca que las señales queden bien amplificadas y definidas.
- El ruido Flicker presente, aunque relevante a la hora de tenerlo en cuenta, es (del orden de 5 a 10μV) en torno a 50 veces menor a una señal de entrada.
- La frecuencia de esquina de este ruido está en torno a los 500 Hz.
- La frecuencia de corte del amplificador se sitúa alrededor de los 13 kHz.

Viendo esto, es preferible elegir una frecuencia en la que la ganancia del amplificador no se atenúe (<12kHz) y que se reduzca el ruido considerablemente sin tener que utilizar un filtro excesivamente restrictivo. Por lo tanto, se tornará la balanza del lado de mantener la ganancia en lugar de reducir aún más el ruido. Con todo lo anterior, la frecuencia de Chopper óptima, suponiendo que se toma 800 Hz como la frecuencia de esquina por si queda algún componente del Flicker mayor al ruido blanco, tendrá que cumplir:

$$13kHz > f_{chopper} > 800Hz; 20\cdot\log_{10}(f_{chopper}) \simeq 48.2dB \tag{4-11}$$

Cumpliendo lo anterior (4-11) la frecuencia de Chopper óptima será la máxima que cumpla estos requisitos. Para ello, hay que fijarse en el diagrama de Bode (figura 4-3) (en el caso real sería en un análisis AC del circuito) y elegir la frecuencia máxima para la que aún se mantienen los 48.2dB de ganancia. Esta frecuencia será, aproximadamente, 2.4kHz.

Por último, para esta frecuencia que se ha considerado óptima, se representará la salida del circuito Chopper con la salida del mismo circuito, pero sin Chopper (es decir, la entrada más el ruido, amplificados) tanto en el dominio temporal (figura 4-10) como en el frecuencial (figura 4-11) para poder realizar una valoración final adecuada de los resultados.

Se utilizará un nuevo modelo de Simulink C_Final.slx, y se modificarán un poco los “scripts” para poder hacer todo lo mencionado. Además, partiendo de que ambas señales sin ruido deberían tener la misma potencia integrada en banda, se integrarán ambas señales (figura 4-12) en banda para calcular cuánto ruido se ha eliminado gracias a la técnica de Chopper. Se debe recordar, que se vuelve a aplicar el factor de escala para que las comparaciones sean los más reales posibles.

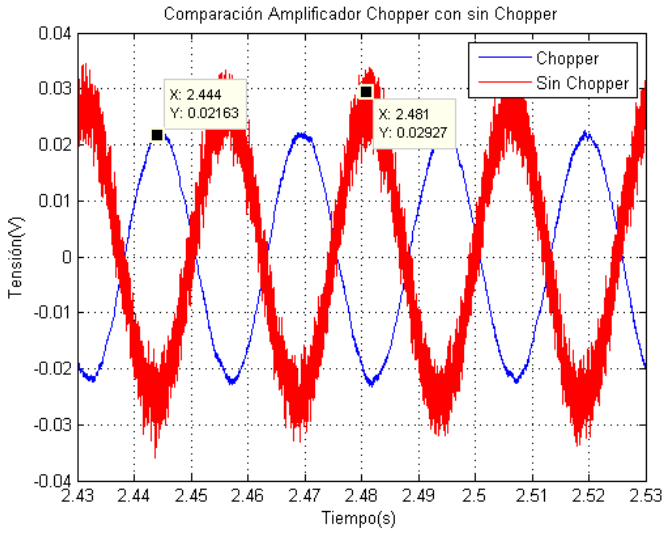


FIGURA 4-10. COMPARACIÓN SALIDA CON CHOPPER Y AMPLIFICADOR SIN CHOPPER DOMINIO TEMPORAL PARA LFP.

En primer lugar, se aprecia que la señal sin Chopper presenta bastante más ruido (luego se verá cuánto) que la señal proveniente del circuito Chopper. Por otra parte, se puede ver que la amplitud de la onda del circuito sin Chopper es mayor (7.36 mV de diferencia) que la que presenta esta técnica. Al margen de que este último resultado pueda parecer negativo, no lo es en absoluto, ya que éste se debe a que se ha ampliado el ruido a la entrada del amplificador que era justamente lo que se pretendía evitar con la topología utilizada, es decir, la señal sin Chopper presenta más amplitud porque se ha amplificado el ruido. Se muestra también que la ganancia del Chopper es de unos 47.5 dB. Esta atenuación de 0.7 dB es debida al filtro, cuya frecuencia de corte es 200 Hz. El desfase de 180º es debido a la modulación y demodulación de la onda.

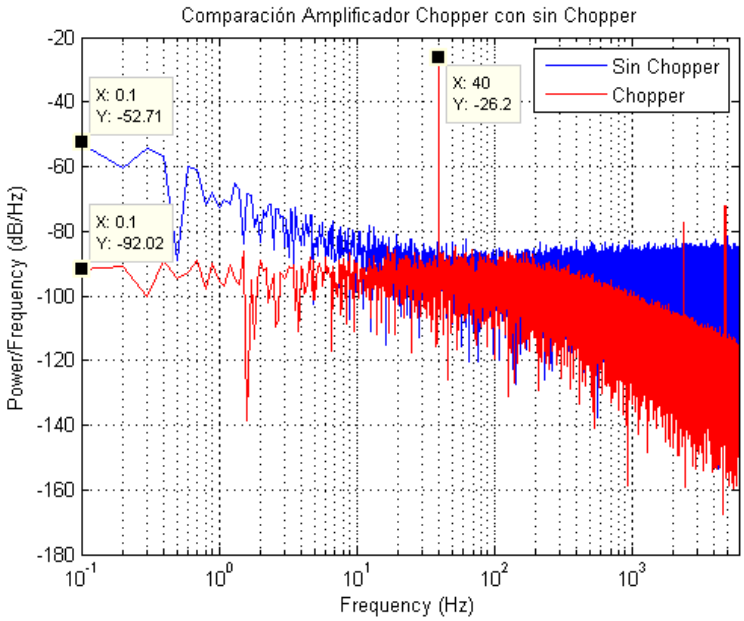


FIGURA 4-11. COMPARACIÓN SALIDA CON CHOPPER Y AMPLIFICADOR SIN CHOPPER DOMINIO FRECUENCIAL PARA LFP.

Con la figura 4-11 se pueden llegar a las mismas conclusiones que con la figura 4-11 tan solo que en esta figura se puede apreciar más claramente como reduce la topología Chopper el ruido a bajas frecuencias, llegando a disminuir en torno a 40 dB/Hz para 0.1 Hz y alrededor de 30dB/Hz para 1Hz.

Se puede ver, también, que el valor en las frecuencias a las que se encuentra modulado el ruido Flicker para el Chopper es tan solo un poco mayor en magnitud que para las mismas frecuencias de ruido blanco en el circuito sin Chopper.

También, en esta figura, comentar la diferencia inapreciable que existe para la magnitud de la señal de entrada (a 40 Hz). Esto corrobora que la mayor amplitud en la figura temporal (figura 4-10) de la señal sin Chopper es debida a la amplificación del ruido, como ya se comentó.

Por último, se calculará el ruido integrado en banda (1 Hz a 5 kHz) de la siguiente forma: Se calculará la potencia espectral en esa banda de la señal de entrada y se le restará a la potencia espectral de la salida del circuito. Realmente, el ruido estará en las frecuencias de Chopper que se hayan elegido, no obstante, es bueno para el análisis saber cómo quedará este a las frecuencias en las que antes se tenía.

La figura 4-12 es, quizás, la gráfica más representativa de lo conseguido durante el trabajo. En ella se realiza un análisis cuantitativo y calificativo de cómo influye la técnica de Chopper en la eliminación de ruido Flicker para el amplificador tratado. Como se puede apreciar, la diferencia en dB de la potencia integrada para bajas frecuencias es del orden de 30 dB, por lo que constituye una reducción del ruido bastante considerable para esta zona del espectro. A la frecuencia a la que se encuentra la señal de entrada (su frecuencia fundamental) se puede ver que justo antes de ésta producirse, la diferencia de potencia de ruido integrado es de 19dB, que, para tratarse una frecuencia de 40 Hz, más que importante. Una vez se pasa el armónico fundamental de la señal de entrada, la diferencia de ruido integrado se iguala y la diferencia es mucho menor debido a que la señal de entrada es varios ordenes de magnitud mayor que la señal de ruido.

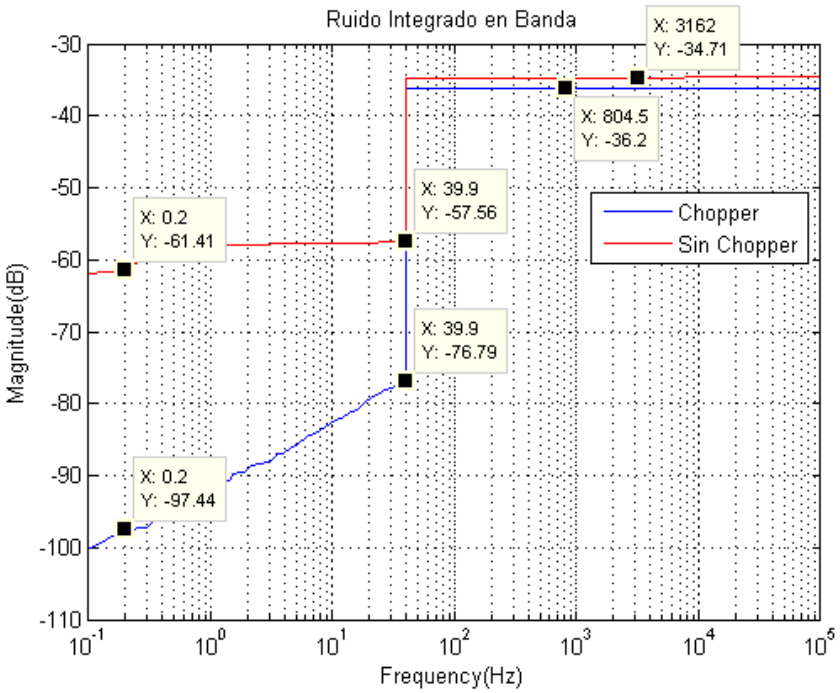


FIGURA 4-12. COMPARACIÓN RUIDO INTEGRADO EN BANDA SALIDA CON CHOPPER Y AMPLIFICADOR SIN CHOPPER PARA LFP.

4.2.2 Simulación Chopper con amplificador MatLab de AP

En este último apartado del trabajo se va a mostrar cómo influye la técnica de Chopper en la amplificación de un potencial de acción para ver si realmente esta técnica es positiva o negativa para este tipo de señales. El potencial de acción que se va utilizar es el mostrado en la figura 3-7 con una frecuencia de 5kHz. Sabiendo esto, la frecuencia de chopper que se va a utilizar será de 10kHz. El filtro usado tendrá una frecuencia de corte de 7kHz.

Los resultados que se mostrarán serán en una figura temporal comparando la salida de la técnica Chopper con la salida sin usar la técnica (figura 4-13) y en una figura frecuencial realizando la misma comparación (figura 4-14).

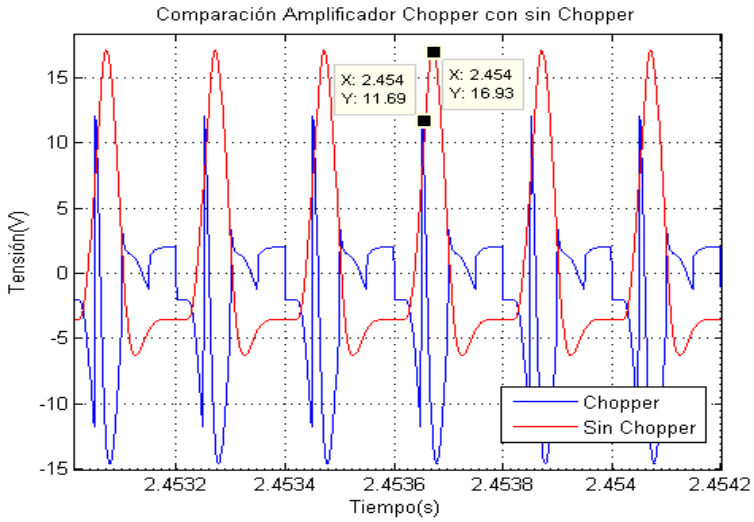


FIGURA 4-13. COMPARACIÓN SALIDA CON CHOPPER Y AMPLIFICADOR SIN CHOPPER DOMINIO TEMPORAL PARA AP.

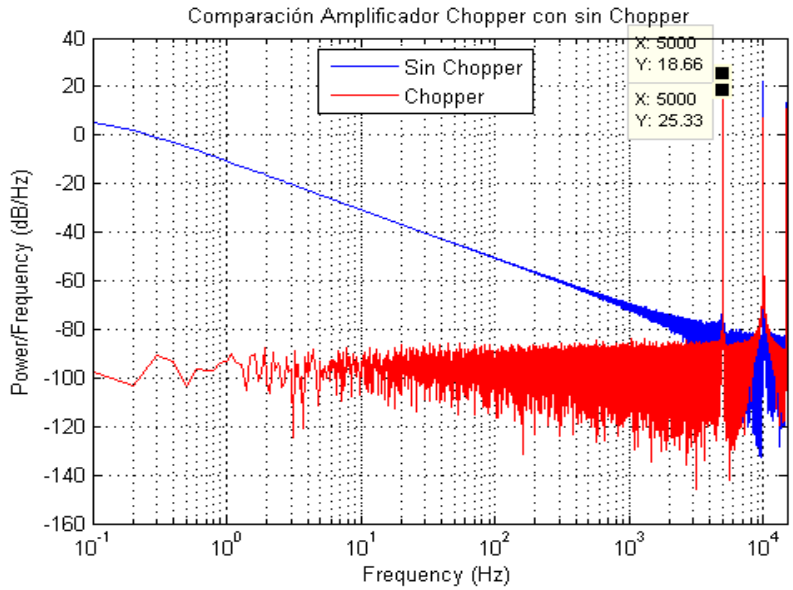


FIGURA 4-14. COMPARACIÓN SALIDA CON CHOPPER Y AMPLIFICADOR SIN CHOPPER DOMINIO TEMPORAL PARA AP.

Analizando ambas gráficas, se deben comentar varios puntos:

- La ganancia del potencial de acción se ve atenuada debido a que la frecuencia de Chopper supera el ancho de banda del amplificador.
- Aunque el ruido Flicker tenga una magnitud elevada en la salida sin Chopper, éste se desprecia debido a que la señal de salida tiene una magnitud bastante mayor y de considerable mayor frecuencia.
- Debido a la modulación y demodulación, al filtro y al ancho de banda finito del amplificador, la señal de salida usando la técnica de Chopper ni se aproxima a un potencial de acción.

Con todo esto, se puede concluir que la técnica de Chopper, tal y como se predijo en el primer capítulo, es perjudicial a la hora de tratar señales como los potenciales de acción, por lo que se deberá encontrar el modo de diferenciar LFP y AP a la hora de procesar las señales o bien encontrar un punto medio en el que sean tratadas positivamente.

5 CONCLUSIONES

5.1 Resumen y conclusiones

Partiendo de una breve Introducción de los conceptos de amplificadores de Neural Recording que son más relevantes para este trabajo se han cubierto los trabajos y se han obtenido los resultados que se listan a continuación.

- Se han implementado un conjunto de algoritmos robustos para representar la densidad espectral de potencia (PSD) y para calcular el ruido integrado para cualquier señal en MatLab-Simulink. Este punto, aunque no fuera uno de los objetivos del proyecto, supone facilitar el trabajo con las señales en el dominio de la frecuencia para futuros trabajos.
- Se han analizado métodos basados en mapas discretos para la generación de ruido: el *saltador* para ruido Flicker y el de Bernoulli para ruido blanco. Alentados por los resultados obtenidos, lo que iba a terminar siendo una mera comprobación de la posibilidad de generar ruido mediante esos mecanismos, cristalizó en una arquitectura híbrida, formada por varios mapas discretos, que puede parametrizarse para conformar ruidos con distintas densidades espectrales de potencia. Viendo las posibilidades que esto suponía para las etapas de diseño donde en muchas ocasiones se precisa de la generación de un ruido concreto para ver cómo afecta en el circuito, se decidió acudir a la literatura para ofrecer técnicas de implementar esos mapas en circuitos "on-chip", que podrían servir como sistemas embebidos del circuito de Neural Recording para realizar simulaciones reales y ver cómo afecta el ruido en ellos.
- Con lo anterior, se generó una distribución de ruido que se aproxima de una forma muy eficiente a la del *active-feedback time constant neural amplifier* en el entorno de MatLab-Simulink. De esta forma, se podrían hacer análisis de alto nivel en el futuro de una manera más fácil y rápida. También se introdujeron estos puntos en Cadence y se hicieron las comprobaciones necesarias de que se estaba generando el ruido deseado.
- Llegados este punto y teniendo el ruido caracterizado, se hizo un breve análisis teórico de las ventajas e inconvenientes de la técnica de Chopper, como paso previo antes de llevarla a la práctica. Una vez hecho esto, se implementó en MatLab-Simulink esta arquitectura y se demostró que era capaz de reducir unos 40 dB/Hz, en el caso ideal, el ruido para frecuencias menores a 1 Hz. Gracias a lo positivo de estos resultados, se implementó en Cadence esta técnica de forma ideal mediante dos métodos distintos: i) uno totalmente analógico mediante llaves y, ii) otro de diseño mixto (para lo que hubo que introducir este tipo de diseño). Las valoraciones que tuvo esta implementación fueron totalmente positivas, aunque se vieron algunas discrepancias con el modelo de MatLab-Simulink y se dedujo que era debido principalmente a que se utilizaban parámetros distintos para realizar la transformada de Fourier discreta.
- En un paso posterior se añadieron fenómenos de segundo orden para comprobar su impacto sobre la técnica de Chopper. Para ello, se introdujo el esquema del *active-feedback time constant neural enhanced amplifier* y se analizó el modelo en pequeña señal del mismo. Gracias al software Mathematica se consiguieron resolver las ecuaciones nodales del modelo de pequeña señal y se consiguió una función de transferencia similar a la del modelo real: la ganancia era 0.2dB mayor, la frecuencia de corte del paso alto se encontraba en 0.116Hz en lugar de 0.129Hz, la del paso bajo en 13kHz en lugar de 7kHz y la pendiente del paso bajo no llegaba a caer 40dB en una década, pero se aproximaba bastante. Sin embargo, estas diferencias podían ser debidas a que los

parámetros que se tenían para la función de transferencia que se utilizaron eran más actuales que con los que se comparaba del modelo real.

- Con un modelo aproximado al real del amplificador y con la implementación del Chopper en MatLab-Simulink, solo quedaba simular el comportamiento del conjunto para ver cómo eran los resultados. Antes de simular, era importante elegir una frecuencia de Chopper que cumpliera con los requisitos que se establecieron durante el capítulo tercero. La frecuencia que se eligió fue 2.4kHz. Los resultados obtenidos fueron más que satisfactorios en los que se pudo observar mediante la gráfica del ruido integrado en banda como para frecuencias cercanas a 0.1 Hz, la potencia de ruido se reducía en unos 30 dB y que esta diferencia se iba reduciendo hasta justo el momento de llegar a la frecuencia fundamental de la señal de entrada, donde se había reducido en 19 dB el ruido en la señal.

Con todo esto, haciendo una valoración global de lo obtenido, se puede concluir que la técnica de Chopper es adecuada para reducir el ruido a baja frecuencia. No obstante, hay que tener muy en cuenta las características de la señal de entrada y del amplificador, ya que usar esta técnica para señales de frecuencias medias/altas resulta contraproducente; a la par que hay respetar los criterios establecidos para elegir la frecuencia a la que se modula. Aplicando esta técnica al amplificador elegido para el trabajo, se ha puesto de manifiesto, aunque con un alto nivel de abstracción, que esta técnica, a priori, resultaría muy positiva implementada en este circuito. Gracias a esto, se le ha ahorrado tiempo y trabajo al diseñador en comprobar experimentalmente (aunque fuera en alto nivel) si conviene realizar una arquitectura Chopper para aplicarla a su circuito.

Por último, como el proceso de abstracción se ha realizado de tal forma que primero se realizaban unas implementaciones generales y después se particularizaba, cualquier diseñador que desee ver cómo influiría, como una primera aproximación, una técnica de Chopper en su circuito puede introducir los datos de su ruido, tal y como se ha definido en el trabajo, y los de su amplificador y ver hasta qué punto le convendría seguir profundizando y utilizar esta técnica en su circuito, lo que le ahorraría tiempo y facilitaría el trabajo.

5.2 Futuras investigaciones

Viendo los resultados positivos expuestos anteriormente, es lógico pensar que el siguiente escalón lógico a seguir en el proyecto es el de reducir el nivel de abstracción y comenzar a realizar el circuito a nivel de transistor. Esto llevaría a empezar a tener en cuenta un efecto negativo introducido en el capítulo dos y que no se ha tenido en cuenta debido a que se tratan de análisis de alto nivel: la inyección de carga.

Además, como se dijo durante el capítulo cuarto, aunque las señales de entrada son de una amplitud bastante pequeña (excepto por los potenciales de acción), podría ocurrir que los puntos de polarización del dispositivo al ser modulados con la técnica de Chopper cambiaran e influyeran negativamente en el diseño.

Otro punto, de vital importancia para el Neural Recording, que no se ha abordado en los análisis que se han realizado es el consumo del dispositivo, ya que, si se tiene que conseguir una modulación mediante transistores que se encuentran funcionando como llaves, lo lógico a pensar es si el consumo va a aumentar considerablemente, lo cual podría resultar contraproducente.

Por tanto, el orden lógico a continuar sería el siguiente:

1. Realización en Cadence de un esquemático totalmente diferencial con transistores de la técnica de Chopper y analizar los resultados obtenidos para un amplificador ideal.
2. En caso que el efecto de inyección de carga sea especialmente importante, recurrir a alguna topología avanzada cómo puede ser la de offset-stabilized Chopper [6].

3. Introducir el amplificador a tratar dentro del esquemático y analizar los resultados obtenidos, principalmente se debe comprobar que los puntos de polarización de los dispositivos siguen siendo los mismos y no se alteran negativamente los parámetros del dispositivo.
4. Comprobar cómo afecta cuantitativamente la técnica de Chopper en la eliminación del ruido Flicker y en el consumo del dispositivo y realizar las valoraciones oportunas.
5. Si es positivo, continuar mejorando los parámetros del amplificador (como el ancho de banda o la ganancia) y realizar el layout del mismo.

Aparte de esto, otra posible línea de investigación sería la realizar un selector para diferenciar los potenciales de campo locales de los potenciales activos, para realizar la técnica de Chopper o no. Esto se podría realizar, por ejemplo, dividiendo el amplificador en dos etapas, una para la cual se utilizará el Chopper y otra etapa para los potenciales de acción con un filtro paso alto contundente a la entrada del amplificador para eliminar los ruidos a baja frecuencia.

Sin duda, el trabajo realizado constituye los cimientos principales ante múltiples posibilidades de implementación directa.

ANEXO A

En este anexo se van a explicar dos funciones que se usarán para calcular y representar las densidades espectrales de potencia en MATLAB de las señales que interese.

El primero de ellos, spectrum.m, calcula la densidad espectral de potencia de una señal de la siguiente forma:

Se le introduce como parámetros de entrada la señal temporal que queremos pasar al dominio frecuencial y la frecuencia de muestreo utilizada para obtenerla. Esta señal se pasa al domino de la frecuencia mediante la función de MATLAB fft que calcula la transformada discreta de Fourier de la señal [23]. Una vez hecho esto, se toman solo los valores para frecuencias positivas y para compensar este efecto en la densidad espectral de potencia multiplicamos el valor de los puntos por dos. Ahora faltaría tener la magnitud en potencia por lo que se eleva el valor de los puntos obtenidos al cuadrado y para normalizarlos y expresarlos en V^2/Hz se dividen estos valores obtenidos cada uno por su frecuencia correspondiente. Los parámetros de salida serán las frecuencias y los valores espectrales de potencia para esas frecuencias. El código empleado es el siguiente:

```
function [f X]=spectrum(x,Fm)
N = length(x);
xdft = fft(x);
xdft = xdft(1:N/2+1);
X = (1/(Fm*N)) * abs(xdft).^2;
X(2:end-1) = 2*X(2:end-1);
f = 0:Fm/length(x):Fm/2;
end
```

Aquí se debe realizar una consideración importante: Como se muestra, el cálculo de la densidad espectral es directamente proporcional al número de puntos que se tomen de nuestra señal, y como estas señales están siendo obtenidas mediante Simulink, van a ser proporcionales al tiempo de muestreo que se tome y al tiempo de simulación en Simulink. En el caso que concierne el trabajo, se deben mostrar con especial interés las bajas frecuencias lo que impone restricciones: o bien se usa un periodo de muestreo muy bajo, o bien se aumenta el tiempo de simulación. Ambas opciones suponen un extra de cálculo computacional por lo que se deben estudiar las distintas posibilidades para obtener los resultados más precisos posibles con un coste computacional alto pero posible. En este caso se ha establecido como óptimo un periodo de muestreo de 1e-6s y un tiempo de simulación de 10s.

Por otro lado, tenemos pintapsd.m que, introduciendo las frecuencias y los valores espectrales de potencia para esas frecuencias, representa en dB/Hz con respecto a las frecuencias (Hz), es decir la distribución espectral de potencia. Es un programa simple pero que ahorrará mucho tiempo y repetición de líneas de código. Se implementa de la siguiente forma:

```
function pintapsd(f,psdx)
figure();
semilogx((f),10*log10((psdx)));
grid on
xlabel('Frequency 10^x (Hz)')
ylabel('Power/Frequency (dB/Hz)')
```

ANEXO B

El cálculo del ruido integrado en banda es un cálculo matemático muy interesante desde el punto de vista electrónico ya que permite comprobar cuantitativamente cuál es la potencia de ruido para una banda de frecuencias determinadas, esto puede ser muy útil para ver si dónde el ruido es más intenso es una banda que influye en nuestro circuito.

Existen diversas maneras para calcular en MATLAB la potencia de una señal integrada en una banda determinada [23]. Sin embargo, se ha preferido calcularlo mediante un sumatorio del área bajo la curva de la señal en el dominio de la frecuencia. Para calcular esta área se ha fragmentado el eje frecuencial en "bins", lo suficientemente pequeños (en el caso que corresponde se toman como "bins" el valor más pequeño que se puede tomar, es decir, el valor de un punto de la longitud de la señal de la que se calcula la integral) y se multiplicará por el valor de la potencia en ese punto concreto. De esta forma se tendrá el área de un rectángulo infinitesimal. El área de la curva en una banda concreta será la suma de todos los rectángulos desde la frecuencia que comienza la banda de interés, hasta la frecuencia donde termina. Matemáticamente sería:

$$P_{total} = \int_{-\infty}^{\infty} S_n(f) \cdot \partial f \; ; \tag{B-1}$$

$$P_{banda} = \int_{fminbanda}^{fmaxbanda} S_n(f) \cdot \partial f \tag{B-2}$$

Si se pasa (B-2) a tiempo discreto se tendría:

$$P_{bandadt} = \sum_{i=fmin}^{fmax} Sn(i) \cdot bin \; ; siendo \; bin = \frac{fs}{2L} \tag{B-3}$$

Siendo fs la frecuencia de muestreo empleada y L la longitud de la señal. Implementando en código (B-3) se obtiene:

```
p=0;
for i=1:1:L/2+1
    p=psdx(i)*fs/(2*L)+p;
end
```

Siendo "p" la potencia de ruido que a calcular, "i" el entero que irá recorriendo todos los bins de la banda, psdx(i) el valor de la potencia para una frecuencia determinada, y "fs/2L" la longitud del bin. Hay que decir que en este caso se está calculando el de la señal entera, y si se quisiera el de una franja concreta bastaría con cambiar el bucle for.

Para comprobar que nuestro código es correcto, se ha entrenado con un ruido blanco de distribución Gaussiana. Se sabe [26] que el valor de la potencia entre menos infinito e infinito de esta distribución es igual al valor de la varianza (desviación típica al cuadrado), por lo que si este código se implementa con una distribución Gaussiana a la que se le atribuye la varianza que se desea, se puede comprobar si el código es correcto. Esto está implementado en Gaussian_Spectrum.m donde se compara el valor de potencia que se consigue mediante una autocorrelación [26] y mediante el sumatorio. Ambos resultados son muy parecidos tal y como se muestra en la figura B-1 por lo que se trata de un buen código para calcular el ruido integrado en una banda determinada.

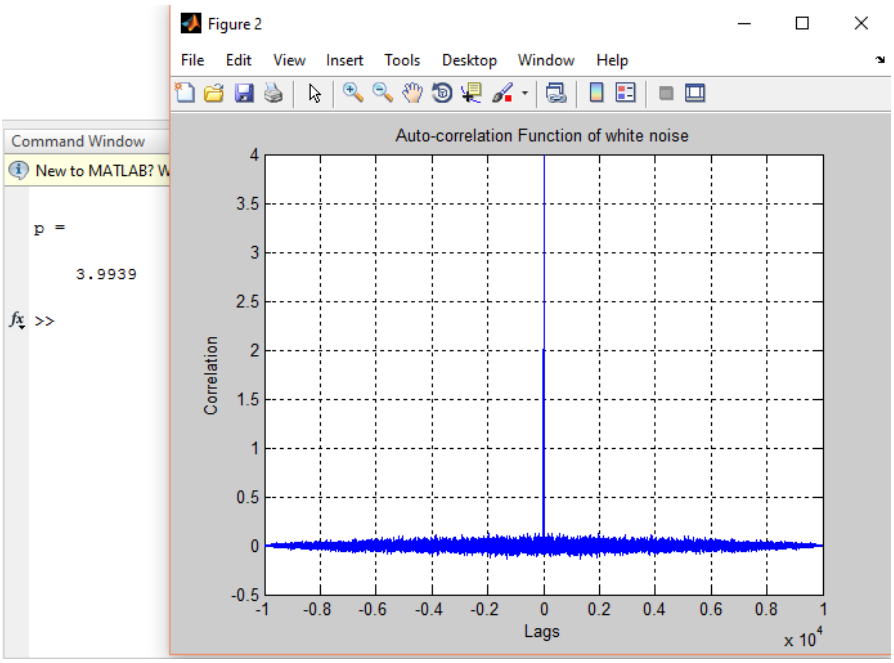


FIGURA B-1. COMPARACIÓN CÁLCULO POTENCIA GAUSSIANA CON VARIANZA = 4 MEDIANTE AUTOCORRELACIÓN E INTEGRACIÓN EN BANDA.

Para calcular el "bin" del Ruido Equivalente a la Entrada que se ha proporcionado del amplificador a estudiar es algo más complicado ya que los datos no están en escala linear, sino logarítmica. Para calcularlo basta con hacer diff(Archivo(:,1)) (suponemos que la segunda columna del archivo son las frecuencias. Con esto, del mismo modo que el anterior,

Por último, es interesante representar el ruido integrado en banda. Para ello se utilizará el comando checksum() y se irá representando el sumatorio (integral discreta) del ruido conforme se vaya aumentando la frecuencia empezando por la frecuencia más baja dada por la simulación y el bin. De esta forma, en un 1Hz representará el sumatorio desde la frecuencia mínima hasta 1 Hz, en 10 Hz, desde la frecuencia mínima hasta 10 Hz, y así sucesivamente hasta la frecuencia de muestreo. Por ello ésta gráfica siempre irá creciendo.

El Script total será el siguiente (Script_RuidoIntegradoEnBanda_JL.m).

```
L= length(X);
N= length(Pot_Flicker_JoseLuis)-1;


bin_JL=diff(Pot_Flicker_JoseLuis(:,1));%mínima distancia entre datos frecuenciales
Pot_bin_JL=Pot_Flicker_JoseLuis((1:end-1),2).*bin_JL; %bin Jose Luis
bin_fl=(Fm/2)/L;
Pot_bin_fl=X*bin_fl;


Pot=0;
Pot_JL=0;
```

```
for i=1:1:10000; %Fm/2*L=0.1 por lo tanto para 1khz Hz L=10000 --> Medimos de esta
forma la banda desde la primera frecuencia observada hasta los 1000 Hz


   Pot=Pot_bin_fl(i)+Pot;


end


Pot_dB=10*log10(Pot) %Mostramos por pantalla lo calculado


for i=1:1:16; %el punto 12 de la frecuencia corresponde aproximadamente con los 1000
Hz


   Pot_JL=Pot_bin_JL(i)+Pot_JL;


end


Pot_JLdB=10*log10(Pot_JL)


figure();
semilogx((f),10*log10(cumsum(Pot_bin_fl))); grid on; hold on;
xlim([0.1 100000]);
ylim([-120 -80]);
semilogx((Pot_Flicker_JoseLuis((1:end-1),1)),10*log10(cumsum(Pot_bin_JL)),'ro-');
grid on;
title('Ruido Integrado en Banda');
xlabel('Frequency(Hz)');
legend('Ruido','Ruido del A-FTCENA');
```

REFERENCIAS

[1] Rodríguez-Pérez, A. (2013). Diseño de Sensores Implantables para la Adquisición de Señales Neurocorticales. Tesis Doctoral Universidad de Sevilla.

[2] Lebedev, M. & Nicolelis, A. (2006). Brain-machine interfaces: past, present and future. En: Trends in Neurosciences, vol. 29, no. 9, pp. 536 –546.

[3] Findings from Mind-Controlled Robot Arm Project (2014). Disponible en: http://www.upmc.com/

[4] Truccolo, W., Donoghue, J. A., Hochberg, L. R., Eskandar, E. N., Madsen, J. R., Anderson, W. S., Brown, E. N., Halgren, E., & Cash, S. S. (2011). Single-neuron dynamics in human focal epilepsy. En: Nature of Neuroscience, vol. 14, pp. 635– 641.

[5] Başar, E. (2013). Brain oscillations in neuropsychiatric disease. Dialogues in Clinical Neuroscience, 15(3), 291–300.

[6] Demosthenous, A. (2014). Advances in Microelectronics for Implantable Medical Devices. En: Advances in Electronics, vol. 2014, DOI:10.1155/2014/981295.

[7] Potenciales de Acción. Hyperphysics, ©2016 [consulta 20 de Junio 2016]. Disponible en: http://hyperphysics.phy-astr.gsu.edu/hbasees/biology/actpot.html

[8] Harrison, R. R. (2008) The Design of Integrated Circuits to Observe Brain Activity En: Proceedings of the IEEE, vol. 96, no. 7, pp. 1203-1216. DOI: 10.1109/JPROC.2008.922581

[9] Valtierra, J.L., Rodríguez-Vázquez, A., & Delgado-Restituto, M. (2016). A 4-Mode Reconfigurable Low Noise Amplifier for Implantable Neural Recording Channels. En: 12th Conference on Phd Research in Microelectronics and Electronics, Lisbon, Portugal.

[10] Fan, Q., Sebastiano, F., Huijsing, H. & Makinwa, K. (2010). A 1.8μW 1μV-offset capacitively-coupled Chopper instrumentation amplifier in 65nm CMOS. En: ESSCIRC, 2010 Proceedings of the, Seville, Spain, pp. 170-173.doi: 10.1109/ESSCIRC.2010.5619902

[11] Wattanapanitch, W., Fee, M. & Sarpeshkar, R. (2007). An Energy-Efficient Micropower Neural Recording Amplifier. En: IEEE Transactions on Biomedical Circuits and Systems, vol. 1, no. 2, pp. 136-147.

[12] Dwivedi, S., & Gogoi, A. K. (2015). Local field potential measurement with low-power area-efficient neural recording amplifier. En: Signal Processing, Informatics, Communication and Energy Systems (SPICES), 2015 IEEE International Conference on, Kozhikode, pp. 1-5.DOI: 10.1109/SPICES.2015.7091507

[13] Chang, J., Abidi, A. A., & Viswanathan, C. R. (1994). Flicker noise in CMOS transistors from subthreshold to strong inversion at various temperatures," En :IEEE Trans. on Electron Devices, vol. 41, no. 11, pp. 1965-1971.

[14] Razavi, B. (2000). Design of Analog CMOS Integrated Circuits.McGraw-Hill Higher Education, ISBN 10: 0072380322 ISBN 13: 9780072380323.

[15] Enz, C. C. & Temes, G. C. (1996). Circuit techniques for reducing the effects of op-amp imperfections: autozeroing, correlated double sampling, and Chopper stabilization, En: Proceedings of the IEEE, vol. 84, no. 11, pp. 1584-1614.doi: 10.1109/5.542410

[16] McGonigal, G. C., & Elmasry, M. I. (1987). Generation of noise by electronic iteration of the logistic map. En: IEEE Trans. Circuits Syst., vol. CAS-34, pp. 981-983.

[17] Chua, L. 0., et al. (1990).Generating randomness from chaos and constructing chaos with desired randomness. En: Int. J. Circuit Theory and Applications, vol. 18, pp. 215-240.

[18] Delgado-Restituto, M., Rodriguez-Vasquez, A., Espejo, S. & Huertas, J. L. (1992). A chaotic switched-capacitor circuit for 1/f noise generation. En: IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, vol. 39, no. 4, pp. 325-328

[19] Devaney, R. L. (1986). An Introduction to Chaotic Dynamical Systems. Benjamin/Cumming.

[20] Nejati, H., Beirami, A., Sahebi, A. G. & Ali, W. H. (2013). Variability analysis of tent map-based chaotic-map truly random number generators. En: IEEE 56th International Midwest Symposium on Circuits and Systems (MWSCAS), Columbus, OH, 2013, pp. 157-160.DOI: 10.1109/MWSCAS.2013.6674609.

[21] Degaldo-Restituto, M., Medeiro, F & Rodriguez-Vazquez, A. (1993) Nonlinear switched-current CMOS IC for random signal generation. En: Electronics Letters, vol. 29, no. 25, pp. 2190-2191.DOI: 10.1049/el:19931471

[22] Delgado-Restituto, M. & Rodriguez-Vazquez, A. (2002). Integrated chaos generators. En: Proceedings of the IEEE, vol. 90, no. 5, pp. 747-767..DOI: 10.1109/JPROC.2002.1015005

[23] The MathWorks, Inc. © 1994-2016. Disponible en: http://es.mathworks.com/

[24]Cadence Design Systems, Inc. © 2016. Disponible en:

https://community.cadence.com/cadence_technology_forums.

[25] Witte, F., Makinwa, K., Huijsing, J. (2009) Dynamic Offset Compensated CMOS Amplifiers. Boston, Springer. ISBN: 978-90-481-2755-9.

[26] Simultation and Analysis of White Noise in MATLAB. Gaussian Waves. [consulta 5 de Junio de 2016]. En: http://www.gaussianwaves.com/2013/11/simulation-and-analysis-of-white-noise-in-MatLab/