Diseño de circuitos para lectura de sensores MEMS en tecnología CMOS

Trabajo fin de Grado

Grado en Ingeniería Electrónica Industrial y Automática

Autor Rubén Garvi Jiménez-Ortiz

Tutor Enrique Prefasi Sen

Agradecimientos

La realización del trabajo de fin de grado supone la culminación de cuatro años de estudios y una importante etapa de mi vida. En este momento me gustaría echar la vista atrás y agradecer su contribución en durante esta etapa a las distintas personas que me han acompañado.

Me gustaría agradecer a la universidad Carlos III de Madrid la oportunidad que me ha brindado de cursar estos estudios con un equipo docente e instalaciones de primera calidad. No me arrepiento de haberla elegido.

Me gustaría agradecer también a los distintos profesores que me han acompañado durante estos años. Sus enseñanzas y consejos han sido de gran valor.

También quiero agradecer a mis compañeros y amigos su apoyo durante los estudios. Hemos compartido grandes momentos juntos que no olvidare.

Así mismo quiero agradecer a mi tutor Enrique, sus consejos y dirección durante el presente trabajo. Sin él no me habría sido posible realizar el proyecto que aquí presento.

Y por último, quiero agradecer a mi familia y en especial a mi madre su paciencia y ayuda durante estos años.

Resumen

En los últimos años el desarrollo de tecnologías *Wireless* ha permitido el desarrollo de redes de sensores. Dichas redes de sensores gozan cada vez más de mayor importancia, y permiten realizar sistemas cada vez más inteligentes al comunicar en tiempo real a distintos dispositivos entre sí y al permitirles a su vez obtener información vital para su funcionamiento.

Todo esto ha permitido desarrollar un nuevo concepto conocido como internet de las cosas o *Internet of Things*. En el internet de las cosas distintos dispositivos se relacionan entre sí y con el medio. En muchos casos ello permite controlar distintos dispositivos así como recibir información de los mismos desde algo tan cotidiano como un teléfono móvil.

Para poder realizar estos sistemas se necesitan sensores de bajo coste y bajo consumo que puedan interactuar con los dispositivos digitales. En el presente trabajo se presenta un diseño a alto nivel de un circuito para mediad de presión mediante un sensor microelectromecánico, MEM. Dicho circuito se caracteriza por su bajo consumo y su alta resolución que alcanza unos 18 bits.

Para la realización de este circuito se emplea una topología de tipo doble rampa debido a su bajo coste y simplicidad. Para obtener una elevada resolución se diseña un modo de operación de dicho circuito que hace que este se comporte como un convertidor de tipo Sigma-Delta de primer orden. Así mismo dicho circuito se conecta directamente al sensor de forma que no son necesarios circuitos de adaptación adicionales.

Por último se trata la implementación física de dicho circuito haciendo hincapié en el problema del ruido flicker. Para solucionarlo se estudia la forma más adecuada de implementación de la técnica para reducir los efectos del ruido flicker conocida como chopping.

Contenido

Índice de Figuras

Índice de Tablas

1 Introducción

En el siguiente trabajo se presenta el diseño e implementación de un conversor de capacidad a digital (CDC) de bajo consumo en tecnología CMOS para la medida de presión mediante sensores MEMS (Sistemas microelectromecánicos o Microelectromechanical systems en inglés).

Dicho diseño se realizará mediante una topología de convertidor ADC (del inglés Analog to Digital Converter) de tipo doble rampa. Para conseguir una mayor resolución en dicha topología usaremos una técnica conocida como *Noise Shaping* o modelado de ruido, típica en los convertidores Sigma-Delta.

A lo largo del presente trabajo se realizará primero un repaso de las tecnologías de sensores MEMS, su importancia actual, así como diferentes topologías. Se continuará haciendo un repaso del estado del arte de los conversores de capacidad a digital (CDC) y algunos conceptos básicos sobre conversores analógico-digital (ADC) del tipo Sigma-Delta.

A continuación se presentará de forma detallada el diseño a nivel de sistema del conversor de capacidad a digital propuesto, así como los modelos de simulación y sus resultados tanto en Cadence como en Simulink.

Posteriormente se procederá a comentar la implementación física del sistema propuesto en tecnología CMOS.

Por último se cerrara el presente trabajo con una conclusión analizando el sistema propuesto y sus resultados obtenidos.

1.1 Sensores MEMS e importancia actual: Avances tecnológicos e internet de las cosas

En las últimas dos décadas la revolución de Internet, la aparición de las redes de sensores y la más reciente irrupción de los teléfonos móviles inteligentes, conocidos comúnmente como *Smartphone* ha cambiado no solo la forma en que nos relacionamos las personas, sino también la forma en que se relacionan los distintos objetos, entre si y con nosotros. Es lo que se conoce como el *internet de las cosas* (*Internet of Things* IoT). [1]

El desarrollo del *internet de las cosas*, ha venido de la mano de los avances tecnológicos en computación, comunicaciones inalámbricas y dispositivos alimentados por baterías. En todo este desarrollo tecnológico han sido de vital importancia la mejora en la capacidad de integración de componentes microelectrónicos y microelectromecánicos, lo que ha permitido fabricar componentes electrónicos más inteligentes y con menor consumo.

Uno de estos componentes fundamentales es el de los sensores para las redes de sensores. Dichos sensores han de tener no solo la capacidad de comunicarse con otros sensores y dispositivos, sino también la capacidad de procesar por ellos mismos información, todo ello con un bajo coste y un bajo consumo.

Todo esto ha sido posible gracias al desarrollo de la tecnología CMOS y al desarrollo de sensores MEMS. Por un lado el la disminución del tamaño de los transistores Mosfet

(transistor de efecto de campo metal-óxido-semiconductor, en inglés *Metal-oxide-semiconductor Field-effect transistor*), ha permitido desarrollar dispositivos electrónicos más baratos y con menor consumo. Dicha tecnología es a día de hoy la predominante en la fabricación de circuitos integrados.

Por otro lado el término MEMS hace referencia al diseño y la fabricación de dispositivos electromecánicos con tamaños del orden del micrómetro. A día de hoy podemos encontrar MEMS en múltiples aplicaciones. Desde sensores de presión o productos químicos a actuadores en las impresoras, hoy en día nos encontramos rodeados de dispositivos MEMS. [2]

En el campo que nos ocupa el de los sensores, los avances en tecnología MEMS, han dado lugar a la disponibilidad de sensores de bajo coste con un desempeño mejor que el de sus equivalentes a macroescala. [3]. Junto a ello es de importancia el hecho de que dichos dispositivos sean fabricados con técnicas similares a las de fabricación de los circuitos integrados, lo cual permite integrar en un mismo dado de silicio dispositivos MEMS y transistores. Ello permite integrar fácilmente en un mismo chip el sensor y la electrónica asociada. Dichos sistemas integrados tienen grandes ventajas, frente a los sistemas realizados con componentes discretos, como son un menor coste y consumo y una mayor inmunidad al ruido.

En el caso de los sensores de presión, en los últimos años se ha producido una transición desde los sensores de tipo piezorresistivo hacia los sensores MEMS capacitivos. Dicha transición se ha debido al hecho de los sensores capacitivos permiten una gran fiabilidad, precisión y sensibilidad, unidos a una baja dependencia de la temperatura y un bajo consumo. [4] Este bajo consumo se debe a que al ser sensores de tipo capacitivo el consumo de potencia en los mismos se produce solo en las resistencias parasitas que se encuentran en contactos y otras partes del dispositivo.

Sin embargo esta naturaleza capacitiva que permite obtener un sensor con un bajo consumo de potencia supone una dificultad a la hora de utilizar dichos sensores en circuitos integrados que están alimentados solo por corriente continua, como ocurre con los chips alimentados por batería. Esto es debido a que los condensadores se comportan como un circuito abierto ante la corriente continua. Es por ello que hay que buscar diferentes técnicas que permitan polarizar estos sensores de forma efectiva. Más adelante en este trabajo se tratara este tema en mayor profundidad.

A continuación se procederá a comentar algunas características de los sensores MEMS capacitivos con más detalle.

1.2 Sensores capacitivos MEMS

Los sensores capacitivos MEMS son utilizados comúnmente para realizar medidas de presión o aceleración. Su principio físico se basa en la variación de la capacidad del sensor ante variaciones de las magnitudes a medir. Como se ha comentado anteriormente entre sus ventajas se encuentra su bajo consumo debido al hecho de que al ser similares a un condensador almacenan energía, pero no la consumen, al contrario que los elementos disipativos como las resistencias.

En el párrafo anterior se hacía referencia a su principio físico de funcionamiento, el cual se basa en la variación de la capacidad del sensor. Es decir los sensores MEMS capacitivos son básicamente condensadores con una capacidad variable.

Un condensador normal se compone de dos electrodos separados por un dieléctrico. Dichos electrodos pueden tener distintas geometrías, desde dos placas paralelas planas hasta dos cilindros concéntricos. En el caso de un condensador de placas paralelas, su capacidad viene dada por la siguiente expresión.

$$C = \varepsilon_0 \varepsilon_r \frac{A}{d} \; [1]$$

Donde

- $\boldsymbol{\varepsilon_0}$ es la permitividad del vacío.
- $\boldsymbol{\varepsilon_r}$ es la permitividad del dieléctrico que se encuentra entre ambas placas.
- $\boldsymbol{A}$ es el área efectiva de las placas.
- $\boldsymbol{d}$ es la distancia entre las placas.

Existen varios métodos para hacer sensores capacitivos. Por un lado se puede variar el área efectiva entre las placas mediante el desplazamiento relativo de las mismas, tal y como se muestra en la siguiente figura.

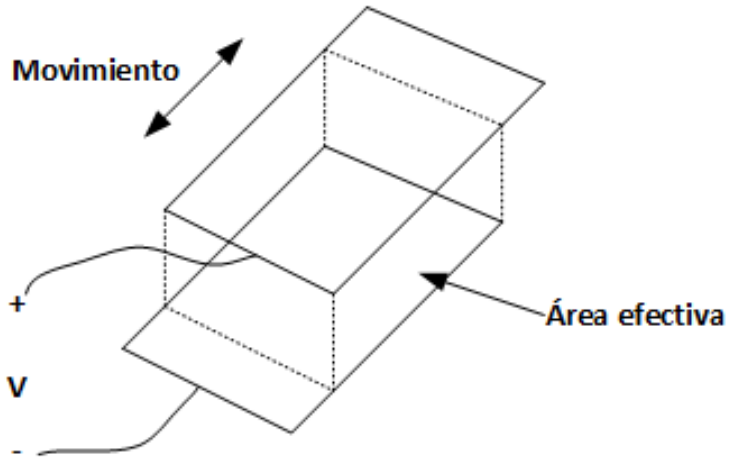


Figura 1.1 Sensor capacitivo de área variable.

En este caso la expresión de la capacidad queda de la siguiente forma

$$C = \varepsilon_0 \varepsilon_r \frac{A + x}{d} \; [2]$$

Donde x representa la variación del área respecto del área nominal.

Otro método para realizar sensores capacitivos consiste en variar el dieléctrico entre las placas tal y como se muestra en la siguiente figura.

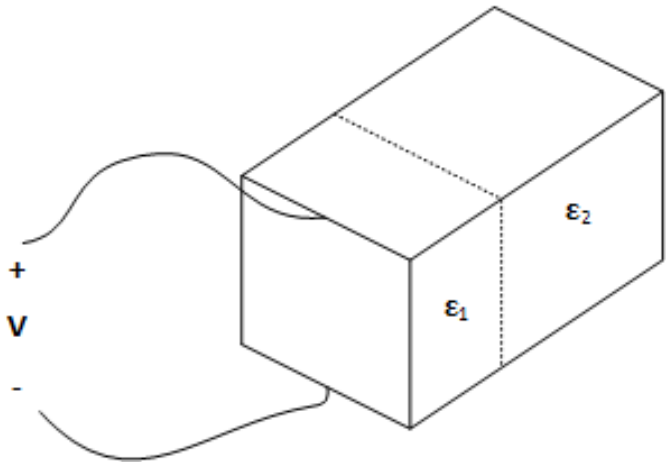


Figura 1.2 Sensor capacitivo con dieléctrico variable.

Siendo la expresión de la capacidad.

$$C = \varepsilon_0 \left(\varepsilon_1 \frac{A_1}{d} + \varepsilon_2 \frac{A_2}{d} \right) \text{ [3]}$$

Es decir es como si tuviésemos en paralelo dos condensadores con distintas capacidades.

Por último podemos variar la distancia entre las placas con la variación de la magnitud a medir para realizar un sensor capacitivo.

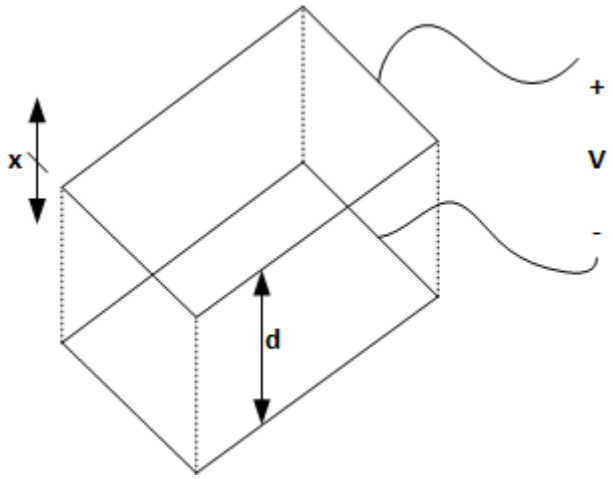


Figura 1.3 Sensor capacitivo con distancia variable.

En este caso la capacidad viene dada por la siguiente expresión.

$$C = \varepsilon_0 \varepsilon_r \frac{A}{d + x} \; [4]$$

Donde x es la variación de la distancia que depende de la magnitud que estamos midiendo.

Este último método es el más usado en la realización de sensores MEMS. De hecho normalmente dichos sensores se suelen realizar con tres placas. Dos de ella permanecen fijas, mientras que una tercera placa se introduce entre ambas. Esta tercera placa es móvil de modo que la distancia entre las placas exteriores con esta tercera placa varía en función de la magnitud a medir.

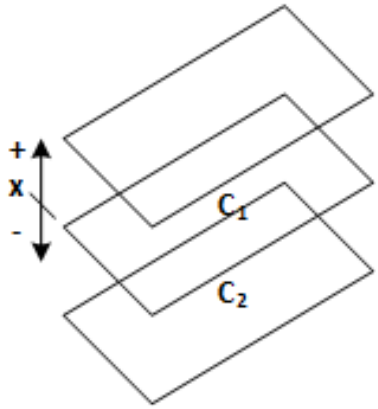


Figura 1.4 Sensor capacitivo diferencial con distancia variable entre las placas.

Dicha configuración se muestra en la Figura 1.4. En ella el condensador C_1 está formado por las placas superior y central, mientras que el condensador C_2 está formado por las placas inferior y central. La placa central es móvil de modo que la expresión de la capacidad es la siguiente.

$$C_T = C_1 + C_2 \; [5]$$

Donde

$$C_1 = \varepsilon_0 \varepsilon_r \frac{A}{d \mp x} = C_0 \frac{1}{1 \mp \frac{x}{d}} \; [6]$$

$$C_2 = \varepsilon_0 \varepsilon_r \frac{A}{d \pm x} = C_0 \frac{1}{1 \pm \frac{x}{d}} \; [7]$$

$$C_0 = \varepsilon_0 \varepsilon_r \frac{A}{d} \; [8]$$

Sabiendo que C_0 es el valor de C_1 y de C_2 en reposo podemos expresar también C_T como

$$C_T = C_0 \pm \Delta C_1 + C_0 \mp \Delta C_2 \; [9]$$

Y como

$$C_1 = C_0 \pm \Delta C_1 \; [10]$$

$$C_2 = C_0 \mp \Delta C_2 \; [11]$$

Suponiendo que la palca central se desplaza en el sentido positivo de las x, podemos expresar ΔC_1 y ΔC_2

$$\Delta C_1 = C_1 - C_0 \; [12]$$

$$\Delta C_2 = C_0 - C_2 \; [13]$$

Por lo que sustituyendo [6], [7] y [8] en [12] y [13] nos queda

$$\Delta C_1 = \varepsilon_0 \varepsilon_r \frac{A}{d-x} - \varepsilon_0 \varepsilon_r \frac{A}{d} = \frac{\varepsilon_0 \varepsilon_r A x}{d^2 - x} \; [14]$$

$$\Delta C_2 = \varepsilon_0 \varepsilon_r \frac{A}{d} - \varepsilon_0 \varepsilon_r \frac{A}{d+x} = \frac{\varepsilon_0 \varepsilon_r A x}{d^2 + x} \quad [15]$$

Que para el caso de x<<d tenemos

$$\Delta C = \varepsilon_0 \varepsilon_r \frac{Ax}{d^2} \; [16]$$

Es decir una variación de la capacidad lineal con x. Por lo tanto, a partir de [9] y [16] podemos expresar C_T de la siguiente forma.

$$C_T = C_0 \pm \Delta C + C_0 \mp \Delta C \; [17]$$

Es decir según [17] es como si tuviésemos dos condensadores cuya capacidad varía con la misma magnitud pero distinto signo en función de la variación de la magnitud que estamos midiendo.

Figura 1.5 muestra la forma en que los sensores diferenciales suelen conectarse en aplicaciones prácticas. Dicha configuración es conocida como medio puente capacitivo. Es importante resaltar que también se pueden hacer sensores diferenciales cuya capacidad varíe en función de otros parámetros distintos a la distancia entre sus placas como por ejemplo el área de las placas.

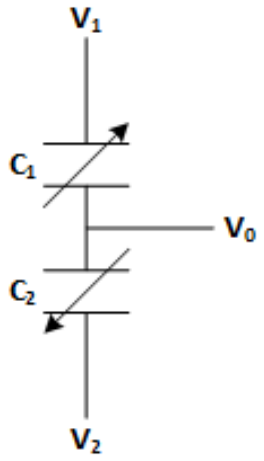


Figura 1.5 Medio puente capacitivo.

Es importante destacar que para que dicho circuito de acondicionamiento funcione es necesario que V_1 y V_2 tengan la misma magnitud y distinto signo. Más adelante en este trabajo se procederá a analizar más en detalle esta topología y a modelar matemáticamente la aplicación concreta de dicho circuito usada en el presente proyecto.

Para finalizar con esta introducción sobre los sensores MEMS capacitivos se va a proceder a comentar como se realizan estos dispositivos físicamente. Con el objeto de incrementar la sensibilidad de los sensores, los electrodos se fabrican con una forma de peine como la que se muestra en la Figura 1.6.

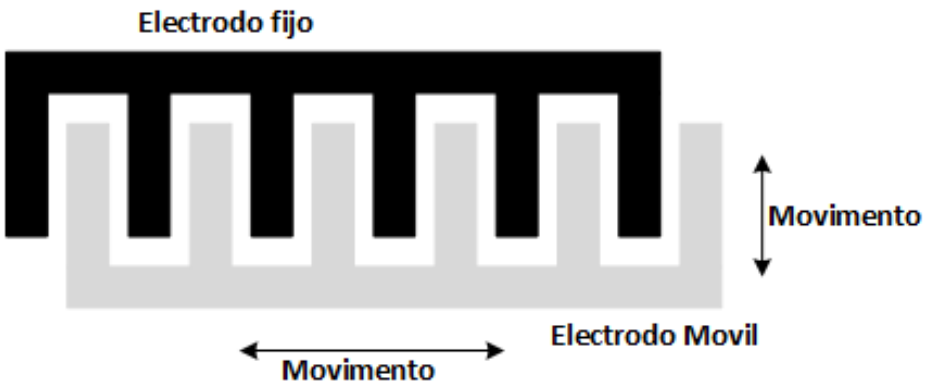


Figura 1.6 Sensor MEMS capacitivo con *fingers*

De esta forma añadiendo un número mayor de púas o *fingers* podemos aumentar la sensibilidad del dispositivo. Es decir si el número de *fingers* es n la sensibilidad del dispositivo se ve multiplicada por n. Esto permite obtener sensores con una mayor sensibilidad en un área menor. [5]

1.3 Topologías de convertidores de Capacidad a Digital (CDCs)

En esta sección se comentarán las distintas topologías existentes de CDCs. Como ya se ha dicho en la introducción el convertidor presentado en este trabajo es un convertidor de tipo doble rampa al cual realizamos Noise-Shaping para aumentar su resolución. Dicho convertidor interactúa directamente con el sensor capacitivo, es decir es un convertidor CDC. El convertidor doble rampa es un convertidor de un bit y el sistema se comporta como un convertidor Sigma-Delta de primer orden.

En los últimos años se han presentado distintas topologías de CDCs. Destacan tres aproximaciones fundamentales. Por un lado encontramos interfaces totalmente digitales, basadas en la conversión de capacidad a tiempo (CTC del inglés *Capacitive to Time Converter*) y después en la conversión de tiempo a un valor digital (TDC del inglés *Time to Digital Converter*) como las presentadas en [6], [7] y [8].

Por otro lado se han presentado varios diseños que usan convertidores Sigma-Delta directamente conectados al sensor. En [9] se emplea un modulador Sigma-Delta de primer orden. Por su parte en [10] se presenta un sensor de humedad basado en un modulador Sigma-Delta de tercer orden.

Por ultimo tenemos técnicas basadas en *Switched-Capacitors*. Dichas técnicas consisten en mover cargas de unos condensadores a otros mediante el uso de interruptores. En [11] se presenta un CDC basado en una técnica de *Switched-Capacitors* con un nuevo esquema que reduce los errores debido a inyección de carga desde los interruptores. En [12] se presenta un sensor capacitivo que consta de un conversor de capacidad a voltaje (C2V) basado en la técnica de *Switched-Capacitors* seguido de un ADC. Por último en [13], en [14] y en [15] se presentan CDCs basados en técnicas de *Switched-Capacitors* con conversores doble-rampa que se conectan directamente al sensor.

1.4 ADC Doble Rampa

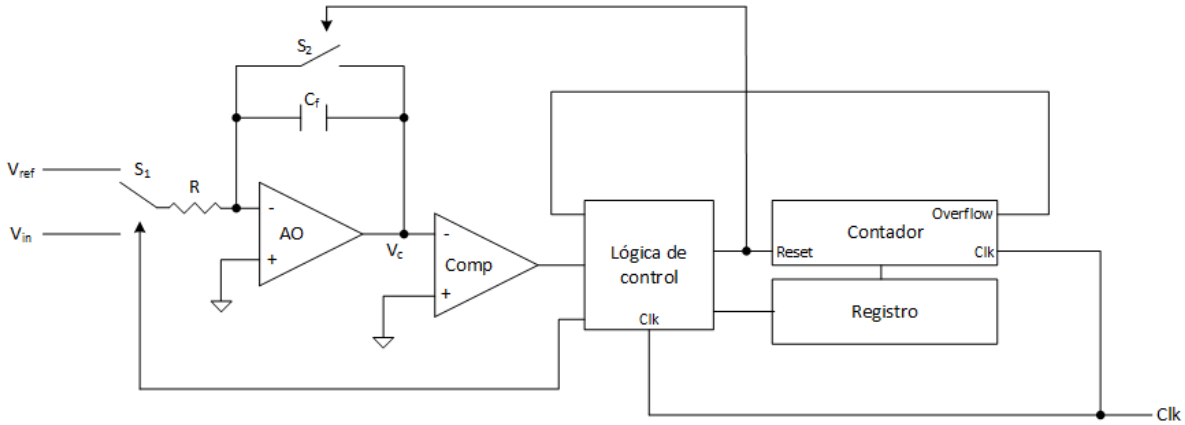


Figura 1.7 Esquema de un ADC de tipo doble rampa.

En la figura 1.7 se muestra el esquema de bloques de un convertidor analógico-digital de tipo doble rampa. Dicho convertidor se compone de un integrador formado por el amplificador operacional y el condensador C_f, que se conecta a un comparador que a su vez va conectado a un circuito digital de control. Además de la salida del comparador, al bloque de control entran la señal de reloj y la señal de *overflow* (desbordamiento) del contador. El bloque de control genera la señal de Reset que resetea el contador y el condensador C_f, así como la señal que hace que el registro guarde la cuenta del contador.

Por su parte el contador tiene como entradas una señal de reloj, *Clk*, y la señal de Reset. El número de bits del contador determina el número de bits del ADC. El registro tiene el mismo número de bits que el contador.

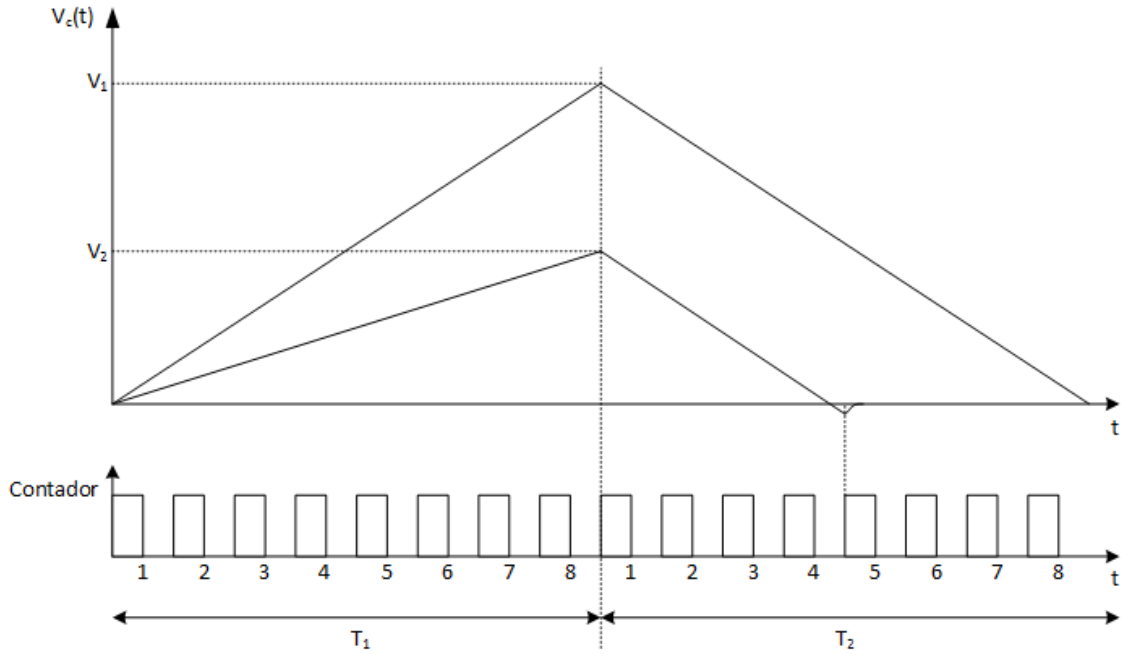


Figura 1.8 Funcionamiento del ADC de doble rampa

En la Figura 1.8 se muestran la salida del contador y la señal $V_c(t)$ de un ADC doble rampa de 3 bits. Como se puede observar el proceso de conversión se divide en dos periodos T_1 y T_2. El primer periodo, T_1, es un periodo fijo. Para un conversor de n bits, dicho periodo durará 2^n pulsos de reloj. Durante dicho periodo T_1, se integra la señal V_{in} que es la tensión analógica que se quiere convertir a digital. En la Figura 1.8 se muestra el valor máximo de tensión en el integrador, V_1 y V_2 para dos tensiones V_{in} distintas. La expresión del valor máximo de tensión en el integrador se obtiene integrando la siguiente expresión.

$$V_c = -\frac{1}{C_f}\int_0^{T_1} \frac{V_{in}}{R}\, dt \quad [18]$$

Donde $\frac{V_{in}}{R}$ es la corriente que carga el condensador C_f, y el signo menos se debe a que dicho condensador se encuentra conectado a la entrada inversora del amplificador operacional. Integrando dicha expresión desde 0 hasta T_1 obtenemos.

$$V_c = -\frac{V_{in}}{C_f R} T_1 \quad [19]$$

Teniendo en cuenta que $T_1 = 2^n \frac{1}{f_{clk}}$ podemos expresar [19] de la siguiente forma.

$$V_c = -\frac{V_{in} 2^n}{C_f R f_{clk}} \quad [20]$$

Es decir durante la fase T_1 la tensión en el integrador aumentará, en el caso de que V_{in} sea negativo, o disminuirá, en el caso de que V_{in} sea positivo, con una pendiente dependiente del valor de V_{in} igual a:

$$-\frac{V_{in}}{C_f R}$$

Una vez pasados 2^n pulsos de reloj, el contador envía una señal de *overflow* que pasa a través del bloque de control y cambia de posición al interruptor S_1, de modo que la entrada del integrador pasa a ser ahora un valor de tensión constante, V_{ref}. En ese momento comienza el periodo T_2. Durante dicho periodo el integrador se descargará desde el valor máximo alcanzado durante T_1 con una pendiente constante. Dicha pendiente se obtiene integrando la siguiente expresión.

$$V_c = -\frac{1}{C_f}\int_0^{T_2} \frac{V_{ref}}{R}\, dt \text{ [21]}$$

Lo que nos da

$$V_c = -\frac{V_{ref}}{C_f R} T_2 \text{ [22]}$$

Y por tanto la pendiente a la que se descarga el integrador durante T_2 es

$$-\frac{V_{ref}}{C_f R}$$

Como la pendiente de descarga del integrador es constante, el tiempo T_2, que tarda el integrador en descargarse depende de la tensión máxima alcanzada durante T_1. Como dicha tensión depende del valor de tensión a convertir V_{in}, si contamos el número de pulsos que tarda el integrador en descargarse, podremos obtener el valor digital de la magnitud V_{in} que estamos convirtiendo.

Por último, una vez realizada la conversión, el integrador es reseteado cerrando el interruptor S_2 y por tanto cortocircuitando C_f.

La ventaja principal de los ADCs de doble rampa, y en general de los ADCs integradores, es que ofrecen una alta resolución a un bajo coste, debido a la simplicidad de sus circuitos. La principal desventaja es su tiempo de conversión. En el caso de un ADC de doble rampa el tiempo de conversión máximo es

$$T_{con(max)} = T_1 + T_{2(max)} = \frac{2 \cdot 2^n}{f_{clk}} = \frac{2^{n+1}}{f_{clk}} \text{ [23]}$$

Dicho tiempo de conversión máximo se da cuando el valor V_{in} es igual al fondo de escala del conversor. A partir de [23] podemos observar que el tiempo de conversión depende del número de bits del ADC, n. Por lo que a mayor número de bits mayor tiempo de conversión.

Otra de las ventajas del ADC de doble rampa es que su salida no depende de los valores de C_f y R, ya que ambos aparecen en cada periodo de integración, T_1 y T_2, y por tanto se cancelan. No obstante, es necesario un valor V_{ref} preciso, para garantizar una conversión correcta. [16]

1.5 Conversores de Sobremuestreo o Sigma-Delta

Los conversores de sobremuestreo son aquellos que muestrean la entrada a frecuencias muy superiores a la frecuencia de Nyquist, esto es a frecuencias muy superiores a 2B, donde B es el ancho de banda de la señal que se quiere convertir. Dichos conversores son capaces de obtener resoluciones muy superiores a las de los conversores que muestrean a la frecuencia de Nyquist.

Además, su precisión no depende tanto de la precisión de los componentes analógicos del conversor, debido al uso de técnicas de procesamiento digital de la señal. Esto permite reducir la cantidad de circuitería analógica en el conversor y su complejidad.

Por otro lado, el hecho de muestrear a una frecuencia muy superior a la de Nyquist, hace que el fenómeno del *aliasing* sea menos problemático, por tanto requiriendo únicamente filtros *anti-aliasing* muy simples. [17]

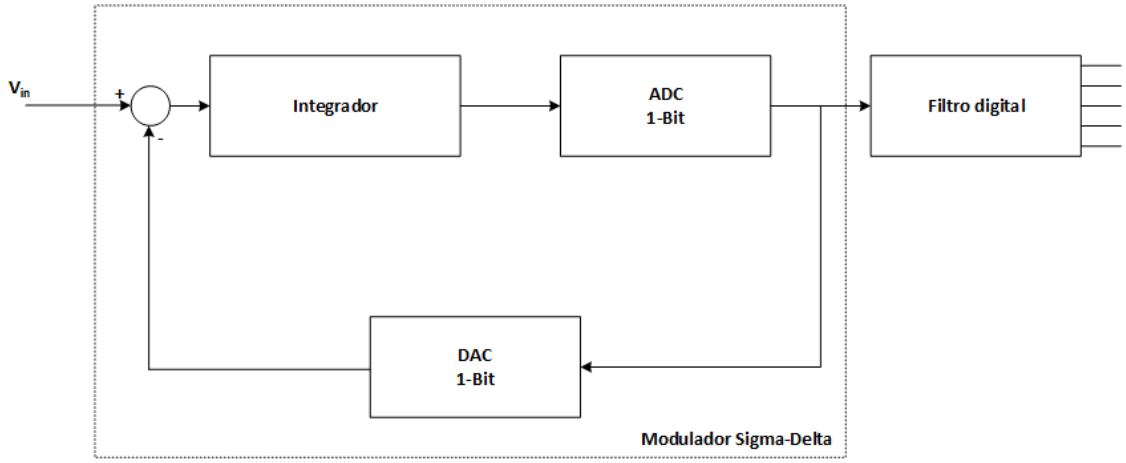


Figura 1.9 ADC Sigma-Delta.

La Figura 1.9 muestra un diagrama de bloques de un conversor Sigma-Delta de primer orden. El funcionamiento de dicho conversor es como sigue. Primero el modulador compuesto por el integrador, el ADC y el DAC (Conversor Digital-Analógico, o en inglés Analog-to-Digital Converter), se encarga de realizar la cuantificación, modulando la señal V_{in}, mediante una serie de pulsos de ancho variable. Dicha señal es similar a una PWM. Esta modulación es conocida como Sigma-Delta.

En dicha señal modulada los pulsos tienen un valor alto durante más tiempo a medida que la señal de entrada tiene un valor mayor. Así mismo los pulsos son más estrechos, a medida que la señal de entrada tiene un valor menor. La densidad de los pulsos representa por tanto el valor medio de la señal V_{in} en un determinado periodo.

Una vez realizada la modulación, la señal es pasada al filtro digital. Dicho filtro se encarga de atenuar el ruido de cuantificación. La señal es después convertida a la frecuencia de Nyquist, siendo esta la salida digital del conversor. Este valor digital representa el valor medio de la señal de entrada durante el periodo de muestreo.

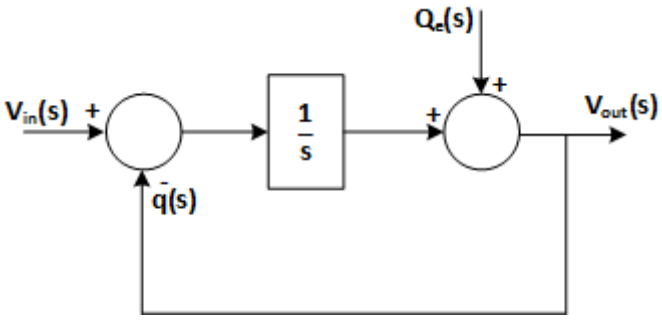


Figura 1.10 Modulador Sigma-Delta [18]

En la Figura 1.10 se muestra un diagrama de bloques del modulador Sigma-Delta de la Figura 1.9. En él, el ADC de un bit ha sido modelado como la suma de ruido de cuantificación $Q_e(s)$ a la salida del integrador. Por otro lado el DAC se ha considerado ideal. Por último el integrador se ha modelado mediante un bloque $\frac{1}{s}$.

La principal ventaja de los conversores Sigma-Delta se puede observar obteniendo las funciones de transferencia de $V_{in}(s)$ y $Q_e(s)$ en la Figura 1.10. La función de transferencia de $V_{in}(s)$ es:

$$\frac{V_{out}(s)}{V_{in}(s)} = \frac{1/s}{1 + 1/s} = \frac{1}{s+1} \quad [24]$$

Es decir la función de transferencia de $V_{out}(s)/V_{in}(s)$ es la misma que la de un filtro paso bajo. En cuanto a la función de transferencia del ruido de cuantificación tenemos:

$$\frac{V_{out}(s)}{Q_e(s)} = \frac{1}{1 + 1/s} = \frac{s}{1+s} \quad [25]$$

O lo que es lo mismo la función de transferencia de un filtro paso alto. Es decir el conversor Sigma-Delta, deja pasar la señal V_{int}, mientras que envía el ruido de cuantificación a altas frecuencias. Este fenómeno es conocido como *Noise-Shaping* o modelado de ruido. Es importante resaltar que el modulador no elimina el ruido de cuantificación sino que lo saca del ancho de banda de la señal V_{in}, enviándolo a frecuencias mayores. Es por ello que es necesario filtrar la señal de salida del modulador para obtener el valor digital de V_{in}.

2 Sistema Propuesto

El sistema propuesto se realizará mediante una topología de conversor de tipo doble rampa, la cual será operada de una forma especial para conseguir un efecto similar al *Noise-Shaping* o modelado de ruido, comentado anteriormente. El proyecto consistirá específicamente en el diseño de la parte analógica del conversor.

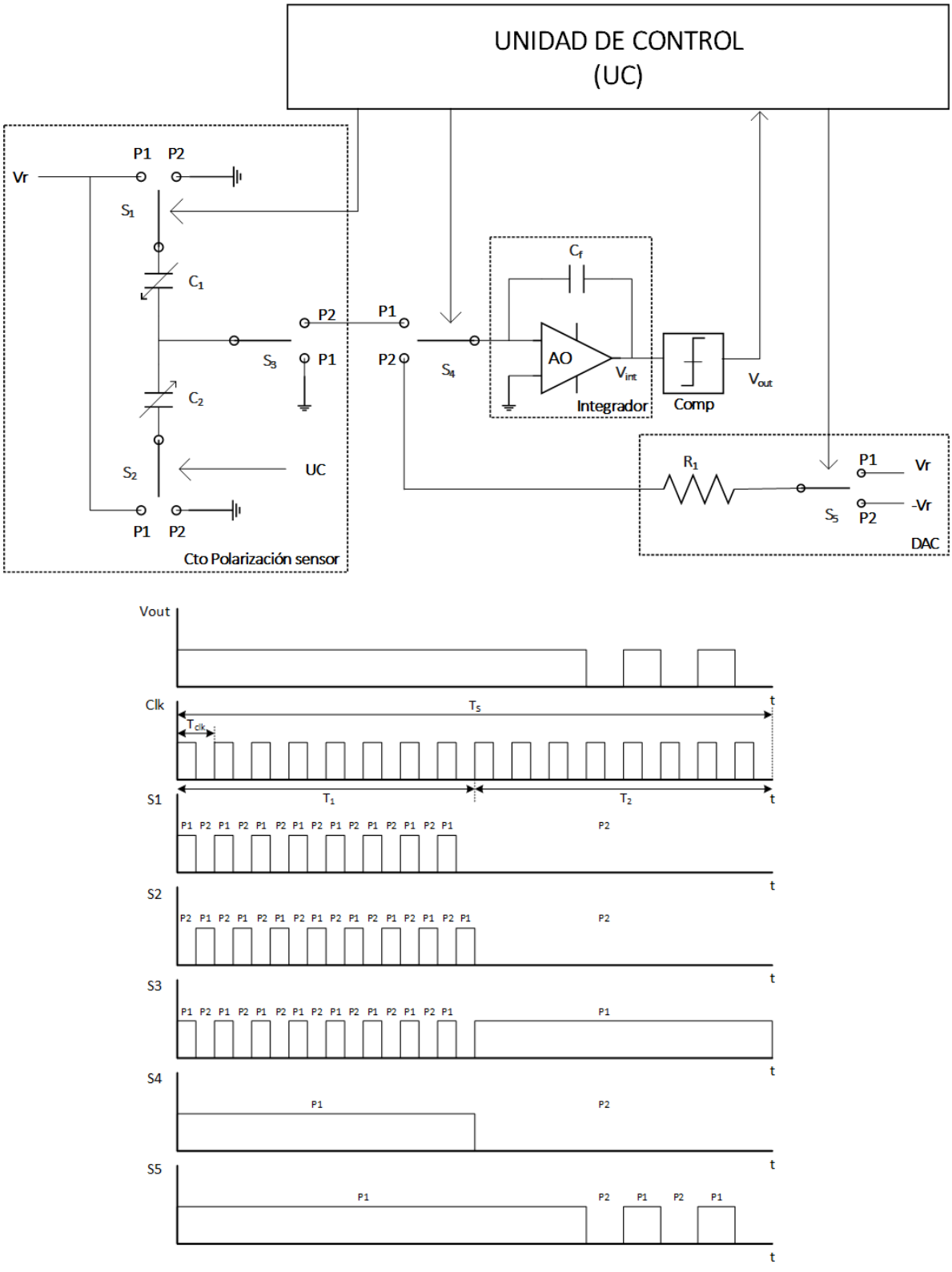


Figura 2.1 Circuito analógico del conversor y cronograma asociado.

Dicho circuito analógico junto a su cronograma asociado se muestra en la Figura 2.1. Como se puede observar el circuito se compone de varios bloques. Por un lado disponemos de un bloque que contiene el circuito de polarización del sensor. Dicho bloque se compone de un sensor MEMS capacitivo de tipo diferencial compuesto por C_1 y C_2, junto a los interruptores S_1 y S_2.

El bloque de polarización del sensor se conecta directamente al ADC de doble rampa compuesto por el integrador, el comparador síncrono que realiza la función de un ADC de 1 bit y el DAC formado por la resistencia R_1 y el interruptor S_5. Por último la unidad de control se encarga de operar los distintos interruptores mediante las señales de disparo correspondientes. V_r es la tensión de referencia.

Como se puede comprobar los bloques del ADC de doble rampa son similares a los de un ADC de tipo Sigma-Delta. Por otro lado la diferencia principal que encontramos en el circuito respecto al conversor de tipo doble rampa convencional es la existencia del bloque del circuito de polarización. En lugar de realizar la carga del condensador C_f a través de R_1 tanto en el periodo T_1 como en el periodo T_2, en el conversor propuesto realizamos la carga de C_f a través del circuito de polarización del sensor durante T_1, mientras que su descarga se lleva a cabo a través de R_1 durante T_2. Esta forma de operación permite conectar el sensor directamente al ADC, de forma que obtenemos un conversor de Capacidad a Digital. El interruptor S_4 se encarga de variar la topología del circuito permitiendo dicho modo de operación.

A continuación se pasara a explicar la operación del circuito y como se realiza el modelado de ruido.

2.1 Operación general del circuito y modelado de ruido

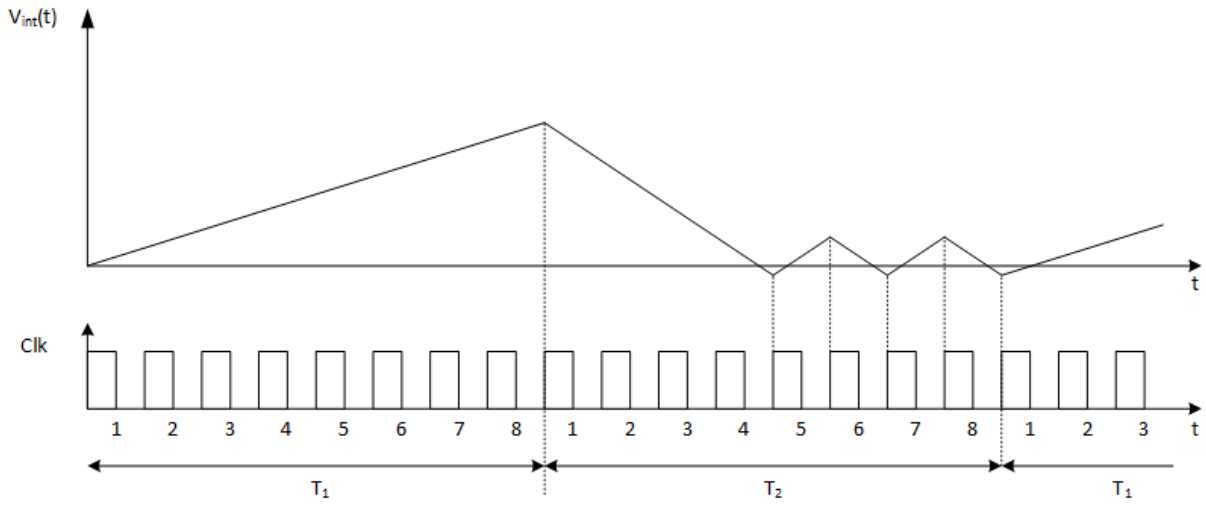


Figura 2.2 Operación del conversor doble rampa con Noise-Shaping.

La Figura 2.2 ilustra el funcionamiento del conversor propuesto. En ella se muestran tanto la señal de reloj como la variable de estado V_{int}. Es importante resaltar que esta es una figura simplificada ya que el comportamiento de V_{int} durante T_1 es escalonado en

lugar de recto. Se comentará más sobre esto cuando se pase a explicar el circuito de polarización del sensor.

Como se pude comprobar el funcionamiento del conversor durante T_1 es similar al de un doble rampa convencional. Así mismo la pendiente con la que se descarga V_{int} durante T_2 es también constante.

Sin embargo se puede observar que hay dos diferencias importantes con la operación de un conversor doble rampa convencional. Por un lado en el caso del sistema propuesto tanto T_1 como T_2 tienen un valor constante independiente de la magnitud a convertir. En el caso de la Figura 2.2 ambos son iguales a $2^n T_{Clk}$.

Por otro lado al finalizar T_2 no se procede a resetear el condensador C_f. En lugar de eso se deja oscilar el valor de V_{int} en torno a cero. De este modo se conserva el valor de tensión almacenado en C_f en el momento en que llega el primer pulso de reloj después de que V_{int} cruce cero. Este valor de tensión es el error de cuantificación de nuestro ADC, y por tanto de esta forma restamos dicho error de cuantificación al valor que vamos a medir en los siguientes periodos T_1 y T_2.

Esta resta se puede ver muy fácilmente en la Figura 2.2. Al empezar a integrar de nuevo en el segundo periodo T_1, no empezamos en cero sino que empezamos en 0-Q_e siendo Q_e el error de cuantificación. De este modo el valor máximo que alcanzará la tensión en el condensador durante la segunda conversión será

$$V_{int(T1)}(2) = V_{int(T1)}(1) - Q_e \text{ [26]}$$

Hay que tener en cuenta que el signo del error de cuantificación, Q_e, depende de si cruzamos por cero durante un periodo de reloj par o impar. Si cruzamos cero durante un periodo de reloj par el error de cuantificación tendrá signo positivo, mientras que si cruzamos cero durante un pulso de reloj impar el error de cuantificación será negativo. Teniendo en cuenta esto y [22] la expresión del error de cuantificación es:

$$Q_e = (-1)^n \frac{V_r}{C_f R_1} T_{err} \text{ [27]}$$

Donde T_{err} es el tiempo que pasa desde que V_{int} cruza por cero hasta que llega el siguiente pulso de reloj, y *n* es el número del periodo del reloj en el que V_{int} cruza por cero (empezando a contar los periodos de reloj desde uno). En el caso de que V_{int} cruce por cero desde valores negativos la expresión [26] seguiría siendo válida, mientras que la expresión [27] quedaría de la siguiente forma.

$$Q_e = (-1)^{n-1} \frac{V_r}{C_f R_1} T_{err} \text{ [28]}$$

Es decir igual que la ecuación [28] multiplicada por -1.

Mediante esta operación especial del conversor de doble rampa es como realizamos el modelado de ruido. Para ver porque de este modo realizamos modelado de ruido vamos a ver qué es lo que ocurre con la señal de salida del circuito.

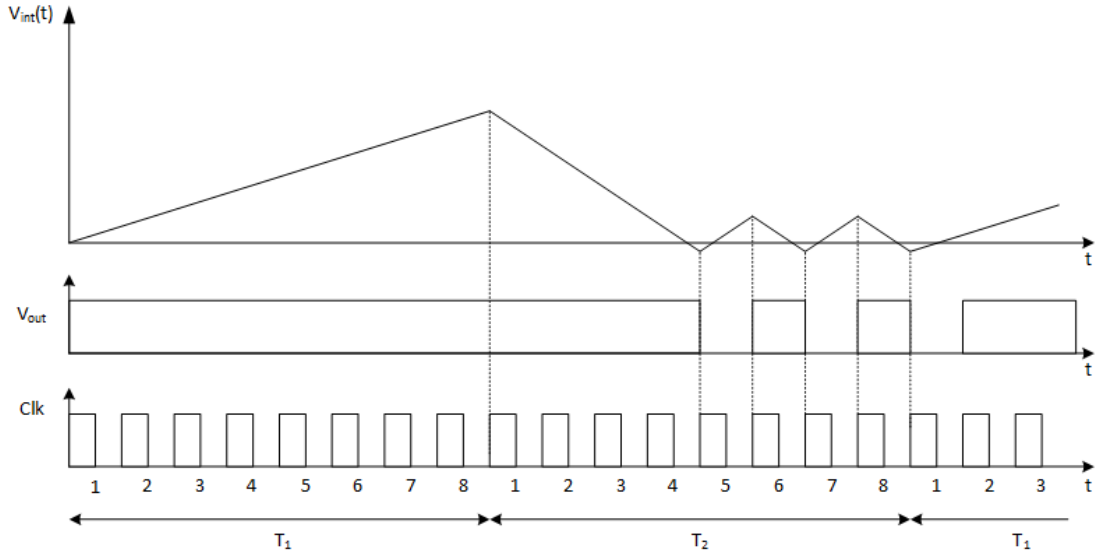


Figura 2.3 Salida del circuito V_{out}.

En la figura 2.3 se muestra la salida del circuito V_{out} además de mostrar de nuevo las señales de reloj, Clk, y la tensión en el integrador V_{int}. Es importante resaltar que solo se tiene en cuenta V_{out} durante T_2, ya que durante T_1, V_{out} tiene un valor alto para toda medida positiva. Del mismo modo tiene un valor bajo para toda medida negativa. De un modo u otro no aporta nada a la medida que se está realizando y por tanto no se tiene en cuenta.

Como se ha comentado anteriormente, un conversor Sigma-Delta genera una señal similar a una PWM dependiendo del valor de la magnitud a convertir. Esto es, genera una señal con periodo constante cuyos pulsos tienen un ancho variable en función de la señal que se esté modulando, o lo que es lo mismo genera una señal cuyo ciclo de trabajo depende de la señal moduladora. El valor medio de dicha señal generada es igual al valor de la señal moduladora en dicho periodo.

En nuestro caso para conseguir el mismo efecto tomamos la señal V_{comp} y restamos el tiempo que dicha señal tiene un valor bajo al tiempo que tiene un valor alto. Es decir hacemos que los periodos de reloj en los que V_{out} tenga un valor alto sean equivalgan a 1, y aquellos en los que V_{out} tenga un valor bajo equivalgan a -1, y sumamos todos los valores de forma que obtenemos el número de pulsos que la señal V_{out} tiene valor alto.

Sin embargo este resultado nos da una señal cuyo valor medio contiene no solo la señal que queremos medir o convertir, sino también su error. Es decir es similar al resultado que obtendríamos con un conversor doble rampa convencional. Para poder generar una señal cuadrada cuyo ciclo de trabajo sea proporcional a la señal que estamos midiendo tenemos que realizar la media del ciclo de trabajo de varios pulsos consecutivos. De esta forma obtenemos un resultado mucho más preciso.

Esto ocurre porque en los distintos ciclos consecutivos el ciclo de trabajo de cada ciclo es proporcional a un valor que va desde el valor de la señal que queremos convertir menos el error máximo de cuantificación, hasta el valor de la señal más el error máximo de cuantificación. Ello es así debido al hecho de que en cada ciclo consecutivo vamos

restando el error de cuantificación del ciclo anterior al no resetear el condensador C_f como se ha explicado con anterioridad.

Es decir llevando a cabo esta operación, y quedándonos con las bajas frecuencias de V_{out}, es decir la media, obtenemos un resultado mucho más preciso. Es decir nos llevamos el ruido de cuantificación a altas frecuencias, realizamos modelado de ruido o *Noise-Shaping*.

2.2 Operación del circuito de polarización del sensor

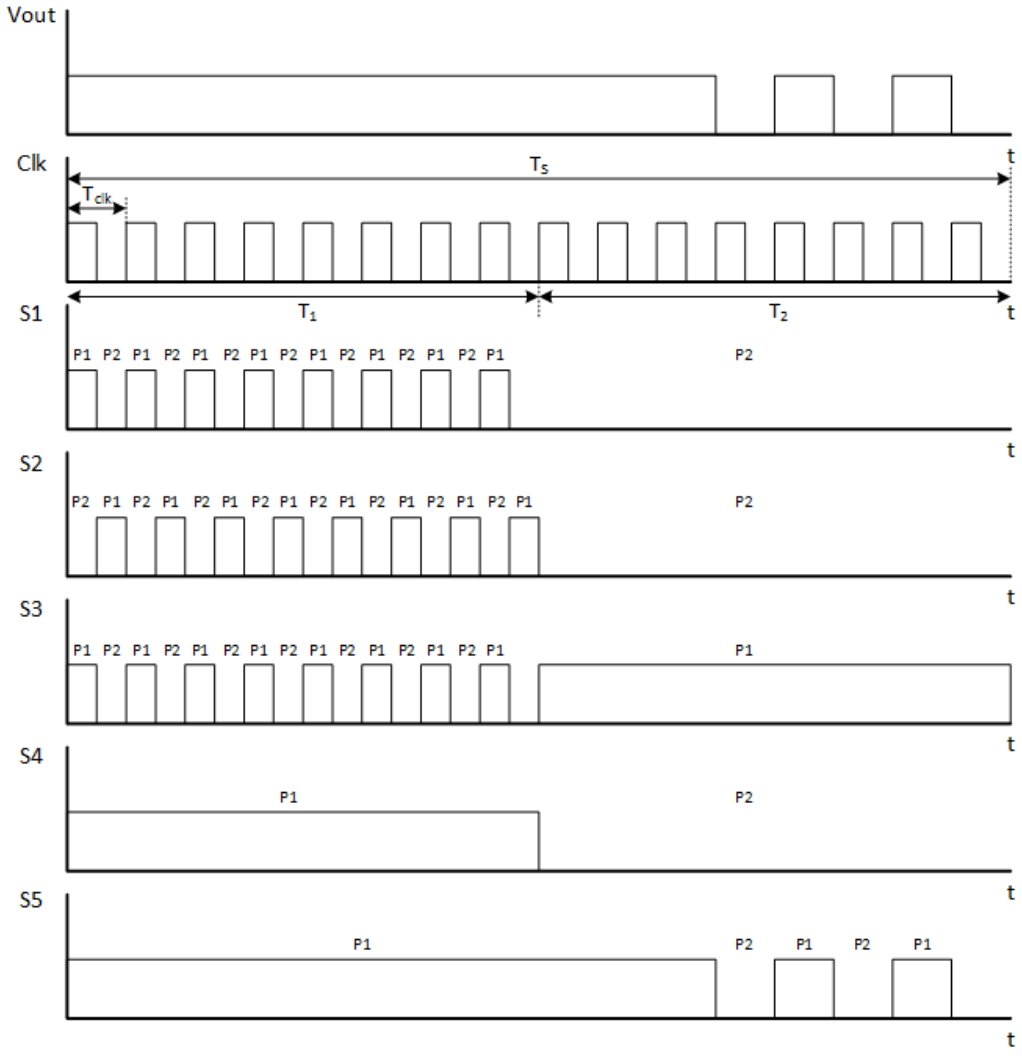


Figura 2.4 Diagrama de tiempos del circuito propuesto.

En la figura 2.4 se muestra de nuevo el diagrama de tiempos del circuito propuesto con el objetivo de facilitar la explicación del circuito de polarización del sensor. En él, además de los interruptores S_1, S_2, y S_3 que forman parte del circuito de polarización del sensor se muestran también los interruptores S_4 y S_5. Los números que aparecen sobre las señales de disparo de los interruptores representan la posición de los mismos durante la duración de dicha señal.

El interruptor S_4 se encarga de cambiar la entrada del integrador entre el circuito de polarización del sensor durante T_1 y el DAC durante T_2. Por su parte S_5 es el interruptor del DAC. En caso de que la salida V_{out} sea positiva dicho interruptor pasa a la posición 1, de forma que la tensión en el integrador disminuye con una pendiente $-\frac{V_r}{C_f R_1}$.

Por otro lado en el caso contrario, es decir cuando la salida V_{out} es negativa, el interruptor S_5 pasa a la posición 2, de forma que el integrador se carga a V_r a través de R_1, y por tanto su tensión aumenta con una pendiente $\frac{V_r}{C_f R_1}$.

En el caso de los interruptores del circuito de polarización del sensor S_1, S_2 y S_3, permanecen sin conmutar en la posición 2 en el caso de S_1 y S_2, y en la posición 1 en el caso de S_3 durante todo T_2. Esto es debido a que durante T_2 el circuito de polarización no se usa, ya que el integrador está conectado al DAC.

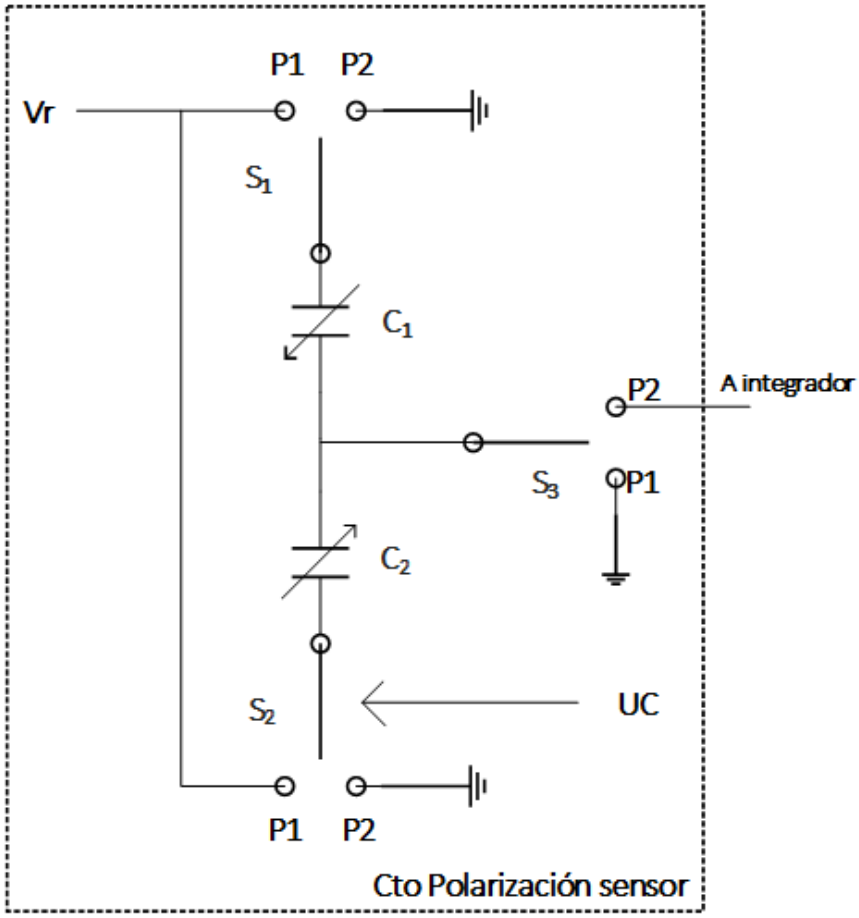


Figura 2.5 Detalle circuito de polarización del sensor.

Por su parte durante T_1 la operación de los interruptores del circuito de polarización del sensor es similar a la presentada en [15]. Cuando el reloj está a nivel alto, S_1 y S_3 pasan a la posición 1, mientras que S_2 pasa a la posición 2. De esta forma C_1 se carga a V_r, mientras que C_2 se descarga a tierra. La carga en el condensador C_1 viene dada por la siguiente expresión.

$$Q_{C1} = C_1 V_r \ [29]$$

Por su parte cuando el reloj pasa a nivel bajo, los interruptores S_1 y S_3 pasan a la posición 2, mientras que S_2 pasa a la posición 1. De esta forma C_1 se descarga a tierra, mientras C_2 se carga a V_r. Por tanto la carga en C_2 viene dada por:

$$Q_{C2} = C_2 V_r \ [30]$$

En el caso de que C_1 sea mayor que C_2, C_1 se descarga primero con la corriente que carga C_2. Una vez C_2 está completamente cargado C_1 se descarga a través de C_f. De esta forma se transfiere a C_f una carga Q_f igual a:

$$Q_f = -(Q_{C1} - Q_{C2}) = -\big((C_1 - C_2)V_r\big) = -2\Delta C \ V_r \ [31]$$

Por tanto y debido a que C_f está conectado a la entrada inversora del amplificador operacional, la salida del integrador V_{int}, aumentará en escalones de valor:

$$V_r \frac{2\Delta C}{C_f}$$

Por otro lado en el caso de que C_2 sea mayor que C_1, C_2 se carga primero a través de la corriente de descarga de C_1. Una vez C_1 se ha descargado completamente, C_2 termina de cargarse a través de C_f. Por tanto la carga transferida a C_f, Q_f, tiene el siguiente valor:

$$Q_f = (Q_{C2} - Q_{C1}) = \big((C_2 - C_1)V_r\big) = 2\Delta C \ V_r \ [32]$$

Es decir, la salida del integrador V_{int} disminuirá en escalones de valor:

$$-V_r \frac{2\Delta C}{C_f}$$

Mediante esta forma de operación se puede conectar directamente el sensor al integrador. El único problema es que para poder realizar la operación descrita en el apartado anterior, 2.1, y por tanto tener un comportamiento similar a un Sigma-Delta de primer orden, necesitamos descargar mediante una recta y no mediante escalones. Es por ello que no podemos utilizar una operación similar a la descrita en [15] durante T_2 y

en su lugar necesitamos usar un DAC con una resistencia R_1. Es debido a esto que tenemos que añadir el interruptor S_4, para poder variar la topología del circuito de acuerdo a estas necesidades.

2.3 Cálculo de los valores de C_f y R_1.

Empezamos calculando el valor de R_1. R_1 se encarga de marcar la pendiente de descarga del integrador. Hay que asegurarse por tanto que su valor permite descargar por completo el integrador para el caso de mayor tensión en el mismo. Recordando que el valor de cada escalón de tensión T_1 es:

$$V_r \frac{2\Delta C}{C_f}$$

Tenemos que la mayor tensión en el integrador se da cuando ΔC es máximo y tiene el siguiente valor:

$$V_{int(max)} = V_r \frac{2\Delta C_{max}}{C_f} N_1 = V_r \frac{2\Delta C_{max}}{C_f} T_1 f_{clk} \text{ [33]}$$

Donde N_1 es el número de escalones durante T_1, o lo que es lo mismo el número de ciclos de reloj que dura T_1.

Por otra parte la descarga del integrador durante T_2 viene dada por [22] donde V_{ref} es igual a V_r y R es igual a R_1. Por tanto el valor de tensión que el integrador alcanzará al finalizar T_2 es igual a:

$$V_{int(final)} = V_{int(max)} - \frac{V_r}{C_f R_1} T_2 \text{ [34]}$$

Es decir, si queremos que el integrador alcance un valor de tensión de cero voltios al finalizar T_2 para todos los casos posibles, tenemos que asegurarnos que $V_{int(final)}$ es igual a cero cuando V_{int} es igual a $V_{int(max)}$. Esto expresado matemáticamente es:

$$V_{int(final)} = 0 = V_r \frac{2\Delta C_{max}}{C_f} T_1 f_{clk} - \frac{V_r}{C_f R_1} T_2$$

Operando y despejando R_1 obtenemos la siguiente expresión:

$$R_1 = \frac{T_2}{2\Delta C_{max} T_1 f_{clk}} \text{ [35]}$$

Sabiendo que el número de pulsos de reloj durante T_1 es igual a $N_1=T_1f_{clk}$ y que el número de pulsos de reloj durante T_2 es igual a $N_2=T_2f_{clk}$, la expresión [35] nos queda de la siguiente forma:

$$R_1 = \frac{N_2}{2\Delta C_{max} N_1 f_{clk}} \quad [36]$$

Y para el caso de que $N_1=N_2$, la expresión de R_1 se convierte en:

$$R_1 = \frac{1}{2\Delta C_{max} f_{clk}} \quad [37]$$

Expresión que solo depende de la frecuencia del reloj y de la variación máxima del sensor.

Cabe resaltar que el mismo procedimiento se puede seguir para un escalón de tensión negativo, teniendo en cuenta que en ese caso [22] ha de ser multiplicada por menos uno. Realizando las operaciones de nuevo se llega a una expresión igual a [36].

Por otro lado el valor del condensador del integrador se obtiene despejando C_f de la expresión de la tensión máxima en el integrador [33], quedándonos la siguiente ecuación:

$$C_f = V_r \frac{2\Delta C_{max}}{V_{int(max)}} T_1 f_{clk}$$

O lo que es lo mismo:

$$C_f = V_r \frac{2\Delta C_{max}}{V_{int(max)}} N_1 \quad [38]$$

Donde se ha tenido en cuenta de nuevo que $N_1=T_1f_{clk}$.

A partir de las ecuaciones [36] y [38] se pueden calcular los valores de C_f y R_1 para distintos valores de N_1 y N_2. A continuación se muestra una tabla (Tabla 2.1) con los resultados de R_1 y C_f con valores de N_1 y N_2 que varían desde 2 hasta 14, manteniendo constante el número total de pulsos durante cada periodo de muestreo $N_{samp}=N_1+N_2$. Para la elaboración de esta tabla se ha empleado una variación máxima del sensor, $\Delta C_{(max)}$, de un picofaradio. El valor de la tensión de referencia, V_r, y el de la tensión máxima en el integrador se ha escogido de dos voltios. Por último, la frecuencia de reloj escogida ha sido de 2MHz mientras que la frecuencia de muestreo escogida, f_{samp}, es de 125kHz, lo que hace un número de dieciséis pulsos de reloj durante cada periodo de muestreo, N_{samp}.

N_1	N_2	R_1 (MΩ)	C_f (pF)
2	14	1,75	4,00
4	12	0,750	8
6	10	0,417	12
8	8	0,250	16
10	6	0,150	20
12	4	0,0833	24
14	2	0,0357	28

Tabla 2.1 Valores de R_1 y C_f para distintos valores de N_1 y N_2. Para la realización de la tabla se han empleado los siguientes datos: f_{clk}=2MHz, f_{samp}=125kHz, V_r=$V_{int(max)}$=2 V y $\Delta C_{(max)}$=1pF.

Como se puede observar en los resultados de la Tabla 2.1, a medida que disminuimos N_1 el valor del condensador C_f requerido disminuye, mientras que si aumentamos N_1, necesitamos un valor mayor de capacidad para el condensador del integrador. Es decir cómo se puede observar fácilmente en [38] el valor de C_f necesario para que el conversor funcione correctamente es directamente proporcional al número de ciclos de reloj durante T_1, es decir es proporcional a N_1.

Por otra parte la ecuación [36] apuntaba a que R_1 es proporcional al ratio entre N_2 y N_1. Es por ello que observamos en la Tabla 2.1, que al aumentar el número de pulsos de reloj durante N_2 a la vez que disminuimos el número de pulsos durante N_1 el valor de R_1 aumenta, mientras que si disminuimos N_2 y aumentamos N_1 ocurre lo contrario.

Hay que señalar que si estuviésemos realizando un convertidor de doble rampa convencional, la resolución de dicho convertidor vendría determinada por el número de pulsos de reloj durante T_2, es decir por N_2 (sin tener en cuenta las limitaciones físicas de los componentes). Es por ello que podríamos aumentar la duración de T_2 para aumentar la resolución del conversor a costa de un mayor tiempo de conversión. Además podríamos llevar a cabo este cambio manteniendo la misma T_1, lo cual disminuiría el aumento del tiempo de conversión, y nos permitiría mantener la misma C_f, a costa de una mayor R_1.

Del mismo modo, si quisiéramos mantener el mismo tiempo de conversión a la vez que aumentamos la resolución del conversor, podríamos aumentar el número de ciclos que dura T_2 a la vez que disminuimos el número de ciclos que dura T_1, de forma similar a lo que se ha hecho en la tabla 2.1. De este modo conseguimos aumentar la resolución del conversor, a la vez que reducimos el valor de C_f, a costa de necesitar una R_1 mucho mayor que en el caso anterior.

En el caso del sistema propuesto, al tratarse de un sistema que se comporta como un Sigma-Delta, la resolución de nuestro conversor vendrá determinada sobre todo por la relación señal a ruido, SNR. Es decir por lo bien que pueda nuestro circuito enviar el ruido de cuantificación a altas frecuencias. Es por ello que no nos merecerá la pena tanto aumentar la resolución de nuestro conversor por los métodos antes descritos, y por ello emplearemos unos valores de N_1=N_2=8, lo cual da un resultado razonable para C_f y R_1. Los valores concretos empleados se especificarán más adelante.

2.4 Sensor *Fully Differencial*

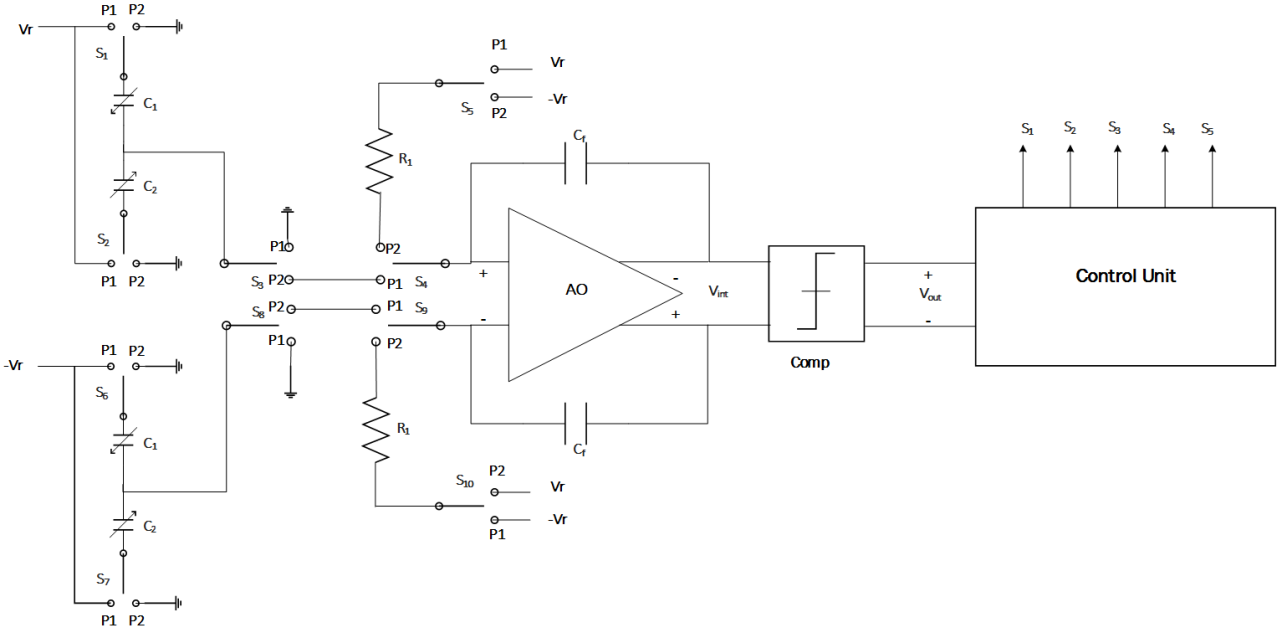


Figura 2.6 Esquema del CDC propuesto en modo *fully differential*.

La Figura 2.5 muestra la implementación en modo *fully differential* del circuito presentado en la Figura 2.1. Las implementaciones de circuitos en modo completamente diferencial (*fully differential*), tienen entre sus ventajas que ayudan a cancelar offsets y a reducir el ruido. El coste de esto es la duplicación de circuitos necesarios, y por tanto la realización de circuitos que ocupan una mayor área y tienen un mayor consumo.

Además en el caso del sensor de tipo *fully differential* el valor de la tensión máxima en el integrador, $V_{int(max)}$ es el doble que el que se obtiene en una implementación de tipo *single ended*, debido a que tenemos dos sensores MEMS. Por tanto la expresión [33] se convierte para el caso de una implementación *fully differential* en:

$$V_{int(max)} = V_r \frac{4\Delta C_{max}}{C_f} N_1 = V_r \frac{4\Delta C_{max}}{C_f} T_1 f_{clk} \text{ [39]}$$

Así mismo la pendiente de descarga del integrador durante T_2 se ve multiplicada por dos debido a que tenemos dos DACs. Por tanto para calcular la R_1 que tenemos que usar en cada uno de los DACs del circuito *fully differential* realizamos el mismo procedimiento que en el apartado 2.3 con las nuevas expresiones de $V_{int(max)}$ y la pendiente de descarga:

$$V_{int(final)} = 0 = V_r \frac{4\Delta C_{max}}{C_f} N_1 - 2 \frac{V_r}{C_f R_1} \frac{N_2}{f_{clk}}$$

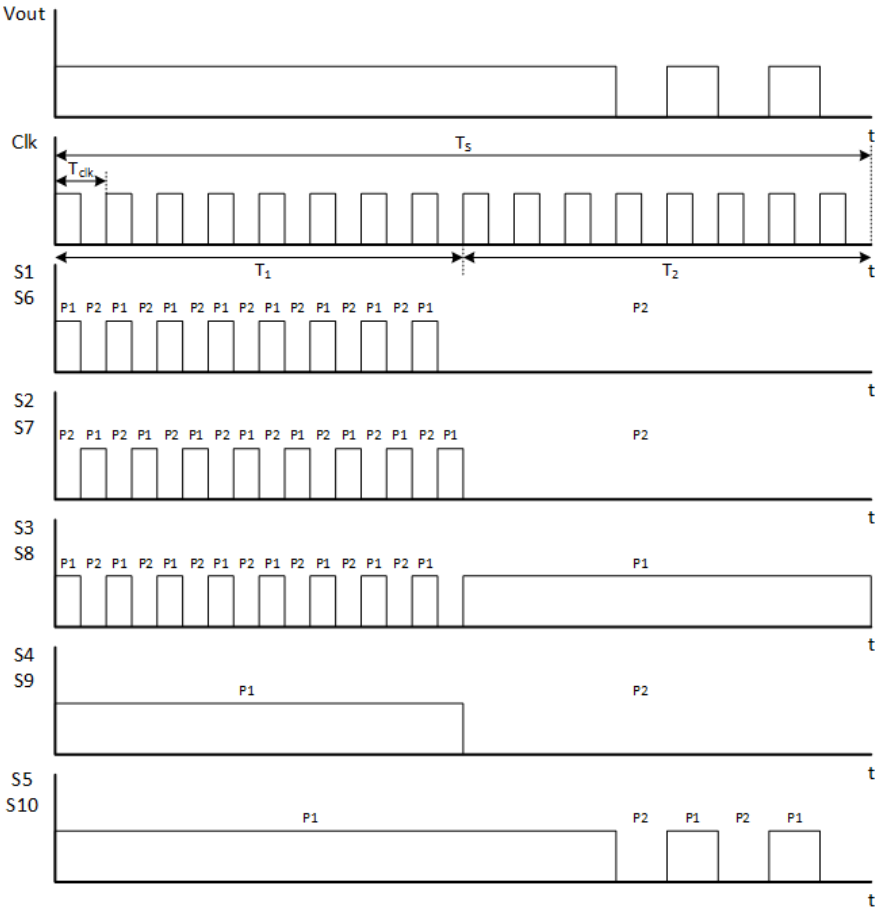


Figura 2.7 Cronograma mostrando las señales de disparo de los interruptores del circuito de la Figura 2.6

Y despejando R_1 obtenemos el mismo resultado que en [36]

$$R_1 = \frac{N_2}{2\Delta C_{max} N_1 f_{clk}} \; [40]$$

Por su parte despejando C_f de [39] obtenemos la siguiente expresión

$$C_f = V_r \frac{4\Delta C_{max}}{V_{int(max)}} \; N_1 [41]$$

Donde vemos que el valor de C_f que hemos de emplear es el doble que el necesario para una implementación *single ended*. Ello se suma al hecho de que hay que usar dos condensadores en el integrador.

En la Figura 2.7 se muestra el cronograma del circuito de la Figura 2.6. Como se puede observar es similar al cronograma asociado al circuito *single ended* mostrado en la Figura2.1

A continuación se va a proceder a modelar el sistema para su implementación y simulación en Matlab Simulink.

2.5 Modelado del sistema

En este apartado se va a modelar el sistema de forma matemática con el objetivo de obtener un modelo en Matlab simulink. De esta forma podremos simular el sistema en Matlab, comprobando su funcionamiento antes de pasar a simular el sistema en Cadence.

Lo primero que modelaremos será el circuito de polarización del sensor. Para ello vamos a recordar brevemente cómo funcionaba dicho circuito. El circuito de polarización del sensor se muestra en la Figura 2.4.

El funcionamiento del circuito era el siguiente. Cuando la señal de reloj tenía un valor alto, cargábamos C_1 a V_r. Para ello poníamos los interruptores S_1 y S_3 en la posición 1 y el interruptor S_2 en la posición 2. Por otro lado cuando la señal de reloj pasaba a nivel bajo, todos los interruptores cambiaban de posición, y descargábamos C_1 a tierra, a la vez que cargábamos C_2 a V_r. La diferencia de la carga entre los dos condensadores C_1 y C_2 se almacenaba en el condensador del integrador C_f.

Teniendo esto en cuenta podemos escribir las siguientes ecuaciones de las corrientes en el circuito. Dichas corrientes se muestran en la Figura 2.8.

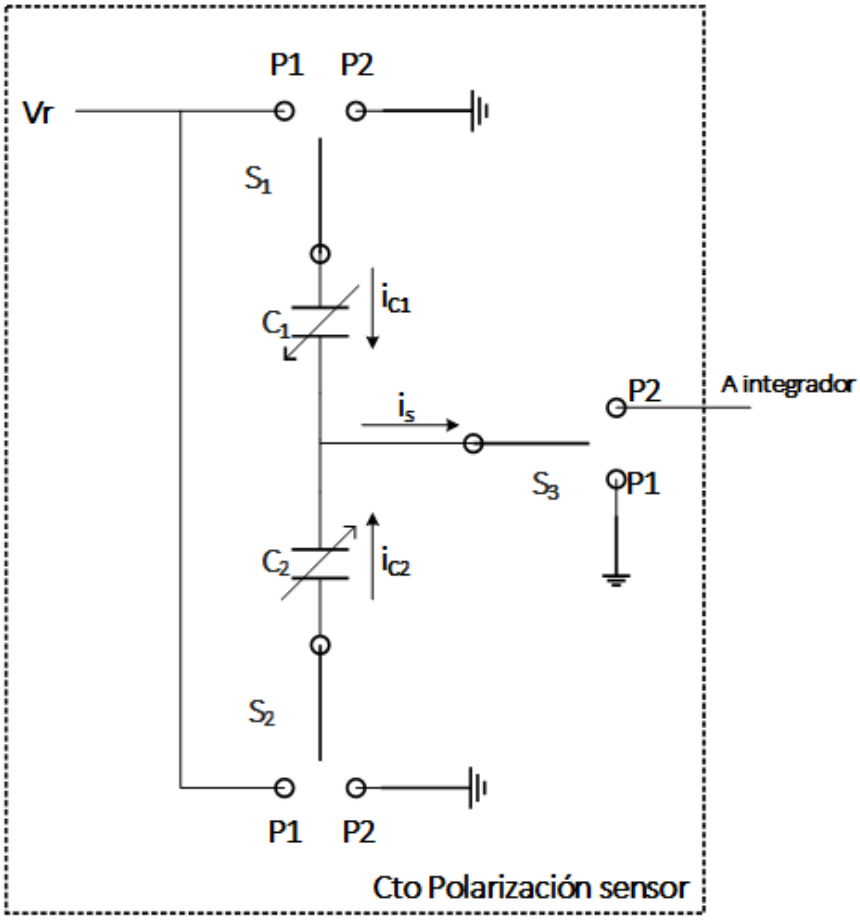


Figura 2.8 Corrientes en el circuito de polarización del sensor.

Para Clk en nivel alto tenemos que:

$$i_s = i_{C1} + i_{C2} \; [42]$$

Como C_2 tiene sus dos terminales a masa

$$i_{C2} = 0$$

Por tanto [42] se convierte en

$$i_s = i_{C1} \; [43]$$

Donde i_{C1} se calcula a partir de la siguiente expresión

$$V_c = \frac{1}{C_f} \int_0^t i_{C1} \; dt \; [44]$$

Que integrando para t y despejando i_{C1} nos queda

$$i_{C1} = C_1 V_r \frac{1}{t} \; [45]$$

Y por tanto, a partir de [44] y [45], podemos calcular i_s cuando Clk tienen valor alto

$$i_s = C_1 V_r \frac{1}{t} \; [46]$$

Donde t es el tiempo que tarda en cargarse el condensador.

Por otro lado cuando Clk tiene valor bajo, tenemos que

$$i_s = i_{C1} + i_{C2} \text{ [47]}$$

Ya que $i_{C2} \neq 0$

En este caso las corriente i_{C1} e i_{C2} son

$$i_{C1} = -C_1 V_r \frac{1}{t} \text{ [48]}$$

$$i_{C2} = C_2 V_r \frac{1}{t} \text{ [49]}$$

Donde i_{C1} tiene signo negativo ya que es corriente de descarga del condensador, mientras que en la Figura 2.8 la corriente i_{C1} mostrada es una corriente que está cargando el condensador. Así mismo t representa el tiempo de carga de los condensadores. Suponemos que dicho tiempo es el mismo.

Por tanto i_s según [47] es la suma de [48] y [49]

$$i_s = -C_1 V_r \frac{1}{t} + C_2 V_r \frac{1}{t}$$

$$i_s = V_r \frac{1}{t}(C_2 - C_1) \text{ [50]}$$

En el caso de un sistema de tipo *fully differential* la corriente i_s se ve multiplicada por dos, ya que hay dos circuitos de polarización del sensor. Por tanto para obtener las expresiones de i_s en un circuito *fully differential* solo tenemos que multiplicar [46] y [50] por dos:

$$i_{shigh} = 2C_1 V_r \frac{1}{t} \text{ [51]}$$

$$i_{slow} = 2V_r \frac{1}{t}(C_2 - C_1) \text{ [52]}$$

El siguiente elemento que tenemos que modelar es el DAC. Dicho elemento consta de una resistencia R_1 y un interruptor. La señal de control de dicho interruptor depende de la salida del comparador síncrono.

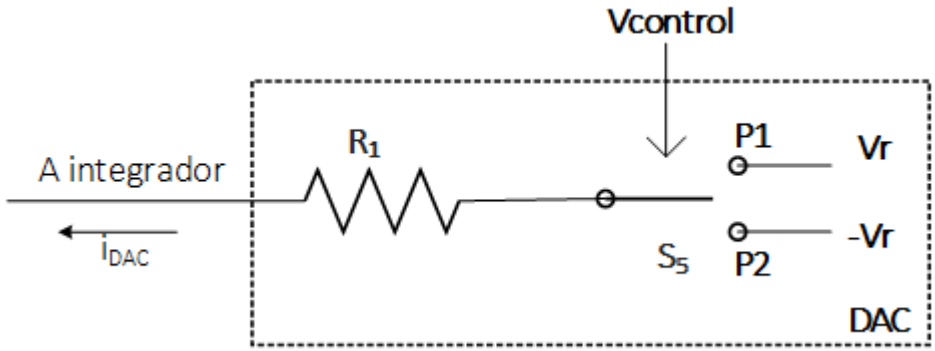


Figura 2.9 Conversor Digital-Analógico (DAC)

Si $V_{Control}$ toma un valor de 1 cuando la salida del comparador tiene nivel alto, es decir cuando V_{int} es positivo; mientras que $V_{Control}$ toma un valor de -1 cuando la salida del comparador tiene un nivel bajo, o lo que es lo mismo cuando V_{int} es negativo, entonces podemos modelar el DAC de la siguiente forma:

$$i_{DAC} = \frac{(-V_r V_{Control})}{R_1} \quad [53]$$

Donde R_1 viene dada por [40].

El siguiente elemento que debemos modelar es el integrador. Para modelarlo más fácilmente tenemos en cuenta la relación que aparece en la Figura 2.10

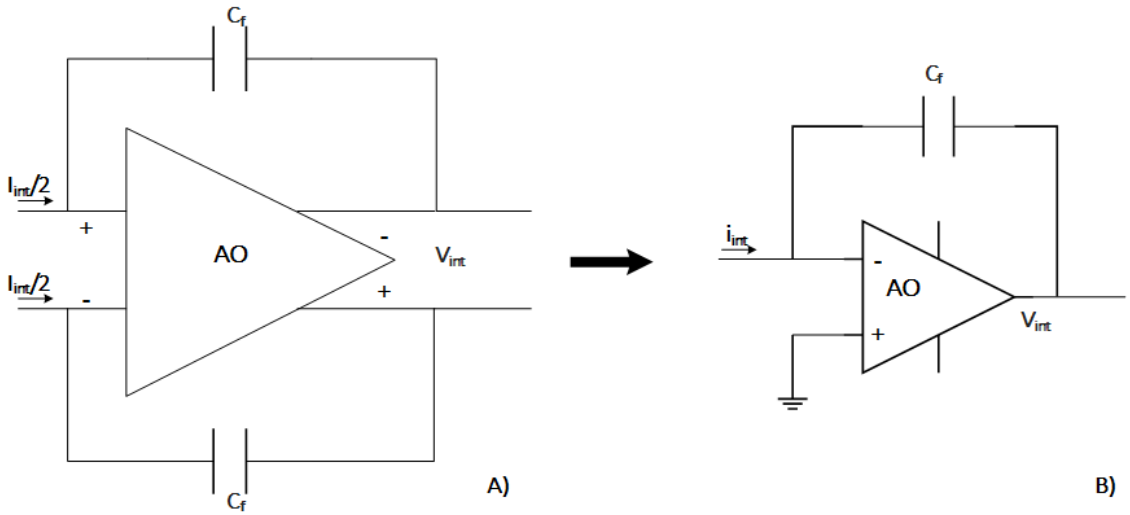


Figura 2.10 Equivalencia entre integrador *fully differential* e integrador *single ended*

Así a partir del circuito B) de la Figura 2.10 podemos modelar el integrador de la siguiente forma. Primero obtenemos la expresión de la impedancia de C_f.

$$Z_{cf}(j\omega) = \frac{1}{C_f(j\omega)}$$

Dicha impedancia se convierte en el plano de Laplace en:

$$Z_{cf}(s) = \frac{1}{C_f s} \ [54]$$

Teniendo en cuenta que en un amplificador operacional ideal la corriente por la entrada es cero y que según el principio de cortocircuito virtual ambos terminales de entrada están a la misma tensión cuando se aplica realimentación negativa (en el caso que nos ocupa ambos terminales están a la tensión del modo común), tenemos que según la ley de Ohm:

$$V_{int}(s) = -i_{int}(s)Z_{cf}(s) \ [55]$$

Por lo que teniendo en cuanta [54] y [55] el integrador se modela mediante la siguiente expresión:

$$\frac{V_{int}(s)}{i_{int}(s)} = -\frac{1}{C_f s} \ [56]$$

Por último solo nos queda modelar el comparador síncrono. Para modelar dicho componente vamos a utilizar un par de bloques de Simulink. El primero de estos bloques es el bloque *Relay*, el cual tiene dos salidas, en nuestro caso 1 y -1. Dicho bloque tendrá 1 como salida cuando su entrada sea mayor que cero, y -1 como salida cuando su entrada sea menor que cero. A la salida del bloque *Relay* se conectará el bloque *Zero-Order Hold* el cual mantendrá el valor de la salida del bloque *Relay*. Pará ello dicho bloque muestreará el valor a la salida de *Relay* con una frecuencia igual a la frecuencia de reloj, Clk. De este modo el conjunto de ambos bloques *Relay* y *Zero-Order Hold* se comportará como un comparador síncrono.

2.6 Implementación del modelo en Simulink y simulación

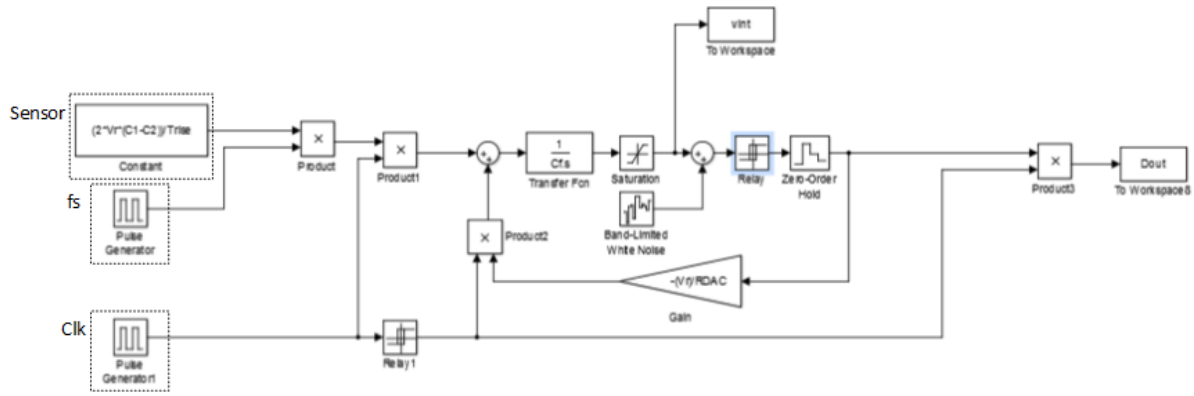


Figura 2.11 Modelo en Matlab Simulink del sistema propuesto en modo *fully differential*

En la Figura 2.11 se muestra el modelo en Simulink del sistema. En él se puede observar que se ha modificado ligeramente el modelado del sistema discutido en el apartado anterior. Básicamente, se ha quitado el signo negativo del modelo del integrador que aparecía en la expresión [56], y en su lugar se ha multiplicado por -1 la expresión del circuito de polarización del sensor [52].

Así mismo es importante comentar también que para modelar el sensor se ha empleado sólo la expresión [52] (multiplicada por -1 como se acaba de explicar). Esto es debido a que el integrador solo ve está corriente durante la fase correspondiente a T_1, ya que la corriente modelada en la expresión [51] (la corriente que pasa por el interruptor S_3 cuando el reloj tiene nivel alto), va a masa.

Por último, a la salida del circuito tenemos un multiplicador y un bloque para adquirir los datos y luego poderlos emplear para realizar los cálculos necesarios en Matlab. Dicho conjunto de multiplicador y bloque de adquisición se encarga de obtener el valor de la salida del circuito durante la fase de desintegración, T_2, que como se ha comentado anteriormente es la que nos interesa para obtener la medida. Dichos datos se muestrean con la frecuencia de reloj de modo que obtenemos un solo valor de salida por cada ciclo de reloj de T_2. Es decir en total obtendremos N_2 datos por cada periodo de desintegración.

Para tener en cuenta solo los datos de T_2, multiplicamos la salida por la inversa de la señal de disparo de S_4, la cual tiene una frecuencia fs. De este modo durante T_1, la señal de disparo de S_4 vale 1, y por tanto su inversa valdrá 0, por lo que durante T_1 obtenemos N_1 ceros que luego eliminaremos en Matlab. Por su parte durante T_2, la señal de disparo vale 0, por tanto al multiplicar la salida por su contraria lo que estamos haciendo es multiplicar la salida por 1, y por tanto al muestrear a la frecuencia de reloj con el bloque *Sink*, obtendremos los datos de interés para realizar la medida.

Para llevar a cabo la simulación, a parte del modelo de Simulink mostrado en la Figura 2.11, se ha empleado también un script de Matlab. Dicho script se muestra en mayor detalle en el Anexo 1. Al ejecutar dicho script lo primero que hacemos es calcular el tiempo T_s dependiendo del sobremuestreo, OSR, que queramos emplear así como del ancho de banda de la señal que estemos midiendo. Del mismo modo se calculan el

número de puntos de la FFT y el tiempo de simulación. Después de ello, calculamos la posición del tono en la FFT

Una vez realizado dicho cálculo, el siguiente apartado del script nos permite establecer los parámetros de nuestro circuito tales como el valor de la resistencia del DAC, llamada aquí RDAC en lugar de R_1; el valor del condensador del integrador, C_f; los valores de C_0 de V_r. Así mismo se indican los valores de la variación del sensor máxima ΔCmax y la tensión en el integrador máxima VCmax.

También hay que especificar el valor Trise, el cual representa el tiempo de carga de los condensadores. En el apartado anterior, apartado 2.5, habíamos llamado a este tiempo simplemente t. El valor de Trise para realizar las simulaciones se ha obtenido a partir de una simulación en Spice donde se carga un condensador de valor nominal similar al del sensor empleado. Dicho valor es de 1 ps.

Este apartado se encarga también de calcular la frecuencia y el periodo de reloj a partir de N_1 y N_2. Además en este apartado se determinan también la frecuencia de chopping, fch; el periodo de chopping, Tch; la potencia del ruido flicker, pFlicker; y la semilla del ruido flicker, Seed_NoiseF. No usaremos estos cuatro últimos paramentos hasta el apartado 3 de esta memoria. No obstante, si usaremos los valores de ditherPower y Seed_Noise para añadir ruido dither como se comentará más adelante.

Por su parte el siguiente apartado del script se encarga de obtener el valor de la medida teórica y de dBin a partir de la variación de la capacidad del sensor en la simulación que vamos a realizar. El valor de dBin se usara para obtener la SQNR a partir de la SNR. Dicha SQNR es el resultado que usaremos para comparar simulaciones con distintas variaciones de sensor. Por ultimo este apartado se encarga también de calcular C_1 y C_2 a partir de la variación de la capacidad del sensor y del valor C_0

Una vez todos los parámetros han sido configurados, el Script se encarga de ejecutar el modelo de Simulink del sistema y adquirir los datos. Para realizar la adquisición de los datos, el Script se encarga primero de modificar el array que contiene los datos provenientes de la simulación, Dout, eliminado los ceros y dándole a dicho array el formato adecuado para realizar a continuación la transformada rápida de Fourier, FFT.

El siguiente paso que realiza el script consiste en realizar y representar la FFT. A partir de los resultados de la FFT, se realiza el cálculo de la relación señal a ruido, SNR, que es la que determina la resolución en los convertidores con modelado de ruido. La SNR se calcula tomando los dos primeros puntos de la FFT como potencia de la señal, y tomando el resto de puntos de la FFT como potencia del ruido y aplicando la siguiente expresión:

$$SNR = 20\log\frac{Potencia\ de\ la\ señal}{Potencia\ del\ ruido}\ [57]$$

A partir de la SNR calculada es posible obtener el número de bits equivalente o ENOB. Dicho número de bits equivalente nos da la resolución final del convertidor teniendo en cuenta el efecto del modelado de ruido. Para obtener el ENOB a partir de la relación señal a ruido empleamos la siguiente expresión:

$$ENOB = \frac{SNR - 1{,}76}{6{,}06} \text{ [58]}$$

Por último sólo nos queda obtener el valor de la medida del sensor. Para ello sumamos los datos almacenados en Dout en grupos de N_2 datos. En nuestro caso como N_2 es igual a ocho, los grupos serán de ocho datos. Una vez sumamos los datos en grupos de ocho, realizamos la media de las sumas de todos los grupos de que disponemos. Esto es equivalente a filtrar la señal a través de un filtro paso bajo. De esta forma obtenemos finalmente el valor que estamos midiendo.

A continuación se muestran los resultados de las simulaciones realizadas con el modelo de simulink mostrado en la Figura 2.11. Dichas simulaciones se han realizado con una OSR de 3000, una frecuencia de reloj de 960 kHz y una N_1 y N_2 de ocho. Primero se muestran dos simulaciones completas, incluyendo sus resultados y la transformada rápida de Fourier. Posteriormente se procederá a mostrar una serie de gráficos y tablas resumen de todas las simulaciones realizadas.

La primera simulación completa que se muestra es para una variación del sensor, ΔC de 0,3pF. Los parámetros empleados en dicha simulación se muestran en la Tabla 2.2. Como ya se ha comentado anteriormente, en las simulaciones se ha añadido también ruido dither. Esta adicción de ruido dither se lleva a cabo a propósito en conversores de tipo Sigma-Delta para que funcionen adecuadamente.

En concreto el dither es necesario debido a que un convertidor sigma-delta es básicamente un oscilador que oscila en torno a una frecuencia. Dicho oscilador es el que se encarga de modular la señal, pero para ello necesita que tanto el oscilador como la señal tengan potencias comparables. Sin embargo, especialmente para señales pequeñas el oscilador suele tener una potencia mucho mayor que la de la señal. Para solucionar esto, se añade ruido dither, de modo que se añade potencia en toda la franja de frecuencias, y de este modo se provoca que tanto la señal como el oscilador tengan potencias comparables.

Parámetro	Valor
Modelo de Simulink	CDC_simulinkABR.slx
Script de Simulink	CDCABR16SimulinkReadout
V_r	1,5 v
$V_{int(max)}$ (VCmax_V)	2 v
OSR	3000
Ancho de Banda de la señal	10 Hz
Frecuencia de reloj	960 kHz
Frecuencia de señal de disparo de S_3 (fs)	60 kHz
Potencia de Dither	10 p
R_{DAC}	521 kΩ
Cf	12 pF
C0	5 pF
ΔC	0,3 pF

Tabla 2.2 Parámetros empleados en la simulación de la Figura 2.12 y Tabla 2.3.

Los resultados de la FFT de la simulación realizada con los parámetros de la Tabla 2.2 se muestra en la Figura 2.12. En ella se observa un comportamiento similar al de un conversor de tipo Sigma- Delta. Se puede observar que la potencia del ruido a bajas frecuencias es menor que a altas. Así vemos que el ruido va disminuyendo su potencia a medida que nos movemos a más bajas frecuencias con una pendiente de unos -20 dB/década.

Si analizamos los resultaos de la simulación vemos que obtenemos una relación señal a ruido de unos 122,89 dB, lo que nos da un numero de bits equivalente de aproximadamente 20,12 bits. De este modo conseguimos medir el valor de continua del sensor, es decir la medida que nos interesa con gran resolución, usando un conversor de doble rampa de sólo tres bits, operado de la forma explicada anteriormente de modo que realizamos el modelado de ruido.

Por último podemos comparar la medida teórica del valor de continua del sensor con la medida del valor de continua del sensor obtenida mediante simulación. Dicha medida teórica ha sido obtenida mediante la siguiente expresión:

$$Medida\ teorica = \frac{\Delta C}{\Delta C_{max}} V_{int(max)} \quad [59]$$

Si comparamos este resultado teórico con el obtenido mediante simulación podemos comprobar que son muy similares, no diferenciándose hasta el cuarto decimal.

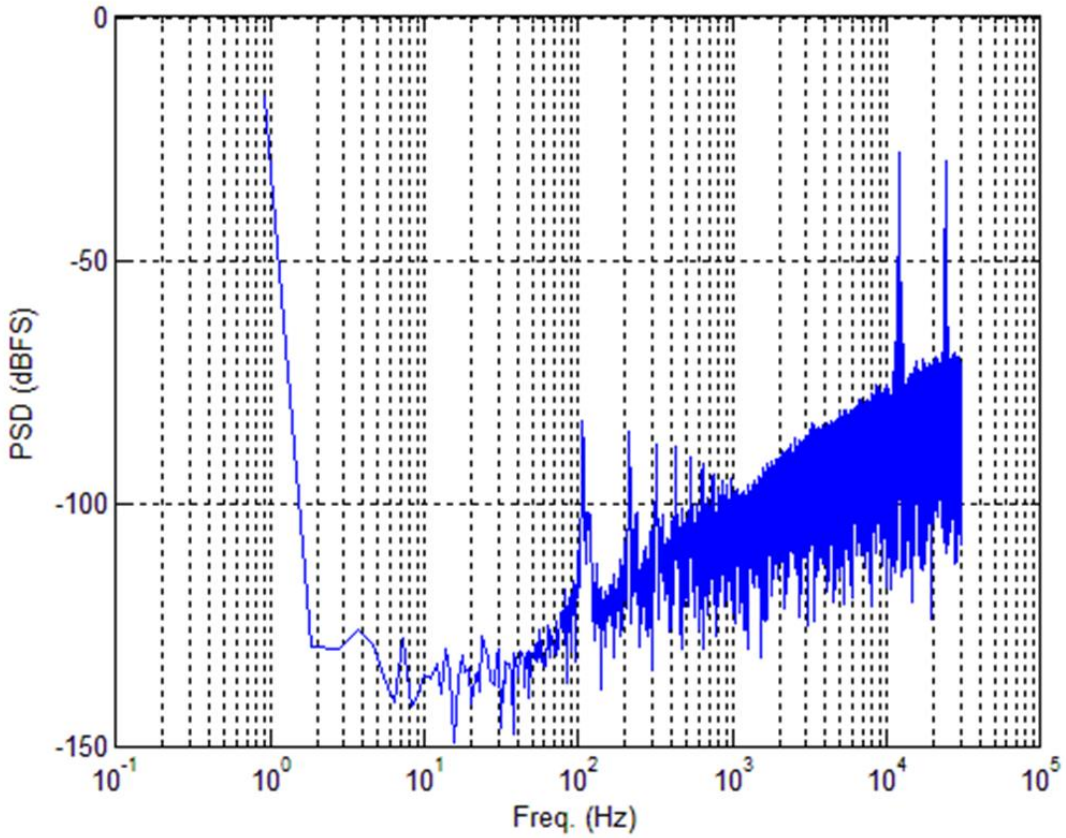


Figura 2.12 FFT de la simulación realizada con los parámetros mostrados en Tabla 2.2.

Parámetro	Resultado
Relación señal a Ruido SQNR	122,89133 (dB)
Número de bits equivalente ENOB	20,12148 (bits)
Medida Teórica	0,600000
Medida Real	0,600189

Tabla 2.3 Resultados de la simulación realizada con los parámetros mostrados en Tabla 2.2.

A continuación se muestran con detalle los resultados de otra simulación realizada con el modelo de Simulink. Dicha simulación se ha realizado con los parámetros que aparecen en la Tabla 2.4.

Parámetro	Valor
Modelo de Simulink	CDC_simulinkABR.slx
Script de Simulink	CDCABR16SimulinkReadout
V_r	1,5 v
$V_{int(max)}$ (VCmax_V)	2 v
OSR	3000
Ancho de Banda de la señal	10 Hz
Frecuencia de reloj	960 kHz
Frecuencia de señal de disparo de S_3 (fs)	60 kHz
Potencia de Dither	10 p
R_{DAC}	521 kΩ
Cf	12 pF
C0	5 pF
ΔC	0.6 pF

Tabla 2.4 Parámetros empleados en la simulación de la Figura 2.13 y Tabla 2.5.

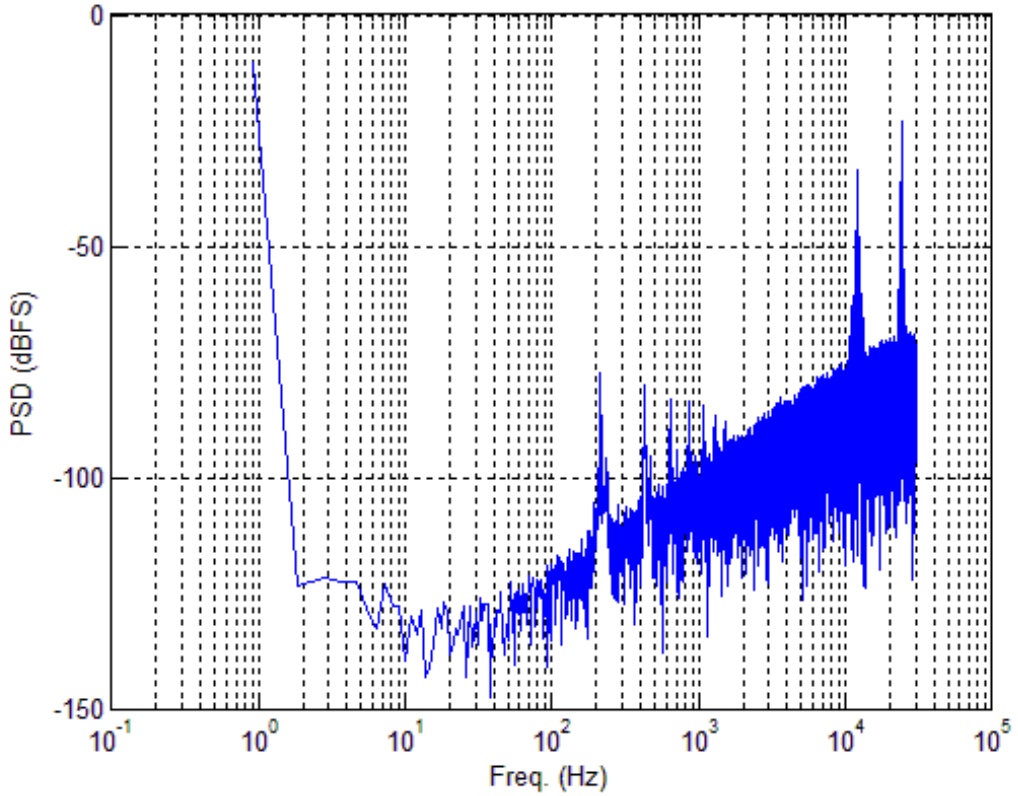


Figura 2.13 FFT de la simulación realizada con los parámetros mostrados en Tabla 2.4.

Parámetro	Resultado
Relación señal a Ruido SQNR	116,87194 (dB)
Número de bits equivalente ENOB	19,12158 (bits)
Medida Teórica	1,200000
Medida Real	1,200371

Tabla 2.5 Resultados de la simulación realizada con los parámetros mostrados en Tabla 2.4.

Como se puede observar en esta segunda simulación se ha obtenido una SNR de 116,872 dB lo que da una resolución de unos 19,122 bits. En cuanto a la medida teórica y la real ambas son iguales hasta el cuarto decimal.

Comparando la FFT de la Figura 2.12 con la de la Figura 2.13, podemos observar que en ambas aparecen dos tonos a altas frecuencias, a aproximadamente 10 kHz y 25 kHz. Así mismo aparecen una serie de tonos a más bajas frecuencias, entre 100 Hz y 1 kHz. Estos últimos tonos aparecen ligeramente desplazados hacia mayores frecuencias en la Figura 2.13. Dichos tonos que aparecen son consecuencia del modelado de ruido.

A continuación se muestran una serie de gráficos con los resultados de las diversas simulaciones realizadas con el modelo de simulink. En el primero de los gráficos, Figura 2.14, se muestran los resultados de la relación señal a ruido del convertidor en función de la variación de la capacidad del sensor. Como podemos observar la SNR se mantiene en torno a 120 dB. El valor máximo de dicha SNR es de 132,01086 dB y se obtiene con una variación de la capacidad del sensor, ΔC, de 0,1 pF. Por su parte la menor SNR la obtenemos para una variación de la capacidad del sensor de 0,85 pF. Dicha SNR es de 113,7653 dB.

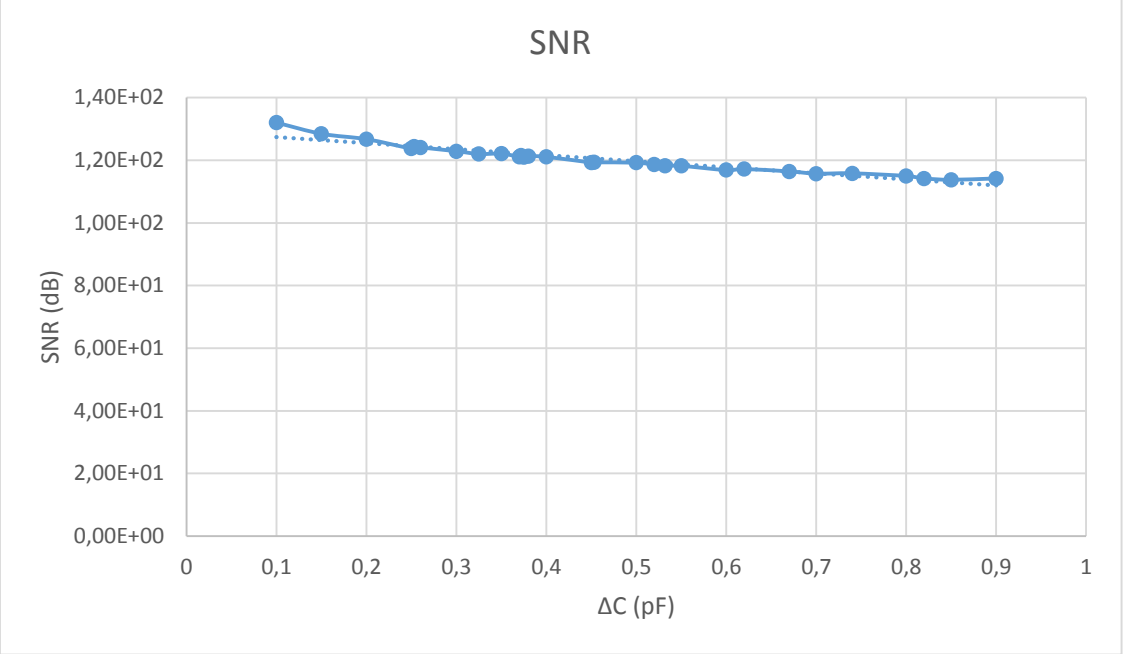


Figura 2.14 Relación señal a ruido SNR.

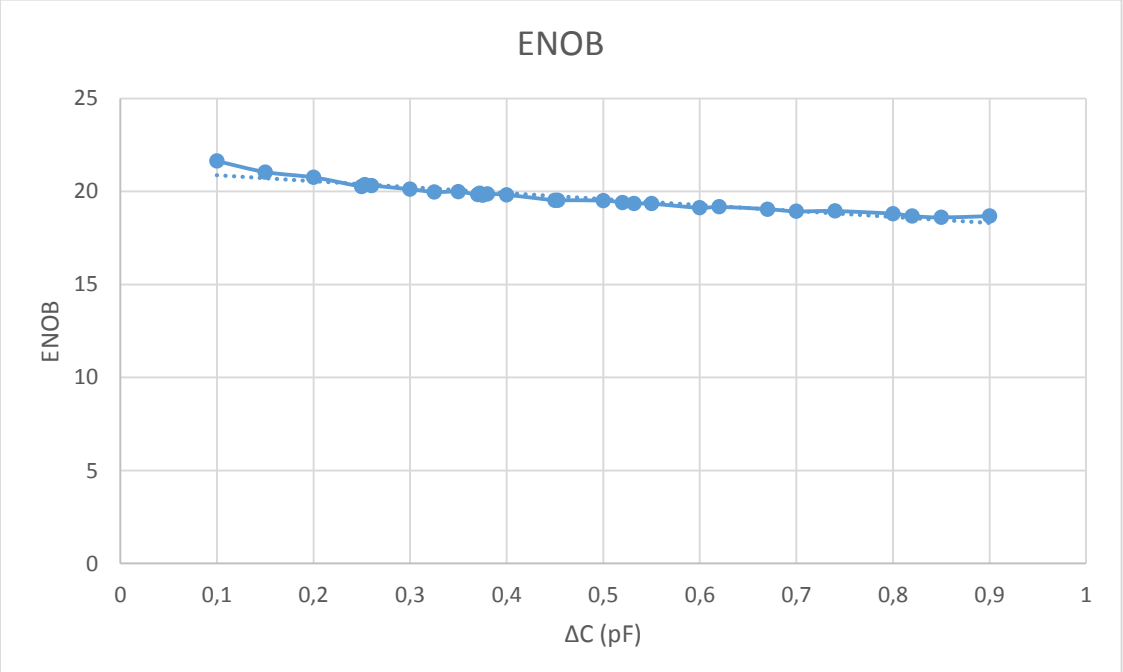


Figura 2.15 Numero de bits equivalente del convertidor ENOB.

En la Figura 2.15 se muestra el número de bits equivalente de nuestro convertidor en función de la variación de capacidad del sensor. Como podemos observar el perfil de esta grafica es similar al de la gráfica de la Figura 2.14. Esto es debido a que como se ha comentado anteriormente el número de bits equivalente se calcula directamente a partir de la relación señal a ruido. Podemos ver que durante todo el rango del sensor, el valor del ENOB se mantiene por encima de 18 bits. El punto donde dicho valor del ENOB es mayor se da para una variación de la capacidad del sensor de 0,1 pF, y es de 21,636 bits. Por su parte el menor valor de ENOB se da para una variación del sensor de 0,85 pF, siendo de 18,606 bits.

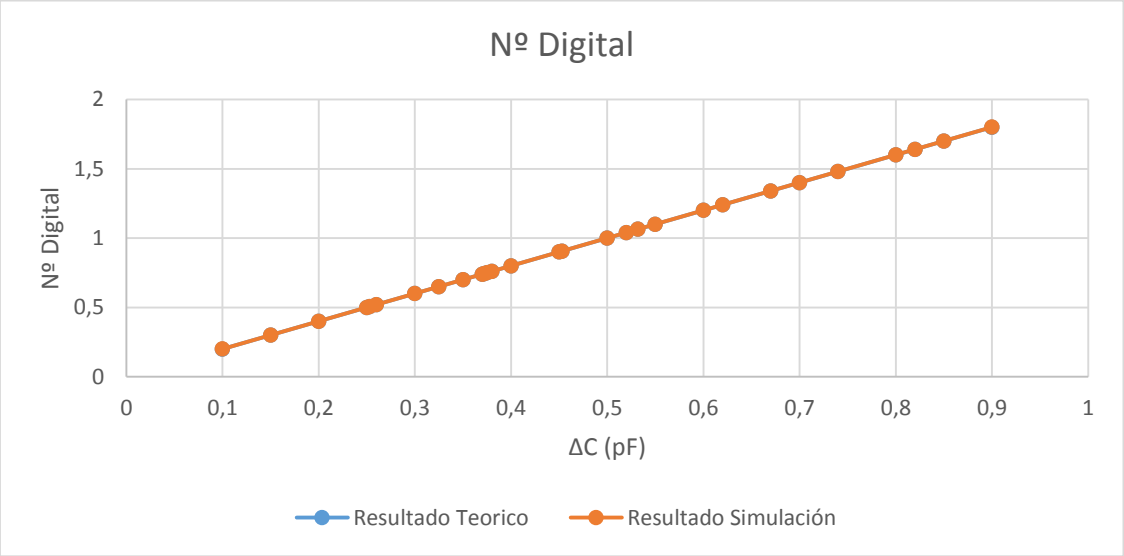


Figura 2.16 Comparación entre la lectura del sensor obtenida mediante simulación y su valor teórico.

Por último Figura 2.16 muestra una comparación entre los resultados de medida obtenido mediante la simulación y los que se obtendrían teóricamente para una misma variación de capacidad del sensor. Como se puede observar en la gráfica de la Figura 2.16, ambos resultados son muy similares y prácticamente se superponen para todo el rango de variación del sensor.

2.7 Simulación del convertidor en Cadence

En este apartado se van a estudiar los resultados obtenidos mediante simulación de un modelo del sistema propuesto implementado en Cadence. Dicho modelo ha sido implementado mediante la herramienta de captura de esquemático Vituoso de Cadence y ha sido simulado mediante el simulador APS.

En la implementación del modelo en Virtuoso se emplea un modelo de amplificador completamente diferencial (*fully differential*) ideal. Por su parte los interruptores se implementan mediante puertas de transmisión realizadas con transistores MOSFET.

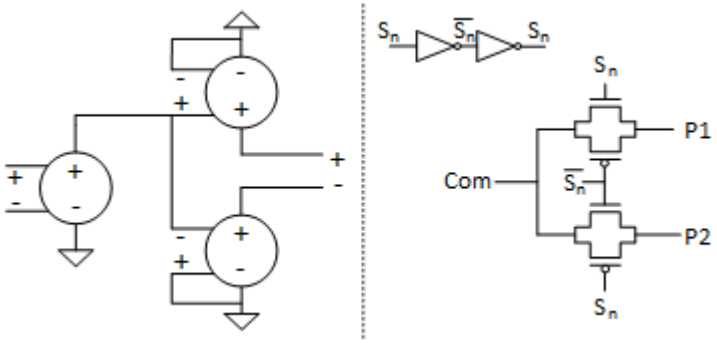


Figura 2.17 Izquierda modelo de amplificador ideal *fully differential*; Derecha implementación de los interruptores del circuito.

Así mismo se ha añadido ruido Dither al modelo de igual manera que se ha realizado en el modelo de Simulink. Dicho ruido se genera mediante una fuente de ruido blanco implementada en Verilog-A.

Una vez se realiza la simulación en APS se exportan los resultados para poder realizar la FFT en Matlab. Dichos datos son exportados en un archivo de texto, el cual es leído por un script de Matlab. Dicho Script se muestra en el Anexo 2. Este script es muy similar al utilizado para simular el modelo de Simulink. Las principales diferencias entre ambos scipts se dan en los apartados de *parámetros Dual-Slope*, *Entrada* y *Cargar archivo de datos*. No obstante el funcionamiento básico del script es similar al explicado en el apartado 2.6.

Así lo primero que hacemos al ejecutar el script es calcular el tiempo de simulación en función del sobremuestreo y el ancho de banda. Calculamos también T_s y el número de puntos de nuestra FFT. A continuación calculamos la posición del tono de la FFT. Continuamos con la introducción de los parámetros del convertidor. Aquí comienzan las diferencias, ya que en este caso los parámetros que tenemos en cuenta son N_1, N_2, la variación del sensor máxima ΔCmax y la tensión en el integrador máxima, VCmax. También calculamos el periodo y la frecuencia del reloj a partir de N_1, N_2 y T_s.

El siguiente paso consiste en introducir el valor de la variación del sensor para la simulación sobre la que vamos a calcular la FFT. A partir de este valor se obtiene la medida teórica mediante la expresión [59] y dBin, que se utiliza para obtener el SQNR, un valor de SNR que podamos comparar entre simulaciones. Una vez hecho esto, lo siguiente es cargar el archivo con los resultados de la simulación. Por último se llevan a cobo los mismos pasos que en el script del apartado anterior para obtener la SNR y demás resultados.

A continuación se procede a mostrar los resultados y las FFTs de dos simulaciones realizadas en Cadence, del mismo modo que se ha hecho para el modelo de Simulink. Así mismo se procederá posteriormente a mostrar un resumen de los resultados de las simulaciones realizadas con el modelo de Cadence.

Al igual que en la simulación en Simulink, la primera simulación que se muestra se ha realizado con una variación de la capacidad del sensor de 0,3 pF. Los parámetros empleados en la simulación se muestran en la Tabla 2.6. Por su parte, la transformada rápida de Fourier de dicha simulación se muestra en la Figura 2.18. Como se puede observar, al igual que para la simulación en Simulink, la FFT muestra un comportamiento de similar al de un convertidor de tipo Sigma-Delta con una caída de la potencia espectral del ruido desde altas frecuencias con una pendiente de -20 dB/década.

Comparando ambas FFTs, la de la Figura 2.12 realizada a partir del modelo de Simulink, y la de la Figura 2.18 realizada a partir del modelo de Cadence, podemos observar que ambas son muy similares. La principal diferencia es que en la FFT de la Figura 2.12 aparecen un mayor número de tonos a frecuencias medias frente a la FFT de la Figura 2.18. Así mismo dichos tonos aparecen desplazados ligeramente hacia frecuencias mayores en la Figura 2.18, y su número es menor. Ello es probablemente debido a diferencias en los motores de simulación.

Parámetro	Valor
Banco de pruebas de Cadence	TB_DualSDiff_dither_ep_v8
Estado simulado	TB_LongSim20160218h
V_r (VDDA)	1,5 v
$V_{int(max)}$ (VCmax_V)	2 v
OSR	3000
Ancho de Banda de la señal	10 Hz
Frecuencia de reloj	960 kHz
Frecuencia de señal de disparo de S_3 (fs)	60 kHz
Potencia de Dither	10 p
R_{DAC}	521 kΩ
Cf	12 pF
C0	5 pF
ΔC	0.3 pF

Tabla 2.6 Parámetros empleados en la simulación de la Figura 2.18 y Tabla 2.7.

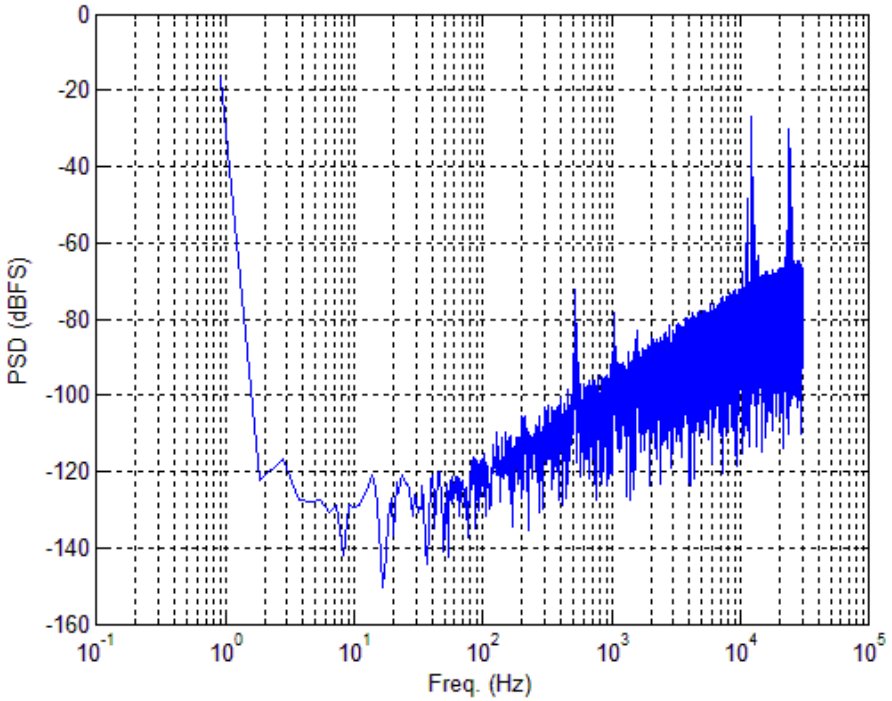


Figura 2.18 FFT de la simulación realizada con los parámetros mostrados en Tabla 2.6.

Parámetro	Resultado
Relación señal a Ruido SQNR	116,4539 (dB)
Número de bits equivalente ENOB	19,05214 (bits)
Medida Teórica	0,600000
Medida Real	0,599136

Tabla 2.7 Resultados de la simulación realizada con los parámetros mostrados en Tabla 2.6.

La Tabla 2.7 muestra los resultados de la simulación de Cadence para 0,3 pF de variación del sensor. Comparando estos resultados con los de la simulación de Simulink para una variación de la capacidad del sensor de 0,3pF, mostrados en la Tabla 2.3 se observa que la SNR obtenida en Cadence es de aproximadamente 6,44 dB menos que la obtenida en Simulink. No obstante dicha diferencia no parece especialmente grande y es posible que se deba a diferencias tanto en los modelos como en los motores de simulación.

Parámetro	Valor
Banco de pruebas de Cadence	TB_DualSDiff_dither_ep_v5
Estado simulado	TB_LongSim20160218e
V_r (VDDA)	1,5 v
$V_{int(max)}$ (VCmax_V)	2 v
OSR	3000
Ancho de Banda de la señal	10 Hz
Frecuencia de reloj	960 kHz
Frecuencia de señal de disparo de S_3 (fs)	60 kHz
Potencia de Dither	10 p
R_{DAC}	521 kΩ
Cf	12 pF
C0	5 pF
ΔC	0.6 pF

Tabla 2.8 Parámetros empleados en la simulación de la Figura 2.19 y Tabla 2.9.

La siguiente simulación que se va a comentar se ha realizado con una variación de la capacidad del sensor de 0,6pF. Los parámetros empleados en dicha simulación se muestran en la Tabla 2.8. Esta simulación es equivalente a la simulación de Simulink cuyos resultados se muestran en la Figura 2.13 y en la Tabla 2.5.

Comparando las FFTs de la Figura 2.19 y la Figura 2.13 observamos que de nuevo la principal diferencia se da en la cantidad de tonos que aparecen a frecuencias medias. No obstante en este caso no parece apreciarse un desplazamiento de los tonos a frecuencias medias de la Figura 2.19 respecto a los de la Figura 2.13.

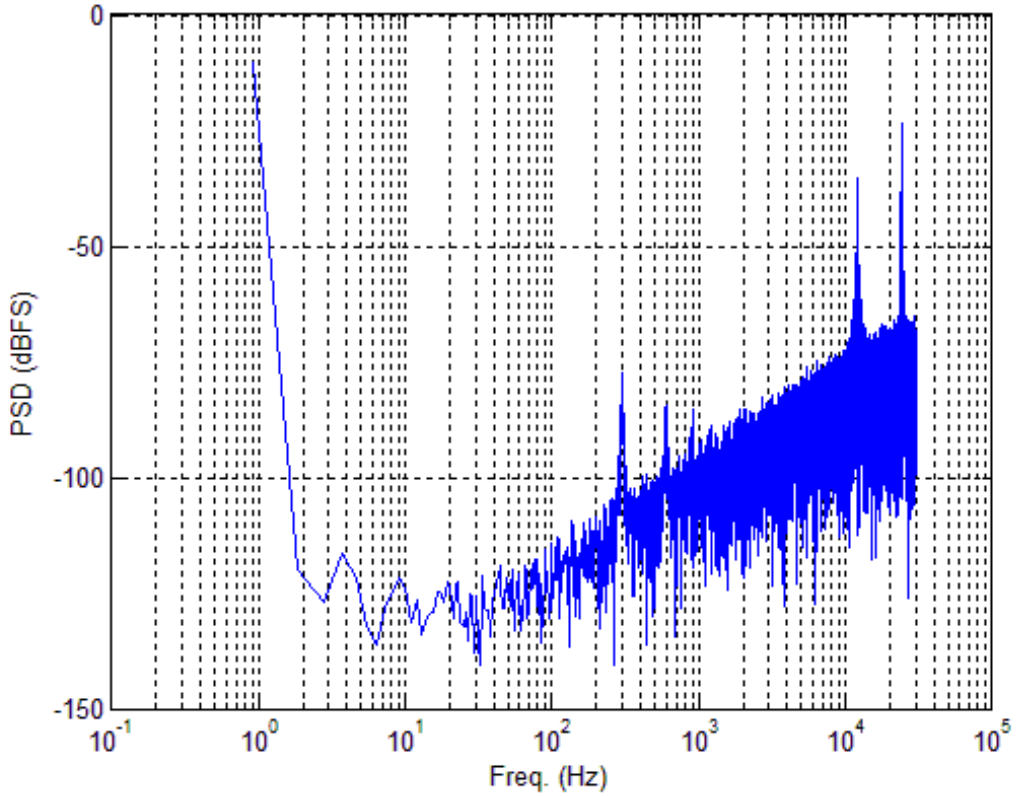


Figura 2.19 FFT de la simulación realizada con los parámetros mostrados en Tabla 2.8.

Parámetro	Resultado
Relación señal a Ruido SQNR	115,0508 (dB)
Número de bits equivalente ENOB	18,81908 (bits)
Medida Teórica	1,200000
Medida Real	1,200500

Tabla 2.9 Resultados de la simulación realizada con los parámetros mostrados en Tabla 2.8.

En cuanto a los resultados mostrados en la Tabla 2.9, se observa que son muy similares a los mostrados en la Tabla 2.5. Es decir los resultados de la simulación de Cadence y de Simulink no difieren de forma importante en este caso. Así ambas SNR son similares, siendo la de Cadence muy ligeramente inferior. Como es lógico lo mismo ocurre para el número de bits equivalente de ambas simulaciones.

A continuación se muestran una serie de gráficas resumen con los resultados de las simulaciones realizadas en Cadence. La primera de estas gráficas (Figura 2.20) muestra los resultados de la relación señal a ruido en función de la variación de la capacidad del sensor. Como se puede observar la SNR es bastante constante a lo largo del rango de variación del sensor. Así tenemos que la mayor SNR se da para una variación del sensor de 0,253 pF y es de 118,9045 dB. Por su parte la menor SNR se da para una variación del sensor de 0,1 pF y es de 111,6747 dB.

Si comparamos estos resultados con los obtenidos mediante el modelo de Simulink, mostrados en la Figura 2.14, se observa que el comportamiento de la SNR en el modelo de Cadence es más constante que en el de Simulink. Esto ocurre principalmente debido a que en el modelo de Simulink la SNR para variaciones del sensor pequeñas es mayor que el obtenido en el modelo de Cadence para las mismas variaciones del sensor. Por su parte para variaciones del sensor mayores ambos modelos se comportan de forma similar.

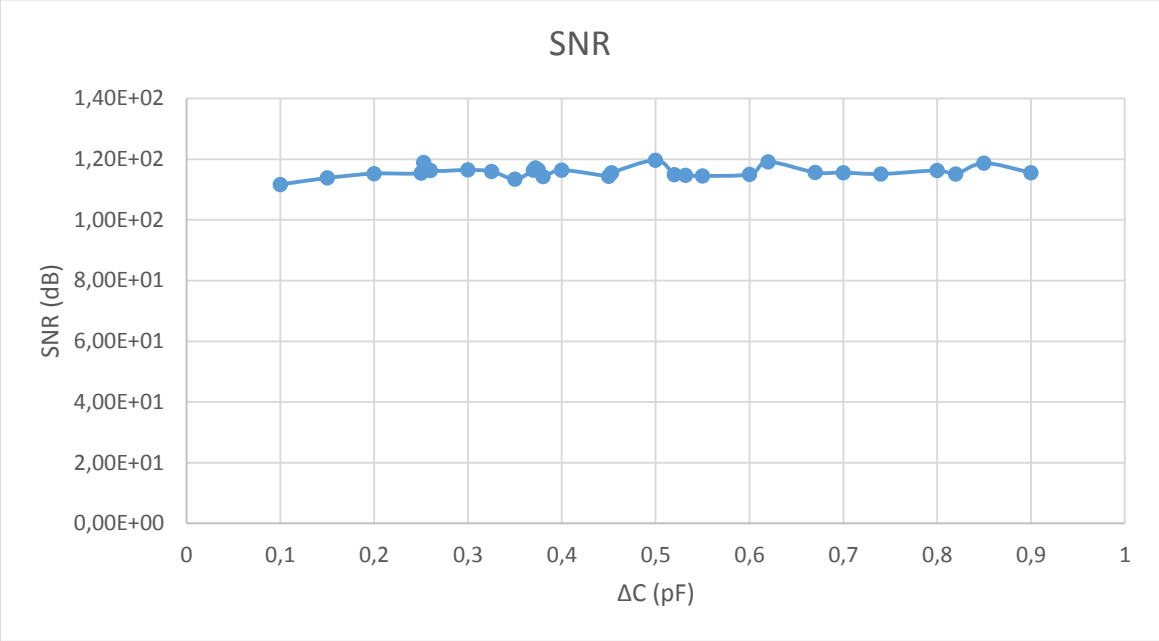


Figura 2.20 Relación señal a ruido SNR.

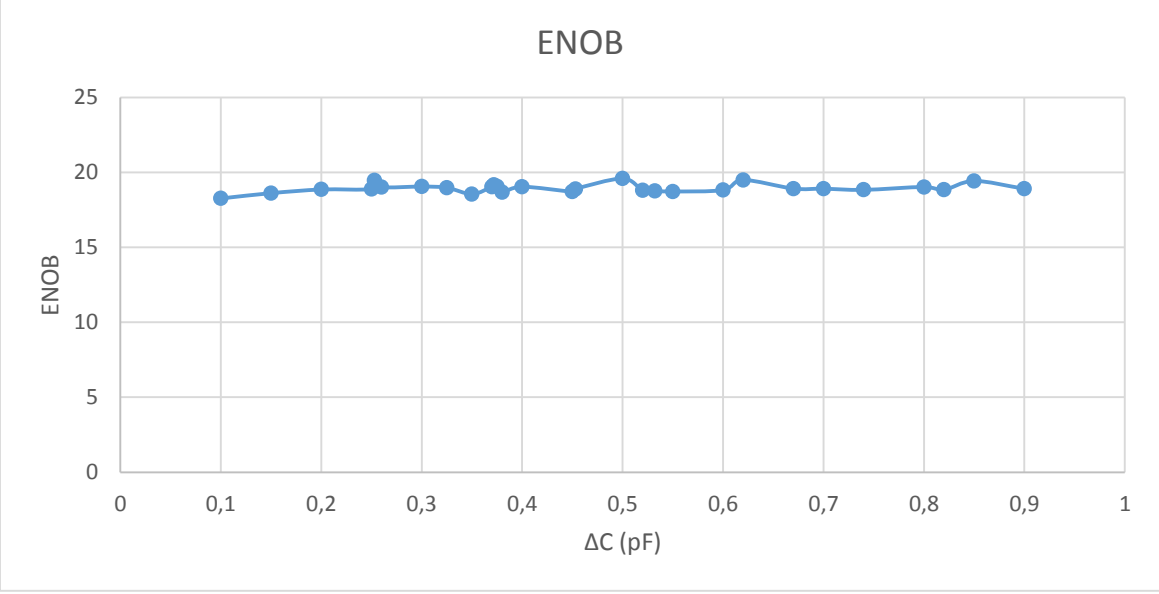


Figura 2.21 Numero de bits equivalente del convertidor ENOB.

En la Figura 2.21 se muestra una gráfica con los resultados del número de bits equivalente del convertidor en función de la variación del sensor. Como se ha comentado anteriormente el perfil de dicha grafica es similar al perfil de la gráfica que muestra los resultados de la relación señal a ruido (Figura 2.20). Para el caso del modelo de Cadence el número de bits del convertidor se mantiene por encima de 18 bits para todo el rango del sensor. El mayor ENOB se da al igual que en el caso de la SNR para una variación del sensor de 0,253 pF y es de 19,4592 bits. Por su parte el menor ENOB es de 18,2583 bits, dándose para una variación del sensor de 0,1 pF.

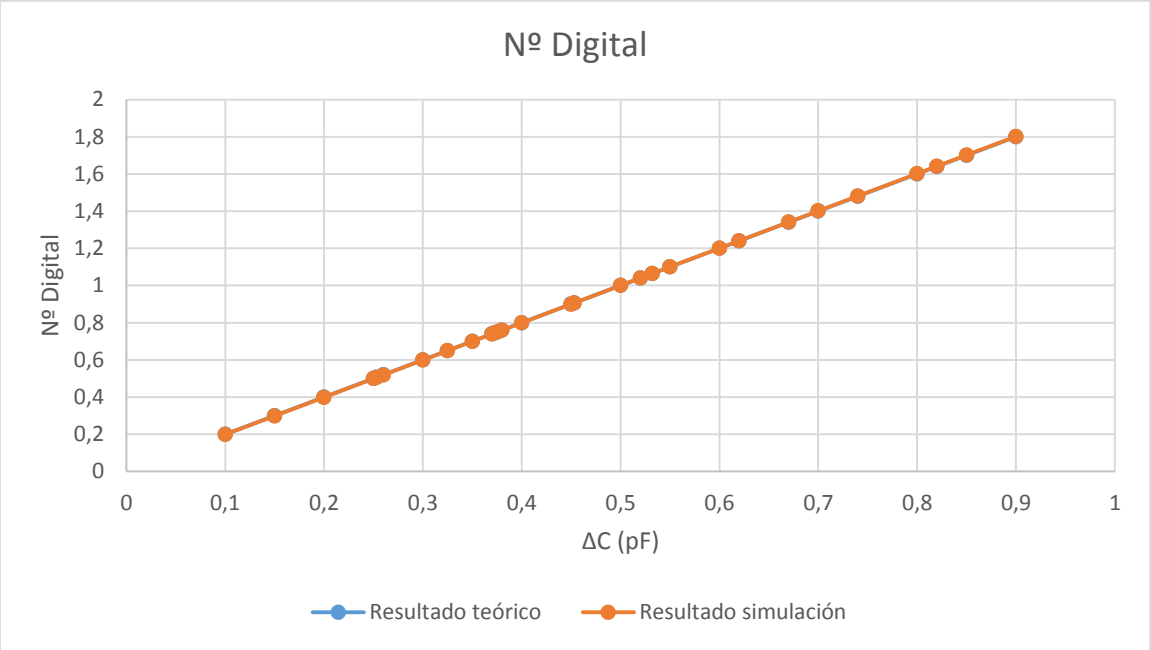


Figura 2.22 Comparación entre la lectura del sensor obtenida mediante simulación y su valor teórico.

Para finalizar la Figura 2.22 muestra superpuestas las gráficas de los resultados teóricos de la medida y los resultados de simulación en función de la variación de la capacidad del sensor. Podemos observar que ambas graficas prácticamente se superponen. Así mismo se observa que el convertidor se comporta de forma lineal respecto a la variación del sensor.

3. Implementación física: Ruido Flicker y Chopping al sistema.

En este apartado se va a tratar el problema del ruido flicker, así como el método escogido para reducir su incidencia. En primer lugar se va a proceder a realizar una pequeña introducción sobre la naturaleza del ruido flicker.

El ruido flicker o ruido 1/f, es un tipo de ruido que afecta de forma importante a sistemas que trabajan con frecuencias bajas. La potencia espectral de dicho ruido tiene un perfil como el mostrado en la Figura 3.1. Como se puede observar el ruido flicker es mayor a bajas frecuencias, desde las cuales va disminuyendo a razón de la relación 1/f, de ahí el nombre de ruido 1/f. Dicho ruido disminuye hasta que su valor es similar al ruido blanco. La frecuencia a la que el ruido flicker tiene un valor similar al ruido blanco se conoce como frecuencia esquina.

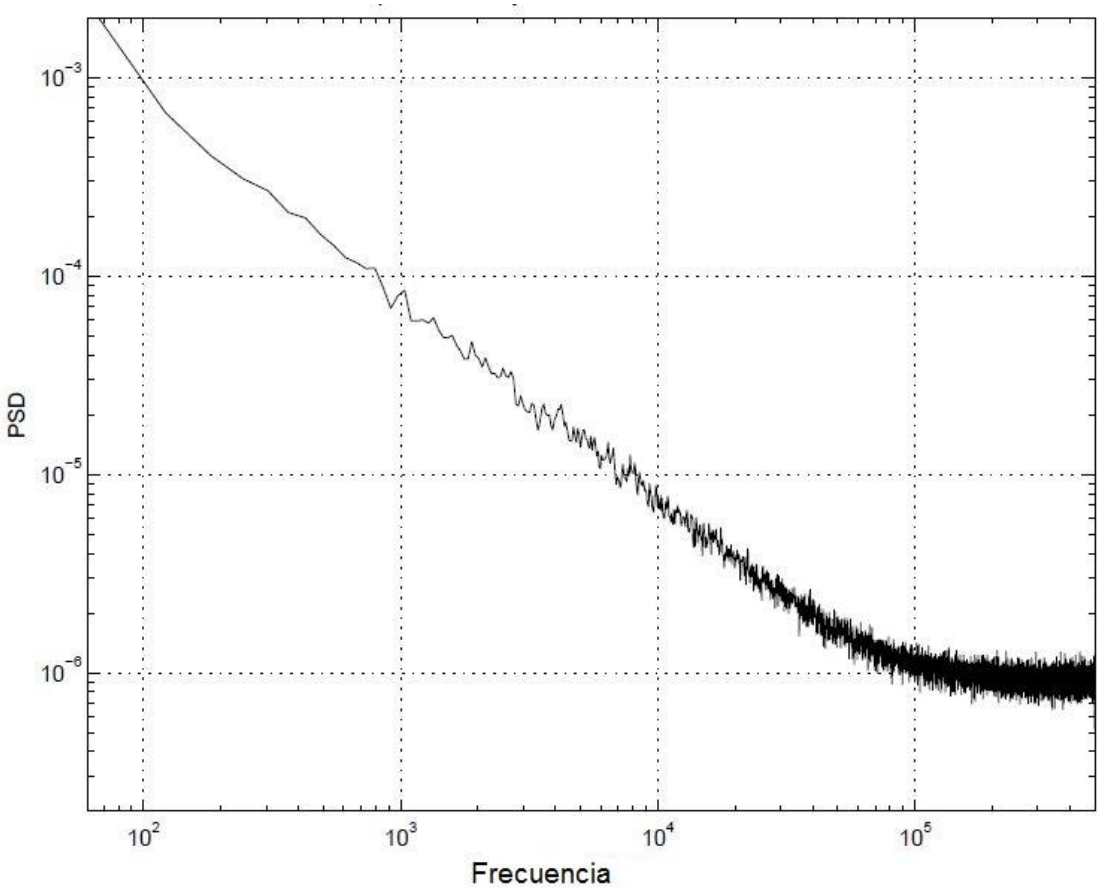


Figura 3.1 Potencia espectral de ruido flicker con ruido blanco superpuesto. De [19].

En circuitos CMOS el origen principal del ruido flicker se encuentra el fenómeno que consiste en la retención y liberación de cargas en la interfaz entre el cristal de silicio y el óxido de silicio, que hace de aislante de puerta de los transistores.

Uno de los métodos fundamentales para reducir la influencia del ruido flicker en un sistema es el método conocido como chopping. Dicho método consiste en hacer pasar la señal a través del circuito que esté generando ruido flicker con polaridades distintas cada ciclo de una señal de reloj, que en nuestro caso será una señal de periodo Tch.

De este modo el chopping modula la señal de interés a la frecuencia de chopping fch, así como a 3fch, 5fch, etc. Por tanto la señal entra al circuito modulada a la frecuencia de chopping. Es entonces cuando el circuito añade el ruido flicker a la señal. Así una vez la señal sale del circuito tiene ruido flicker a bajas frecuencias y la información de interés modulada a frecuencia fch.

El siguiente paso consiste en volver a invertir la polaridad de la señal, es decir de deshacer el chopping. De esta forma, la información o señal de interés que se encontraba modulada a la frecuencia de chopping, es devuelta a su frecuencia original, mientras que el ruido flicker es modulado a la frecuencia de chopping. Es decir el chopping no elimina en sí mismo el ruido flicker, sino que lo envía a altas frecuencias, donde se puede eliminar mediante un filtro paso bajo.

A continuación se va a describir la implementación del chopping en nuestro sistema, así como se va a comprobar su efectividad. Para ello se describirá como se ha modelado el ruido flicker en Matlab Simulink. Dicho modelo se usará en una simulación en la que se mostrará el efecto que tiene el ruido flicker en el sistema.

3.1 Adicción de ruido flicker al modelo de Simulink.

El primer paso para añadir ruido flicker a nuestro modelo de Matlab Simulink es conseguir modelar dicho ruido. Las versiones más modernas de Matlab Simulink cuentan con generadores de ruido flicker. Sin embargo la versión usada para realizar estas simulaciones no dispone de dicho generador. Es por ello que necesitamos generar ruido flicker manualmente a partir de ruido blanco, del que si disponemos un generador, el cual ha sido usado con anterioridad para generar el ruido dither.

Se puede generar ruido flicker a partir de ruido blanco mediante una serie de filtros RC en cascada. Dichos filtros han de contener cada uno un polo y un cero, y los polos y los ceros han de estar equiespaciados a lo largo de la escala logarítmica de frecuencias. En Matlab Simulink dicho filtro será modelado mediante un bloque Zero-Pole. Los polos y los ceros de dicho bloque serán generados mediante un script obtenido de [19].

Dicho script se muestra en el Anexo 3 y en el Anexo 2 bajo el título "*Polos y ceros filtro generación Flicker*". Además de calcular los polos y los ceros del filtro, el script también calcula la ganancia necesaria de dicho filtro generar el ruido flicker adecuadamente.

Este script se ha empleado para generar ruido flicker en nuestro modelo de Simulink, de modo que podamos observar sus efectos. La figura 3.2 muestra el modelo de Simulink con generador de ruido flicker añadido. Para simular este modelo se emplea el script del Anexo 1, el cual ya ha sido explicado en el apartado 2.6 de esta memoria.

En esta ocasión el valor de pFlicker es importante, ya que marca la potencia de la fuente que genera el ruido blanco empleado para generar el ruido flicker. El valor de pFlicker ha sido establecido en $1 \cdot 10^{-23}$. Así mismo el valor de la frecuencia máxima del ruido flicker, fmax se ha establecido en $4 \cdot 10^{5}$ Hz.

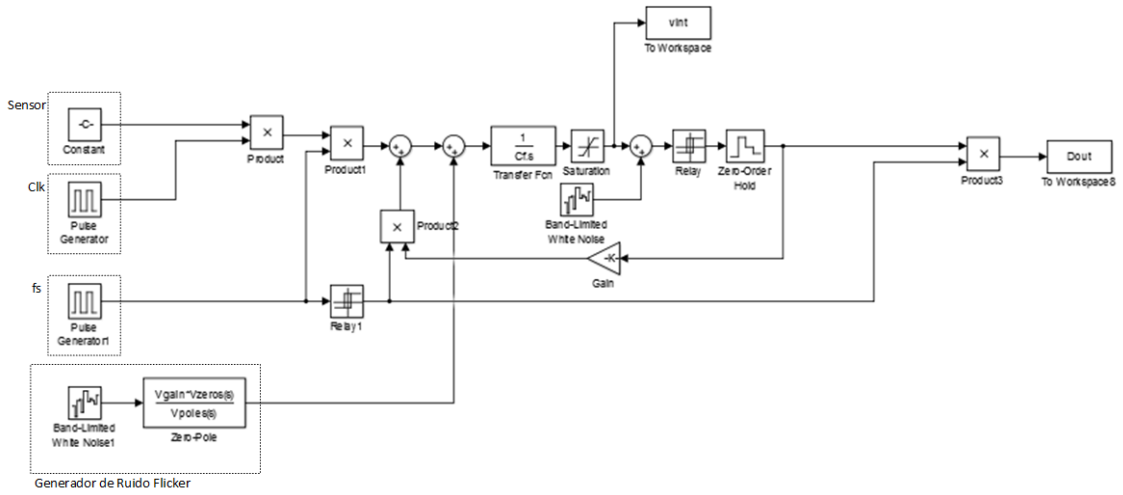


Figura 3.2 Modelo Simulink con Flicker.

Parámetro	Valor
Modelo de Simulink	CDC_simulinkABRFlick.slx
Script de Simulink	CDCABR16SimulinkReadout
V_r	1,5 v
$V_{int(max)}$ (VCmax_V)	2 v
OSR	3000
Ancho de Banda de la señal	10 Hz
Frecuencia de reloj	960 kHz
Frecuencia de señal de disparo de S_3 (fs)	60 kHz
Potencia de Dither	10 p
Potencia Flicker	$1 \cdot 10^{-23}$
Frecuencia Máxima Flicker	$4 \cdot 10^{5}$ Hz
R_{DAC}	521 kΩ
Cf	12 pF
C0	5 pF
ΔC	0,3 pF

Tabla 3.1 Parámetros de la simulación en Simulink con ruido flicker

La figura 3.3 muestra la transformada rápida de Fourier de los resultados de la simulación con ruido flicker realizada con los parámetros mostrados en la tabla 3.1. Como se puede observar aparece ruido a bajas frecuencias con un perfil similar al mostrado en la Figura 3.1. Dicho ruido es el ruido Flicker. A altas frecuencias dicho ruido flicker queda tapado por el ruido de cuantificación que nos hemos llevado a dichas frecuencias al realizar la modulación Sigma-delta.

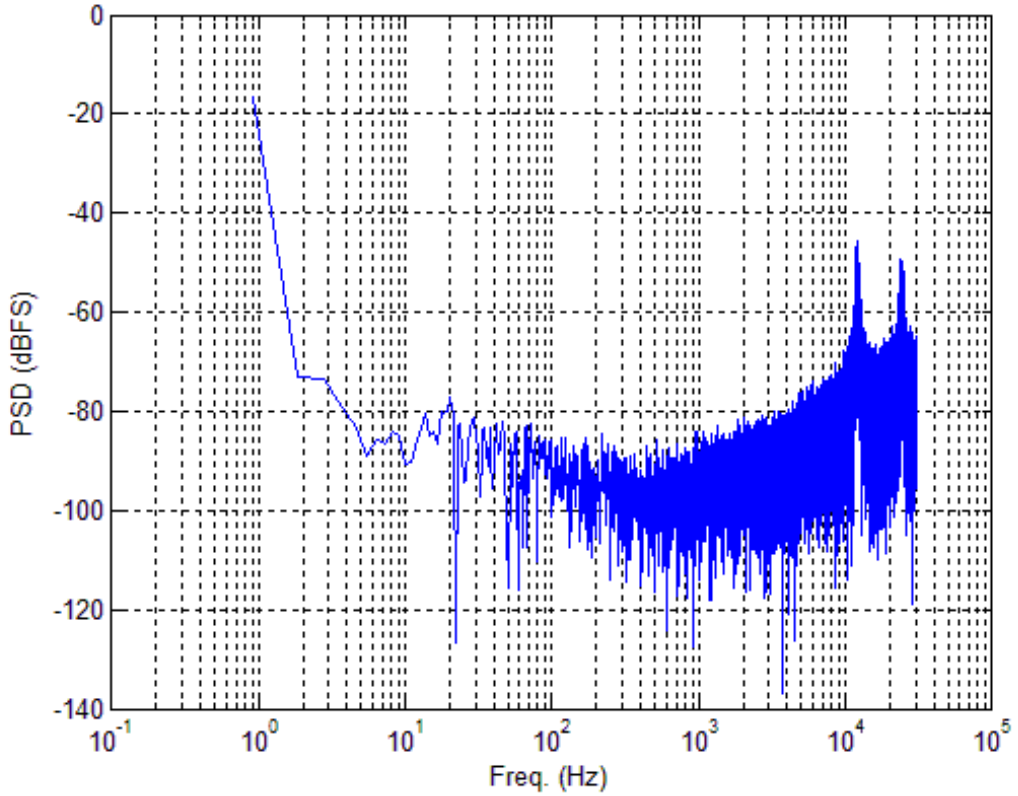


Figura 3.3 FFT de la simulación realizada con los parámetros de la tabla 3.1.

Como podemos observar en la Tabla 3.2 dicha aparición de ruido a bajas frecuencias tiene el efecto de disminuir la relación señal a ruido. Si comparamos el resultado mostrado en la Tabla 3.2 con el mostrado en la Tabla 2.3, que se corresponde a la misma simulación sin ruido flicker, se observa que la SQNR empeora en -52.01814 dB para el caso con flicker.

Parámetro	Resultado
Relación señal a Ruido SQNR	70,87319(dB)
Número de bits equivalente ENOB	11,48059 (bits)
Medida Teórica	0,600000
Medida Real	0,600037

Tabla 3.2 Resultados simulación realizados con los parámetros de la tabla 3.1.

3.2 Sistema propuesto con chopping

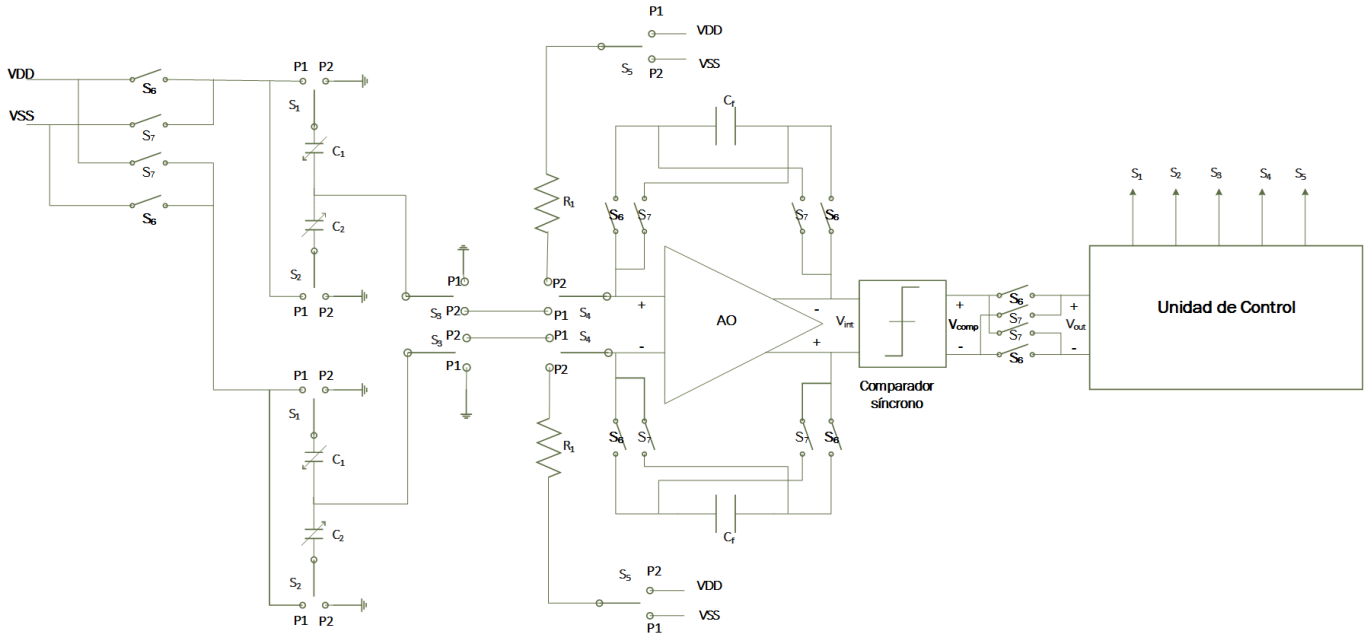


Figura 3.4 Esquema del sistema propuesto con chopping.

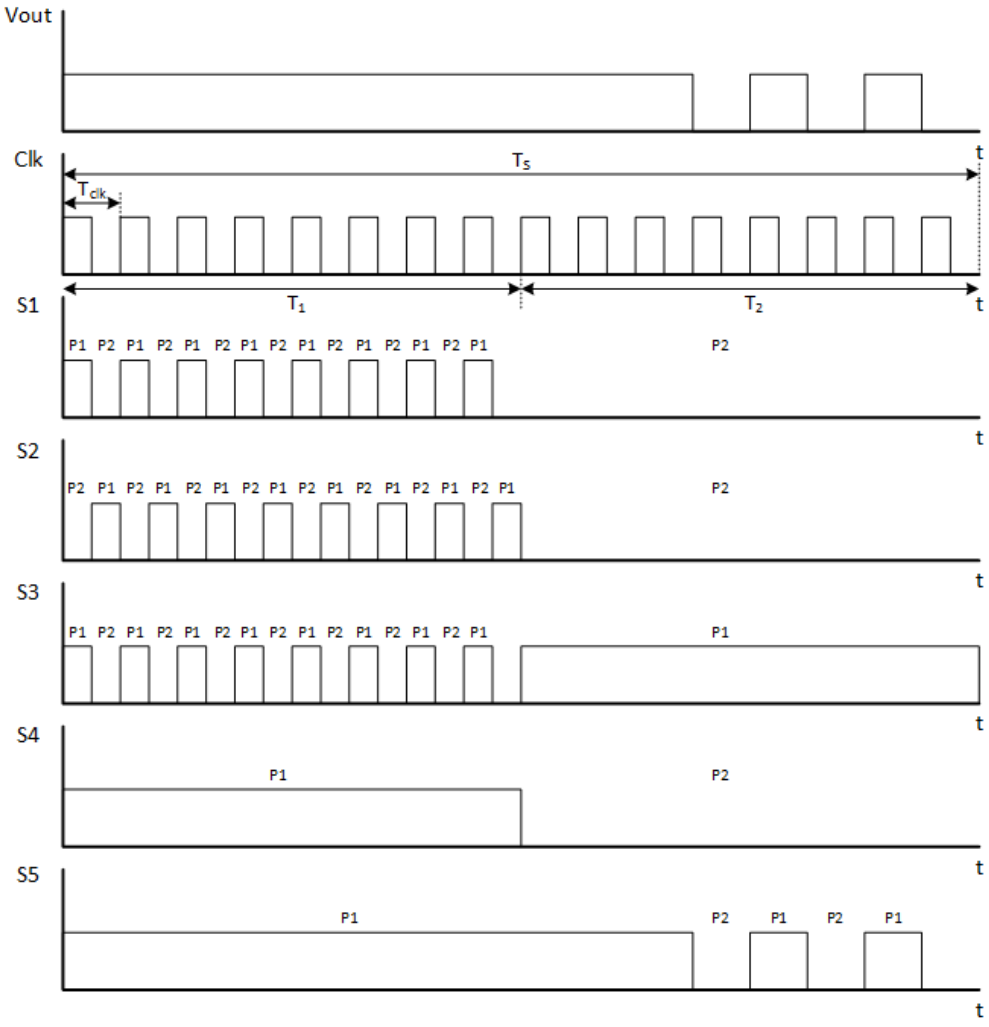


Figura 3.5 Cronograma del sistema mostrado en la Figura 3.4.

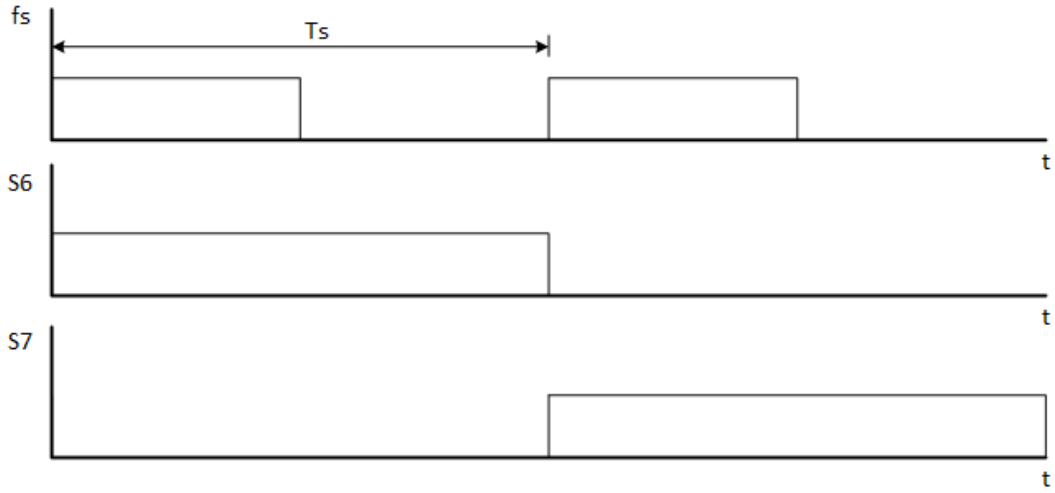


Figura 3.6 Cronograma de las señales de disparo de los interruptores que implementan el chopping en la Figura 3.4.

La Figura 3.4 muestra el esquema del sistema propuesto con chopping. Por simplificar el esquema se han llamado con el mismo nombre todos los interruptores que comparten señal de disparo. Las señales de disparo de los interruptores que se encargan de la operación básica del circuito (esto es la operación que se ha descrito en los apartados 2.1 y 2.2 de esta memoria) se muestran en el cronograma de la Figura 3.5. Dicho cronograma es equivalente a los mostrados en las Figuras 2.1 y 2.7.

Por su parte la Figura 3.6 muestra las señales de disparo asociadas a los interruptores encargados de realizar el chopping. Para realizar el chopping en nuestro circuito lo que hacemos es alimentar cada uno de los sensores con una tensión VDD o VSS durante un ciclo de fs y con la tensión contraria durante el siguiente ciclo de fs.

Así para el sensor ubicado en la parte superior izquierda de la Figura 3.4, usamos durante el primer ciclo del reloj fs la alimentación VDD, es decir la alimentación positiva que en nuestro circuito será de 1,5 V. Para ello cerramos todos los interruptores que aparecen con la etiqueta S_6 durante este primer ciclo de fs.

Por su parte para el sensor ubicado en la parte inferior izquierda de la Figura 3.4 usamos la alimentación con signo contrario, es decir VSS que en nuestro circuito es de 0 v (la tensión común de nuestro circuito es de 750 mV). Esto lo realizamos cerrando también los interruptores asociados con la etiqueta S_6. Hay que recordar cómo se ha dicho al principio de este apartado que los interruptores que funcionan con la misma señal de disparo han sido etiquetados con el mismo nombre por simplicidad.

Como se puede observar el funcionamiento durante el primer ciclo del reloj fs es similar al funcionamiento del circuito explicado en el apartado 2.4. La diferencia ocurre en el segundo ciclo de reloj de fs. Es entonces cuando alimentamos al sensor superior con VSS en lugar de VDD y al sensor inferior con VDD en lugar de VSS. Para realizar este cambio de polaridad, abrimos los interruptores que van asociados a la etiqueta S_6, mientras que cerramos los interruptores asociados a la etiqueta S_7.

En los ciclos sucesivos de reloj fs repetimos este patrón. Primero alimentamos el sensor superior con VDD y el inferior con VSS cerrando los interruptores asociados a S_6 y abriendo los interruptores asociados a S_7. En el siguiente ciclo realizamos la operación

inversa. Alimentamos el sensor superior a VSS y el inferior a VDD, mediante el cierre de los interruptores asociados a S_7 y la apertura de los interruptores asociados a S_6. Repitiendo este procedimiento durante toda la operación del circuito conseguimos modular la señal del sensor a fs, hacerla pasar por el amplificador donde toma ruido que se ubica a bajas frecuencias, y posteriormente devolver la señal del sensor a la banda de interés mientras que enviamos el ruido del amplificador a altas frecuencias. Además de todo esto continuamos enviando también el ruido de cuantificación del convertidor a latas frecuencias mediante el modulado de ruido.

La demodulación de la señal de interés y el envio a altas frecuencias del ruido flicker se realiza mediante los interruptores ubicados a la salida del comparador síncrono. Dichos interruptores hacen que las entradas del bloque *Unidad de Control* que contiene la parte digital del convertidor vean siempre la misma polaridad. Así la entrada superior de este bloque verá siempre una señal positiva, mientras que la entrada inferior verá siempre una señal negativa.

Para ello durante el primer ciclo de fs, los interruptores asociados a S_6 se cierran de forma que la salida superior del comparador y la salida inferior del comparador están conectadas con la entrada superior y la entrada inferior del bloque *Unidad de Control* respectivamente. Como durante este ciclo de fs, la salida superior del comparador es positiva y la salida inferior es negativa, las entradas superior e inferior del bloque *Unidad de Control* ven, como corresponde, una señal positiva y una señal negativa respectivamente.

Por su parte durante el segundo ciclo de fs, la salida superior del comparador lleva una señal negativa (El sensor superior esta alimentado con la tensión negativa o VSS), mientras que la salida inferior del comparador lleva un tensión negativa. Es por ello que es necesario invertir las conexiones entre ambos bloques para que las entradas del bloque *Unidad de Control* sigan viendo una señal positiva para la entrada superior y una negativa para la inferior. De ello se encarga al cerrarse los interruptores asociados a S_7. De esta forma cuando dichos interruptores se cierran, la salida superior del comparador (negativa en este ciclo) se conecta a la entrada inferior de la *Unidad de Control* (negativa), mientras que la salida inferior del comparador (negativa en este ciclo) se conecta a la entrada superior del bloque digital (positiva).

Es importante destacar un aspecto de la implementación del chopping en el sistema. Para que el sistema realice no solo chopping sino también modulación Sigma-Delta de forma correcta, es necesario asegurarnos que los condensadores C_f ven siempre una señal con la misma polaridad. Es por ello que se han añadido interruptores asociados a la operación del chopping junto a estos condensadores. Dichos interruptores se encargan de que el condensador vea una señal de la misma polaridad durante todos los ciclos de fs. Para ello se encargan de “dar la vuelta” a dichos condensadores durante los ciclos en los cuales se invierten las polaridades de la alimentación de los sensores. Para ello se cierran los interruptores asociados a S_7 que se encuentran junto a estos condensadores a la vez que se abren los asociados a S_6 durante los ciclos en que las alimentaciones están invertidas.

Cabe preguntarse el porqué de este ajuste. La explicación es bien sencilla. Como se ha comentado en el apartado 2.1, el sistema propuesto realiza modelado de ruido almacenado el valor del error de cuantificación en el condensador C_f al final de un ciclo de fs, y restando dicho valor al valor de tensión máximo del condensador en el siguiente ciclo de fs.

Ello tiene un efecto similar a un sigma delta convencional en el cual cómo podemos recordar íbamos restando el error al valor a convertir o medir, de forma que aproximábamos el resultado. Sin embargo si no nos aseguramos de que el condensador vea la misma polaridad en cada ciclo de fs, lo que ocurrirá es que en lugar de restar el valor del error en el siguiente ciclo lo sumaremos. Esto se puede observar si en la expresión [26] multiplicamos $V_{int(T1)}(1)$, que de manera general puede representar la medida del sensor, por -1. De esta forma nos queda:

$$V_{int(T1)}(2) = -V_{int(T1)}(1) - Q_e$$

$$-V_{int(T1)}(2) = V_{int(T1)}(1) + Q_e \ [60]$$

Es decir nos da un número negativo que será la suma de la tensión de medida del sensor y el error de cuantificación. Si al desintegrar con una pendiente constante desde $V_{int(T1)}(1)$ nos había quedado un valor de tensión en el condensador de $-Q_e$, al desintegrar con la misma pendiente constante (aunque con distinto signo ya que desintegramos desde un valor negativo) desde un valor de tensión $V_{int(T1)}(1)+Q_e$, la tensión final en el condensador será cero. Para ver esto mejor llamamos V_{des} a la variación de tensión que sufre C_f hasta que el sistema detecta un paso por cero para un valor de tensión máxima en el condensador de $V_{int(max)}$. Así mismo seguimos llamando Q_e a la tensión que queda almacenada en el condensador al final del cilclo fs. De esta forma tenemos que:

$$Q_e = V_{int(max)} - V_{des} \ [61]$$

Si el valor de tensión máxima en el condensador $V_{int(max)}$ es positivo, en el segundo ciclo de fs, dicho valor pasara a ser negativo por efecto del chopping lo que provocará a su vez que V_{des} cambie de signo (ya que estamos desintegrando un valor negativo). Por tanto tendremos:

$$-V_{int(max)} + Q_e + V_{des} = V_{int(max)} - V_{int(max)} + V_{des} - V_{des} = 0$$

Es decir si no hacemos que el condensador C_f vea la señal siempre con la misma polaridad al realizar chopping al sistema, lo que hacemos es resetear el sistema cada dos ciclos de fs, por lo que dejamos de hacer modulado de ruido.

En la Figura 3.7 se muestra una modificación del modelo de Simulink de la Figura 2.11 en la que se realiza chopping. En dicho modelo el condensador ve los cambios de polaridad provocados por el chopping. El chopping se realiza multiplicando la señal por 1 en el primer ciclo y por -1 en el segundo mediante uso de multiplicadores. La frecuencia del chopping es fch.

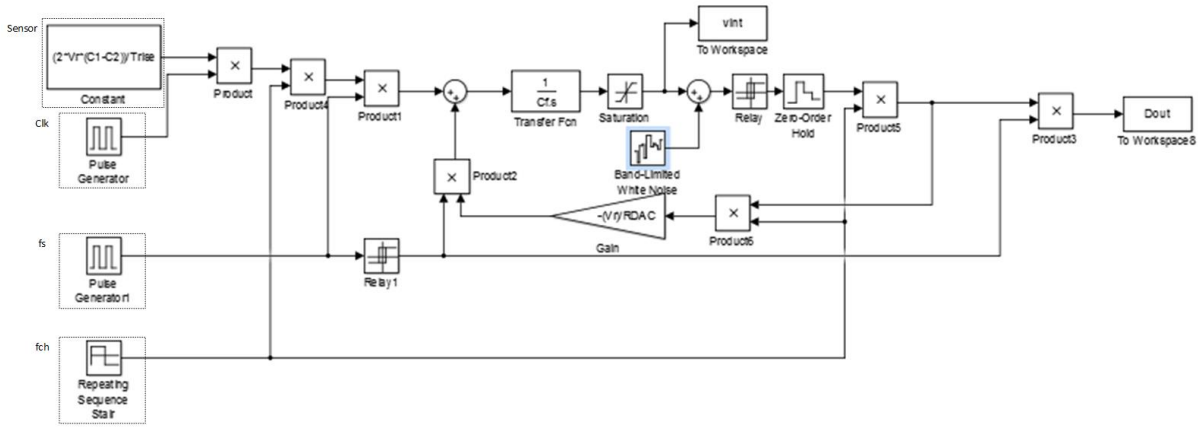


Figura 3.7 Modelo de Simulink con chopping sin anular efecto en el condensador.

Parámetro	Valor
Script de Simulink	CDCABR16SimulinkReadout
V_r	1,5 v
$V_{int(max)}$ (VCmax_V)	2 v
OSR	3000
Ancho de Banda de la señal	10 Hz
Frecuencia de reloj	960 kHz
Frecuencia de señal de disparo de S_3 (fs)	60 kHz
Potencia de Dither	10 p
Frecuencia Chopping fch	30 kHz
R_{DAC}	521 kΩ
Cf	12 pF
C0	5 pF
ΔC	0,2 pF

Tabla 3.3 Parámetros usados en la simulación realizada con el modelo de la Figura 3.7

La FFT de una simulación realizada con este modelo se muestra en la Figura 3.8. Los parámetros de dicha simulación se muestran en la Tabla3.3. Como se puede observar en la simulación el sistema parece funcionar correctamente. Sin embargo han desaparecido los diversos tonos que aparecían a frecuencias medias en las otras simulaciones.

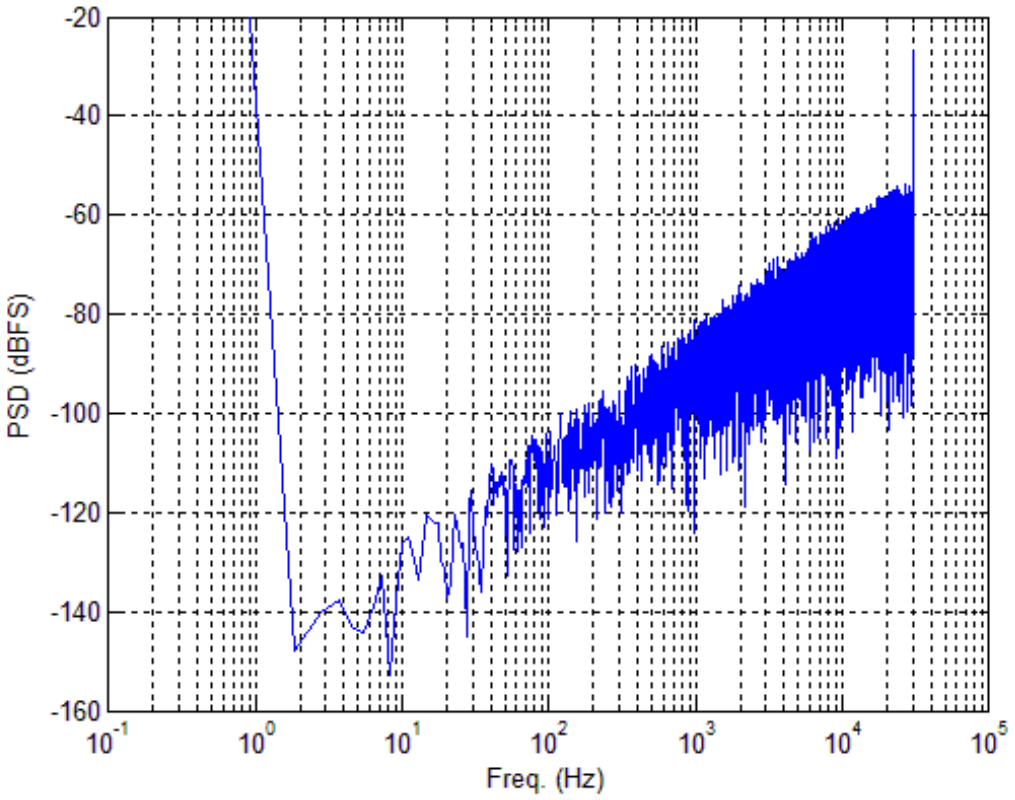


Figura 3.8 FFT de la simulación realizada con el modelo de la Figura 3.7 y los parámetros de la Tabla 3.3

Si observamos los resultados de esta simulación mostrados en la Tabla 3.4 podemos observar no obstante que pese a que la SQNR obtenida se encuentra en 131,64738 dB dándonos un ENOB de 21,57598 bits, la medida real y la teórica difieren en 0,1. Este resultado no se corresponde evidentemente con un conversor de 27 bits.

Parámetro	Resultado
Relación señal a Ruido SQNR	131,64738(dB)
Número de bits equivalente ENOB	21,57598 (bits)
Medida Teórica	0,400000
Medida Real	0,500000

Tabla 3.4 Resultados de la simulación realizada con el modelo de la Figura 3.7 y los parámetros de la Tabla 3.3.

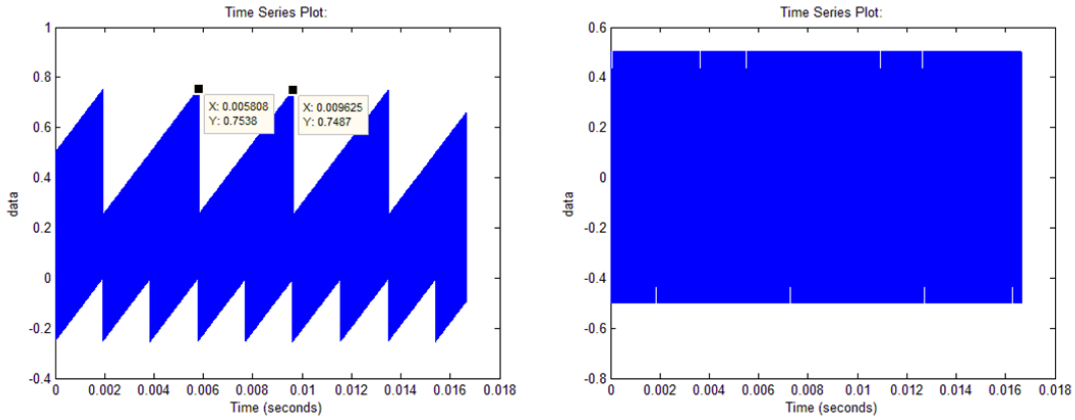


Figura 3.9 Resultados de la variable de estado V_{int} en función del tiempo para dos simulaciones con ΔC 0,251 pF, con chopping a la derecha y sin chopping a la izquierda. El chopping se ha realizado afectando a C_f

En la Figura 3.9 se muestran los resultados en función del tiempo de V_{int} para una simulación del sistema propuesto con chopping y otra sin chopping. El chopping se ha realizado sin evitar su efecto en el condensador C_f, es decir con el modelo de la Figura 3.7.

Como se puede observar a simple vista el perfil de ambas señales es diferente. La señal de la derecha, a la que se he realizado chopping muestra un perfil totalmente plano. Por otra parte la señal de la izquierda, a la que no se ha realizado chopping, muestra un perfil de diente de sierra. Este perfil se produce porque al realizar modelado de ruido estamos variando la tensión máxima que alcanza el condensador C_f en cada ciclo de fs al ir restando el error, tal y como se ha explicado en el apartado 2.1.

Dicho perfil es además el responsable de que aparezcan los armónicos a frecuencias medias que observábamos en las FFTs de las simulaciones sin chopping. Es por ello que estos armónicos no aparecen en la FFT con chopping realizado a partir del modelo de la Figura 3.7, ya que este presenta en su variable de estado un perfil plano. Además, la frecuencia de dichos armónicos depende del error, ya que el perfil de dientes de sierra que observamos en las simulaciones sin chopping depende del error.

3.3 Modelo en Simulink del sistema propuesto con chopping y simulaciones.

El modelo de Simulink del sistema propuesto con chopping se muestra en el Anexo 4. En dicho modelo se han modelado con cada una de las dos ramas del circuito diferencial separadas. Es por ello que el modelo tiene dos partes simétricas. Dicho modelo se ha realizado así para poder intercambiar los integradores al realizar chopping.

Este cambio se realiza de forma similar al cambio de los condensadores C_f explicado en el apartado anterior. Su objetivo es simular este cambio en Matlab. Como en Simulink tenemos en un mismo bloque modelado todo el integrador, no podemos hacer este efecto solo sobre el condensador, y lo tenemos que realizar sobre todo el integrador.

Así mismo como los bloques de Simulink tienen una entrada y una salida fijas en lugar de dar la vuelta al bloque como hacíamos con el condensador, lo que hacemos es que el integrador este siempre en el camino de una señal positiva o de una señal negativa. De este modo el integrador de la parte superior verá siempre una señal positiva, mientras que el integrador de la parte inferior verá siempre una señal negativa. Para ello durante el segundo ciclo de fs, y el resto de ciclos pares, el integrador superior estará conectado al camino inferior en lugar de al superior, y el integrador inferior estará conectado al camino superior en lugar de al inferior.

No obstante, cabe preguntarse si modelar así el circuito tiene sentido, ya que en este modelo estamos haciendo que todo el integrador, lo que incluye el amplificador diferencial además de los condensadores C_f quede fuera de los efectos del chopping. Teniendo en cuenta que dicho amplificador operacional es la principal fuente de ruido flicker del sistema no parece tener mucho sentido realizar este modelo. Sin embargo hay que tener en cuenta que en Simulink no hay transistores y que por tanto el integrador no genera ruido flicker. Dicho ruido flicker ha de ser generado como se ha explicado en el apartado 3.1 se añadirá en un punto donde sí se ve afectado por el chopping.

El modelo mostrado en el Anexo 4 se puede simular usando el script mostrado en el Anexo 1, el cual ya ha sido explicado con anterioridad en esta memoria (ver Apartado 2.6). Según configuremos los parámetros de dicho script podemos hacer simulaciones distintas. Por ejemplo podemos hacer simulaciones con o sin flicker. Para no añadir flicker solo tenemos que poner la potencia del ruido flicker pFlicker a 0, mientras que si queremos añadir ruido de este tipo tenemos que especificar su valor de la potencia mediante ese mismo parámetro.

También podemos hacer que el sistema no realice chopping simplemente haciendo que la frecuencia de chopping fch sea mucho menor que la inversa del tiempo de simulación, de forma que durante toda la simulación no dé tiempo a cambiar de polaridad al sistema. A continuación se muestran los resultados de una serie de simulaciones realizadas con este modelo.

Parámetro	Valor
Modelo de Simulink	CDC_simulinkABRChoppingFlick20160420.slx
Script de Simulink	CDCABR16SimulinkReadout
V_r	1,5 v
$V_{int(max)}$ (VCmax_V)	2 v
OSR	3000
Ancho de Banda de la señal	10 Hz
Frecuencia de reloj	960 kHz
Frecuencia de señal de disparo de S_3 (fs)	60 kHz
Frecuencia de chopping (fch)	30 kHz
Potencia de Dither	10 p
Potencia Flicker	0
R_{DAC}	521 kΩ
Cf	12 pF
C0	5 pF
ΔC	0,3 pF

Tabla 3.5 Parámetros empleados en la simulación de la Figura 3.10 y Tabla 3.6.

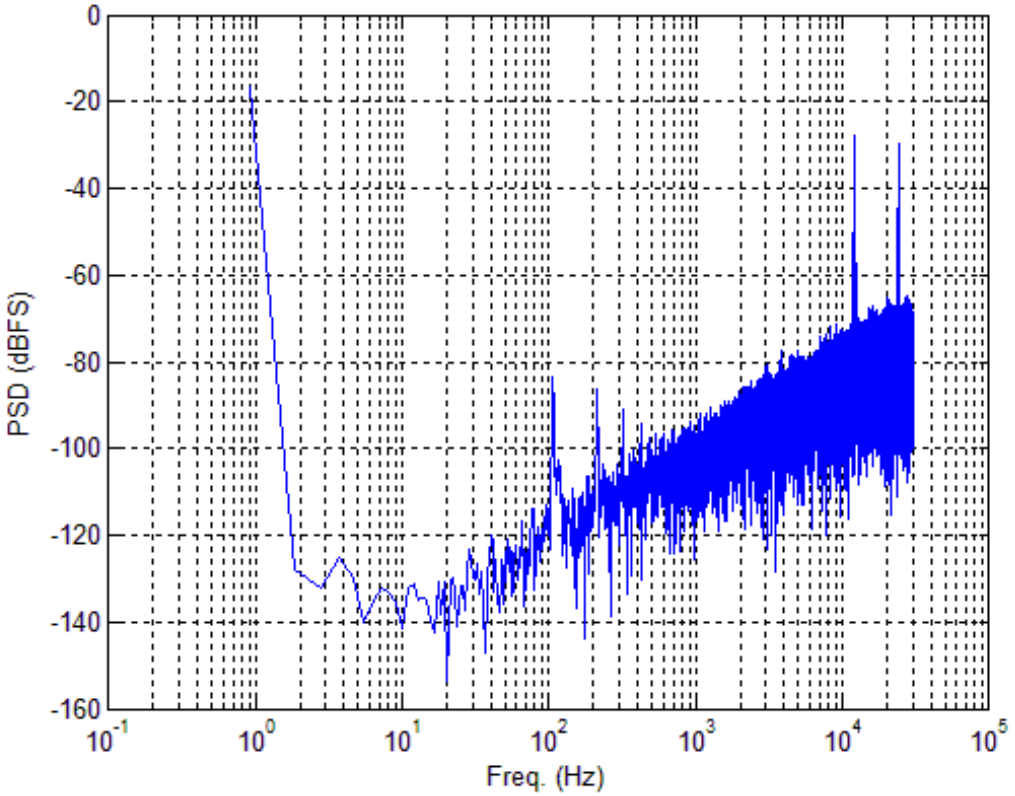


Figura 3.10 FFT de la simulación realizada con los parámetros mostrados en Tabla 3.5.

La Figura 3.10 muestra la transformada rápida de Fourier de la simulación realizada con el modelo de Simulink mostrado en el Anexo 4 y los parámetros de la Tabla 3.5. Como se puede observar el resultado es similar al obtenido en la FFT de la Figura 2.12, en la cual no realizábamos chopping y usábamos una variación de la capacidad del sensor de igual valor. Cabe destacar que esta simulación tiene también los tonos a frecuencias medias. Hemos hablado sobre esto en el Apartado 3.2.

Parámetro	Resultado
Relación señal a Ruido SQNR	122,96378 (dB)
Número de bits equivalente ENOB	20,13352 (bits)
Medida Teórica	0,600000
Medida Real	0,600189

Tabla 3.6 Resultados de la simulación realizada con los parámetros mostrados en Tabla 3.5.

Los resultados de la simulación con los parámetros de la Tabla 3.5 se muestran en la Tabla 3.6. En ella vemos que obtenemos una SQNR de 122,96 dB, lo que nos da una resolución del convertidor de 20 bits para esta variación de la capacidad del sensor. Si comparamos los resultados con los de la simulación equivalente sin chopping mostrados en la Tabla 2.3, podemos observar que tanto el número de bits como la SQNR obtenidos en aquella simulación son similares a los obtenidos en esta.

Parámetro	Valor
Modelo de Simulink	CDC_simulinkABRChoppingFlick20160420.slx
Script de Simulink	CDCABR16SimulinkReadout
V_r	1,5 v
$V_{int(max)}$ (VCmax_V)	2 v
OSR	3000
Ancho de Banda de la señal	10 Hz
Frecuencia de reloj	960 kHz
Frecuencia de señal de disparo de S_3 (fs)	60 kHz
Frecuencia de chopping (fch)	1/(4·1,0924333) Hz
Potencia de Dither	10 p
Potencia Flicker	10 y
Frecuencia Máxima Flicker	$4 \cdot 10^5$ Hz
R_{DAC}	521 kΩ
Cf	12 pF
C0	5 pF
ΔC	0,3 pF

Tabla 3.7 Parámetros empleados en la simulación de la Figura 3.11 y Tabla 3.8.

En ambos casos con y sin chopping se obtiene una medida correcta como demuestra la comparación entre la medida teórica y la real. Es decir el chopping no nos está "estropeando" el modelado de ruido de nuestro convertidor. Queda por ver no obstante si dicha implementación elimina el ruido flicker. Para ello lo primero que vamos a ver es el efecto del ruido flicker en una simulación con este modelo sin realizar el chopping de modo que luego podamos comparar.

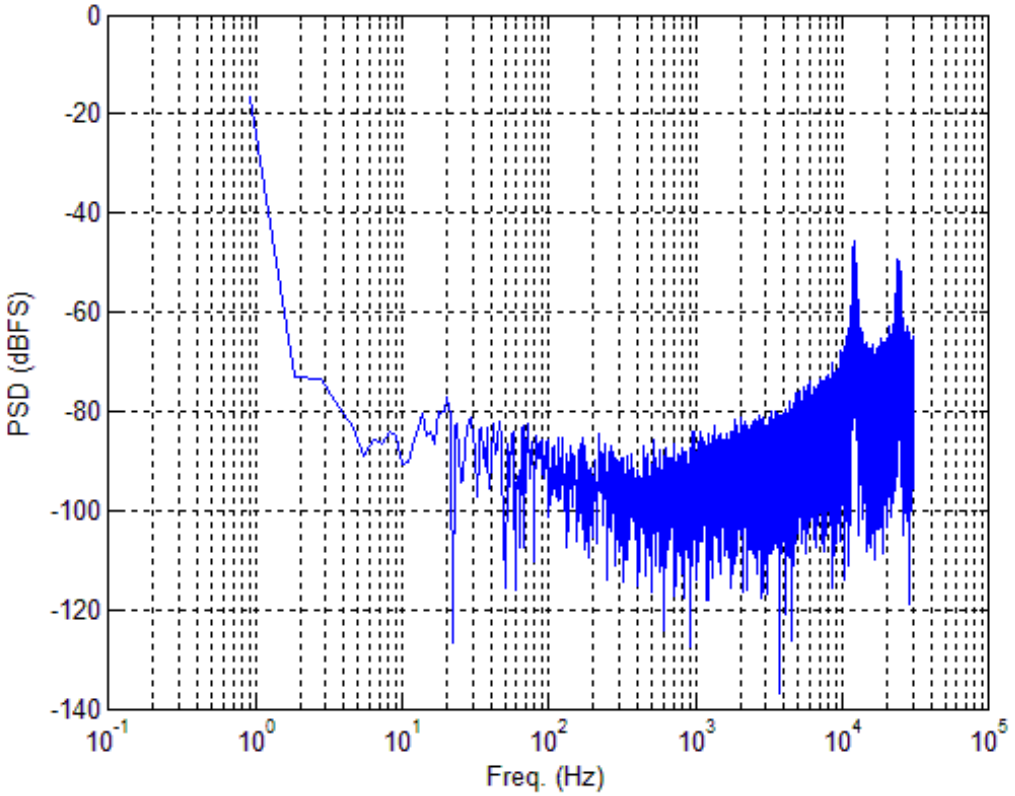


Figura 3.11 FFT de la simulación realizada con los parámetros mostrados en Tabla 3.7.

La Figura 3.11 muestra los resultados de la transformada rápida de Fourier para una simulación realizada con los parámetros de la Tabla 3.7. En dicha simulación se ha añadido ruido flicker. Podemos ver que este se observa a bajas frecuencias en la FFT de la Figura 3.11, desde las cuales va bajando hasta encontrase con el ruido de cuantificación que el modulador Sigma-Delta ha enviado a altas frecuencias. Esta FFT es similar a la mostrada en la Figura 3.3 donde añadíamos flicker al modelo descrito en el apartado 2.6.

En cuanto a los resultados de la simulación cuya FFT se muestra en la Figura 3.11, estos aparecen en la Tabla 3.8. Podemos ver que así mismo son similares a los mostrados en la Tabla 3.2 que se corresponde con la Figura 3.3. Esto nos muestra que tanto el modelo del Anexo 4 como el modelo del apartado 2.6 se comportan de igual forma. Esto ya lo habíamos podido observar al comparar los resultados de la Tabla 3.6 con los de su equivalente en el modelo del apartado 2.6, los de la Tabla 2.3.

Si lo comparamos con los resultados obtenidos con flicker a los obtenidos sin flicker vemos que pasamos de unos 123 dB de SQNR y 20 bits en la simulación sin flicker, a tan solo unos 71 dB de SQNR y 11 bits en la realizada con flicker. Es decir el ruido flicker nos empeora la SQNR en 52,09059 dB, reduciendo la resolución del convertidor en casi 9 bits.

Parámetro	Resultado
Relación señal a Ruido SQNR	70,87319 (dB)
Número de bits equivalente ENOB	11,48059 (bits)
Medida Teórica	0,600000
Medida Real	0,600037

Tabla 3.8 Resultados de la simulación realizada con los parámetros mostrados en Tabla 3.7.

A continuación procedemos a realizar una simulación con el modelo del Anexo 4 en la cual añadimos ruido Flicker a la vez que realizamos chopping al sistema. Para dicha simulación empleamos los parámetros de la Tabla 3.9.

Los resultados de la transformada rápida de Fourier de dicha simulación se muestran en la Figura 3.12. Podemos ver en dicha gráfica que el ruido flicker a bajas frecuencias ha desaparecido en comparación con el resultado obtenido cuando no realizábamos chopping mostrado en la Figura 3.11. No obstante se observa que en los tonos que aparecen a frecuencias medias ha entrado algo de ruido.

En cuanto a los resultados de esta simulación se muestran en la Tabla 3.10. En ella vemos que al realizar chopping en el sistema con flicker obtenemos una relación señal a ruido de 110,54543 dB, frente a los 70,87319 dB que obteníamos de SQNR al no hacer chopping. Esto es una mejora de 39,67224 dB para el sistema con chopping frente al sistema sin chopping cuando nuestra señal se ve afectada por ruido flicker.

En cuanto al número de bits equivalente de nuestro convertidor, observamos que obtenemos una resolución de hasta 18,07067 bits al realizar chopping frente a los 11,48059 bits que obteníamos en la simulación de la Tabla 3.8. Es decir al realizar chopping a nuestro sistema cuando este se ve afectado por ruido flicker, mejoramos la resolución en 6,59017 bits.

Por último si comparamos estos resultados con los obtenidos en las simulaciones sin ruido flicker, podemos ver que los resultados para la simulación con chopping son casi tan buenos como cuando no tenemos ruido. De hecho la resolución es apenas 2 bits menor y la SNQR unos 11 dB menor. Así mismo estamos obteniendo una medida real correcta, lo que nos demuestra que el chopping realizado de este modo no anula el efecto del modelado de ruido.

Parámetro	Valor
Modelo de Simulink	CDC_simulinkABRChoppingFlick20160420.slx
Script de Simulink	CDCABR16SimulinkReadout
V_r	1,5 v
$V_{int(max)}$ (VCmax_V)	2 v
OSR	3000
Ancho de Banda de la señal	10 Hz
Frecuencia de reloj	960 kHz
Frecuencia de señal de disparo de S_3 (fs)	60 kHz
Frecuencia de chopping (fch)	30 kHz
Potencia de Dither	10 p
Potencia Flicker	10 y
Frecuencia Máxima Flicker	$4 \cdot 10^5$ Hz
R_{DAC}	521 kΩ
Cf	12 pF
C0	5 pF
ΔC	0,3 pF

Tabla 3.9 Parámetros empleados en la simulación de la Figura 3.12 y Tabla 3.10.

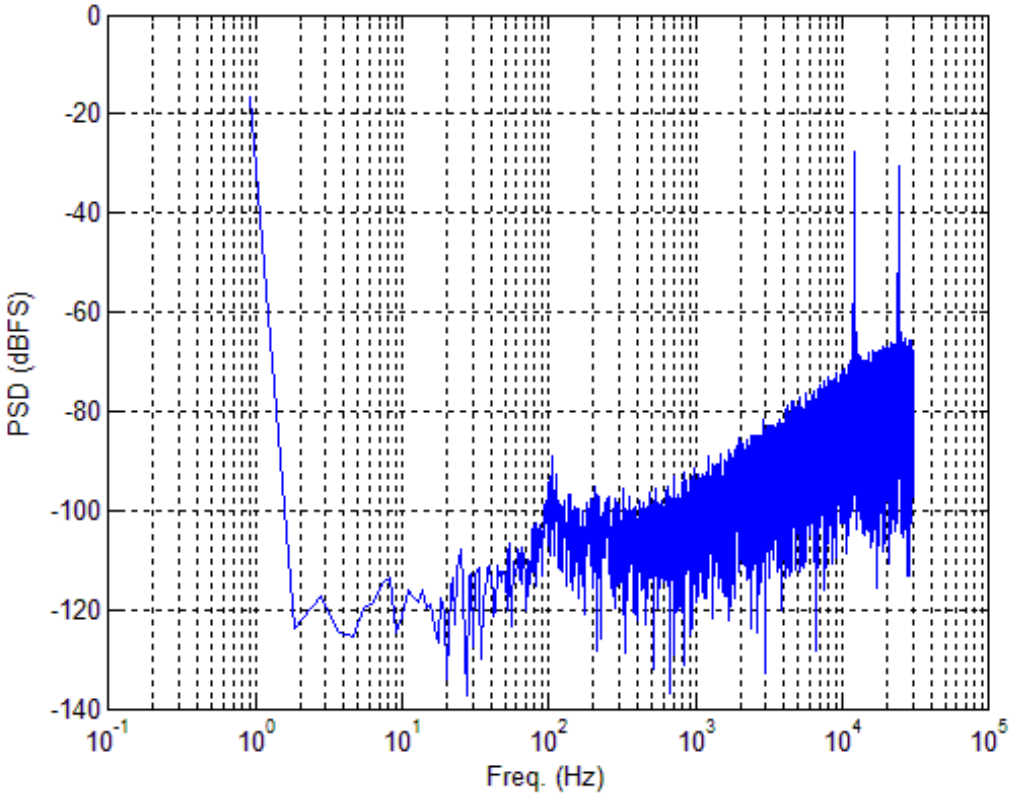


Figura 3.12 FFT de la simulación realizada con los parámetros mostrados en Tabla 3.9.

Parámetro	Resultado
Relación señal a Ruido SQNR	110,54543(dB)
Número de bits equivalente ENOB	18,07067 (bits)
Medida Teórica	0,600000
Medida Real	0,600189

Tabla 3.10 Resultados de la simulación realizada con los parámetros mostrados en Tabla 3.9.

Por último cabe preguntarse si es posible implementar físicamente esta operación del chopping. En el siguiente apartado intentaremos resolver esta duda implementado el chopping en el modelo de Cadence.

3.4 Modelo de Cadence Virtuoso con chopping y simulaciones

En este apartado vamos a proceder a implementar el circuito mostrado en la Figura 3.4 en Cadence Virtuoso. Una vez implementado dicho circuito de manera similar a como se ha implementado en Virtuoso el sistema sin chopping, procederemos a simularlo usando el simulador APS de la plataforma Cadence. Hay que tener en cuenta que estas simulaciones no incluyen ruido flicker ya que no ha sido añadido al modelo de Cadence.

Parámetro	Valor
Banco de pruebas de Cadence	TB_DualSDiff_dither_chopping_20160422
Estado simulado	TB_Chop20160422_OSR3000_C0pF3_Dit10p
V_r (VDDA)	1,5 v
$V_{int(max)}$ (VCmax_V)	2 v
OSR	3000
Ancho de Banda de la señal	10 Hz
Frecuencia de reloj	960 kHz
Frecuencia de señal de disparo de S_3 (fs)	60 kHz
Frecuencia de chopping	30 kHz
Potencia de Dither	10 p
R_{DAC}	521 kΩ
Cf	12 pF
C0	5 pF
ΔC	0.3 pF

Tabla 3.11 Parámetros empleados en la simulación de la Figura 3.13 y Tabla 3.12.

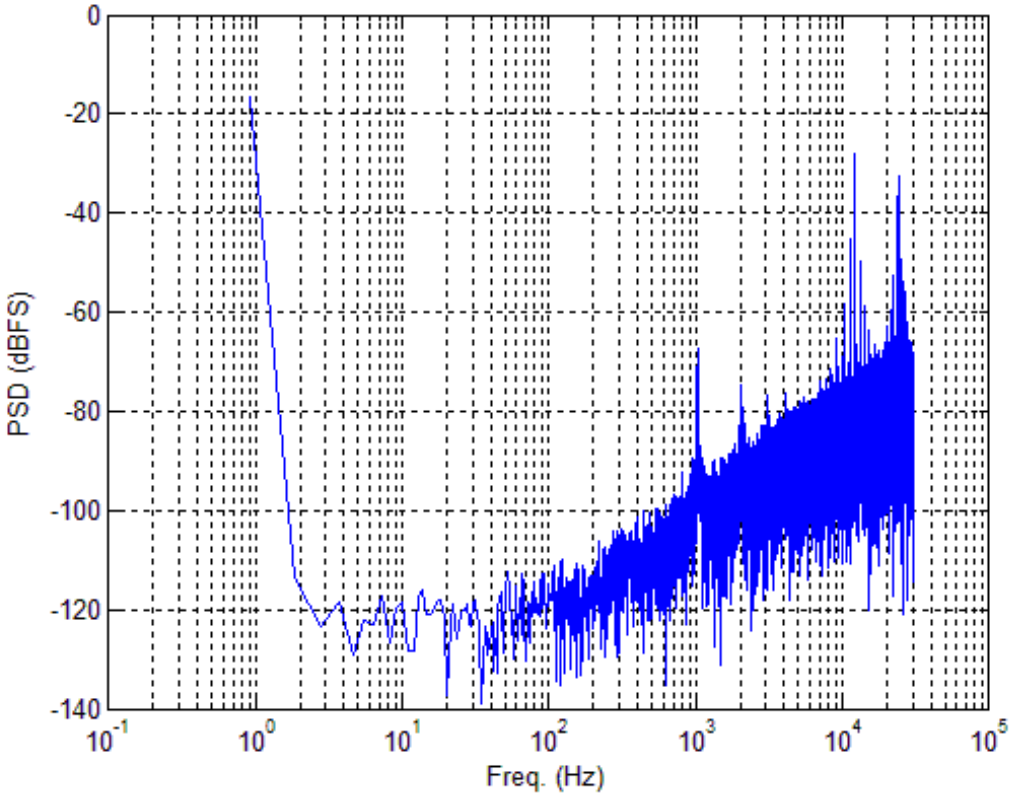


Figura 3.13 FFT de la simulación realizada con los parámetros mostrados en Tabla 3.11.

Los parámetros empleados en esta simulación se muestran en la Tabla 3.11. En cuanto a la FFT obtenida esta se muestra en la Figura 3.13. Podemos observar que el perfil de dicha FFT es similar al obtenido cuando no realizábamos chopping al modelo de Cadence como podemos como podemos comprobar si comparamos la Figura 3.13 con la Figura 2.18. Así mismo podemos ver en la Figura 3.13 que los tonos a frecuencias medias no desaparecen al realizar chopping de este modo.

Parámetro	Resultado
Relación señal a Ruido SQNR	111,91619(dB)
Número de bits equivalente ENOB	18,29837 (bits)
Medida Teórica	0,600000
Medida Real	0,601700

Tabla 3.12 Resultados de la simulación realizada con los parámetros mostrados en Tabla 3.11.

En cuanto a los resultados de la simulación en Cadence con chopping, estos se muestran en la Tabla 3.12. En ellos comprobamos que la relación señal a ruido obtenida es de 111,92 dB. Esta SQNR es solo ligeramente inferior a la obtenida cuando no realizábamos chopping. En cuanto a la resolución obtenida en el modelo con chopping, alcanzamos un valor del número de bits equivalente de 18,29 bits, frente a los 19,05 que obteníamos cuando no realizábamos chopping.

Por último, los resultados mostrados en la Tabla 3.12 demuestran que pese a realizar chopping obtenemos un valor de medida correcto. Esto se puede observar al comparar la medida real con la teórica y ver que ambas coinciden hasta el tercer decimal.

4. Conclusión

A lo largo de esta memoria se ha presentado un convertidor de capacidad a digital CDC compacto y con alta resolución basado en el principio de la modulación Sigma-Delta. Dicho convertidor tiene como objetivo servir como sistema de medida de presión mediante sensores capacitivos MEMs.

Para ello se ha procedido primero a realizar una breve introducción teórica acerca de los sensores capacitivos MEMs usados en este sistema. Se ha continuado con la explicación de los principios de funcionamiento de los convertidores de tipo doble rampa y de los convertidores de sobremuestreo Sigma-Delta en los que se basa nuestro CDC propuesto.

A continuación se ha presentado convertidor de capacidad a digital diseñado. Primero se ha procedido a una explicación teórica de su funcionamiento. Después se ha procedido a explicar el modelado del convertidor para su simulación en Matlab Simulink. Seguidamente se ha mostrado el modelo de Simulink realizado así como los resultados de simulación obtenidos. Dichos resultados has sido analizado poniendo especial interés en los aspectos que afectan a la resolución del convertidor.

Una vez analizados los resultados de las simulaciones del modelo de Matlab Simulink, se ha procedido a presentar el sistema modelado en Cadence virtuoso. Una vez presentado el modelo de Cadence se ha procedido de nuevo a analizar los resultados de una serie de simulaciones realizadas con dicho modelo. Dichos resultados se han comparado con los obtenidos mediante simulación en Simulink observando que ambos son similares.

En el tercer apartado se ha presentado el problema del ruido flicker y como este afecta especialmente a sistemas que tiene las señales de interés en frecuencias muy bajas, como es el caos que nos ocupa. Se ha procedido también a presentar un posible método para solucionar este problema, el chopping. Una vez presentado dicho método se ha explicado la forma concreta de implementación de este método, haciendo hincapié en la principal dificultad de dicha implementación, como este meto interfiere con el modelado de ruido.

Por último se ha presentado un modelo de Simulink capaz de simular la operación del chopping escogida. Los resultados de las simulaciones con dicho modelo han demostrado que la implementación del chopping mejora de forma sustancial el funcionamiento del convertidor propuesto cuando este se ve afectado por ruido flicker. Así mismo se ha mostrado que la implementación física del chopping propuesto es posible y da resultados satisfactorios mediante la simulación de un modelo de Simulink con chopping.

Con todo ello se puede concluir que el conversor de capacidad a digital presentado en esta memoria obtiene una alta resolución con un uso reducido de área. Dicha resolución incluso en el caso de operación más desfavorable es de 18 bits. Así mismo el convertidor es compatible con el uso de chopping para lograr reducir los efectos de ruido generado por el circuito. Del mismo modo es de suponer que el chopping debería ser capaz de anular posibles offsets debidos a los componentes del circuito.

Los siguientes pasos a realizar para continuar el desarrollo de dicho convertidor pasan por realizar un diseño a nivel de transistor, seguido por la fabricación de un prototipo y pruebas del mismo. Dichos pasos exceden el alcance del presente trabajo.

Anexo 1. Script de Matlab utilizado para simular el modelo de Simulink.

```
clear all;

%% Simulation resolution:

format long
warning off;
Nexp=16; % 2^N numero de puntos en la FFT @Fs
BW=10;
OSR=3000;
fs=OSR*2*BW;
Ts=1/fs;
goal_lsim=2^Nexp;
BW_fft=fs/goal_lsim;
Points_fft_BW=BW/BW_fft;
Number_samples_to_remove=10;
Simulation_time=Ts*goal_lsim+(Ts*Number_samples_to_remove);
%% Numero de puntos para integrar y tono de FFT:
lsim=goal_lsim;
binbw=floor(lsim/OSR/2);
fbin=fs/lsim;
bint=floor(binbw/2);
ft=fbin*(bint-1);
%% Parametros Dual-Slope:
N1=8;
N2=8;
Tclk=Ts/(N1+N2);
fclk=1/Tclk;
Trise=1e-12;%1e-12;
P_Width=(Trise/Tclk)*100;
ACmax_pF=1;
VCmax_V=2;
VFS=VCmax_V;
Vr=1.5;
Cf=12e-12;
RDAC=521e3;
C0=5e-12;
Tch=(2*Ts);%(4*Simulation_time);%(50*Ts);
fch=1/Tch;
ditherPower=10e-12;
pFlicker=1e-23;%1e-23;%1e-21;%1e-25
Seed_Noise=37845;
Seed_NoiseF=19542;
%% Entrada:
AC_pF=0.3;%((C1-C2)*1e12); %% CAPACITIVE INPUT DIFFERENCE
C1=C0+(AC_pF*1e-12);
C2=C0;
DCin=(AC_pF/ACmax_pF)*VCmax_V;
amp=DCin;
dBin=20*log10(amp/VFS);
%% Polos y ceros filtro generación Flicker
divideDecade=8;
fMax=4e5;%fs/2;
iMax=floor(log10(fMax)*divideDecade);

Vpoles = [];
```

```
Vzeros = [];
for k=1:4:iMax
    Vpoles=[Vpoles -10^((k+1)/divideDecade)];
    Vzeros=[Vzeros -10^((k+3)/divideDecade)];
end

Vpoles;
Vzeros;
Vgain=1/10^(1/divideDecade);
%% Cargar fichero de datos:
%sim('CDC_simulinkABR.slx');
%sim('CDC_simulinkABRChopping.slx');
%sim('CDC_simulinkABRFlick.slx');
%sim('CDC_simulinkABRChoppingFlick.slx');
%sim('CDC_simulinkABRChoppingMod.slx');
%sim('CDC_simulinkABRChoppingFlickMod.slx');
%sim('CDC_simulinkABRNoNS.slx');
%sim('CDC_simulinkABRSimp.slx');
sim('CDC_simulinkABRChoppingFlick20160420.slx');

adc=Dout(Dout~=0);
%plot(Vint)

%% Pasar Data de Fclk a Fs:
 i=0;
 j=0;
 ns_fix=lsim*N2;
 lsim_adc=length(adc);
 offsetSim=11;
 adc_fix=adc(((lsim_adc-ns_fix+1)-offsetSim):((lsim_adc)-offsetSim));
 data=zeros(lsim,1);

 for i=1:lsim
     j=i;
     data(j)=sum(adc_fix((((i-1)*N2)+1):(i*N2)));
 end

%% FFT of multibit Data @Fs:

figure(2)
freqA=linspace(0,fs/2,lsim/2);
yfft=esph(data);
semilogx(freqA,yfft-max(yfft(1:binbw))+dBin,'b');
%axis([0.5 30e3 -170 0])
xlabel('Freq. (Hz)');
ylabel('PSD (dBFS)');
grid on

Data_fft=abs(fft(data.*hanning(length(data),'periodic')));
noise_Q=Data_fft(3:binbw);

%% SNR con DC
DCfft=Data_fft;
sigDC=DCfft(1:2);
sigDCreal=[sigDC(2) sigDC(1) sigDC(2)];
noiseDC=DCfft(3:binbw);
SNR_DC=10*log10((sigDCreal'*sigDCreal)/(noiseDC'*noiseDC));
SNR_DC2=20*log10(norm(sigDCreal)/norm(noiseDC));
SQNR=SNR_DC2-dBin
ENOB_DC=(SNR_DC2-dBin-1.76)/6.02
```

```
max_data=max(data)
min_data=min(data)
amp=DCin
Digital=((mean(data)*VFS)/N2)
```

Anexo 2. Scrip de Matlab utilizado para analizar los resultados de la simulación en Cadence.

```
clear all;

%% Simulation resolution:

format long
warning off;
Nexp=16; % 2^N numero de puntos en la FFT @Fs
BW=10;
OSR=3000;
fs=OSR*2*BW;
Ts=1/fs;
goal_lsim=2^Nexp;
BW_fft=fs/goal_lsim;
Points_fft_BW=BW/BW_fft;
Number_samples_to_remove=10;
Simulation_time=Ts*goal_lsim+(Ts*Number_samples_to_remove);
%% Numero de puntos para integrar y tono de FFT:
lsim=goal_lsim;
binbw=floor(lsim/OSR/2);
fbin=fs/lsim;
bint=floor(binbw/2);
ft=fbin*(bint-1);
%% Parametros Dual-Slope Cadence:
N1=8;
N2=8;
Tclk=Ts/(N1+N2);
fclk=1/Tclk;
ACmax_pF=1;
VCmax_V=2;
VFS=VCmax_V;
Vr=VFS;
%% Entrada:
AC_pF=0.3; %% CAPACITIVE INPUT DIFFERENCE
DCin=(AC_pF/ACmax_pF)*VCmax_V;
amp=DCin;
dBin=20*log10(amp/VFS);
%% Cargar fichero de datos:
load TB_20160518_OSR3000_C0pF3_Dit10p_Pfck10p_Ffck400k_Chopp.txt -
ASCII
adc=TB_20160518_OSR3000_C0pF3_Dit10p_Pfck10p_Ffck400k_Chopp;

%% Pasar Data de Fclk a Fs:
i=0;
j=0;
ns_fix=lsim*N2;
lsim_adc=length(adc);
offsetSim=9;
adc_fix=adc(((lsim_adc-ns_fix+1)-offsetSim):((lsim_adc)-offsetSim));
data=zeros(lsim,1);

for i=1:lsim
    j=i;
    data(j)=sum(adc_fix((((i-1)*N2)+1):(i*N2)));
end
```

```
%% FFT of multibit Data @Fs:

figure(2)
freqA=linspace(0,fs/2,lsim/2);
yfft=esph(data);
semilogx(freqA,yfft-max(yfft(1:binbw))+dBin,'b');
%axis([0.5 30e3 -170 0])
xlabel('Freq. (Hz)');
ylabel('PSD (dBFS)');
grid on

Data_fft=abs(fft(data.*hanning(length(data),'periodic')));
noise_Q=Data_fft(3:binbw);

%% SNR con DC
DCfft=Data_fft;
sigDC=DCfft(1:2);
sigDCreal=[sigDC(2) sigDC(1) sigDC(2)];
noiseDC=DCfft(3:binbw);
SNR_DC=10*log10((sigDCreal'*sigDCreal)/(noiseDC'*noiseDC));
SNR_DC2=20*log10(norm(sigDCreal)/norm(noiseDC));
SQNR=SNR_DC2-dBin
ENOB_DC=(SNR_DC2-dBin-1.76)/6.02
max_data=max(data)
min_data=min(data)
amp=DCin
Digital=((mean(data)*VFS)/N2)
```

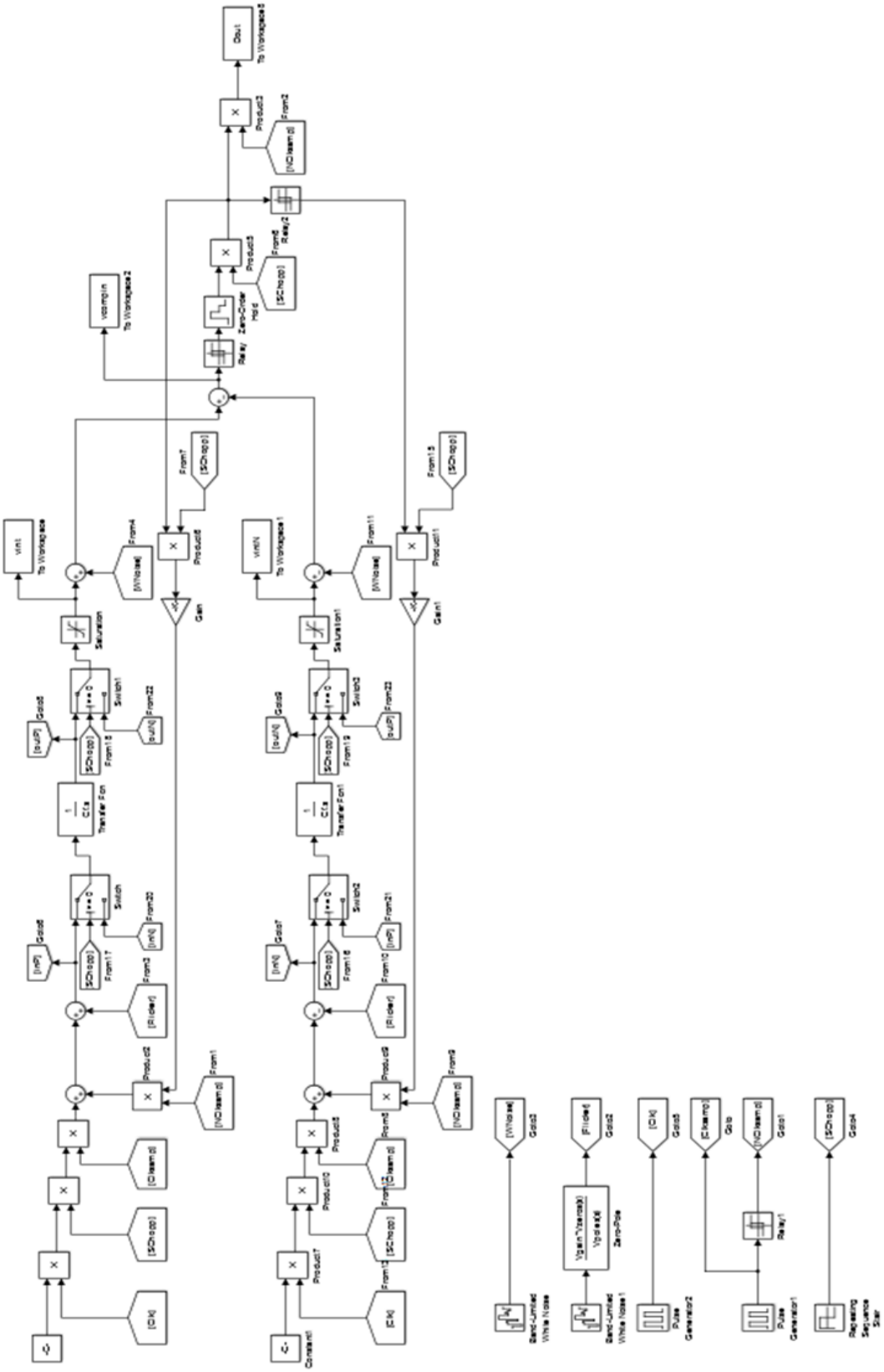
Anexo 3. Script de Matlab para calcular los polos y ceros del filtro del generador de ruido flicker.

```
divideDecade=8;
fMax=4e5;%fs/2;
iMax=floor(log10(fMax)*divideDecade);

Vpoles = [];
Vzeros = [];
for k=1:4:iMax
    Vpoles=[Vpoles -10^((k+1)/divideDecade)];
    Vzeros=[Vzeros -10^((k+3)/divideDecade)];
end

Vpoles;
Vzeros;
Vgain=1/10^(1/divideDecade);
```

Anexo 4. Modelo Matlab con Chopping al sistema.



Bibliografía

[1] S. Paniagua, «La Revolución de los datos:sensores e Internet de las Cosas,» *bit,* pp. 40-45, 2013.

[2] T. M. Adams y R. A. Layton, «Chapter 1. Introduction,» de *Introductory MEMS Fabrication and Applications*, New York, Springer, 2010, pp. 3-4.

[3] «MEMSnet,» [En línea]. Available: https://www.memsnet.org/about/what-is.html. [Último acceso: 20 Mayo 2016].

[4] Y. Wang y P. Vamsy, «Wide-temperature range CMOS capacitance to digital convertor for MEMS pressure sensors,» *Sensors and Actuators,* vol. A, nº 233, pp. 302-309, 2015.

[5] T. M. Adams y R. A. Layton, «Chapter 9: Capacitive transducers,» de *Introductory MEMS*, New York, Springer, 2010, pp. 236-238.

[6] A. Savaliya y B. Mishra, «A 0.3V, 12nW, 47fJ/conv, Fully Digital Capacitive Sensor Interface in 0.18 um CMOS,» de *International conference on VLSI Systems, Architecture, Technology and Applications (VLSI-SATA)*, 2015.

[7] W. Jung, S. Jeong, S. Oh, D. Sylvester y D. Blaauw, «A 0.7 pF-to-10nF Fully Digital Capacitance-to-Digital Converter Using Iterative Delay-Chain Discharge,» de *IEEE International Solid-State Circuits Conference*, 2015.

[8] A. Beriain, H. Solar, R. Berenguer, J. Montiel-Nelson, J. Sosa, R. Pulido y S. García-Alonso, «A very low power 7.9 bit MEMS pressure sensor suitable for batteryless RFID applications,» de *978-1-4799-4132-2/14/$31.00 2014 IEEE*, 2014.

[9] D.-Y. Shin, H. Lee y S. Kim, «A Delta-Sigma Interface Circuit for Capacitive Sensors With an Automatically Calibrated Zero Point,» *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS-II: EXPRESS BRIEFS,* vol. 58, nº 2, pp. 90-94, 2011.

[10] Z. Tan, R. Daamen, A. Humbert, Y. V. Ponomarev, Y. Chae y M. A. Pertijs, «A 1.2-V 8.3-nJ CMOS Humidity Sensor for RFID Applications,» *IEEE JOURNAL OF SOLID-STATE CIRCUITS,* vol. 48, nº 10, pp. 2469-2477, 2013.

[11] B. Li, L. Sun, C.-T. Ko, A. K.-Y. Wong y K.-P. Pun, «A High-Linearity Capacitance-to-Digital Converter Suppressing Charge Errors From Bottom-Plate Switches,» *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS-I: REGULAR PAPERS,* vol. 61, nº 7, pp. 1928-1941, 2014.

[12] S. Wang y C. Dehollain, «Desing and Implementation of a 46-kS/s CMOS SC Dual-Mode Capacitive Sensor Interface With 50-dB SNR and 0.7% Nonlinearity,» *IEEE SENSORS JOURNAL,* vol. 15, nº 2, pp. 1077-1090, 2015.

[13] S. V y B. George, «A Novel Switched-Capacitor Capacitance-to-Digital Converter for Single Element Capacitive Sensors,» *978-1-4799-6144-6/15/$31.00 2015 IEEE.*

[14] N. Madhu Mohan, A. Ravikant Shet, S. Kedarnath, J. Kumar y V., «Digital Converter for Differential Capacitive Sensors,» *IEEE TRANSACTIONS ON INSTRUMENTATION AND MEASUREMENT,* vol. 57, nº 11, pp. 2576-2581, 2008.

[15] B. George y V. Jagadeesh Kumar, «Analysis of the Switched-Capacitor Dual-Slope Capacitance-to-Digital Converter,» *IEEE TRANSACTIONS ON INSTRUMENTATION AND MEASUREMENT,* vol. 59, nº 5, pp. 997-1006, 2010.

[16] R. J. Baker y H. Li, «Chapter 29 Data Converter Architectures,» de *CMOS Circuit Design, Layout, and Simulation*, Hoboken, New Jersey, Wiley, 2010, pp. 1000-1002.

[17] R. Baker y H. Li, «Chapter 29 Data Converter Architectures,» de *CMOS Circuit Design, Layout, and Simulation*, Hoboken, New Jersey, Wiley, 2010, pp. 1007-1009.

[18] R. Baker y H. Li, «Chapter 29 Data Converter Architectures,» de *CMOS Circuit Design, Layout, and Simulation*, Hoboken, New Jersey, Wiley, 2010, p. 1011.

[19] H. Schmid, «Offset, flicker noise, and ways to deal with them,» 2008.