UNIVERSITAT POLITÈCNICA DE VALÈNCIA

TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

DISEÑO DE UN SAR ADC PARA APLICACIONES MULTICANAL CON TECNOLOGÍA CMOS DE 180nm

José Juan Cerdá

Tutor: Vicente Herrero Bosch

Trabajo Fin de Grado presentado en la Escuela Técnica Superior de Ingenieros de Telecomunicación de la Universitat Politècnica de València, para la obtención del Título de Graduado en Ingeniería de Tecnologías y Servicios de Telecomunicación

Curso 2019-20

Valencia, Septiembre de 2020

Dedicado a mis abuelos y mis padres

Resumen

Los Conversores Analógico a Digital sirven como interfaces que conectan el mundo analógico al binario de los ordenadores. Aquellos que son enfocados a aplicaciones médicas requieren una larga duración de batería y una velocidad y resolución moderada.

Este trabajo presenta el diseño de un SAR ADC para aplicaciones multicanal con tecnología CMOS de 180nm. El conversor propuesto está formado por 3 bloques, un DAC que utiliza una estructura BWC y se encarga del proceso de muestreo de la señal de entrada, un bloque digital que se ocupa de la lógica del circuito y un comparador latch dinámico.

Con este diseño se busca conseguir un ADC de bajo consumo y lo más pequeño posible que tenga una resolución de 9 bits o más y un tiempo de conversión de unos pocos microsegundos. Para conseguirlo se utiliza el kit de Cadence XFAB para diseño analógico/señal-mixta de tecnología CMOS de 180nm y una alimentación de 1.8V.

Este proyecto será empleado para convertir señal analógicas a digital que provenien de un sensor de luz aplicado a la obtención de imagenenes de un PET. El ADC está diseñado para señales que pueden llegar hasta un fondo de escala de 1V.

Abreviaciones

ADC	Conversor Analógico-Digital
SAR	Aproximación Sucesiva
DAC	Conversor Digital-Analógico
CMOS	Semiconductor Complementario de Óxido Metálico
BWC	Condensadores de Pesos Binarios
TWC	BWC de Dos Etapas
PET	Tomografía por Emisión de Positrones
DSP	Procesador Digital de Señales
LSB	Bit Menos Significativo
MSB	Bit Más Significativo
RMS	Valor Cuadrático Medio
DNL	No-Linealidad Diferencial
INL	No-Linealidad Integral
SNR	Relación Señal a Ruido
SINAD	Relación Señal a Ruido y Distorsión
SFDR	Rango Dinámico Libre de Espurios
ENOB	Número Efectivo de Bits
FFT	Transformada de Fourier
T/H	Track & Hold
FE	Fondo de Escala
CDAC	DAC Capacitivo
OpAmp	Amplificador Operacional
FF	Flip-Flop
VIN	Señal de entrada
VREF	Señal de referencia
GND	Señal de tierra
PMOS	Semiconductores de óxido de metal de tipo P
NMOS	Semiconductores de óxido de metal de tipo N

Índice general

Índice de figuras

Índice de tablas

Parte I

Memoria

Capítulo 1

Introducción

1.1. Objetivo

Se requiere diseñar un conversor analógico digital para integrarlo en un sistema multicanal para un dispositivo de aplicación en medicina. Un sistema multicanal requiere dispositivos de bajo consumo ya que hay tantos convertidores que van a consumir potencia como canales hay en paralelo.

En este trabajo diferentes arquitecturas e implementaciones de convertidores ADC son estudiadas. En particular la de aproximaciones sucesivas de tipo capacitivo. Dichos convertidores son los que presentan un mejor balance precisión-consumo para estas aplicaciones.

Los objetivos principales son minimizar el tiempo de conversión hasta unos pocos microsegundos y conseguir una resolución de al menos 9 bits pero realizando al mismo tiempo una buena solución al compromiso con su consumo y tamaño.

Para diseñar el sistema se utiliza el kit de Cadence XFAB - XH018 de 180nm. XH018 está especialmente optimizado para aplicaciones médicas, industriales y de automoción, por lo que es perfecto para la nuestra. Además, se trata de una tecnología que soporta operaciones a altas temperaturas y de bajas pérdidas, y proporciona kits y módulos que nos permiten diseñar circuitos robustos y optimizados para bajo consumo, rendimiento y area.

1.2. Organización de este trabajo

Los capítulos de este trabajo se organizan de la siguiente manera:

Capítulo 2: presenta los principios básicos de un ADC tales como fundamentos básicos y sus caracterísitcas estáticas y dinámicas. Además, al final del capítulo se explican y comparan las cinco arquitecturas de ADC más populares.

Capítulo 3: en este capítulo se realiza más detenidamente un estudio del SAR ADC. En él se analizan y discuten las posibles arquitecturas tanto del ADC como de los bloques que lo componen para nuestra aplicación.

Capítulo 4: está dedicado a la implementación del SAR ADC con Cadence. En él se explica la metodología de diseño empleada y cómo se ha aplicado paso a paso hasta conseguir las prestaciones

deseadas.

Capítulo 5: muestra el desempeño del SAR ADC final diseñado analizando mediante simulaciones las características estudiadas en el capítulo 2.

Capítulo 6: contiene las conclusiones y propuesta de trabajo futuro.

1.3. Metodología de trabajo del TFG

La metodología de trabajo empleada ha sido GTD (Getting Things Done). GTD es una metodología de productividad personal que aplico y me ha permitido realizar el TFG al mismo tiempo que he trabajado a tiempo parcial en una empresa y he estudiado el master de electrónica (MUISE) de la UPV.

En resumen GTD es una serie de hábitos productivos que permiten al usuario gestionar sus tareas y proyectos de una forma eficaz, productiva y sin estrés si se aplica correctamente.

El diagrama temporal de cómo ha ido avanzando el proyecto des de su formalización hasta su entrega en Noviembre ha sido el siguiente:

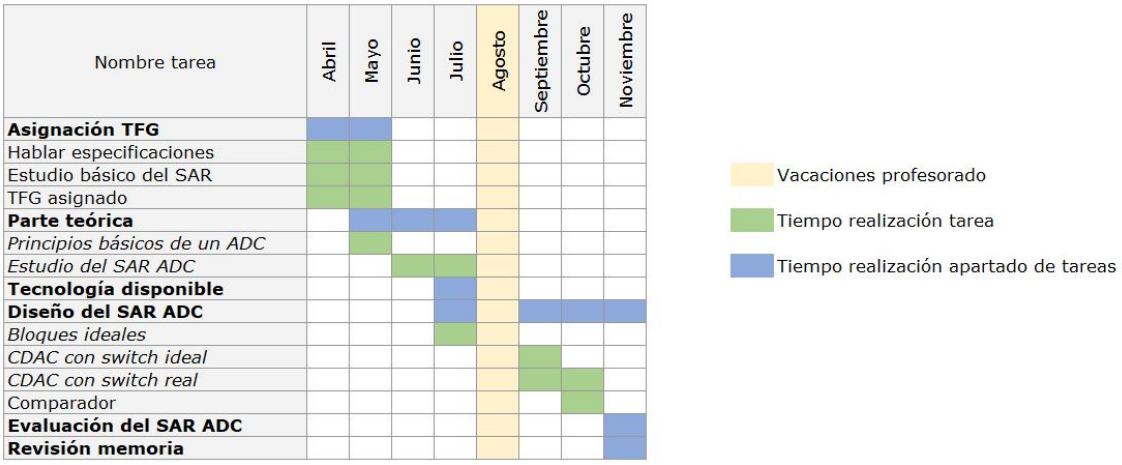


Figura 1.1: Diagrama temporal del TFG

El diagrama muestra en negrita las secciones principales. Las que ocurren durante varios meses se desglosan por tareas/apartados.

El reto de este TFG ha sido que debido al COVID-19 no se pudo empezar a diseñar hasta el 10/07/2020. Además, al tratarse de una nueva tecnología empleada en la universidad y de que en Agosto son las vacaciones de los profesores, no fue hasta Septimebre entonces que pude empezar a diseñar de forma eficaz el proyecto.

Capítulo 2

Principios Básicos de un ADC

2.1. Base Teórica

La función de un ADC como indica su nombre, es la de transformar señales analógicas en digitales. Esta conversión facilita el procesado de datos y la señal resultante se vuelve más inmune al ruido y otras interferencias en las que las señales analógicas son más sensibles.

Los conversores tienen un rol importante en el procesamiento de señales y son altamente usados en campos tales como audio, control, comunicación y medicina.

En la figura 2.1 se muestra el diagrama de bloques de un sistema DSP. Anterior a la conversión de analógico a digital, la señal analógica generalmente pasa a través de algún tipo de circuito de acondicionamiento de señal que realiza funciones tales como amplificación, atenuación y/o filtrado. Además se requiere un filtro de paso bajo/paso de banda para eliminar señales no deseadas fuera del ancho de banda de interés y evitar aliasing [1].

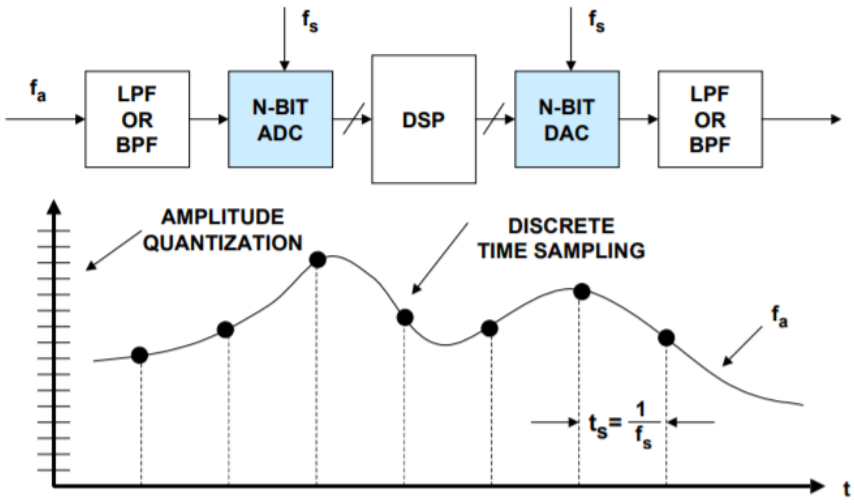


Figura 2.1: Diagrama Sistema DSP [1]

Respecto al proceso que sigue un ADC, básicamente mesura la amplitud de una señal (normalmente voltaje) de forma periódica, redondea los valores a un set de valores preestablecidos (niveles de cuantificación) y los registra en cualquier tipo de memoria o soporte:

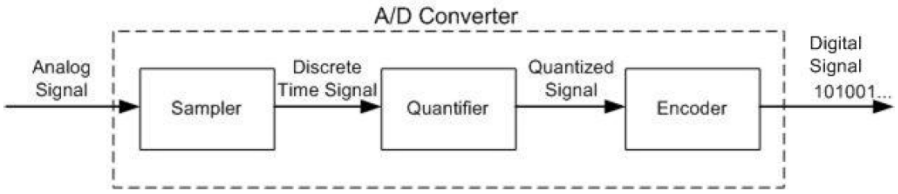


Figura 2.2: Proceso de Conversión de un ADC [2]

2.2. Fundamentos de un ADC

2.2.1. Resolución

La resolución es el número de bits o números de salidas de un ADC. El paso más pequeño es definido como LSB y se obtiene en base a la siguiente ecuación:

$$V_{LSB} = \frac{V_{REF}}{2^N} \tag{2.1}$$

Donde V_{REF} es el voltaje de referencia del conversor.

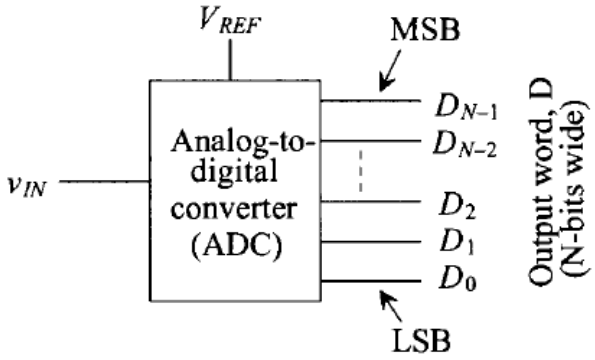


Figura 2.3: Diagrama de un ADC [3]

Las salidas D_N de la figura 2.3 representan el valor de entrada detectado:

$$V_{detectado} = (\frac{D_0}{2^N} + \frac{D_1}{2^{N-1}} + ... + \frac{D_{N-1}}{2}) \cdot V_{REF} \tag{2.2}$$

Siendo D_0 el LSB y D_{N-1} el MSB.

2.2.2. Aliasing

El fenómeno de aliasing puede ocurrir durante el muestreo de una señal. Más concretamente cuando la frecuencia de muestreo no es por lo menos dos veces la de la señal que queremos discretizar. Esta regla es conocida como el criterio de Nyquist. Cuando hay aliasing el resultado que obtenemos al muestrear una señal puede resultar en una frecuencia diferente y por lo tanto hacer difícil la recuperación de la señal original.

La figura 2.4 muestra un caso de aliasing por submuestreo.

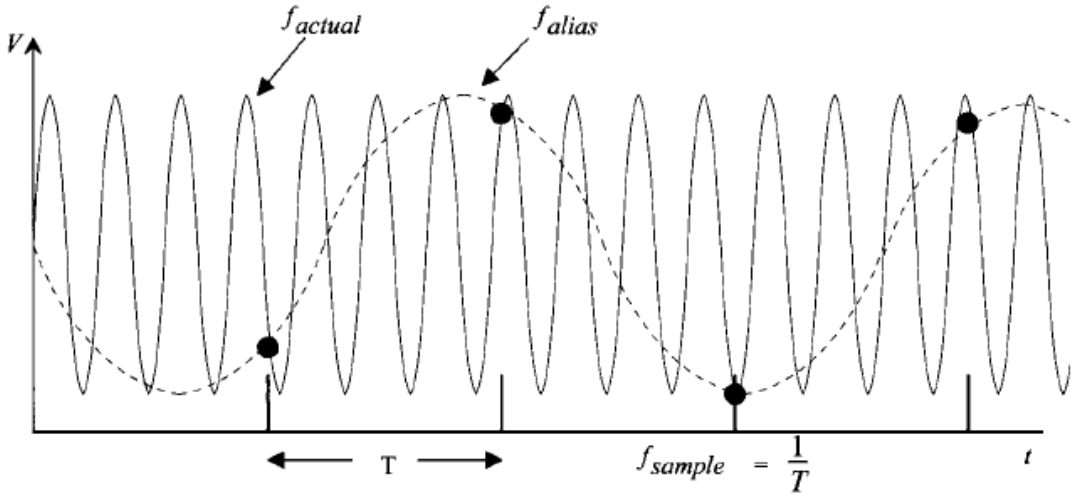


Figura 2.4: Aliasing causado por submuestreo [3]

Para tener una visión más amplia del aliasing, la figura 2.5 muestra un análisis en el dominio de frecuencia. Como resultado, la señal con frecuencia f es muestreada a fs. En frecuencia esto se traduce a que el espectro de la señal sea duplicada a frecuencias multiplicadas por fs. Si fs <2·f los espectros se solaparán y ocurrirá aliasing.

El aliasing puede ser combatido de dos formas, muestreando a frecuencias mayores que cumplan como mínimo el criterio de Nyquist o filtrando la señal analógica antes de que sea sampleada y quitando por lo tanto las frecuencias que sean mayores de la mitad de la frecuencia de muestreo.

El filtrado de una señal antes de muestrearla ya lo habíamos visto en el diagrama de bloques de un sistema DSP de la figura 2.1. Es una buena práctica realizar este filtrado premuestro ya que elimina señales no deseadas fuera del ancho de banda de interés y evita el aliasing.

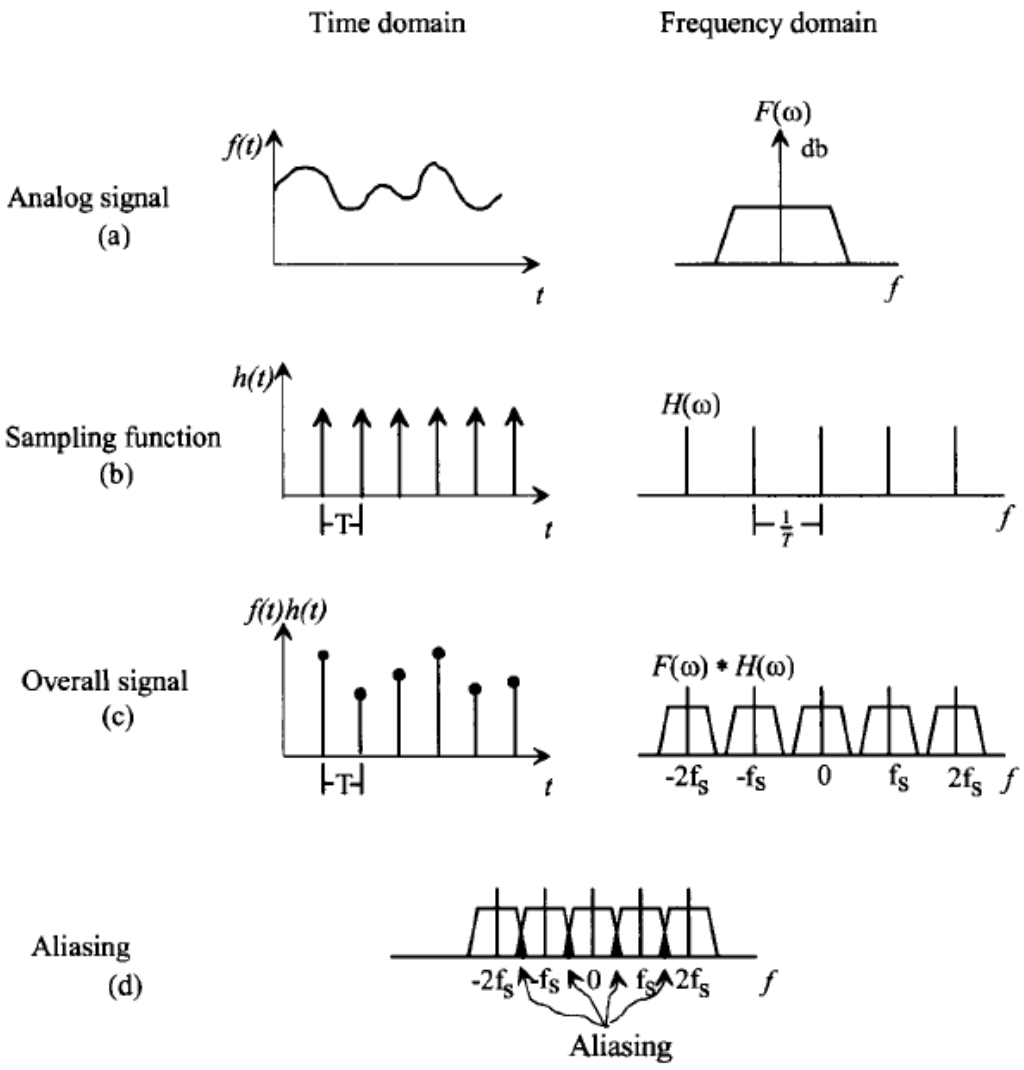


Figura 2.5: Aliasing en el dominio de frecuencia[3]

2.2.3. Error de cuantificación

Cuando convertimos una señal analógica a unos valores preestablecidos se produce un error como resultado de esta cuantificación. Este error es conocido como error de cuantificación y aparece en todos los ADC, ya que en menor o mayor medida la cantidad de niveles de cuantificación que tienen es finita.

Este error se calcula en base a la diferencia entre la señal que obtenemos a la salida y la que debería ser en un conversor analógico digital ideal. El error máximo al que puede llegar es de 1 LSB. La figura 2.6 muestra la curva de transferencia de un ADC ideal de 3 bits y su correspondiente error de cuantificación.

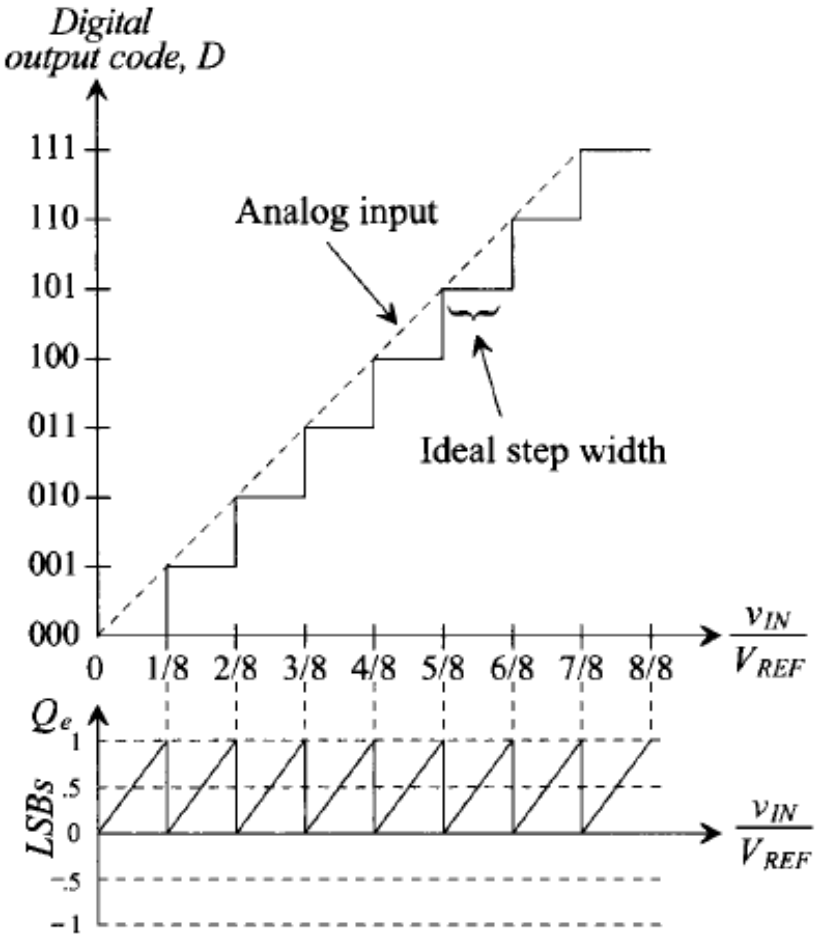


Figura 2.6: Aliasing en el dominio de frecuencia[3]

Una forma de reducir el efecto de cuantificación es centrarlo en cero de tal forma de que el error máximo sea de ± 0.5 LSB. Esto se consigue desplazando la respuesta real de la gráfica hacia la izquierda 0.5LSB como se muestra en la figura 2.7.

La señal de entrada no suele tener un valor constante ya que normalmente vendrá con un movimiento senoidal, cosenoidal u otro. Esto produce que el error de cuantificación vaya cambiando constantemente entre unos mismos valores.

Si asumimos que el rango de entrada es más amplio que el error de cuantificación, como estos valores no estan directamente controlados o afectados por el valor a la entrada podemos modelar el error de cuantifiación como un valor RMS. Este valor se calcula con la siguiente expresión[4]:

$$Q_{e,RMS} = \frac{V_{LSB}}{\sqrt{12}} \tag{2.3}$$

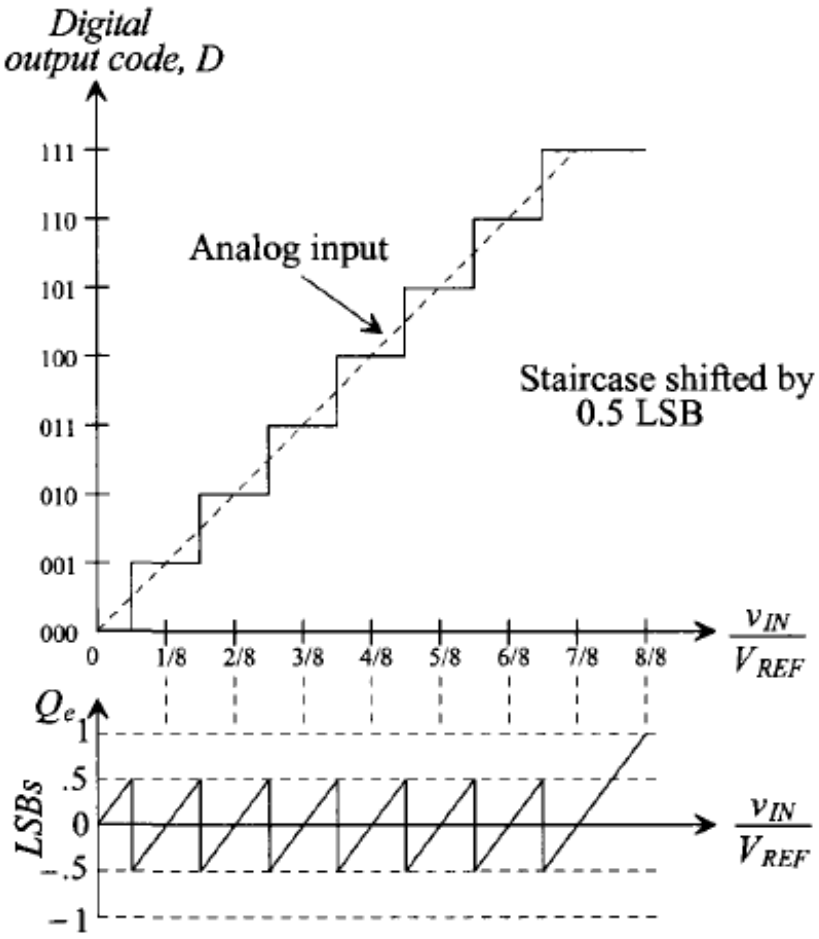


Figura 2.7: Aliasing en el dominio de frecuencia[3]

2.3. Características Estáticas

Las características estáticas o DC de un ADC nos permite analizar cuánto se desvían sus características de transferencia de la curva ideal. Estas especificaciones son particularmente importantes en aplicaciones donde el ADC es usado para medir eventos físicos lentos tales como la temperatura, la presión o el peso. Los parámetros que se van a tratar en esta sección son los siguientes: el Error de Offset y de Full-Scale, la No-Linealidad Diferencial, la No-Linealidad Integral y Códigos Perdidos.

2.3.1. Error de Offset y de Full-Scale

El error de Offset ocurre cuando hay una diferencia entre el valor de la primera transición de código y el valor de 0.5LSB. La figura 2.8 muestra este error y un dato interesante se obtiene al analizar también el error de cuanticación, ya que se vuelve ideal justo al pasar el voltaje de offset inicial.

Respecto al error de Full-Scale, este se obtiene calculando la diferencia entre la última transición de código del ADC a medir y la del ideal. Este efecto lo podemos observar en la figura 2.9 y ocurre debido al error de ganancia o de factor de escala, que es la diferencia entre las pendientes de las dos curvas representadas.

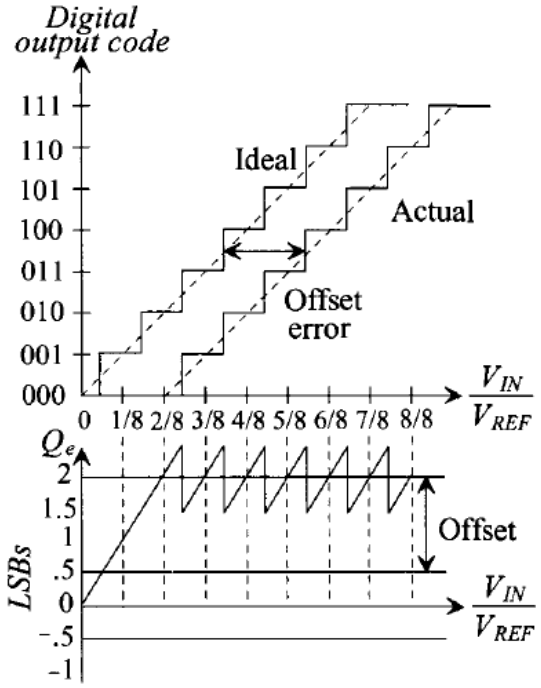


Figura 2.8: Error de Offset[3]

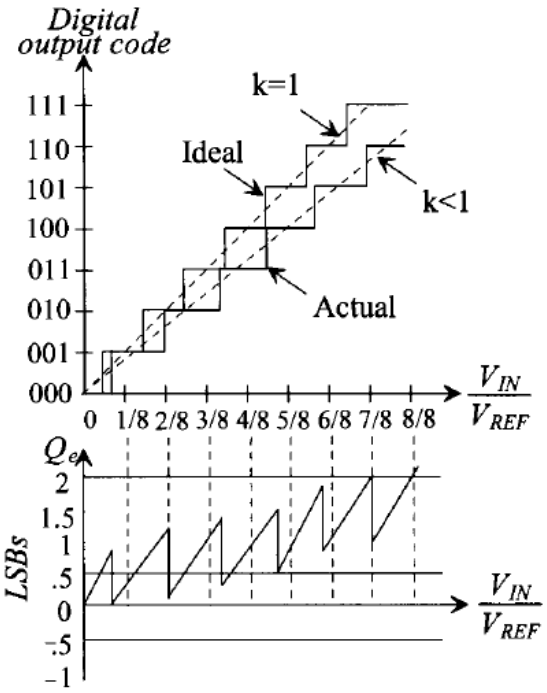


Figura 2.9: Error de Full-Scale[3]

2.3.2. No-Linealidad Diferencial (DNL)

La anchura de un valor de código de un ADC ideal es de 1LSB, pero en uno real podría no ser así. Con DNL estudiamos este error para cada valor de código:

$$DNL = Ancho_{ADCreal} - Ancho_{ADCideal} \tag{2.4}$$

Entonces cuando la anchura del código sea más pequeña que 1LSB, el DNL será negativo y si es más grande, positivo.

En la figura 2.10 se muestran estos dos casos. El código 001 tiene un ancho de 0.5LSB por lo que DNL = -0.5LSB. Respecto al código 101, observamos que su anchura es de 1.5LSB, obteniéndo así un DNL = +0.5LSB.

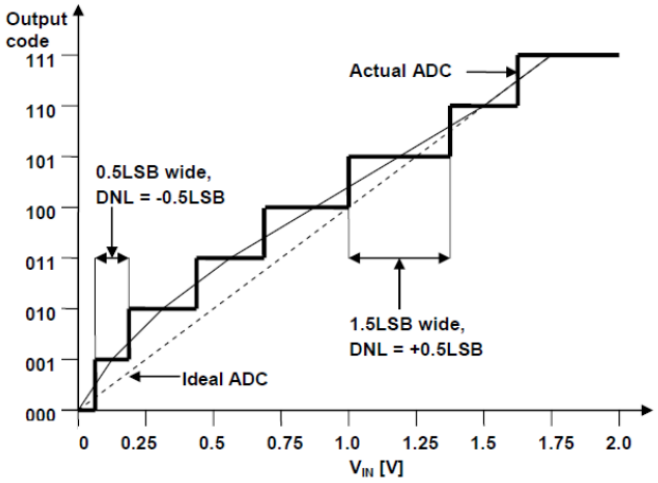


Figura 2.10: Error DNL[5]

2.3.3. No-Linealidad Integral (INL)

El INL en un conversor nos indica la cantidad de desviación de la curva de éste con la ideal. Además, puede ser interpretado como la suma de los DNLs. Por ejemplo, varios DNLs negativos elevarían la curva de transferencia y el INL en este caso disminuiría su valor.

Para analizar el INL, primero dibujamos una línea que une los puntos de la transición del primer y el último código en la función de transferencia. Hecho esto, se calcula la diferencia entre esta recta y los puntos de transición del conversor que queremos analizar. La figura 2.11 muestra un análisis de INL.

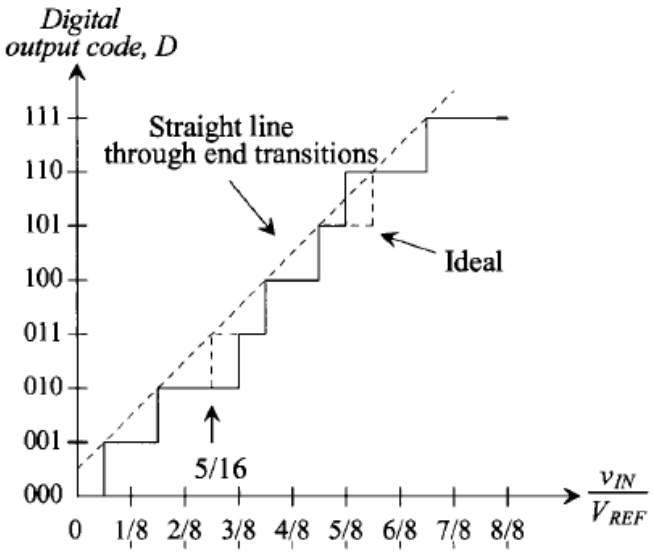


Figura 2.11: Error INL[3]

2.3.4. Código Perdido

Como su nombre indica, puede ser que cuando analicemos la curva de transferencia de un conversor, no aparezca la transición de un código y por lo tanto esté "perdido". Este efecto ocurre como consecuencia de tener un DNL de -1LSB.

En la figura 2.12 podemos observar como el paso de 101 no está y por lo tanto hay un código perdido. Otra deducción que podemos obtener en base a esta gráfica es que un paso de 2LSB (y por lo tanto un DNL de +1LSB) no asegura que haya un código perdido (aunque podría ocurrir).

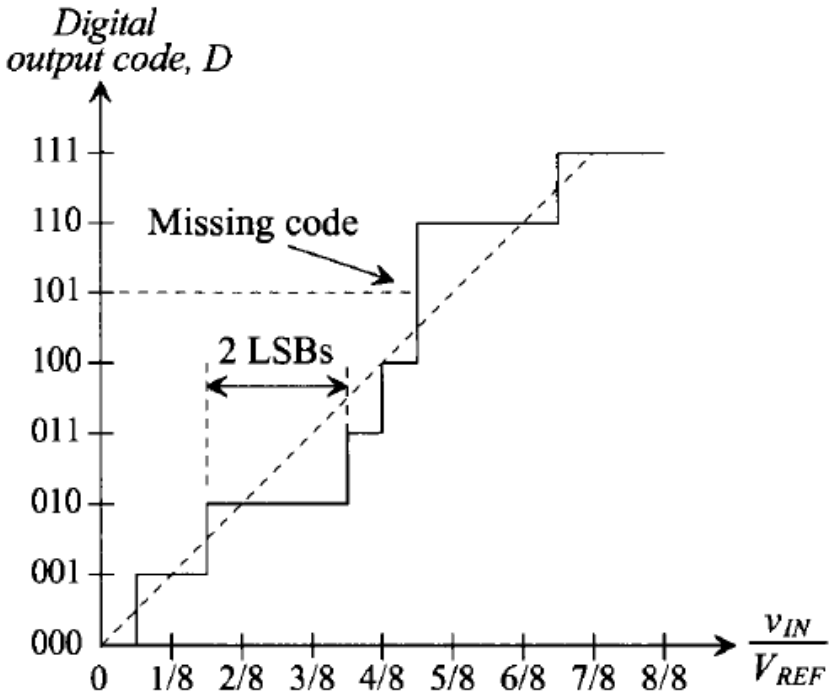


Figura 2.12: Código Perdido[3]

2.4. Caracterísitcas Dinámicas

Cuando la frecuencia de la señal de entrada incrementa, aparecen otras medidas que deben ser tratadas para determinar el rendimiento de un ADC. En estas caracterísitcas, el análisis del dominio de la frecuencia toma más importancia ya que las imperfecciones que aparecen en ella introducen ruido y distorsión en la muestra de salida.

Las características dinámicas o AC de un conversor nos indican por lo tanto cuanto ruido y distorsión ha sido introducido en una señal muestreada y la precisión del conversor para cierta frecuencia de entrada y de muestreo.

2.4.1. Relación Señal a Ruido (SNR)

La relación señal a ruido representa la proporción entre la señal que se transmite y el ruido que la corrompe. Esta es su expresión:

$$SNR = 20log(\frac{V_{inMAX}}{V_{noise}}) \tag{2.5}$$

Donde V_{inMAX} es la señal de entrada del ADC y V_{noise} (si consideramos que el conversor es ideal) será equivalente al error de cuantificación en RMS ($Q_{e,RMS}$). Si asumimos que la señal de entrada tiene un valor de pico a pico que equivale al valor de Full-Scale del voltaje de referencia del conversor, el valor RMS de V_{inMAX} es el siguiente:

$$V_{inMAX} = \frac{V_{REF}}{2\sqrt{2}} = \frac{2^N V_{LSB}}{2\sqrt{2}} \tag{2.6}$$

Si substituimos 2.6 y 2.3 en la ecuación 2.5 y la simplificamos, obtenemos la siguiente expresión:

$$SNR = 6{,}02N + 1{,}76 \tag{2.7}$$

Obteniendo así, una expresión que relaciona directamente el SNR y la resolución del conversor.

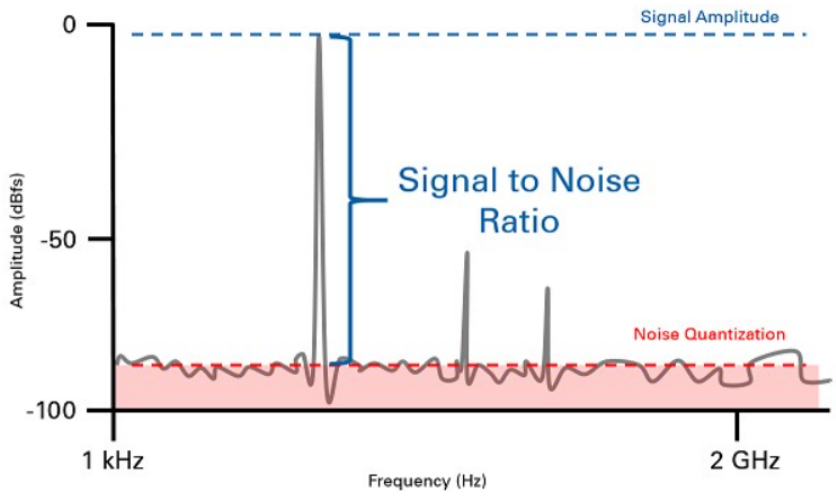


Figura 2.13: SNR en la FFT[6]

2.4.2. Distorsión Armónica (THD)

En sistemas eléctricos de corriente alterna aprarecen armónicos. Los armónicos son frecuencias múltiplos de la frecuencia fundamental de trabajo. THD mide la relación entre la suma de las potencias de los armónicos y la fundamental de la señal.

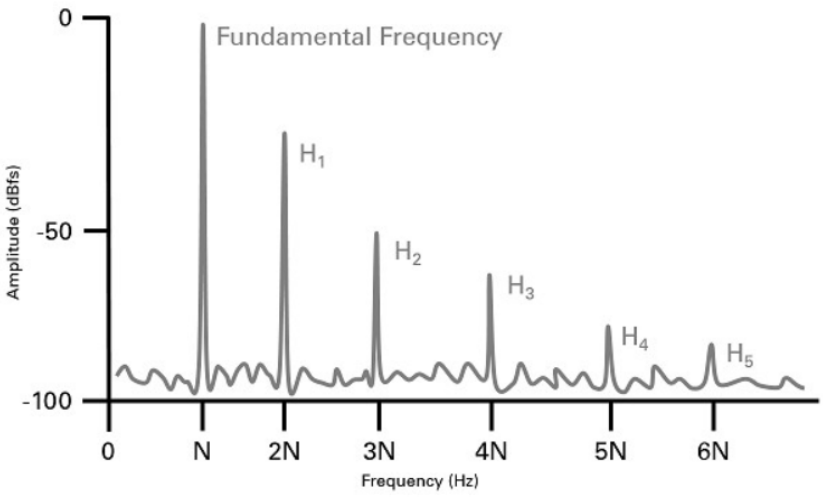


Figura 2.14: FFT con la frecuencia fundamental y cinco armónicos[6]

2.4.3. Relación Señal a Ruido y Distorsión (SINAD)

SINAD relaciona el valor RMS de la señal de entrada con el RMS equivalente de todo el ruido y distorsión en el análisis de la FFT. Su expresión:

$$SINAD = 20log(\frac{V_{inMAX}}{V_{NOISE} + V_{THD}}) \tag{2.8}$$

Además, también puede calcularse a través de los valores de SNR y THD:

$$SINAD = -10log(10^{-\frac{SNR}{10}} + 10^{\frac{THD}{10}}) \tag{2.9}$$

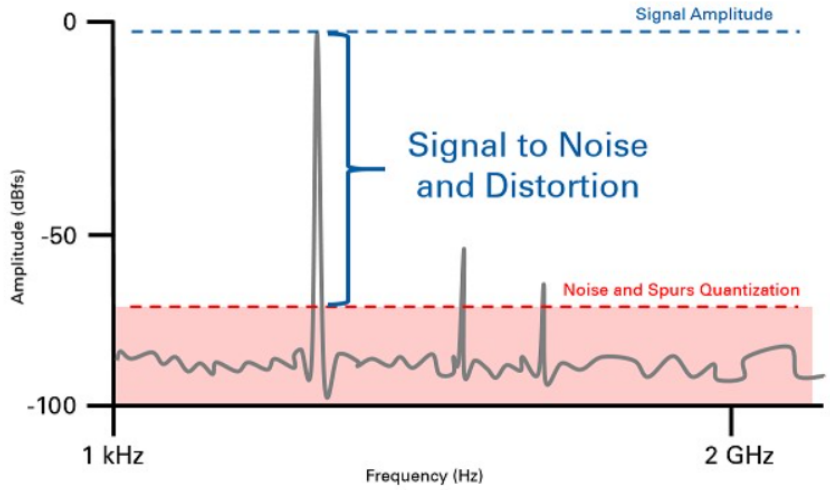


Figura 2.15: SINAD en la FFT[6]

2.4.4. Rango Dinámico Libre de Espurios (SFDR)

El SFDR es la relación entre nivel de la señal de entrada frente al espurio de mayor amplitud en la FFT. En la imagen 2.16 observamos un caso:

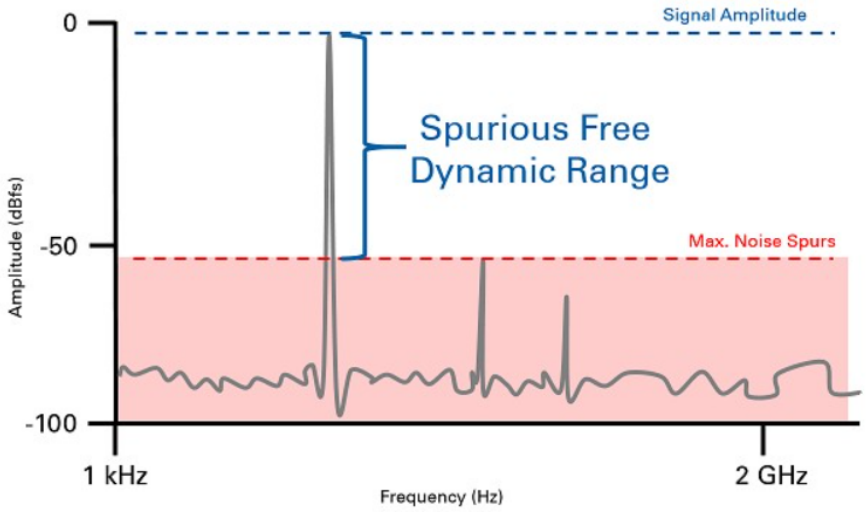


Figura 2.16: SFDR en la FFT[6]

La ecuación es por lo tanto:

$$SFDR = 20log(\frac{V_{inMAX}}{V_{SPUR}}) \tag{2.10}$$

2.4.5. Número Efectivo de Bits (ENOB)

El número efectivo de bits lo obtenemos en base a la ecuación 2.7 substituyendo N por ENOB y SNR por SINAD. De tal forma que:

$$ENOB = \frac{SINAD - 1{,}76}{6{,}02} \tag{2.11}$$

Este cálculo nos muestra como de cercano a un conversor ideal nuestro dispositivo está funcionando. Por ejemplo, si tenemos un conversor de 16 bits con un SINAD de 88dB, el valor de ENOB es de 14.32 bits y por lo tanto está produciendo aproximadamente la resolución de un conversor de 14 bits.

2.5. Arquitecturas ADC

En esta sección se discuten las cinco arquitecturas de ADC más utilizadas en el mercado y sus características. Además, al final se recogen los datos de cada ADC en un tabla para comparar sus prestaciones y aplicaciones.

2.5.1. Flash ADC

El conversores Flash destacan por su velocidad de conversión, ya que pueden llegar a los 5GS/s con 8 bits. Esto se consigue gracias a su arquitectura, mostrada en la figura 2.17, que tiene $2^N - 1$ comparadores de alta velocidad.

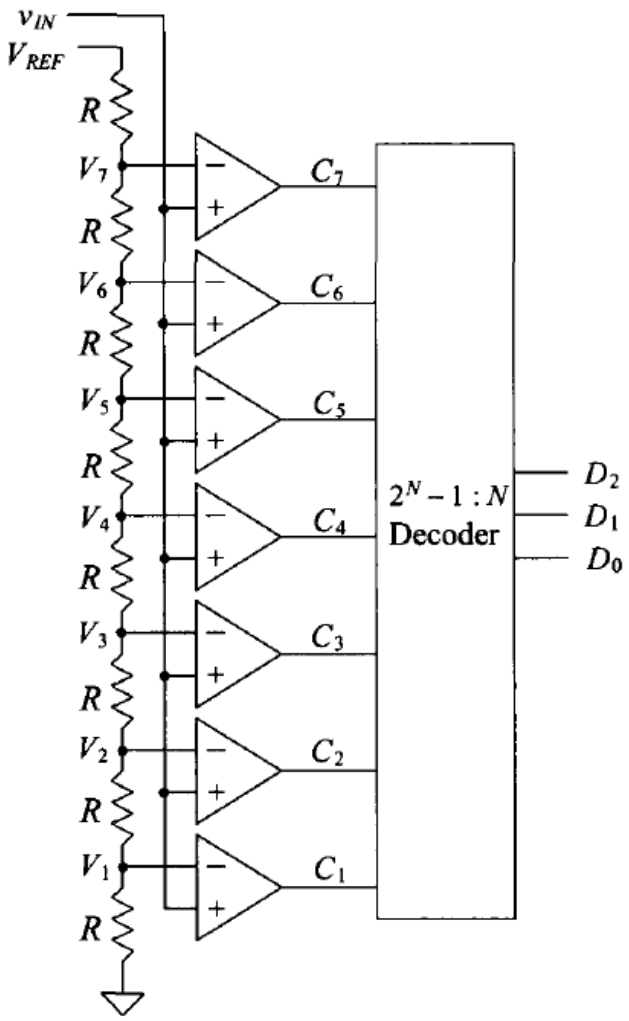


Figura 2.17: Arquitectura Flash ADC[3]

El hecho de que el número de componentes aumente exponencialmente con la resolución produce que el precio y la potencia consumida de estos dispositivos sea muy alta. Es por ello que suelen llegar hasta a los 8 bits de resolución.

2.5.2. Pipelined ADC

El tipo Pipelined se trata de un ADC de N etapas en serie, donde 1 bit está siendo convertido en cada estado. La figura 2.18 muestra la arquitectura de este conversor. Como se puede observar, cada etapa contiene un ADC de 1 bit, un Sample & Hold, un sumador y un amplificador de dos de ganancia.

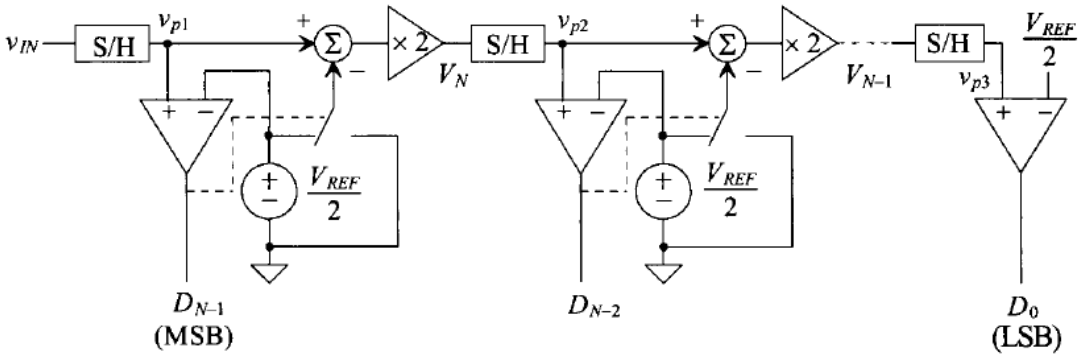


Figura 2.18: Arquitectura Pipelined ADC[3]

Al utilizar una arquitectura de tipo pipeline, esto implica que para que el conversor esté trabajando a una conversión por ciclo de reloj, tenga que tener un retraso inicial de N ciclos de reloj. Aun así destacan por su rapidez de conversión, pudiendo llegar a más de 200MHz de muestreo con resoluciones de no más de 6 a 8 bits.

2.5.3. SAR ADC

El ADC de aproximación sucesiva o SAR utiliza un algoritmo de búsqueda binaria para aproximar el valor de entrada al más cercano de los preestablecidos que tiene. Esta búsqueda se realiza durante N ciclos de reloj (figura 2.19. Destacan porque se pueden conseguir consumos y tamaños muy pequeños y porque su resolución suele abarcar de 8 a 16 bits fácilmente. Respecto a la velocidad de conversión, puede llegar hasta los 10MS/s.

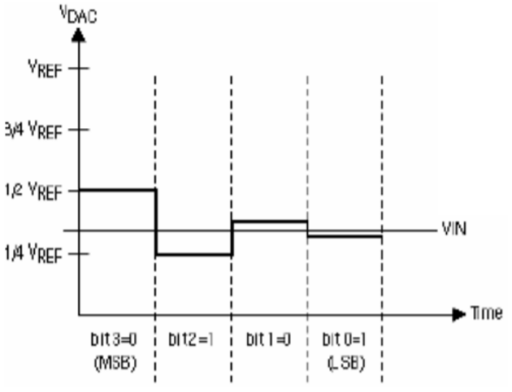


Figura 2.19: Ejemplo de conversión en un SAR de 4 bits[7]

Un SAR ADC está formado por un registro de aproximaciones sucesivas (SAR), un DAC, un comparador rápido y un circuito de sample & hold (aunque puede ir incluido en el circuito del DAC). En la figura 2.20 se muestra el diagrama de bloques comentado.

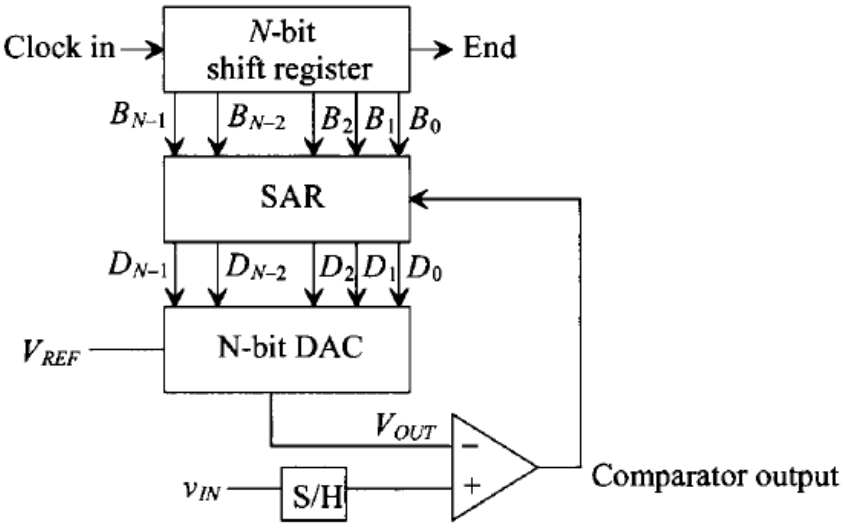


Figura 2.20: Arquitectura Pipelined ADC[3]

En el próximo capítulo entraré en más detalle sobre las arquitecturas posibles del SAR ADC y cómo funciona su algoritmo binario de búsqueda.

2.5.4. ADC Rampa o Integrador

Otro tipo de conversor analógico digital es el de Rampa o Integrador. Como indica su nombre, realiza la conversion integrando la señal de entrada para que posteriormente sea interpretada por un contador digital. Estos conversores destacan por su precisión pero tienen conversiones muy lentas.

Destacan dos tipos de arquitecturas en los integradores, la de una rampa y la de doble rampa. La figura 2.21 y la 2.23 muestran estos dos tipos de arquitecturas con sus respectivas funciones de transferencia 2.22 y 2.24.

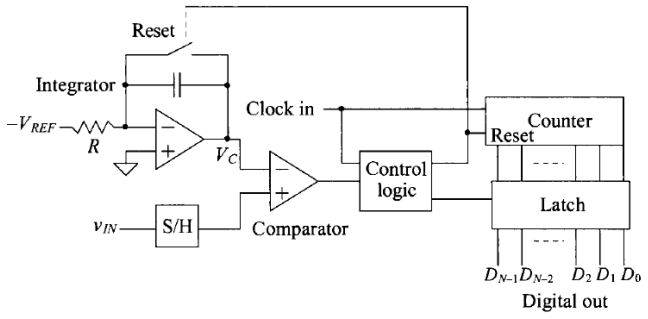


Figura 2.21: ADC de Una Rampa[3]

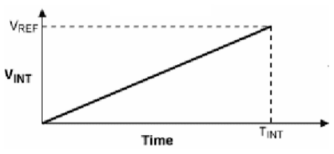


Figura 2.22: Gráfica[7]

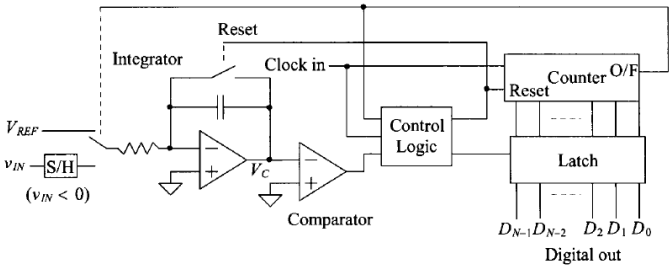


Figura 2.23: ADC de Doble Rampa[3]

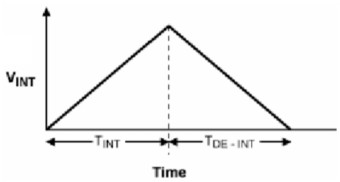


Figura 2.24: Gráfica[7]

En el de una rampa se integra el voltaje de entrada hasta que el de la salida del integrador sea igual que el de referencia. Además, la lógica para interpretarlo será más sencilla de implementar por su forma lineal. El problema de esta arquitectura es que la precisión depende de la frecuencia de reloj, la estabilidad de Vref y la tolerancia de los componentes pasivos R y C.

Una solución a este problema es utilizar una arquitectura de doble rampa, que es más compleja pero es robusta a la tolerancia de los componentes y a la frecuencia del reloj. El ADC doble rampa añade un período de descarga a la función de transferencia después de integrar V_{IN} un tiempo fijo.

2.5.5. ADC Sigma-Delta

El ADC Sigma-Delta destaca por proporcionar la resolución más alta (de 16 a 24 bits) para señales de baja frecuencia. Al igual que el ADC de doble pendiente, es otro tipo de digitalizador integrador. Los bloques que lo componen son un sumador, un integrador, un comparador, un DAC y un Filtro Digital. En la figura 2.25 se muestra la arquitectura.

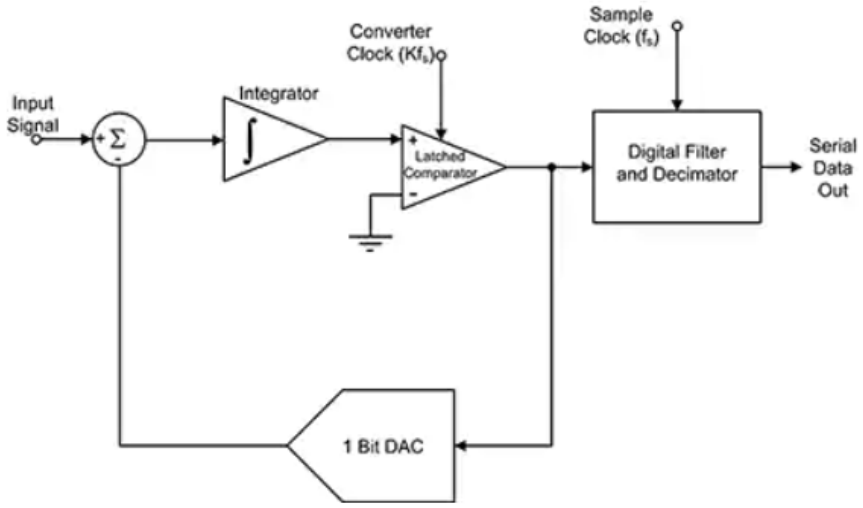


Figura 2.25: Arquitectura ADC Sigma-Delta[8]

Además de proporcionar resoluciones muy altas, el ADC Sigma-Delta se muestrea a una velocidad mucho más alta que la requerida para la velocidad de muestreo diseñada. Esto se entiende mejor observando la 2.25, ya que el reloj del comparador es más rápido por un factor de K que el de muestreo. Estas muestras adicionales del "sobremuestreo" se utilizan para proporcionar un filtrado digital de la salida del conversor. Y un diezmador restaura la $fmuestreo$ de salida a la de muestreo especificada.

2.5.6. Dispación de Potencia

A la hora de valorar un comparador es más conveniente analizar la potencia consumida por comparación que por operación. Esto es debido a que como hemos visto, muchas arquitecturas realizarán varios ciclos antes de tener una conversión final.

La disipación de energía por comparación depende del consumo del comparador y del número de iteraciones requeridas para realizar la conversión. Por otro lado, la de conversión dependerá de la arquitectura, la tecnología y las especificaciones requeridas.

La siguiente tabla muestra los consumos para cada tipo de arquitectura.

Arquitectura	Comparaciones por conversión	Consumo por comparación	Consumo por conversión
Flash	Alto	Alto	Alto
Pipelined	Medio	Medio	Medio
SAR	Medio	Alto	Bajo-Medio
Rampa	Bajo	Bajo	Bajo
Sigma-Delta	Alto	Medio	Medio

Tabla 2.1: Tabla Comparativa Consumo[9]

2.5.7. Comparación

A continuación, se muestra a modo de resumen una tabla con las características normales de uso de cada arquitectura nombrada en esta sección. Además, se nombran algunas aplicaciones de interés de cada una.

Arquitectura	Velocidad (muestras/s)	Resolución (bits)	Consumo	Aplicaciones
Flash	10M - 10G	4 - 8	Alto	Comunicaciones, Radar Osciloscopios digitales
Pipelined	5 - 200M	8 - 16	Medio	Captura de vídeo y audio Aplicaciones instrumentación
SAR	1 - 10M	8 - 18	Bajo - Medio	Adquisición de datos Aplicaciones de bajo coste
Rampa	2 - 100k	12 - 24	Bajo	Transductores de baja frecuencia (presión, temperatura...)
Sigma-Delta	10 - 100k	16 - 24	Medio	Audio digital Medición industrial

Tabla 2.2: Tabla Comparativa[7][10]

A la vista de los resultados, la elección del ADC vendrá propiciada principalmente entre la compensación de “velocidad - resolución”. Además, para elegir el ADC que mejor se desempeñe en cierta aplicación, se deben considerar características tales como el consumo, el tamaño y evidentenement el precio. Para nuestra aplicación el SAR ADC es la mejor elección que disponemos.

Capítulo 3

Estudio del SAR ADC

3.1. Introducción

Como había comentado en el capítulo de Introducción, los objetivos principales para este diseño son minimizar el tiempo de conversión y conseguir una resolución de 9 bits o más pero manteniendo un consumo y tamaño pequeño. La arquitectura que más satisface estas características es el SAR ADC debido a que permite desarrollar chips de tamaños pequeños, con poco consumo y que pueden llegar a velocidades y resoluciones moderadas.

En este capítulo voy a analizar las diferentes arquitecturas del SAR ADC y las de los bloques que lo componen tales como el DAC, el comparador y el bloque digital.

3.2. Arquitecturas del SAR ADC

3.2.1. SAR ADC con bloque de Track & Hold

Este tipo de SAR realiza el muestreo en un bloque de Track & Hold (o Sample & Hold). En la figura 3.1 se muestra su diagrama de bloques.

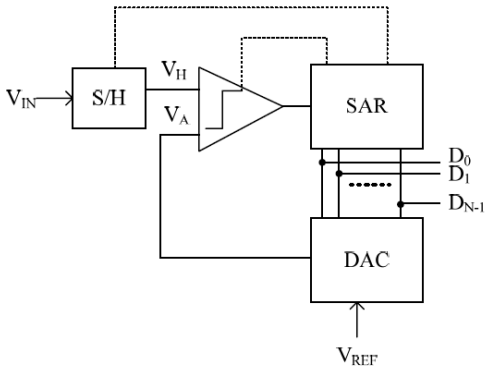


Figura 3.1: SAR ADC con bloque de T/H [11]

Se basa en el algoritmo de búsqueda binaria y va realizando comparaciones y aproximaciones durante Nbits ciclos de reloj. Para entender mejor como funciona, voy a mostrar un ejemplo de conversión para un SAR de 4 bits.

Datos: $V_{REF} = 6V$ $V_{IN} = 2{,}5V$ Resolución = 4 bits

Nota: $V_{DAC} = \frac{V_{SARb} \cdot V_{REF}}{2^{bits}}$

Empieza el SAR poniendo el MSB a 1:

1. SAR = 1000 $\rightarrow$ $V_{DAC} = \frac{8 \cdot 6}{16} = 3V$ $\rightarrow$ $3V > 2{,}5V$ $\rightarrow$ $Comp = 0$

2. SAR = 0100 $\rightarrow$ $V_{DAC} = \frac{4 \cdot 6}{16} = 1{,}5V$ $\rightarrow$ $1{,}5V < 2{,}5V$ $\rightarrow$ $Comp = 1$

3. SAR = 0110 $\rightarrow$ $V_{DAC} = \frac{6 \cdot 6}{16} = 2{,}25V$ $\rightarrow$ $2{,}25V < 2{,}5V$ $\rightarrow$ $Comp = 1$

4. SAR = 0111 $\rightarrow$ $V_{DAC} = \frac{7 \cdot 6}{16} = 2{,}625V$ $\rightarrow$ $2{,}625V > 2{,}5V$ $\rightarrow$ $Comp = 0$

5. SAR = 0110 $\rightarrow$ $Vdetectado = 2{,}25V$

Como se puede observar, el resultado no es exacto debido a que un ADC aproxima a uno de sus nivel de cuantificación el voltaje de entrada. Por lo que cuanto mayor sea la resolución, mayor será la precisión de este. El LSB de este conversor sería de: LSB = $\frac{FE}{2^{bits}} = \frac{6V}{16} = 0{,}375V$.

La principal ventaja de este SAR es que la Capacidad de Entrada es independiente del DAC, reduciendo así el ruido kick-back del reloj a la entrada.

El kickback es el ruido que añade el comparador durante las grande variaciones de voltaje que ocurren en una de sus fases de funcionamiento. Este ruido se acopla a la línea en la que están conectadas sus entradas, añadiendo así parásitos e introduciendo un offset en la medida. Cuanto mayor sea la impedancia de la línea, más grande será su efecto.

Sin embargo, el hecho de tener un bloque extra para realizar el muestreo produce un aumento de la potencia consumida y el tamaño del chip. Y como el consumo es uno de los objetivos principales a minimizar, no usaré esta arquitectura y por lo tanto no estudiaré las diferentes formas de implementar un Track & Hold.

3.2.2. SAR ADC sin bloque de Track & Hold

En esta arquitectura, el bloque de Track & Hold va incluído en el DAC del sistema. El mejor tipo de conversor digital-analógico para incluir el muestreo es el capacitivo, es decir, un CDAC.

Los CDAC utilizan técnicas de conversión con condesadores de pesos binarios, esto nos proporciona varias ventajas frente a otros tipos de DAC. Entre ellas, los errores de fabricación suelen ser menores debido a las relaciones binarias entre condensadores y además, los condensadores nos permiten utilizar técnicas de carga - distribución que nos proporcionan un consumo muy pequeño.

En la figura 3.2 se muestra un ejemplo de este tipo de arquitectura con el CDAC con muestreo incluido.

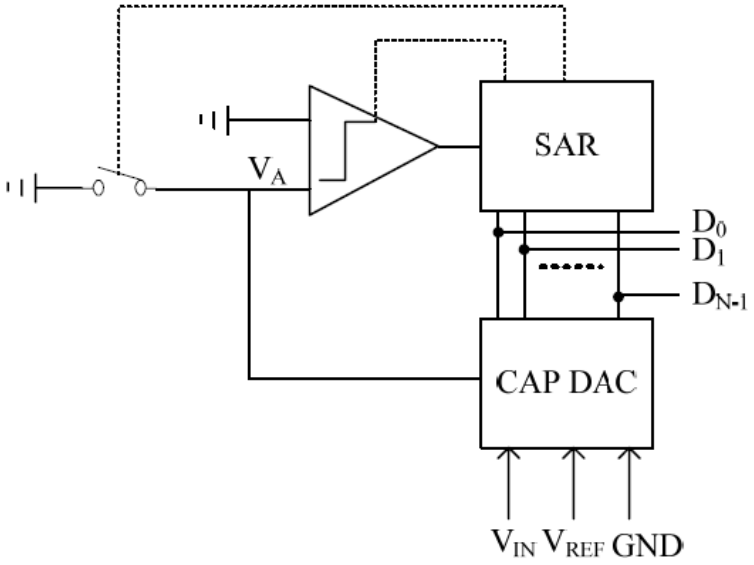


Figura 3.2: SAR ADC sin bloque de T/H [11]

Respecto al proceso de conversión, este variará dependiendo de la arquitectura elegida en el CDAC por lo que entraré en más detalle en el capítulo de implementación.

Las ventajas y desventajas de esta arquitectura son las contrarias que las del SAR con bloque de T/H. Ya que evidentemente al tener un bloque menos, consumirá menos potencia y su tamaño será menor. Sin embargo, C_{IN} estará directamente en relación con el DAC y como presenta mayor impedancia, el kickback tomará más importancia.

A la vista de los resultados, esta es la mejor arquitectura para las características que necesitamos en el ADC de este TFG.

3.3. DAC Capacitivo (CDAC)

3.3.1. Condensadores de Pesos Binarios (BWC)

Este arquitectura consiste en una serie de condensadores relacionados entre sí por potencias de dos donde el primer condensador que va conectado de forma diferente que los demás es el Dummy, y toma el valor de C. Como el valor total de estos condensadores aumenta de forma exponencial con la resolución, la precisión del DAC BWC suele estar limitada entre 8 y 10 bits. La figura 3.3 muestra esta arquitectura.

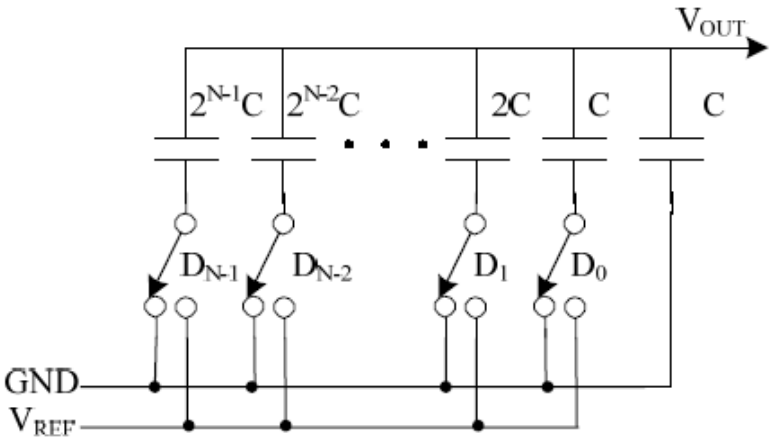


Figura 3.3: Arquitectura del DAC BWC [11]

3.3.2. BWC de Dos Etapas (TWC)

TWC soluciona el problema del BWC introduciendo un condensador de acople en medio del array de condensadores, reduciendo así el valor de los condensadores más grandes. La figura 3.4 muestra un ejemplo de un DAC de 8 bits usando TWC.

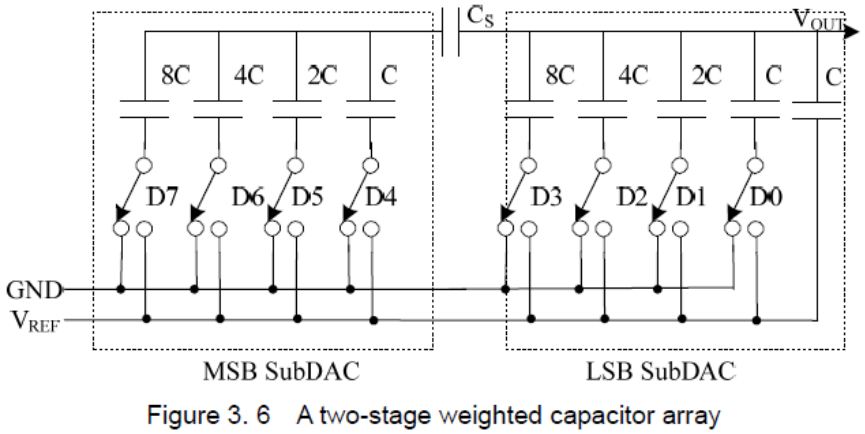


Figure 3. 6 A two-stage weighted capacitor array

Figura 3.4: Arquitectura del DAC TWC [11]

3.3.3. C-2C Ladder

Como se puede observar, la arquitectura C-2C mostrada en la figura 3.5 facilita el matching de los condensadores.

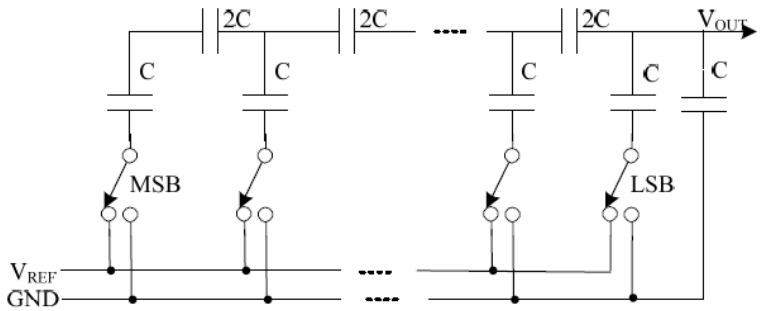


Figura 3.5: Arquitectura del DAC C-2C [11]

Además, como tiene una RC menor que las otras arquitecturas, se obtienen mejores velocidades y su baja capacidad reduce el consumo. Sin embargo, las capacidades parásitas que aparecen en los nodos producen una pérdida de la linealidad.

3.4. Comparador

El comparador es uno de los bloques más críticos en los ADC ya que se encarga de comparar el valor de dos entradas analógicas para obtener una salida digital en base a qué entrada es mayor. A lo largo de esta sección se van a discutir las tres formas básicas de implementar este bloque.

3.4.1. Comparador de Lazo-Abierto

El comparador de lazo-abierto u open-loop en inglés es básicamente un amplificador con entrada diferencial y alta ganancia, y una salida single ended de gran variación. Un ejemplo de comparador de lazo-abierto sería un OpAmp de dos etapas sin compensación como el ilustrado en la figura 3.6.

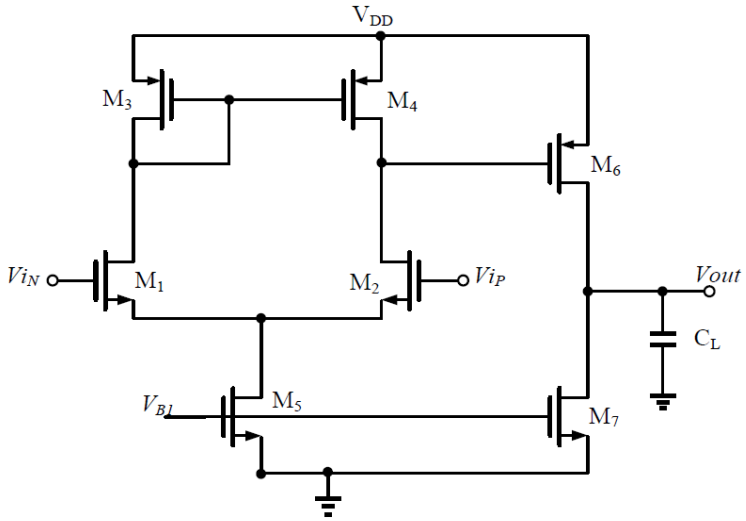


Figura 3.6: Comparador de Lazo-Abierto [12]

Estos tipos de comparadores suelen presentar varias limitaciones. La diferencia mínima de voltaje es decidida en base a la resolución conversor, por lo que voltajes muy pequeños requerirán grandes ganancias en los comparadores. En la figura 3.7 se muestra la respuesta en frecuencia de comparadores en lazo-abierto de una etapa. Como se puede observar, para conseguir grandes ganancias, tenemos que tener bajas frecuencias para mantener la relación de ganancia/unidad.

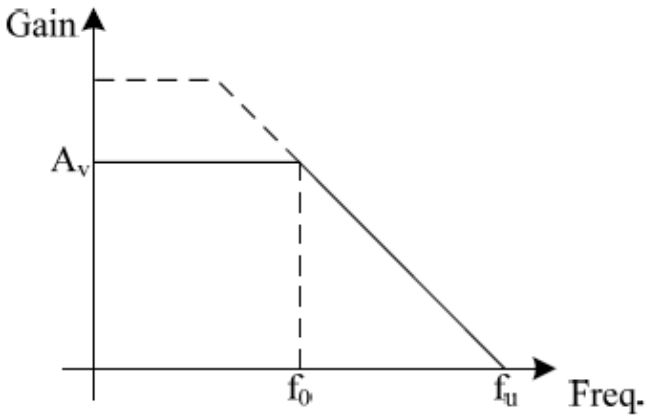


Figura 3.7: Función de transferencia de un OpAmp de una etapa [11]

Para solucionar el problema de la baja ganancia se pueden utilizar varios comparadores en cascada con baja ganancia pero al mismo tiempo estaríamos introduciendo más offsets que podrían inducir más problemas estáticos. Además de que consumiríamos más potencia. Es por estas razones que estos tipos de comparadores no son los indicados para conversores analógicos-digital que requieran altas velocidades y bajos consumos.

3.4.2. Comparador Latched con Pre-Amplificador

Este tipo de comparadores están formados por un circuito latch y una etapa de pre-amplificación, la figura 3.8 muestra un ejemplo.

En esta configuración, la etapa de amplificación atenúa el offset del latch por su ganancia. Esto beneficia a aplicaciones que requieran un offset de entrada bajo y que requieran la detección de pequeñas diferencias de voltaje en sus entradas.

La desventaja es que al usar latches estáticos pues consumen potencia estática y esto no es atractivo para aplicaciones de baja potencia.

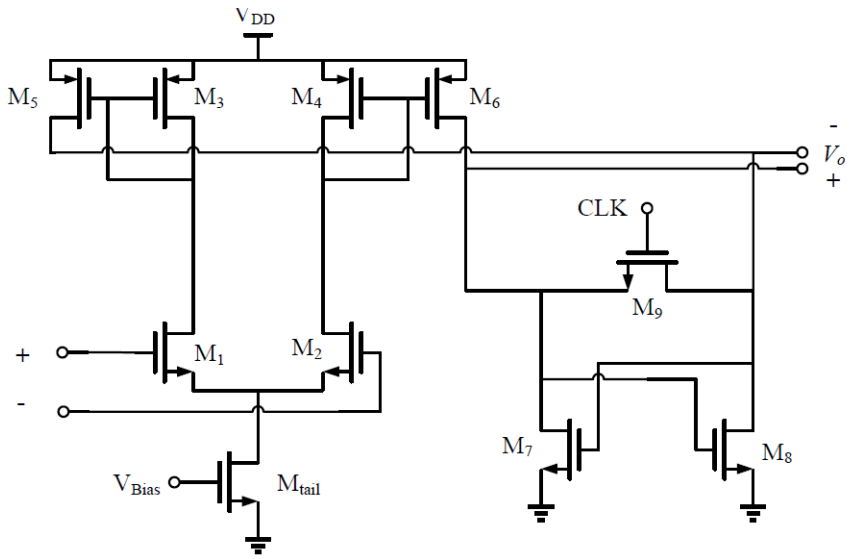


Figura 3.8: Comparador Latched con Pre-Amplificación [12]

3.4.3. Comparador Latched Dinámico

Los comparadores implementados con latches hacen uso de la combinación de amplificación y de realimentación positiva. En la figura 3.9 se muestra como funcionan estos comparadores.

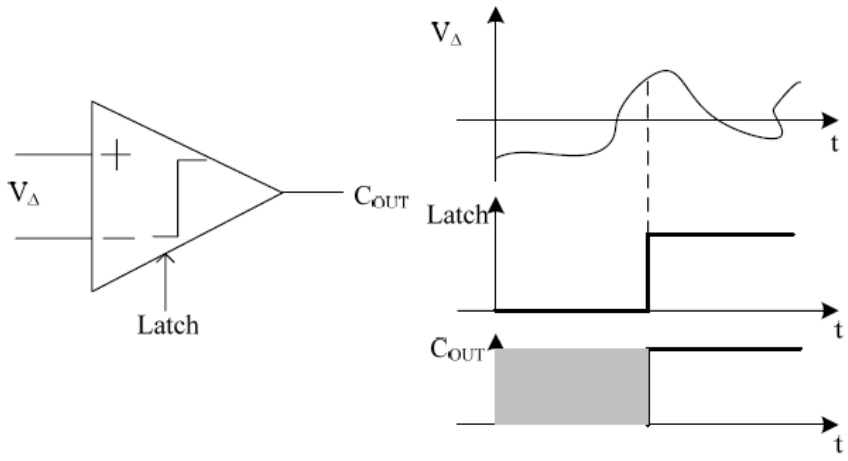


Figura 3.9: Funcionamiento del Comparador Latched Dinámico [11]

Su operación se divide en dos fases de funcionamiento, una de reseteo en la que el comparador carga o descarga los nodos dependiendo de la configuración y otra de regeneración en la que se hace la comparación a la que se le aplica la realimentación positiva. Obteniéndo así un valor digital

a la salida del comparador. En la figura 3.10 se muestra un posible esquemático para implementar esta arquitectura.

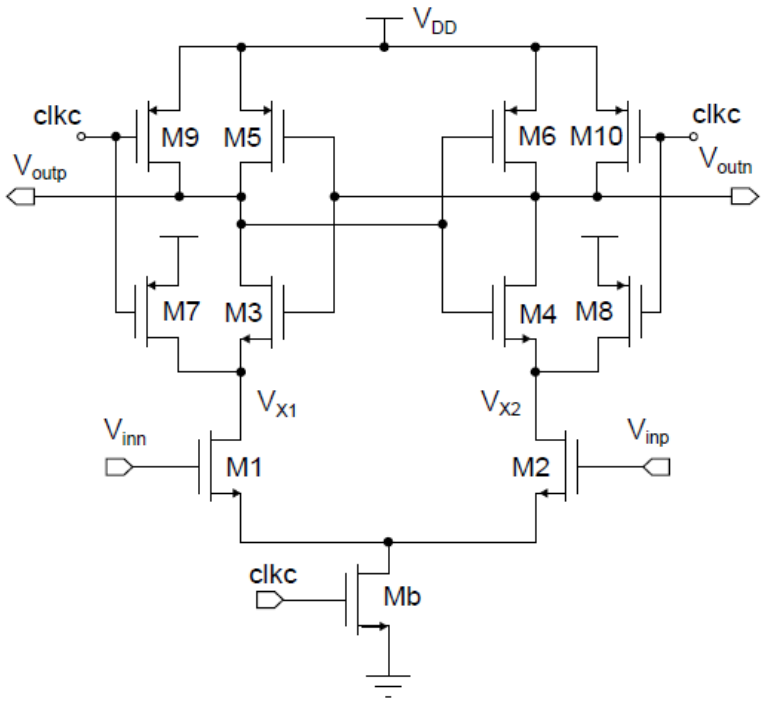


Figura 3.10: Comparator Latched Dinámico [13]

Una de las ventajas de esta arquitectura es que no consume potencia en la fase de regeneración y además, en la de reset no se consume potencia estática. Es por ello que los comparadores latched dinámicos son los más eficientes en consumo de potencia. Sin embargo, introducen offsets de entrada grandes reduciéndo así su atractivo para aplicaciones de conversores de alta resolución.

Para solucionar los problemas de offset en estas arquitecturas se emplean técnicas de cancelación de offset con varactores, más etapas o podríamos usar un pre-amplificador también para reducirlo. Obviamente la elección de qué método usar dependerá de la aplicación a la que será sometido.

3.5. Bloque Digital

El bloque digital de un SAR es un registro de aproximaciones sucesivas que se encarga de la lógica de control. El algoritmo que emplea se basa en uno de búsqueda binaria. En esta sección se van a discutir las dos principales formas de implementar este bloque, la propuesta de Anderson y la de Rossi y Fucili.

3.5.1. Lógica propuesta por Anderson

En la figura 3.11 se muestra la lógica propuesta por Anderson. Está formada por un contador de anillo y registros de desplazamiento. Y como se puede observar, emplea un total de 2N flip-flops.

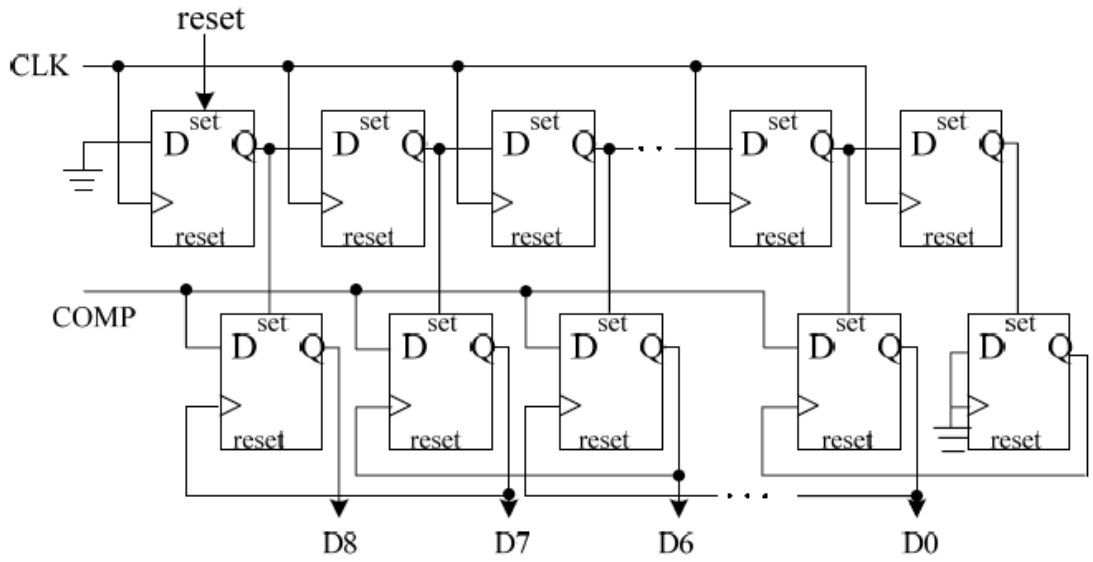


Figura 3.11: Esquemático de la lógica de Anderson [11]

La lógica empieza poniendo la señal de reset a 1 para iniciar todos los flip flops a cero. Luego se pone a 1 el primer flip-flop, suponiendo así un valor de output para que sea comparado con el de entrada. Este valor se compara y se decide si el MSB es 1 o 0. Cuando tenemos el valor del MSB, se pone y se desplaza el 1 al siguiente número, y así sucesivamente hasta tener todos los números.

Para entenderlo mejor, he creado la tabla 3.1 a modo de ejemplo para un convesor de 3 bits. Donde los elementos C son los valores que se comparan con el de la entrada.

Ciclo	Reset	Output			Comparación
0	1	0	0	0	-
1	0	1	0	0	C2
2	0	D2	1	0	C1
3	0	D2	D1	1	C0
4	0	D2	D1	D0	-

Tabla 3.1: Lógica de Anderson

3.5.2. Lógica propuesta por Rossi y Fucili

Para reducir el número de registros de la lógica de Anderson, Rossi y Fucili propusieron la mostrada en la figura 3.12. Reduciendo así, el número de registros que se utilizan a la mitad.

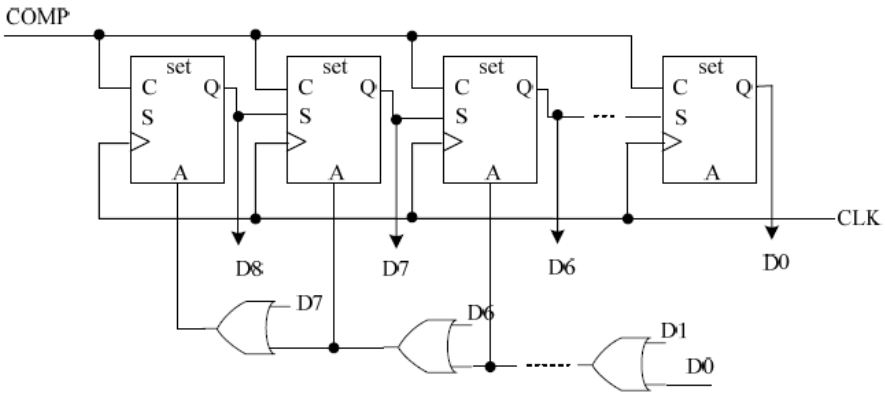


Figura 3.12: Esquemático de la lógica de Rossi y Fucili [11]

En la figura 3.13 se puede observar la estructura de los registros. Estos añaden a los FF tipo D, un multiplexor y un decodificador. La lógica de decodificación se muestra en la tabla 3.2.

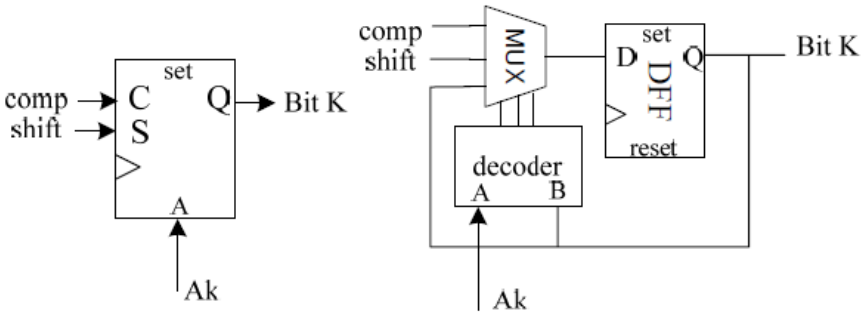


Figura 3.13: Estructura interna de los registros [11]

A	B	Operación
1	-	Memoriza
0	1	Carga el dato Comp
0	0	Desplaza a la derecha

Tabla 3.2: Lógica de Decodificación

Capítulo 4

Diseño del SAR ADC

4.1. Introducción

En este capítulo voy a mostrar como he diseñado el SAR ADC propuesto junto al kit de Cadence XFAB y la metodología empleada para realizarlo, explicando paso a paso las decisiones y cambios que se han ido haciendo para conseguir las prestaciones deseadas.

El SAR propuesto es de 10 bits para asegurar un número efectivo de 9 bits o más. Añadir un bit más solo añade un ciclo más a la conversión y nos baja el V_{LSB} de 1.953125mV a la mitad, 0.9765625mV, dando así un margen de [-LSB LSB] para el error de cuantifiación.

La figura 4.1 muestra el diagrama de bloques del SAR ADC. Como se puede observar, el CDAC realiza también el muestreo de la entrada. Encima de él se encuentra el bloque digital de aproximaciones sucesivas y al lado el bloque comparador que contiene un comparador latch dinámico y un SR latch que actúa como buffer de salida y mantiene el dato hasta que se realiza otra conversión.

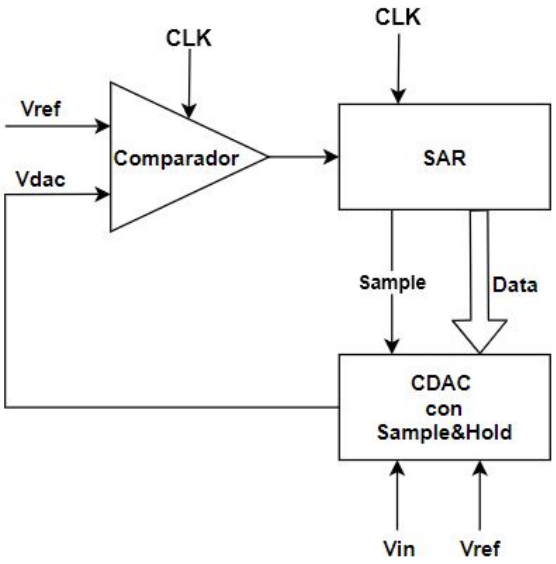


Figura 4.1: Diagrama del SAR ADC diseñado

4.2. CDAC con muestreo

Antes de comenzar con la metodología empleada en el diseño, voy a explicar cómo funciona el bloque CDAC con muestreo elegido. Para implementarlo, se ha utilizado la arquitectura de Pesos Binarios (BWC) con muestreo. Esta arquitectura diseñada para 10 bits la podemos observar en la figura 4.2:

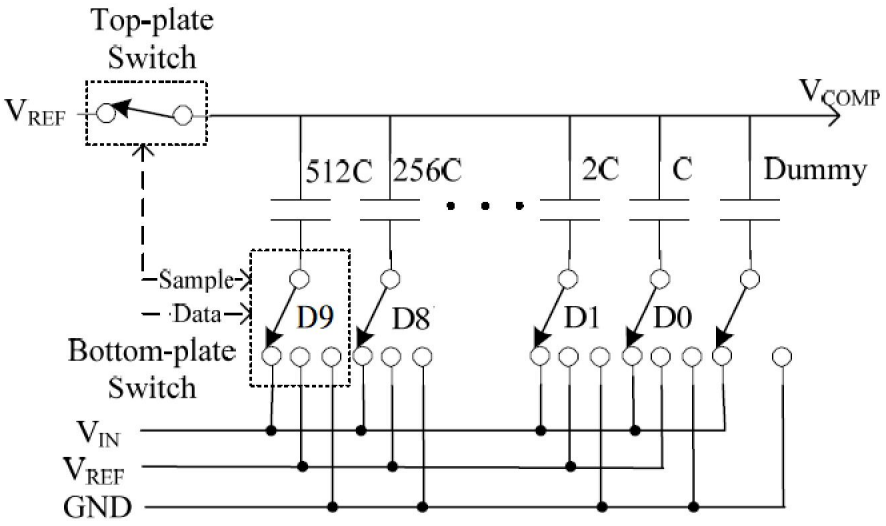


Figura 4.2: Diagrama del BWC con muestreo

Como se puede observar, la figura 4.2 añade a la circuitería mostrada en 3.3 la señal de entrada y un switch en la parte superior que conecta la línea que va al comparador a V_{REF}. Su funcionamiento tiene tres fases:

- Muestreo: cuando la señal de control Sample es 1:
  - Switch de Vcomp: conecta la línea a V_{REF}.
  - Switch Condensador: se conectan a V_{IN} todos los condensadores.
  - De esta forma se cargan los condensadores a $V_{REF} - V_{IN}$.
- Hold: cuando Sample vuelve a 0 y Data es 0:
  - Switch de Vcomp: deja la línea sin conectar.
  - Switch Condensador: se conectan a tierra.
  - De esta forma los condensadores cargados se mantienen a $V_{REF} - V_{IN}$.
- Redistribución: es la fase en la que empieza la conversión:
  - Switch de MSB se conecta a V_{REF} poniendo su señal de control Data a 0. Como vale la mitad de la capacidad total Vcomp se carga a $V_{REF} - V_{IN} + 0{,}5 \cdot V_{REF}$.
  - Este valor se compara con V_{REF} en el comparador y el bloque SAR decide si mantener Data a 0 o 1.

- Después se hará el mismo proceso para cada condensador excepto para el Dummy que tiene el mismo valor que C y se mantiene a tierra durante la conversión.

- Una vez acabada la conversión, en Vcomp deberíamos tener el valor de:

$$Vcomp = V_{REF} - V_{IN} + D9 \cdot \frac{V_{REF}}{2} + D8 \cdot \frac{V_{REF}}{2^2} + \cdots + D1 \cdot \frac{V_{REF}}{2^9} + D0 \cdot \frac{V_{REF}}{2^{10}}$$

Una vez conocido el funcionamiento de cada bloque a diseñar paso a mostrar la metodología empleada para realizar el diseño de la figura 4.1.

4.3. Metodología de diseño

El enfoque empleado ha sido de arriba hacia abajo, lo que en inglés se llamaría “top-down approach”, es decir, se ha partido de bloques ideales y se han ido detallando y refinando uno a uno hasta lograr el diseño completo.

Por lo tanto empecé diseñando el bloque ideal para los switches y el comparador mediante Verilog-A y les creé un símbolo a cada uno. Más tarde, implementé el SAR con Verilog y una vez hecho esto, pasé a diseñar el CDAC real y con la ayuda de los otros bloques ideales pude ir paso a paso validando su funcionamiento.

Una vez diseñado y validado un bloque real, se pasa al siguiente y así paso a paso hasta a lograr un sistema real.

4.4. Bloques ideales

A continuación se muestra en cada subsección el código, el testbench y el gráfico obtenido después de una simulació transient de cada bloque ideal diseñado.

4.4.1. Comparador

El comparador toma las entradas vin y vref. Si vin >vref la salida es igual a la señal conectada en vdd del comparador, sino es igual a la conectada en gnd.

El código de la figura 4.3 implementa el comparador:

```
// VerilogA for Prueba, Comparator, veriloga

`include "constants.vams"
`include "disciplines.vams"

module Comparator(vin, vout, vdd, vss, ref);

input vin, ref;
inout vss, vdd;
output vout;

parameter delay = 0, ttime = 1p;

electrical vin, ref, vdd, vout, vss;
real result;

    analog begin
    @(cross((V(vin) - V(ref)),0))
        if(V(vin) > V(ref))
            result = V(vdd);
        else
            result = V(vss);

    V(vout) <+ transition(result, delay, ttime);

    end


endmodule
```

Figura 4.3: Código del comparador ideal

Una vez diseñado, le creé un símbolo y un testbench para comprobar su funcionamiento:

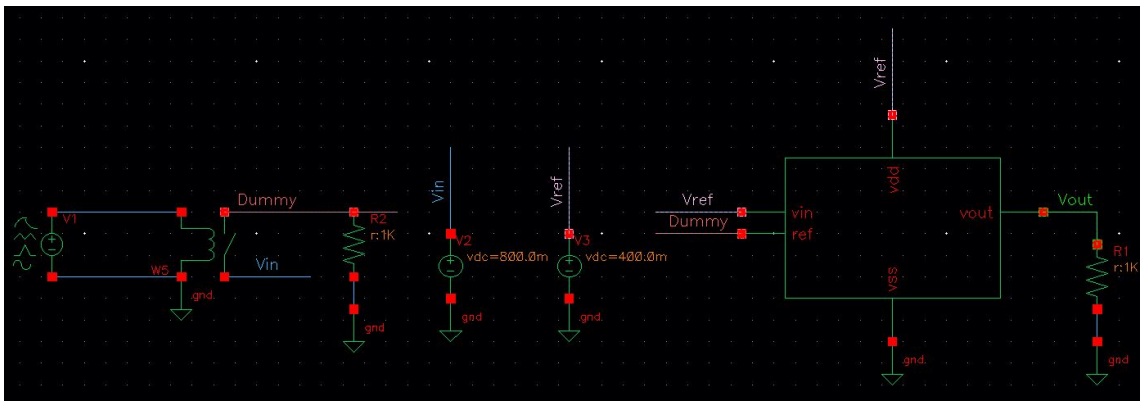


Figura 4.4: Testebench del comparador ideal

Y con un transient analicé su respuesta:

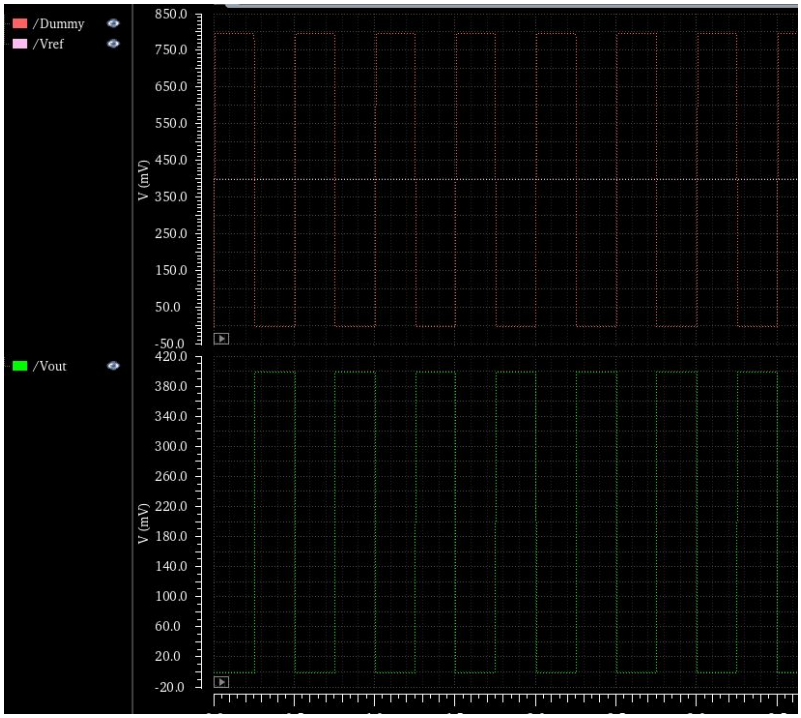


Figura 4.5: Simulación del comparador ideal

A la vista de los resultados funciona correctamente ya que cuando Vref >Dummy la salida es igual a “1”, y “0” en caso contrario.

4.4.2. Switch

Se ha diseñado solamente el switch que decide entre las entradas Vin, Vref y GND del CDAC debido a que los demás se pueden implementar fácilmente con los componentes de Cadence.

Antes de mostrar su código y verifiación voy a explicar cómo funciona. Su diagrama de bloques se muestra en la figura 4.6 y está formado por un inversor, una puerta nand, una nor, dos puertas de transmisión para las señales V_{IN} y V_{REF} y un NMOS para GND.

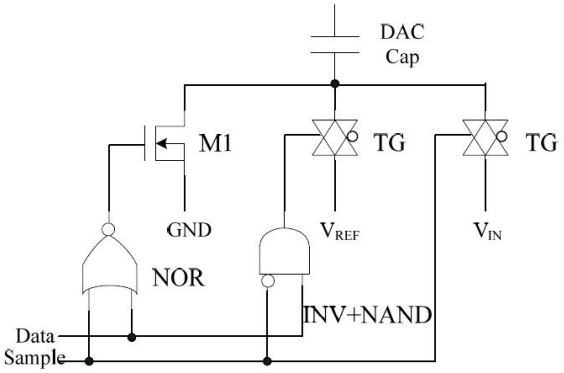


Figura 4.6: Diagrama de bloques del switch [11]

Su tabla de verdad es la siguiente:

Sample	Data	Salida
1	-	V_{IN}
0	0	GND
0	1	V_{REF}

Tabla 4.1: Lógica del switch

Una vez explicado su funcionamiento, paso a mostrar su código y verificación.

```
// VerilogA for Prueba, SWITCH_VerilogA, veriloga

`include "constants.vams"
`include "disciplines.vams"

module SWITCH_VerilogA(vin, vref, gnd, sample, data, vout);

input sample, data;
inout vin, vref, gnd;
output vout;

parameter delay = 0, ttime = 1p;

electrical vin, vref, gnd, sample, data, vout;
real result;

    analog begin
        if(V(sample) > V(gnd))
            result = V(vin);
        else
            begin
            if(V(data) > V(gnd))
                result = V(vref);
            else
                result = V(gnd);
            end

    V(vout) <+ transition(result, delay, ttime);

    end


endmodule
```

Figura 4.7: Código del switch ideal

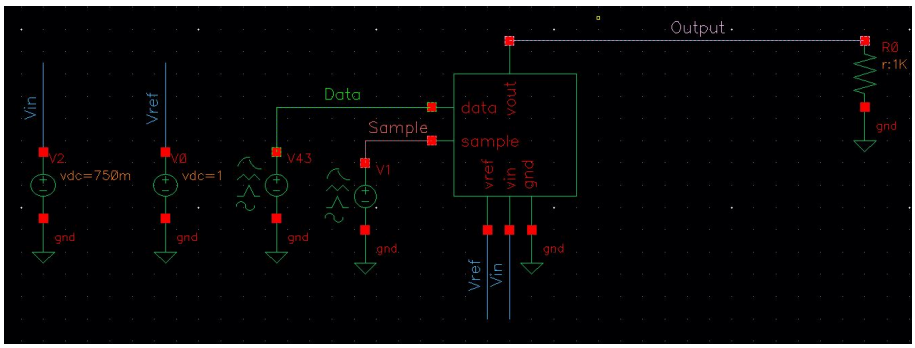


Figura 4.8: Testebench del switch ideal

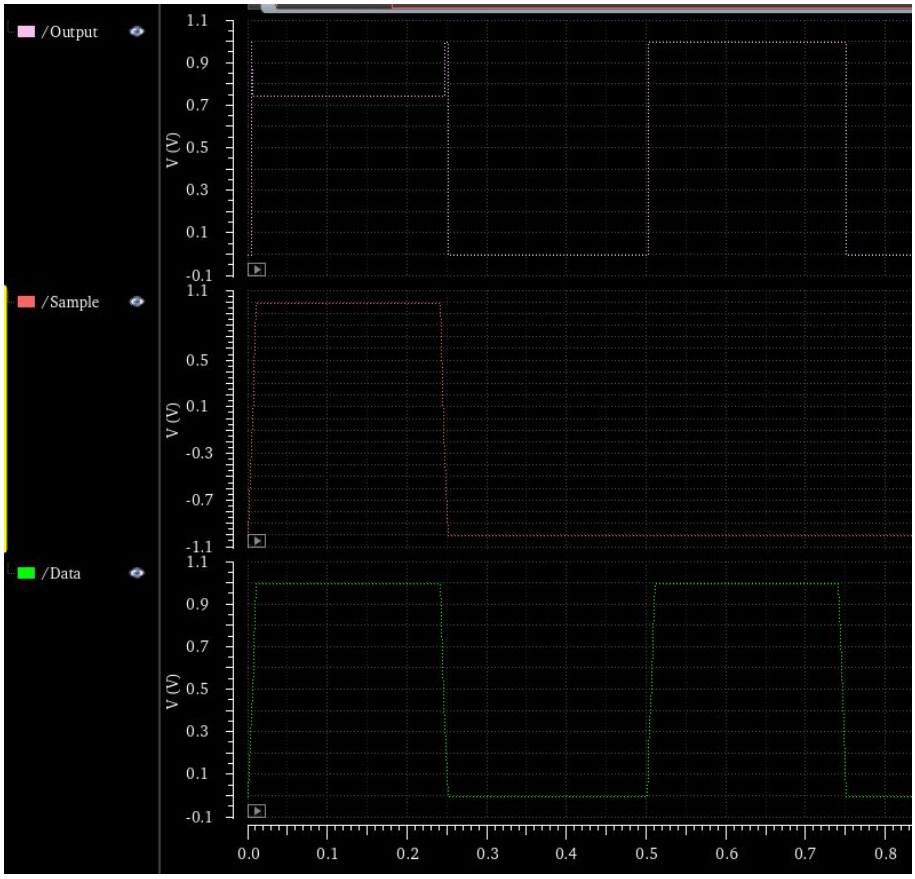


Figura 4.9: Simulación del switch ideal

Como se puede observar, cuando Sample está activo, se toma V_{IN} que es igual a 750mV. Con Sample a 0, si Data es 0 se toma GND y si no, toma V_{REF} que es 1V en este caso.

4.5. SAR

El código está basado en el que utilizan en este proyecto [14] y adaptado para este diseño.

El bloque SAR está formado por 3 entradas y 4 salidas. Las entradas son:

- clk: el reloj
- go: mientras go = 1 se realiza la conversión
- cmp: la salida del comparador

Las salidas:

- sample y data: señales de control del CDAC
- result: resultado final de la conversión

- stateOUT: indica en qué estado se encuentra el SAR
- valid: indica cuando se ha terminado la conversión

El código es básicamente una máquina de 3 estados:

- sWait: si go=0 mantiene el estado / si go=1 pasa a sWait
- sSample: inicializa y resetea el resultado y la máscara
- sConv: realiza la conversión durante 10 ciclos y luego vuelve a lanzar sSample para comenzar otra

```
//Verilog HDL for "Prueba", "SAR" "functional"


module SAR (clk, go, valid, result, sample, data, cmp, stateOUT);
input clk; // clock input
input go; // si go=0 -> empieza conversion
output valid; //valid=1 -> conversion ha finalizado
output [9:0] result; // resultado 10 bits
output sample; // senyal de control para el CDAC
output [9:0] data; // senyal de control para el CDAC
input cmp; // salida del comparador
reg [1:0] state; // estados
reg [9:0] mask; // mascara para ir testeando bit a bit
reg [9:0] result;
output [1:0] stateOUT; // para ver en que estado estoy

// Asignacion de estados
parameter sWait=2'b00, sSample=2'b01, sConv=2'b10;

// Busqueda binaria del SAR
always @(posedge clk) begin
    if (!go) state <= sWait; // espera si go != 0
    else case (state) // si go=0 empieza conversion
        sWait : state <= sSample; //estado de espera -> en el que sample=1
        sSample : //inicializacion y reseteo para hacer la conversion
            begin
            state <= sConv;
            mask <= 10'b1000000000; // reset mask a MSB
            result <= 10'b0; // clear result
            end

        sConv : // conversion SAR
            begin
            // Si Vdac < Vref -> Vcomp = 0 -> set bit
            if (!cmp) result <= result | mask;
                // desplazamos bit al siguiente que vamos a analizar
                mask <= mask>>1;
            if (mask[0]) state <= sSample; // cuando llegue al LSB se acaba
            end
    endcase
end

assign sample = (state==sSample); // sample = 1 cuando este en el estado sSample o acabe una conversion
assign data = result | mask;  // data control
assign valid = (state==sSample);  // ha finalizado cuando esta en sSample otra vez
assign stateOUT = state;
endmodule
```

Figura 4.10: Código del switch ideal

Como el SAR se ha diseñado con Verilog he creado una vista config que abra el esquemático para poder realizar una simulación con 'ams' en vez de 'spectre' (el predeterminado de Cadence). A continuación se muestra la vista config y la simulación que verifica su funcionamiento.

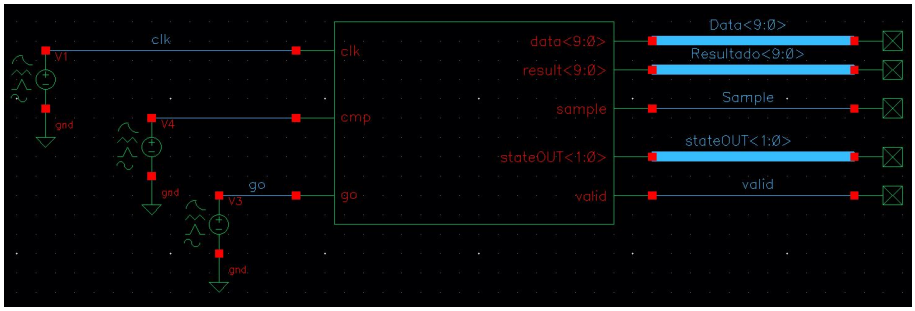


Figura 4.11: Testebench del SAR

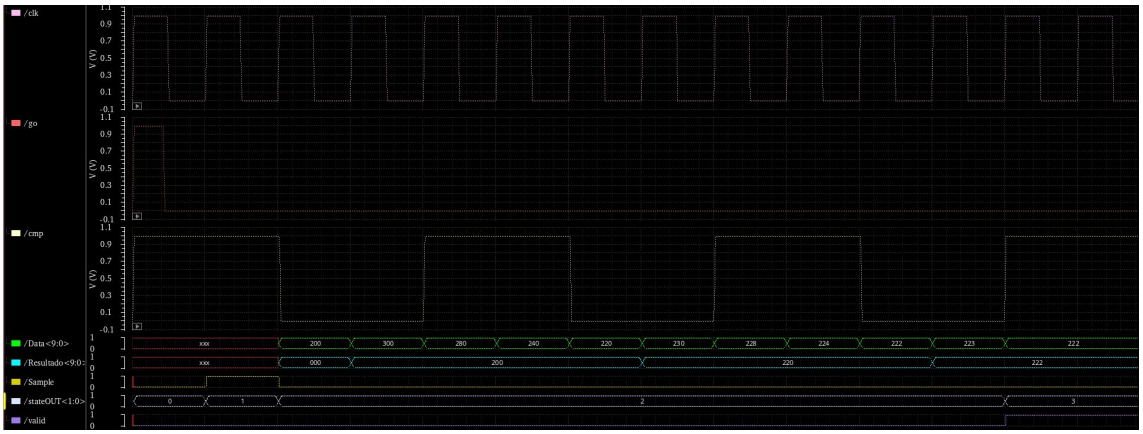


Figura 4.12: Simulación del SAR

El SAR pasa por todos los estados y se puede apreciar cómo varía su valor dependiendo de si cmp es 1 (disminuye) o 0 (aumenta) en data.

El resultado final es 0x222 que equivale a 533,203125mV.

4.6. CDAC

El primer bloque a diseñar es el CDAC. Para mantener la metodología top-down approach se diseña el circuito con condensadores reales y una vez validado su funcionamiento se diseñan, testean e integran los interruptores reales.

4.6.1. CDAC con condensadores reales

El tipo de condensador elegido para diseñar el circuito ha sido el componente cmm6t de la librería de XFAB. Este condensador de tipo MIM (se implementa entre Metal-Aislamiento-Metal) se encuentra en metal 5 y metaltop según la descripción de XFAB. Se ha elegido el que esté en los metales más superiores para que la distancia al sustrato sea lo más grande posible y por lo tanto haya menos parásitos.

Una vez elegido el tipo de condensador, se elige un valor para el condensador unitario (C0), es decir, el del LSB. Cuanto menor sea su valor pues más rápido se cargará/descargará y la potencia consumida será menor. Sin embargo, este valor se debe limitar ya que las capacidades parásitas y el ruido térmico empiezan a cobrar más importancia cuanto más pequeño sea.

Para una tecnología de 180nm y 10 bits he elegido un valor de 20f Faradios. El resto de valores se obtiene en base a este substituyendo la C de la figura 4.2. Los valores se recogen en la tabla 4.2:

Condensador	Valor
Dummy	20fF
C0	40fF
C1	80fF
C2	160fF
C3	320fF
C4	640fF
C5	1.280pF
C6	2.560pF
C7	5.120pF
C8	10.240pF
C9	20.480pF

Tabla 4.2: Valor de los condensadores del CDAC

Para reducir la discrepancia entre sus valores, en las instancias del componente en Cadnece se le añaden a todos el valor unitario. Luego, individualmente se le pone el factor de multiplicación para obtener el resultado final. Es como si estuviéramos poniendo tantos condensadores unitarios en paralelo hasta conseguir el valor deseado.

Una vez elegidos y calculados sus valores, se diseña un primer test con el comparador y los interruptores ideales. Mediante switches de Cadence simulo el funcionamiento del bloque SAR:

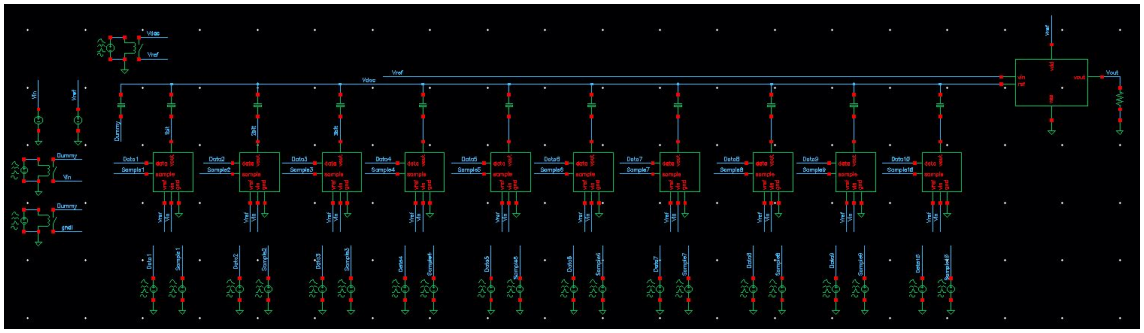


Figura 4.13: Testbench del array de condensadores

El sistema simula que hay una entrada de 750mV (1100000000 en binario). Como se observa en la respuesta de la figura 4.14 se obtiene el resultado deseado:

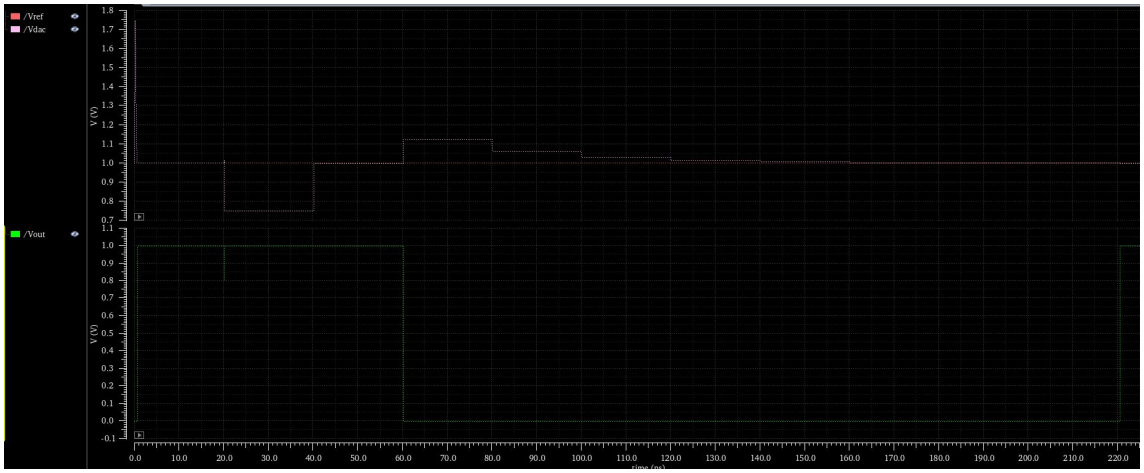


Figura 4.14: Respuesta para una entrada de 750mV

Posteriormente, automatizo el sistema agregando el bloque del SAR:

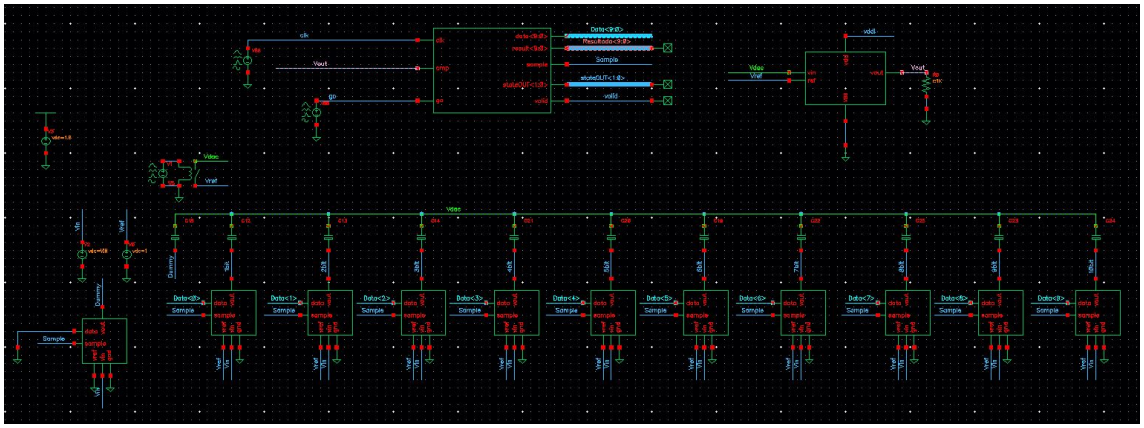
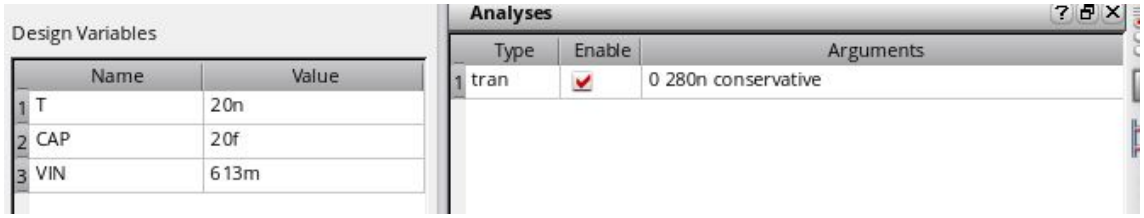


Figura 4.15: Testbench del sistema completo

Para mostrar cómo funciona el sistema completo he creado una simulación con valores parametrizables (condensador unitario, periodo de reloj y voltaje de entrada):



Design Variables

	Name	Value
1	T	20n
2	CAP	20f
3	VIN	613m

Analyses

	Type	Enable	Arguments
1	tran	☑	0 280n conservative

Figura 4.16: Configuración de la simulación

Para esta configuración y valores se obtiene esta respuesta:

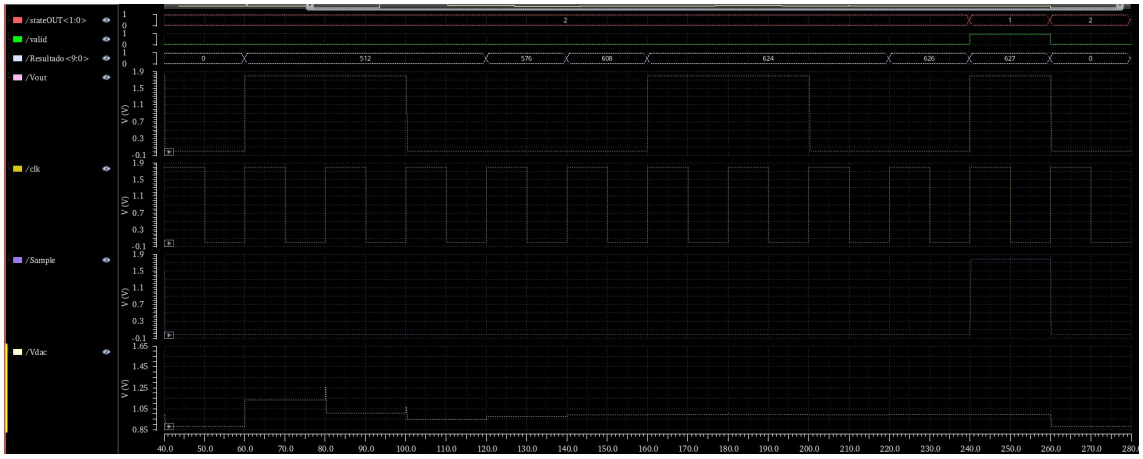


Figura 4.17: Configuración de la simulación

El resultado para una entrada de 613mV es de 627 en decimal. Este valor se corresponde a 612.793mV, que tienen un error de 0.212·LSB y por lo tanto está dentro del error de cuantificación deseado [-0.5·LSB, 0.5·LSB].

Para observar si el diseño funciona en todo el rango de entrada (de 0 a 1V) he creado un "Parametric SWEEP" de 101 puntos equispaciados 10mV para observar su respuesta y comprobar si está en el rango de error diseñado:

Variable	Value	Sweep?	Range Type	From	To	Step Mode	Step Size	Inclusion List	Exclusion List
VIN	101m	☑	From/To	0m	1000m	Linear Steps	10m		

Figura 4.18: Configuración del SWEEP

Una vez obtenidos los resultados, mediante una hoja excel analizo el error de cuantificación del sistema completo con condensadores reales. En la gráfica de la figura 4.19 se recogen los resultados satisfactorios:

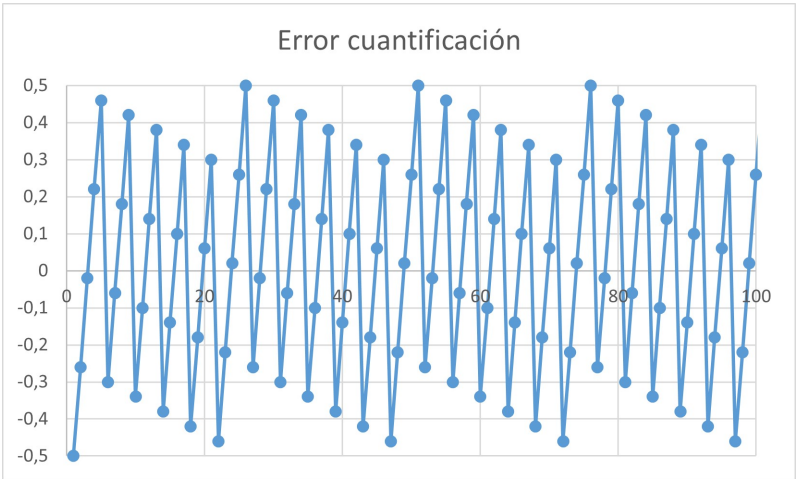


Figura 4.19: Error cuantificación del sistema completo

4.6.2. CDAC con interruptores reales

Los interruptores a diseñar son los conectados a los condensadores del CDAC, el del condensador Dummy y el de la línea del comparador:

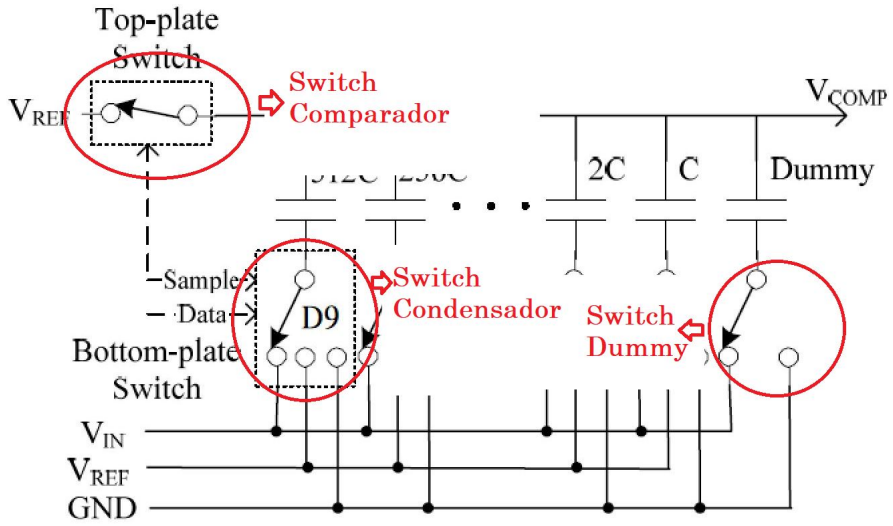


Figura 4.20: Interruptores a diseñar

La integración de interruptores reales supone un cambio importante por su inyección de carga y el compromiso entre su tamaño, consumo, parásitos y tiempo que necesitan para cargar un condensador al voltaje que queremos. Es decir, cuanto más grande será más rápido pero inyectará más parásitos a la línea y consumirá más.

El fenómeno de inyección de carga ocurre porque cuando los interruptores estan activos, almacenan cierta carga que posteriormente al estar en OFF pasan a la línea por el surtidor y el drenador. Introduciendo así, parásitos a la línea como se ve en la figura 4.21.

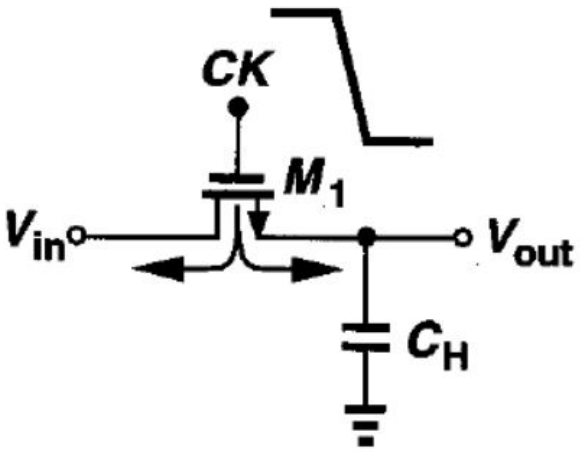


Figura 4.21: Inyección de carga [15]

Para reducir este efecto he usado puertas de transmisión para implementar los interruptores que pasan V_{IN} y V_{REF}. Una puerta de transmisión es la combinación de dos transistores en paralelo, uno PMOS y otro NMOS.

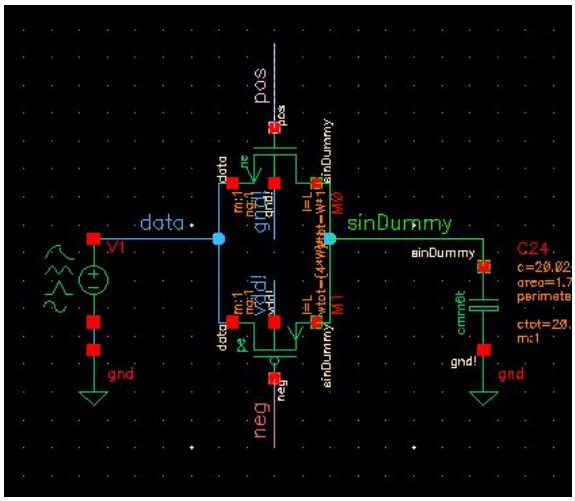


Figura 4.22: Puerta de transmisión

Para conseguir un tiempo de carga y descarga equivalente con estos interruptores se requiere un dimensionamiento adecuado. Esto es mantener un ratio adecuado entre la anchura (que llamaremos W) y longitud (L) del PMOS y NMOS. La longitud de todos los interruptores las mantengo al mínimo que permite la tecnología (180nm) y la relación entre las anchuras a WP = 6*WN, es decir, el PMOS requiere ser 6 veces más grande (a misma L). Esto se obtiene observando la relación que hay entre los factores de ganancia tecnológicos (KpN y KpP) y el voltaje umbral (Vtp y Vtn) entre otros de los NMOS y PMOS de la tecnología usada.

Sin embargo al realizar pruebas con el sistema completo, la reducción de inyección de carga requería ser mayor. Para combatir este problema se han añadido interruptores “Dummy”. Esto es añadir a continuación del interruptor otro con la mitad de tamaño, que funcione a la inversa y que tenga el surtidor y drenador unidos. En nuestro caso sería lo siguiente:

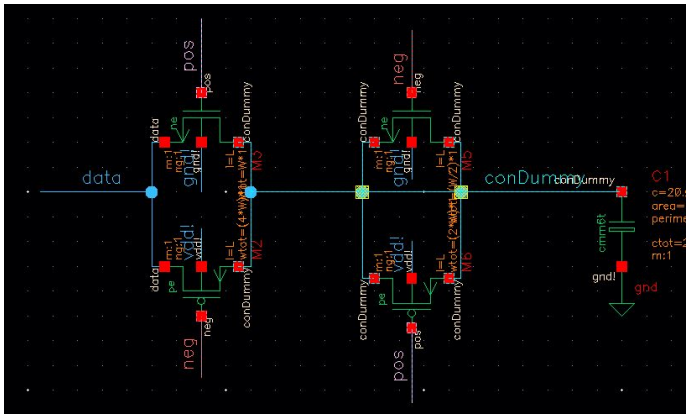


Figura 4.23: Puerta de transmisión con Dummy

El interruptor dummy lo que hace es almacenar la carga que inyecta el interruptor al apagarse. Al tener la mitad de tamaño, sus dimensiones serán de WP*3 y WN/2.

Una vez explicado las consideraciones que se han tomado con los interruptores, paso a mostrar los tres tipos que he diseñado. Todos ellos se han diseñado parametrizando su W (respetando el ratio entre los NMOS y PMOS) con una variable para que luego al simular se pueda ir variando y ajustando su valor para obtener el mejor resultado posible.

4.6.2.1. Switch Condensadores

Su diagrama es el de la figura 4.6. Implementado en Cadence y agregando los dummys queda el interruptor como el de la figura 4.24. Las puertas lógicas son de la libreria de XFAB.

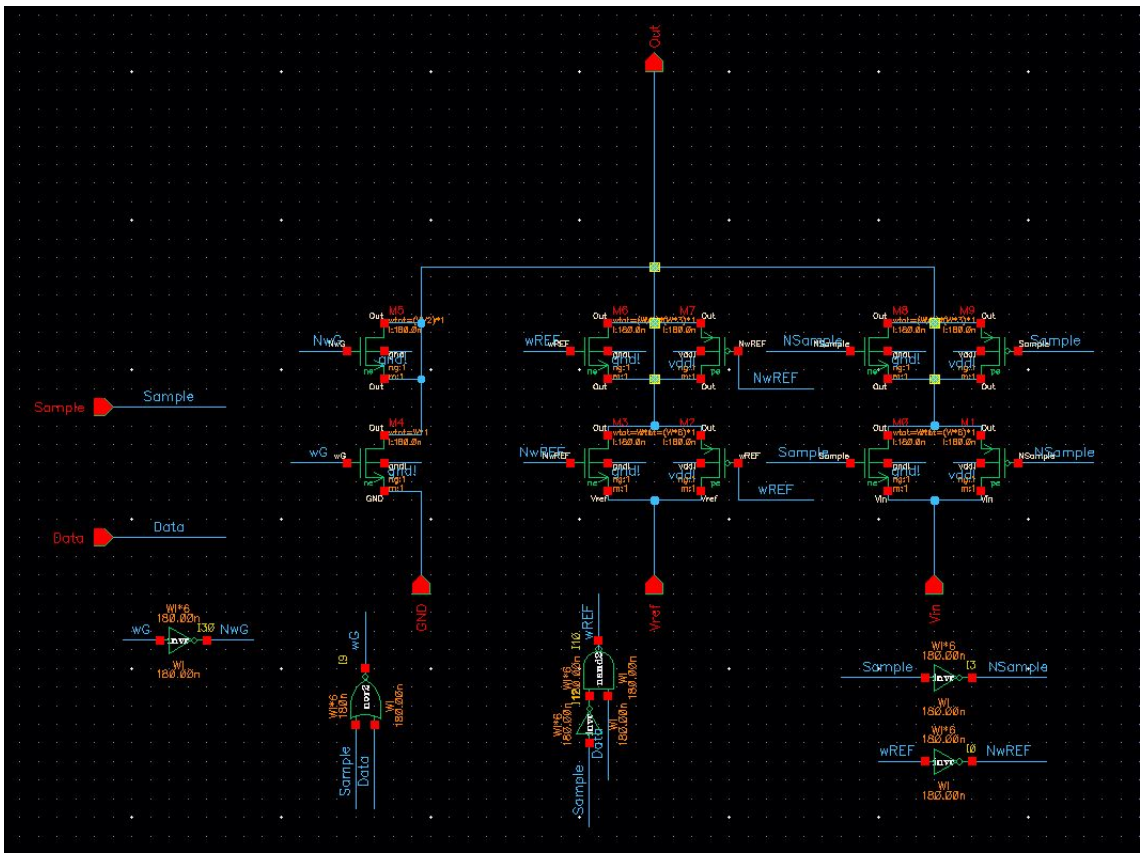


Figura 4.24: Switch Condensadores real

Al comprobar su funcionamiento (figura 4.25) crear los otros interruptores es más sencillo ya que este switch contiene todas las posibles entradas que puede tener un interruptor en el sistema. Además, estas primeras simulaciones sirven para establecer los primeros valores de W que se van a necesitar para poder cargar los condensadores más grandes del CDAC.

Figura 4.25: Funcionamiento Switch Condensadores real

4.6.2.2. Switch LSB

Como posibles entradas tiene Vin (se activa Sample=1) y GND (se activa cuando Sample=0).

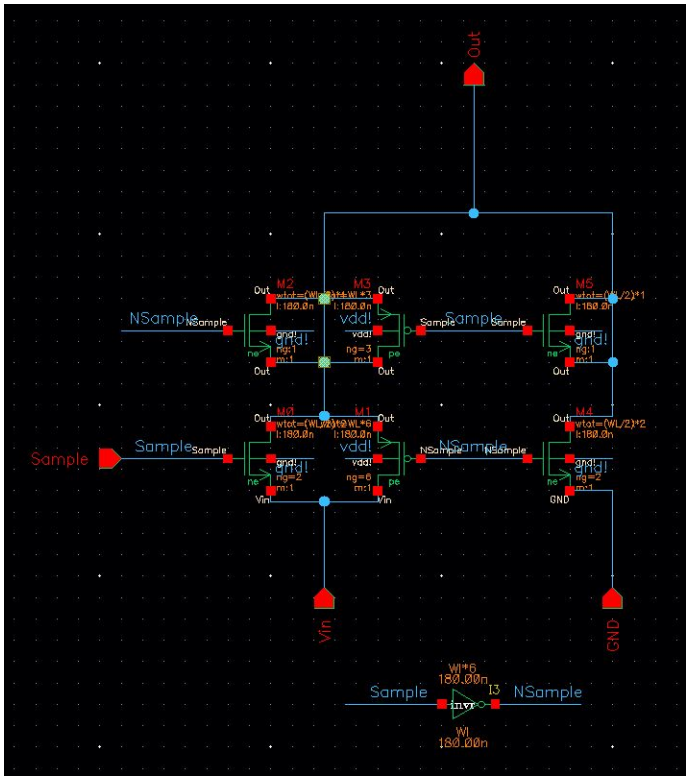


Figura 4.26: Switch LSB real

4.6.2.3. Switch Comparador

Este switch conecta la línea del comparador a Vref cuando se realiza el muestreo (Sample=1), sino deja la línea sin conectar.

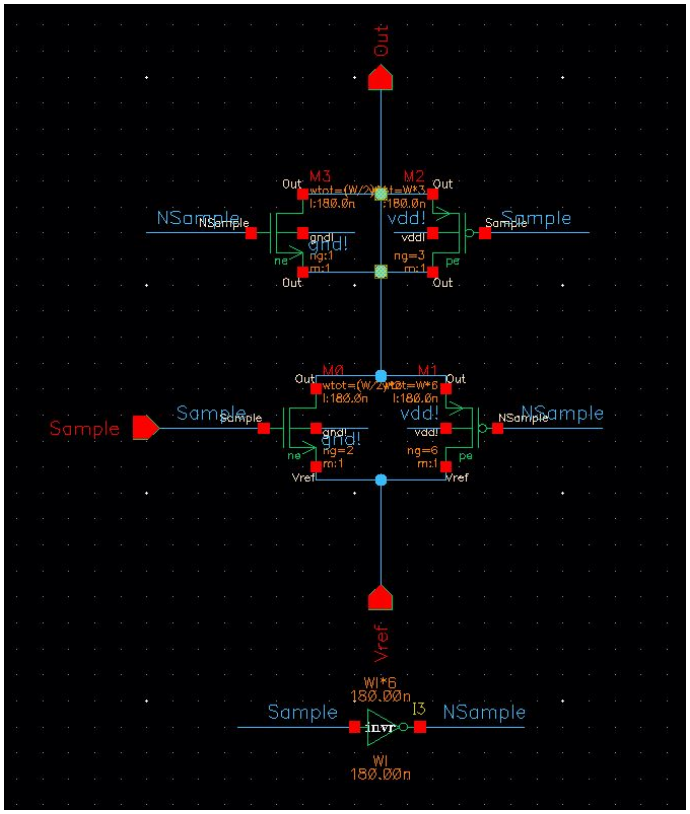


Figura 4.27: Switch Comparador real

4.6.3. Integración en el sistema completo

Una vez diseñados y testeados los interruptores se integran en el sistema completo. Ahora habrán condensadores e interruptores reales en el circuito. La figura 4.28 muestra el nuevo esquemático.

En esta etapa del diseño se debe conseguir un resultado parecido al error de cuantificación del sistema con condensadores reales ya que a medida que se avanza con la integración de más componentes reales, el error aumenta. Principalmente aumenta debido a los tamaños de los transistores, que como había comentado introducen a la línea parásitos debido a sus dimensiones.

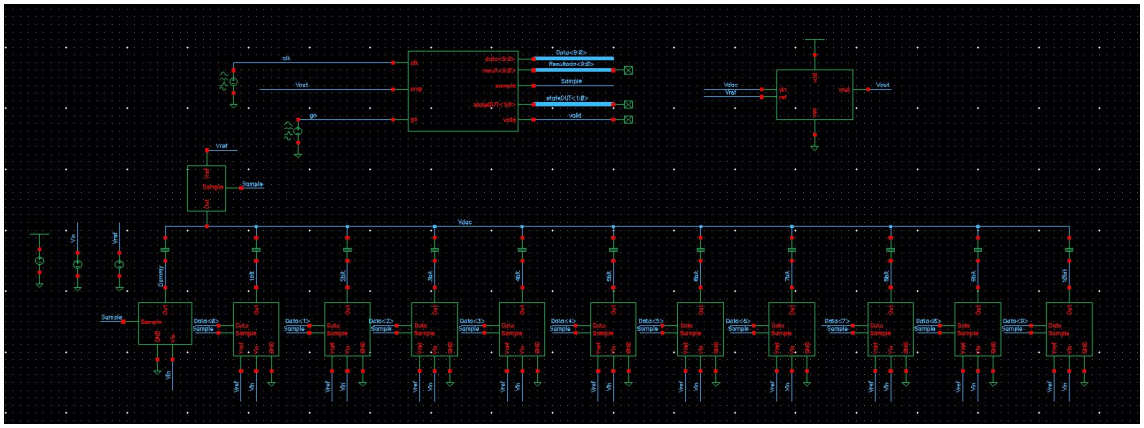


Figura 4.28: Testbench del sistema completo

Es por ello que antes de diseñar para un reloj interno rápido, primero lo diseño para uno de 2MHz para detectar los componentes críticos del sistema y empezar a dimensionarlos correctamente.

Los parámetros que más influyen en la precisión de la medida son:

- W del “switch comparador”: ya que está conectado directamente a la línea de entrada al comparador.
- Tamaño del condensador unitario: porque de él dependen los valores de todos los condensadores del sistema.
- W del “switch condensador”: tiene que ser lo suficientemente grande como para cargar su respectivo condensador al voltaje deseado a cierta velocidad.

Para determinar el valor de cada parámetro crítico se han realizado barridos con el “parametric sweep” de cada uno de ellos y luego de forma más precisa se han ajustado teniendo en cuenta los siguientes aspectos:

1. Cuando se realiza el muestreo, el “switch comparador” conecta la línea a Vref. Hay que asegurar que para cuando el ciclo ha acabado el valor que hay en la línea del comparador es de Vref. Si no se carga a Vref automáticamente el valor que se carga en todos los condensadores será erroneo durante toda la conversión. El tamaño del “switch comparador” y los de los condensadores influyen directamente en el tiempo necesario para estabilizar este valor. La figura 4.29 muestra el instante comentado.

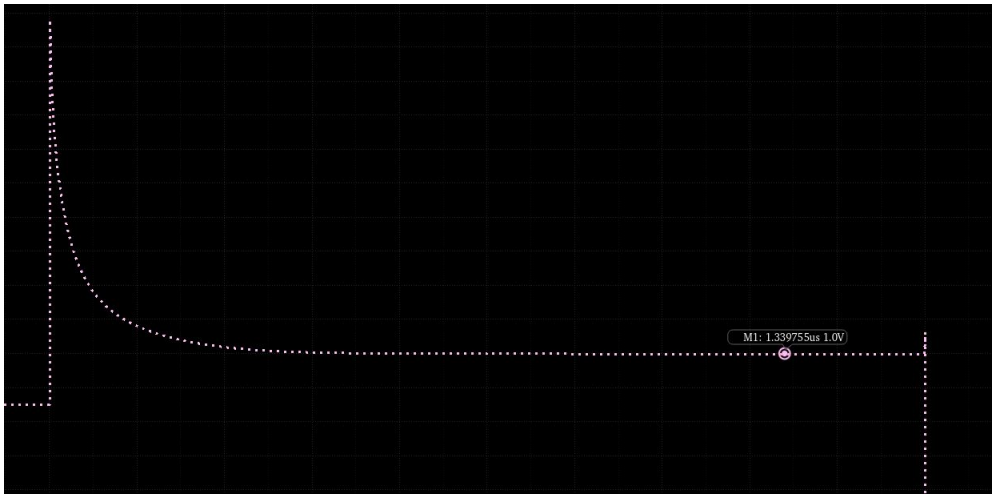


Figura 4.29: Conexión de la línea del comparador a Vref

2. El otro aspecto a tener en cuenta es el del valor de la línea del comparador durante todo el estado de conversión. Al igual que antes, hay que asegurar que se estabilice el valor para cada intervalo de tiempo. Los valores ya estables que se obtengan en la línea en cada comparación de bit se comparan con los que deberían ser idealmente, y de esa forma se puede detectar los parásitos añadidos en cada bit y ajustar los valores del "switch comparador", "switch condensador" y condensador unitario para que el error sea lo menor posible. La figura 4.30 muestra el análisis comentado.

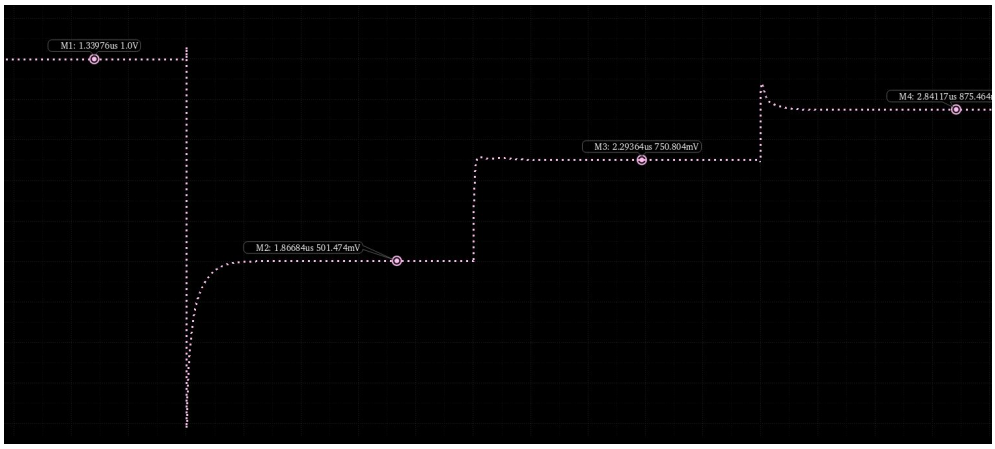


Figura 4.30: Valor de Vcomp en cada comparación

Una vez ajustados los valores, genero un barrido de señal de entrada para obtener el error de cuantificación. Para la siguiente configuración:

- f = 2MHz
- C = 10f Faradios
- WN "switch comparador" = 2μm

- WN “switch condensador” = 1μm
- WN “puertas lógicas” = 1μm

Se obtiene la siguiente respuesta:

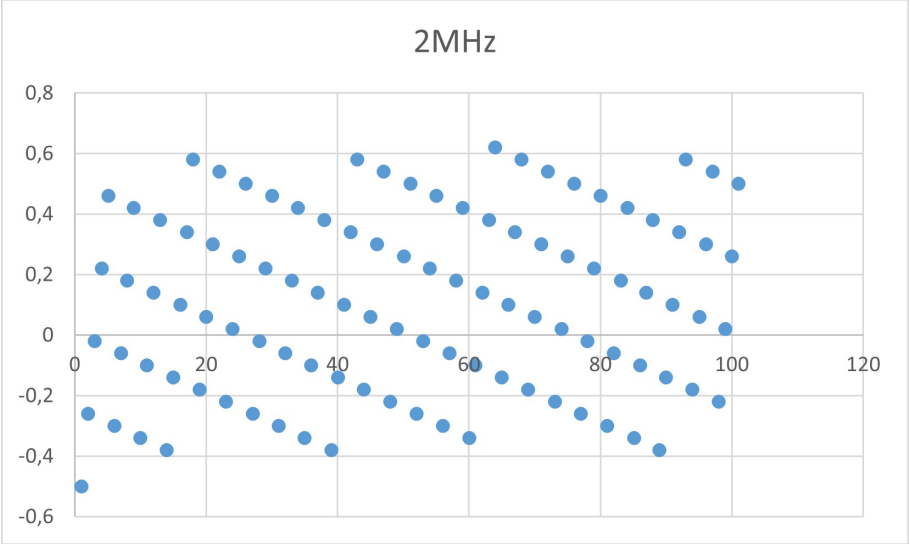


Figura 4.31: Error de cuantificación 2MHz

A la vista de los resultados ya se empieza a salir del intervalo [-0.5LSB 0.5LSB], pero todavía hay margen suficiente entre el de [-1LSB 1LSB] con el que queremos trabajar.

A continuación, aplicando la misma forma de trabajar, se intenta bajar la frecuencia lo máximo posible para conseguir un tiempo de conversión de unos pocos microsegundos. La frecuencia interna a la que apunto es de 10MHz.

Para conseguir esta frecuencia el condensador unitario requiere ser bajado todo lo posible. Para conseguirlo, los condensadores de valor unitario los he implementado como dos en serie del doble de valor. De esta forma se obtiene el mismo valor y el condensador más pequeño del sistema será dos veces el del LSB.

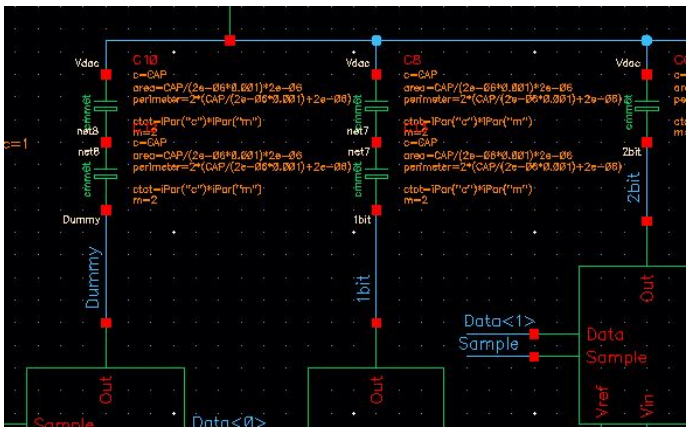


Figura 4.32: Condensador LSB con condensadores en serie

Otro aspecto que he tenido en cuenta es que posiblemente el condensador que uso podría estar mal modelizado y aparecieran parásitos porque no se aplica bien el factor de multiplicación al valor unitario en cada condensador del CDAC. Para comprobarlo se introducen manualmente los condensadores unitarios en paralelo. La conclusión es de que están bien modelizados.

Sin embargo, al realizar las simulaciones con condensadores más pequeños, la conexión de la línea del comparador a Vref durante el muestreo no se realizaba a tiempo. Si se incrementa el tamaño del "switch comparador" para solucionarlo, aparecen errores fuera del rango de trabajo. La solución a este problema ha sido incrementar el muestreo un ciclo más, esto se consigue modificando el código del SAR añadiendo un estado más después del sSample y que sample esté activo durante este ciclo también:

```
// Asignacion de estados
parameter sWait=2'b00, sSample=2'b01, sSample2=2'b10, sConv=2'b11;

// Busqueda binaria del SAR
always @(posedge clk) begin
    if (!go) state <= sWait; // espera si go != 0
    else case (state) // si go=0 empieza conversion
        sWait : state <= sSample; //estado de espera -> en el que sample=1
        sSample : //inicializacion y reseteo para hacer la conversion
            begin
            state <= sSample2;
            mask <= 10'b1000000000; // reset mask a MSB
            result <= 10'b0; // clear result
            end
        sSample2 : //inicializacion y reseteo para hacer la conversion
            begin
            state <= sConv;
            end
        sConv : // conversion SAR
            begin
            // Si Vdac < Vref -> Vcomp = 0 -> set bit
            if (!cmp) result <= result | mask;
                // desplazamos bit al siguiente que vamos a analizar
                mask <= mask>>1;
            if (mask[0]) state <= sSample; // cuando llegue al LSB se acaba
            end
    endcase
end

assign sample = (state==sSample) || (state==sSample2); // sample = 1 cuando este en el estado sSample o acabe una conversion
assign data = result | mask; // data control
assign valid = (state==sSample);  // ha finalizado cuando esta en sSample otra vez
assign stateOUT = state;
endmodule
```

Figura 4.33: Añadir un ciclo al muestreo

Además, como la frecuencia ha aumentado, el tiempo requerido para cargar los condensadores es menor. Esto afecta solo a unos pocos de los bits de más orden. Es por ello que para ahorrar en tamaño y consumo he hecho parametrizable la intancia del "switch condensador" con el parámetro pPar de Cadence. De esta forma a cada switch le puedo asignar un tamaño diferente.

Una vez ajustados los valores, vuelvo a generar un barrido de señal de entrada para obtener el error de cuantificación. Para la siguiente configuración:

- f = 10MHz
- C = 5.5f Faradios
- WN "switch comparador" = 2μm

- "switch condensador"
  - WN 10bit = 6μm
  - WN 9bit = 4μm
  - WN 8bit = 3μm
  - WN 7bit = 2μm
  - WN resto = 1μm
- WN "puertas lógicas" = 1μm

Se obtiene la siguiente respuesta:

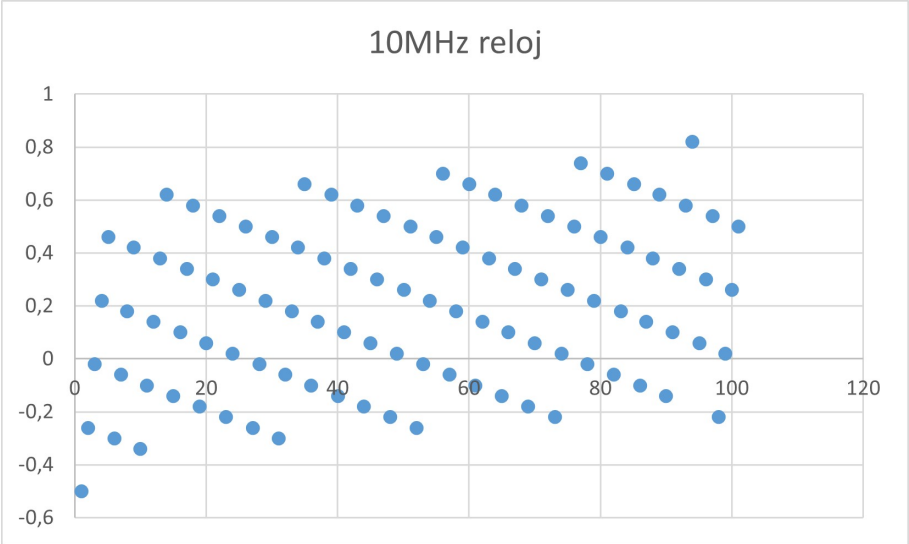


Figura 4.34: Error de cuantificación 10MHz

Una vez todos los interruptores han sido integrados, adaptados y testeados paso a diseñar e introducir el comparador real.

4.7. Comparador

El bloque del comparador está formado por un comparador latch dinámico y un SR latch de tipo NOR que se encarga de mantener el resultado de la comparación durante un ciclo de conversión.

4.7.1. Diseño del comparador y SR latch

El comparador dinámico a diseñar es el de la figura 3.10 debido a su bajo consumo de potencia. La figura 4.35 muestra su implementación en Cadence.

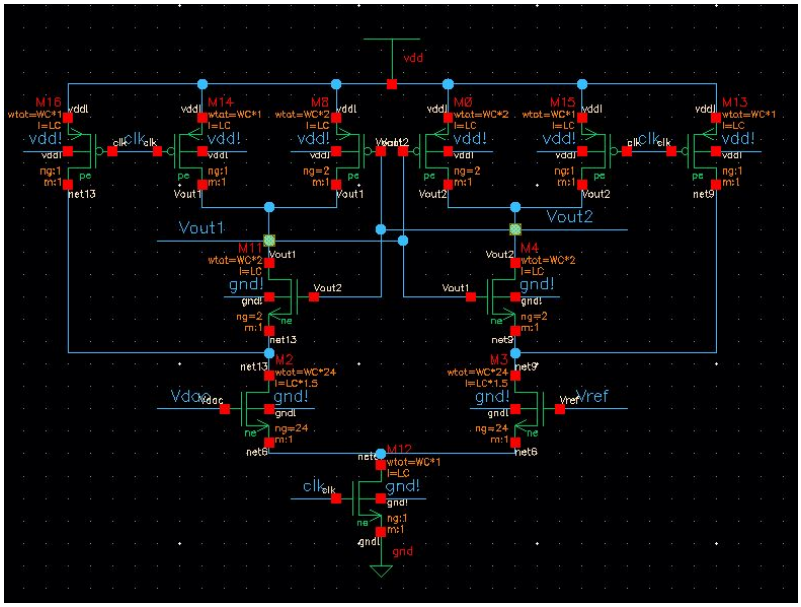


Figura 4.35: Comparador Latched Dinámico real

Los transistores más críticos a diseñar son los que hay situados a la entrada del comparador. A más tamaño del par diferencial a la entrada, más ganancia y por lo tanto más resolución en la comparación. Sin embargo, cuanto más grande sean, la capacidad parásita también lo será y por lo tanto el ruido de retroceso (en inglés el ruido ”kickback”) será mayor. Este ruido aparecerá directamente en las líneas del comparador.

El ruido kickback lo acopla el comparador a la línea durante las grande variaciones de voltaje durante la fase de regeneración. Para combatirlo se pueden añadir interruptores a las entradas que pasen la entrada durante su fase de reset y que corten la línea en el momento que empiece la comparación. Sin embargo, al probarlo en el diseño, los interruptores añadían más error del que quitaban debido a sus propios parásitos. Por lo tanto se ha dimensionado el comparador para tener una buena solución al compromiso con nuestro diseño.

Por otro lado, el SR latch de tipo NOR se muestra en la figura 4.36.

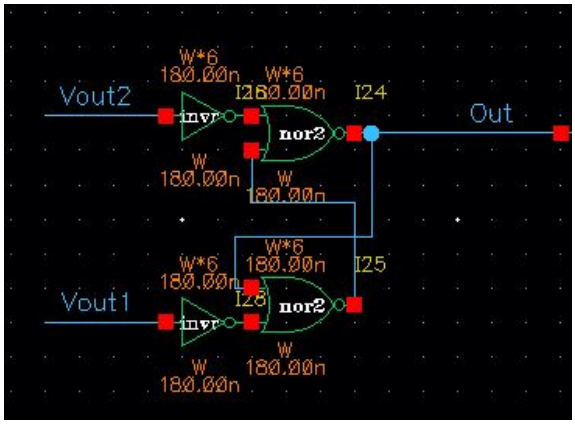


Figura 4.36: SR latch de tipo NOR

Su tabla de verdad es la siguiente:

$\overline{Vout2}$	$\overline{Vout1}$	Out
0	0	Latch / Mantiene datos
0	1	0
1	0	1
1	1	Estado inválido

Tabla 4.3: Lógica del SR latch NOR

Las salidas del comparador se invierten porque sino podríamos entrar en el estado inválido del SR latch NOR.

Una vez montados, se diseña un testbench para comprobar su funcionamiento:

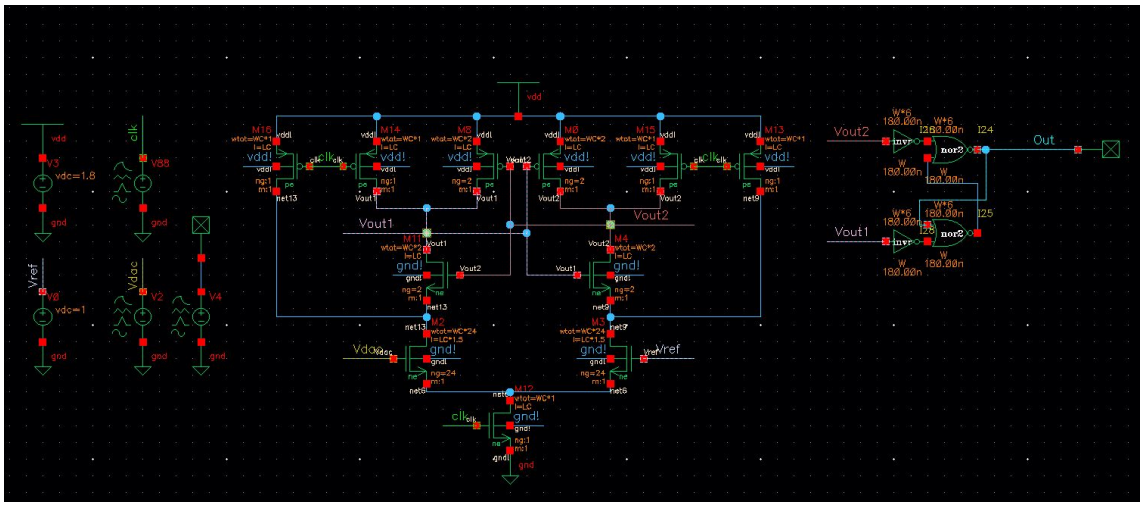


Figura 4.37: Testbench del bloque comparador

Y mediante un pulso que varíe 1LSB o menos la entrada Vdac observo si en un primera instancia debería funcionar en el sistema completo para una resolución de 10 bits.

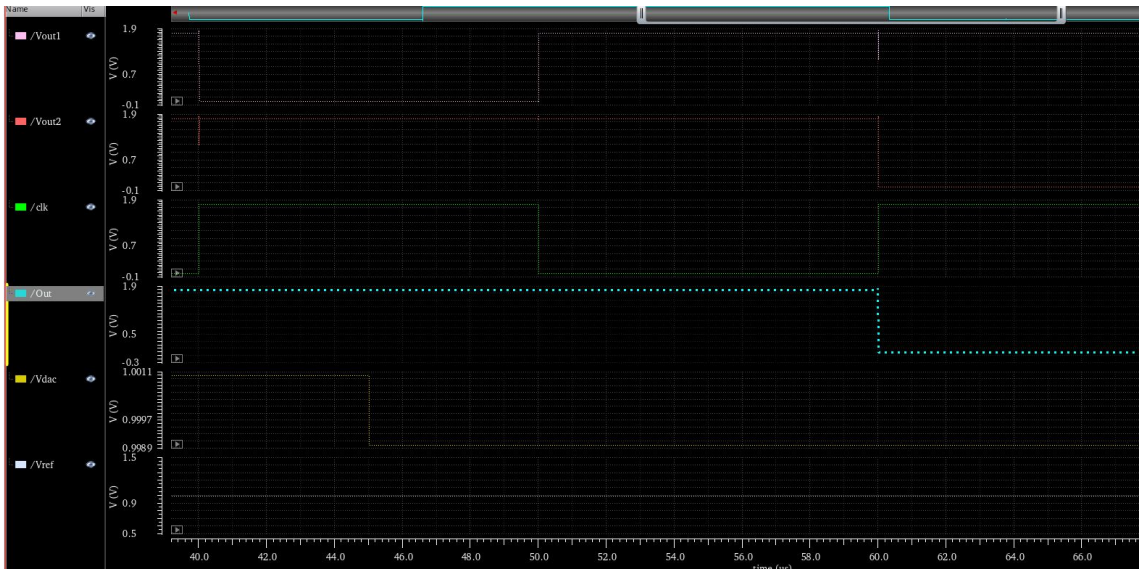


Figura 4.38: Funcionamiento del comparador

Una vez diseñado se crea un símbolo para este bloque y se integra en el sistema completo:

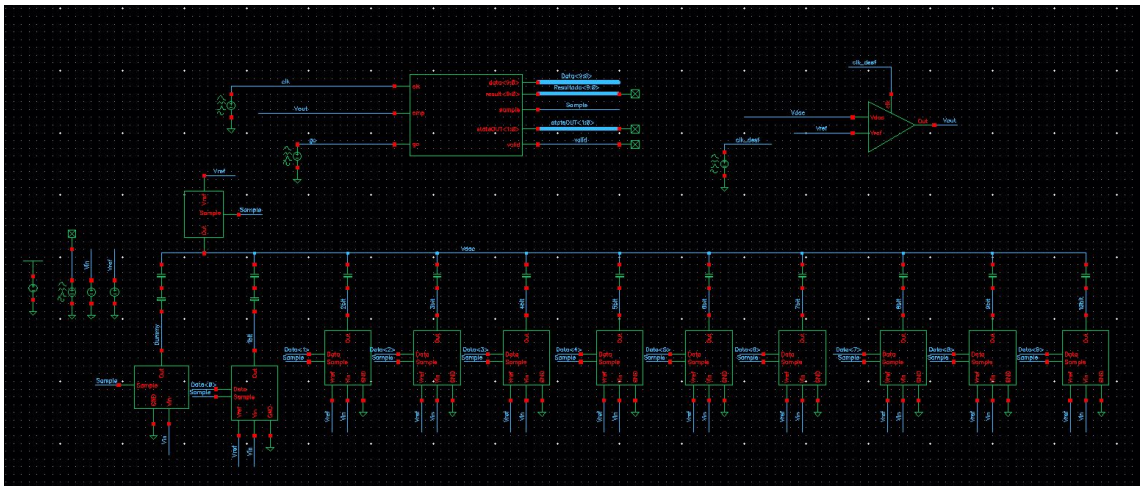


Figura 4.39: Sistema completo con comparador real

Como se puede observar en el esquemático, el comparador va conectado a un reloj diferente del clk. Este reloj es el mismo que clk pero desfasado cierto tiempo para asegurarnos que el comparador compare cuando la señal de la línea de Vdac está cargada y estable.

Para los mismos valores obtenidos en el diseño con interruptores reales se aplica el mismo procedimiento para obtener los del comparador. En este caso las variables a diseñar son el tamaño de los transistores del comparador y tiempo de desfase de su clk.

Las dimensiones finales de los transistores del comparador para un largo de 260nm y siguinedo el diagrama de la figura 3.10:

Transistor	**W** (μm)
M7/M9/M8/M10	0.45
M5/M6	0.9
M3/M4	1.8
M1/M2	2.25
Mb	0.9

Tabla 4.4: Tamaño transistores del comparador dinámico

Una vez ajustados todos los valores ya tenemos el sistema completo montado. El siguiente capítulo muestra un análisis completo mediante simulaciones del desempeño del diseño final propuesto del SAR ADC.

Capítulo 5

Evaluación del SAR ADC

Este capítulo está enfocado a mostrar mediante simulaciones el desempeño del SAR ADC diseñado en el capítulo 4. Los parámetros que analizo son su error de cuantificación, tiempo de conversión, su consumo, sus características estáticas y las dinámicas. Las simulaciones se realizan en Cadence y se exportan para procesarlas en Matlab.

5.1. Tiempo de conversión

La frecuencia de muestreo del sistema final será igual a la inversa tiempo de conversión. Como tiene un reloj interno de 10MHz y 12 ciclos para realizar una conversión, esto se traduce en un tiempo de 1.2μs lo cual está dentro de lo que queríamos conseguir. Por lo tanto la fs=834kHz aproximadamente.

5.2. Consumo

Para analizar el consumo, realizo una simulación con N conversiones y calculo la potencia de la fuente que alimenta al circuito durante ese intervalo de tiempo. La potencia tiene la siguiente expresión:

$$P(W) = Vfuente \cdot Ifuente$$

Donde Vfuente será el voltaje de alimentación del sistema que en esta tecnología es de 1.8V. Por otro lado la Ifuente la obtengo con Cadence integrando su intensidad en el tiempo de simulación y dividiendo ese valor por tantos ciclos de conversión que se hayan hecho en ese intervalo:

$$Ifuente = \frac{1}{N_{conversiones}} \cdot \int_{t_1}^{t_2} I\,dx$$

Para una simulación con 10 conversiones he obtenido un valor de Ifuente de 700nA, obteniendo así una **P = 1.26μW**.

5.3. Características estáticas

Para poder obtener una buena aproximación de las características estáticas se ha de realizar una simulación de tipo parametric de 0V al fondo de escala del conversor (1V). El paso tiene que ser menor de 1LSB entre muestras y múltiplo del LSB. Esto es debido a que el DNL analiza el ancho de los valores de código del ADC con el ideal (1 LSB) por lo que cuantas más muestras haya entre 1 LSB pues mayor precisión habrá en la medida. Yo elijo un paso de LSB/4, lo que implica un total de 4096 ($2^{10} \cdot 4$) muestras que se van a obtener con la simulación.

Una vez acabada la simulación exporto sus resultados de Cadence a Matlab y desarrollo un algoritmo para obtener el resultado obtenido cuando ha acabado cada conversión. De esta forma se obtienen los datos mostrados en la figura 5.1.

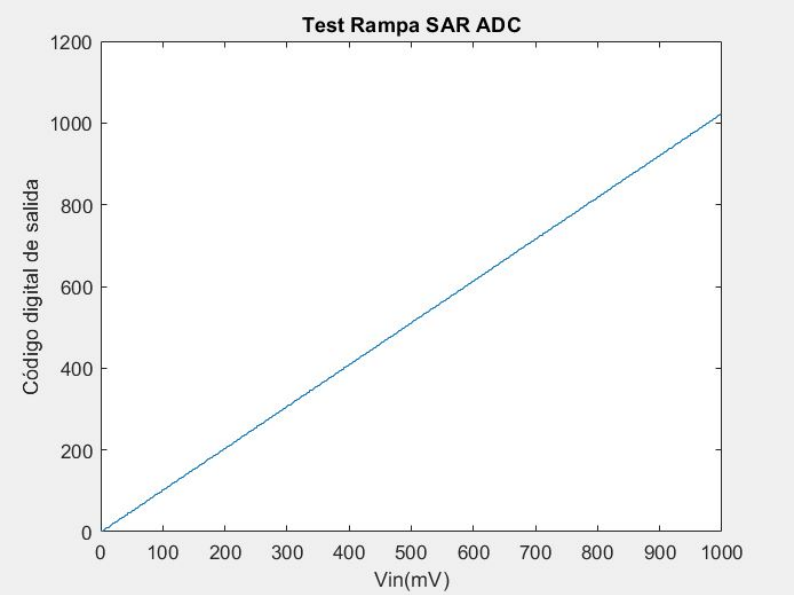


Figura 5.1: Resultado del parametric con paso LSB/4

De la figura 5.1 ya se obtiene la conclusión de que no hay error de offset ni de full scale ya que la señal no está desplazada ni tiene una pendiente diferente al que debería tener un ADC ideal.

5.3.1. DNL

La figura 5.2 muestra el DNL del sistema. Como se puede observar, se encuentra entre [-0.5LSB 0.5LBS]. Como el DNL no supera en ningún momento -1LSB podemos llegar a la conclusión de que no hay ningún código perdido.

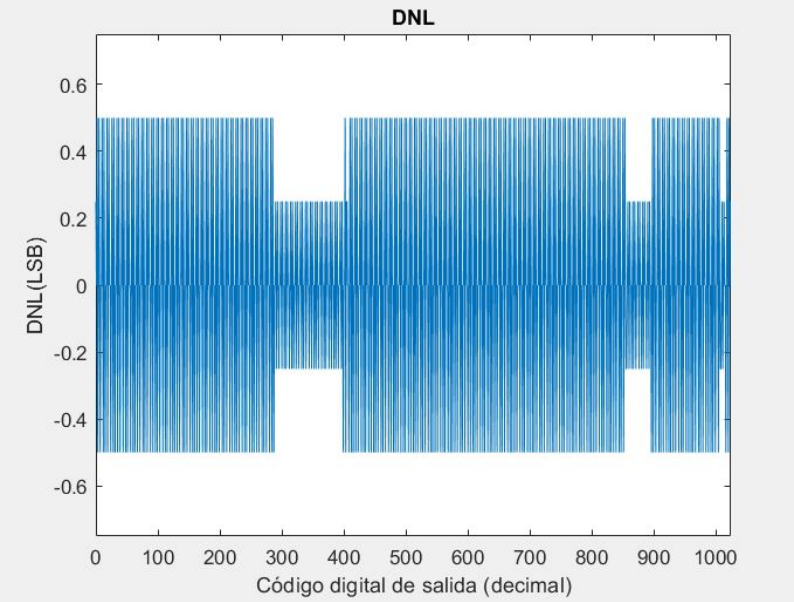


Figura 5.2: DNL del ADC

5.3.2. INL

A través del DNL obtenemos el INL, ya que cada código del INL se obtiene realizando el sumatorio hasta ese código del DNL. La figura 5.3 muestra el INL obtenido en el sistema.

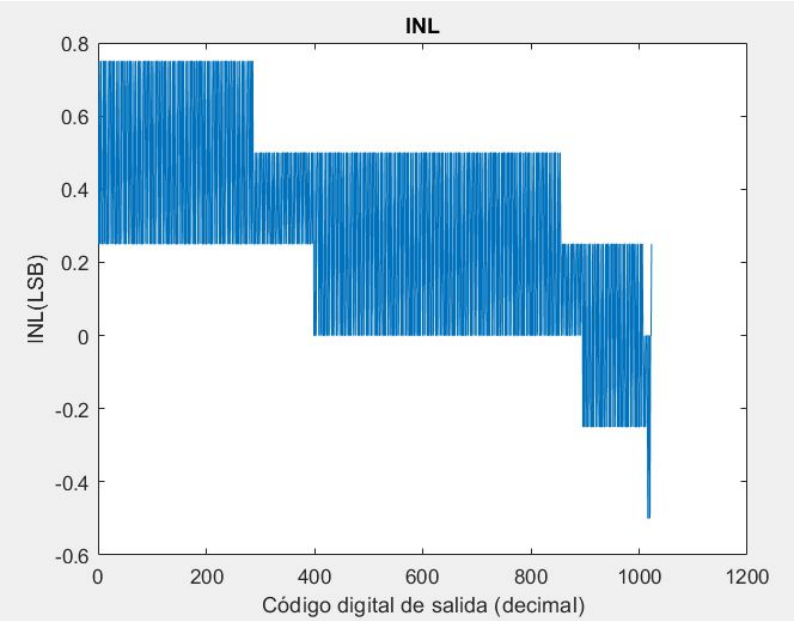


Figura 5.3: INL del ADC

5.4. Error de cuantificación

Comparando los datos obtenidos en la figura 5.1 con los que deberían ser en un conversor ideal, obtengo el error de cuantificación:

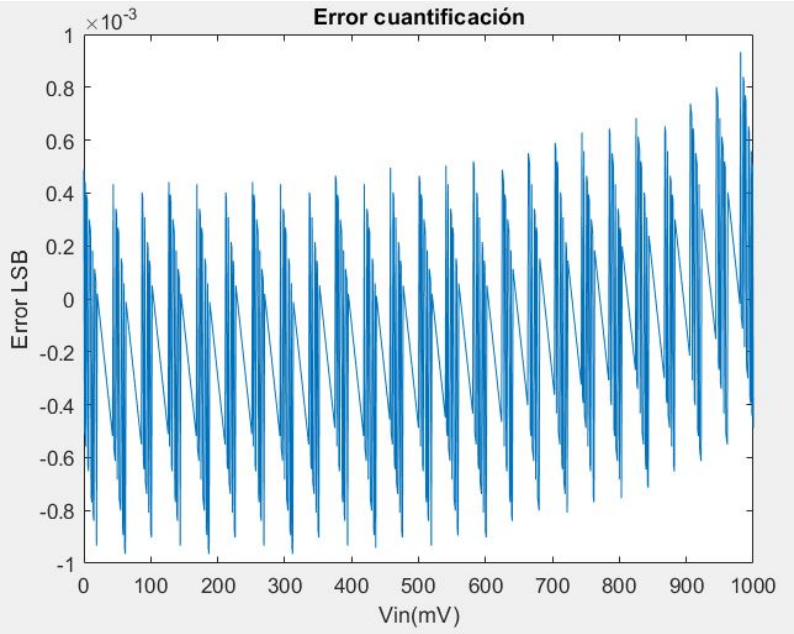


Figura 5.4: Error de cuantificación del sistema completo

A la vista de los resultados, el SAR ADC final se encuentra entre el rango [-1LSB 1LSB] por lo que el sistema deberída tener más de 9bits efectivos.

Además, se puede observar que el resultado obtenido se corresponde con los que hay en las características estáticas. Cuando el error de cuantificación es negativo, eso significa que una señal de entrada ha sido detectada como un valor que es más pequeño del que debería ser, por lo tanto el ancho del código se ensanchará. Al ser más ancho que un LSB, el DNL será por lo tanto positivo y como el INL se obtiene realizando sumatorios del DNL, pues aumentará su valor. El comportamiento es a la inversa para un error de cuantificación positivo.

5.5. Características dinámicas

Para obtener estas medidas introduzco al SAR una señal senoidal centrada en 500mV que vaya en todo el rango de media (de 0 a 1V). Después exporto estos datos a Matlab, extraigo los resultados de las conversiones y realizo la FFT (Transformada de Fourier). A través del espectro de la señal senoidal obtenida a la salida del conversor puedo fácilmente obtener las características dinámicas con Matlab.

La senoidal que se tiene que introducir al sistema tiene que tener una frecuencia determinada para que cuando obtengamos la FFT a la salida no veamos la fuga espectral. Este efecto aparece cuando realizamos la FFT de N puntos de una señal y produce que una componente frecuencial de la señal

pueda afectar a múltiples muestras de la FFT cercanas a su frecuencia. Si las muestras obtenidas las repetimos indefinidamente, las unimos y aparece discontinuidad entre ellas, habrá fuga espectral tal y como muestra la figura 5.5.[16]

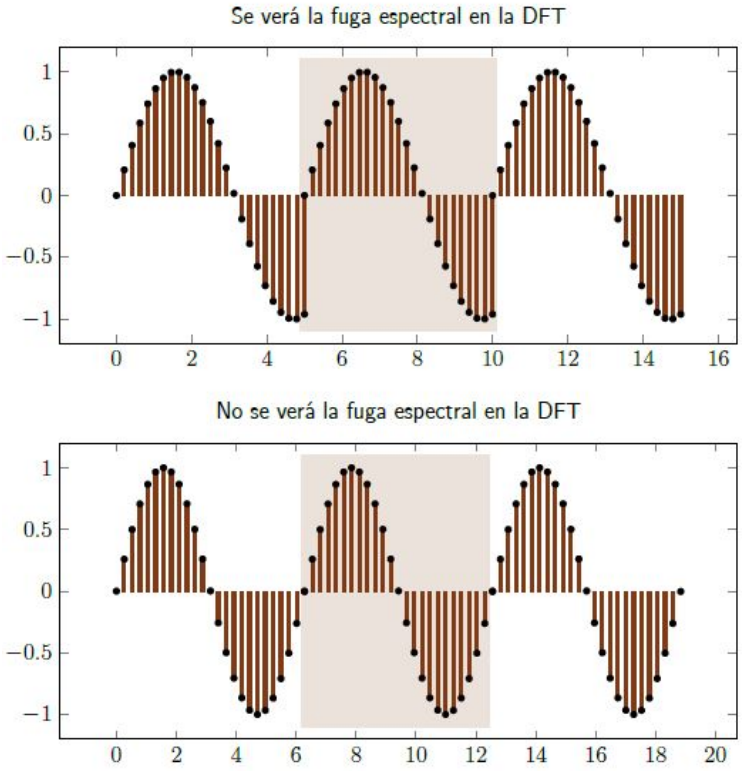


Figura 5.5: Fuga espectral [16]

Matemáticamente para elegir una frecuencia que no tenga fuga espectral debe seguir la siguiente expresión:

$$f = \frac{Np}{N} \cdot fs$$

Donde:

- **f:** frecuencia de la señal de entrada
- **Np:** número de periodos de la señal de entrada
- **N:** número de puntos de la FFT
- **fs:** frecuencia de muestreo del sistema

La N que he elegido es de 1024 y el número de periodos de la señal de entrada (Np) 24. Para estos valores y una fs de 834kHz se obtiene una frecuencia de entrada de 19.53125kHz.

Una vez exportados y obtenidos los resultados de las conversiones que se observan en la figura 5.6 se puede obtener su FFT (figura 5.7).

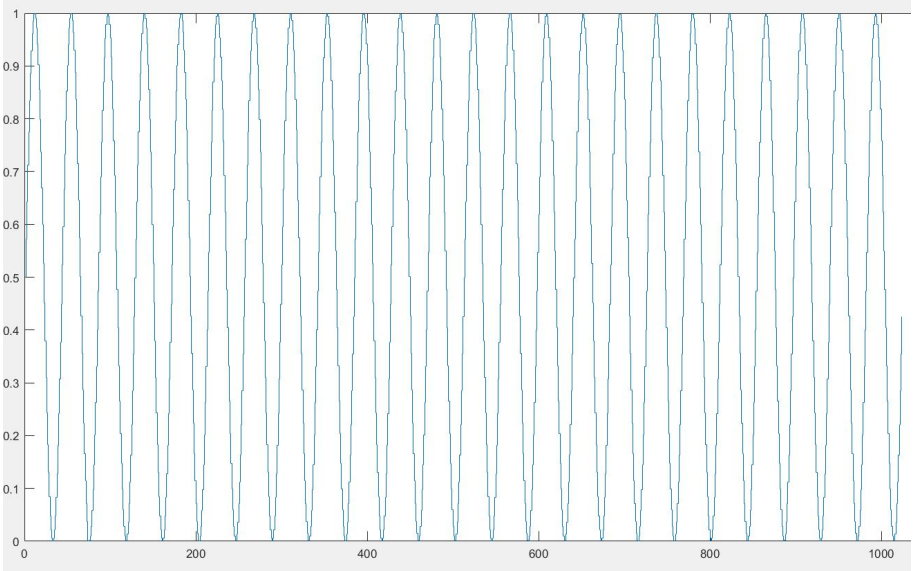


Figura 5.6: Señal de entrada muestreada

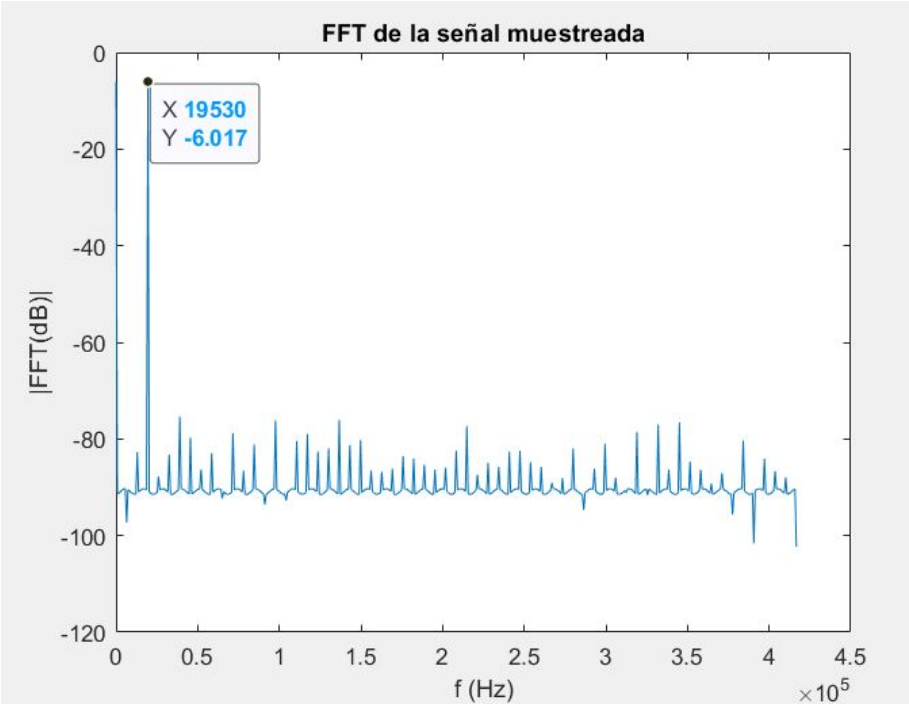


Figura 5.7: FFT de la señal muestreada

En la FFT se observa el contenido frecuencial de una senoidal ya que tiene una delta (el pico marcado en la gráfica) en su frecuencia.

A continuación obtengo con Matlab las características dinámicas.

5.5.1. SNR

SNR = 59.95dB

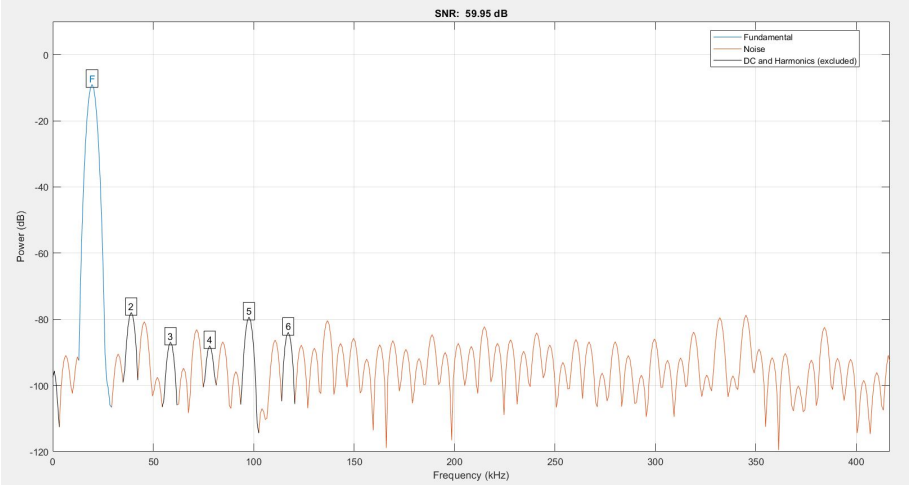


Figura 5.8: SNR de la FFT

5.5.2. THD

|THD| = 65.48dB

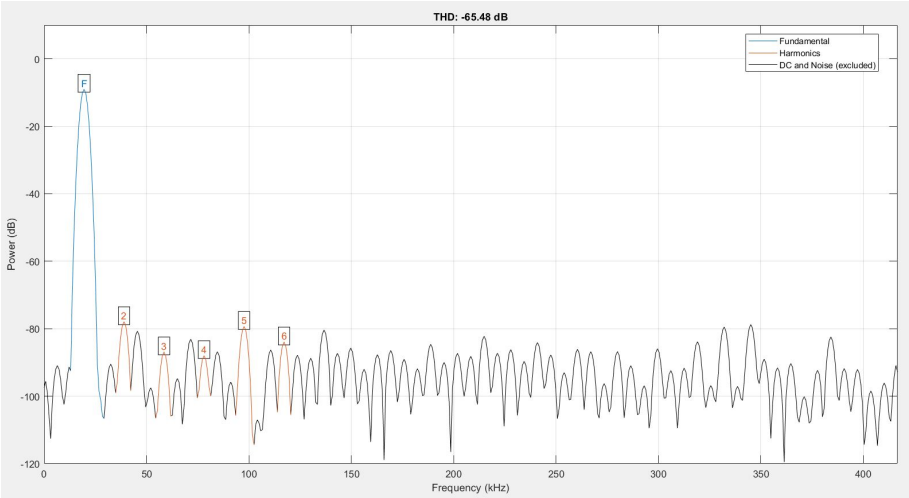


Figura 5.9: THD de la FFT

5.5.3. SINAD y ENOB

SINAD = 58.98dB

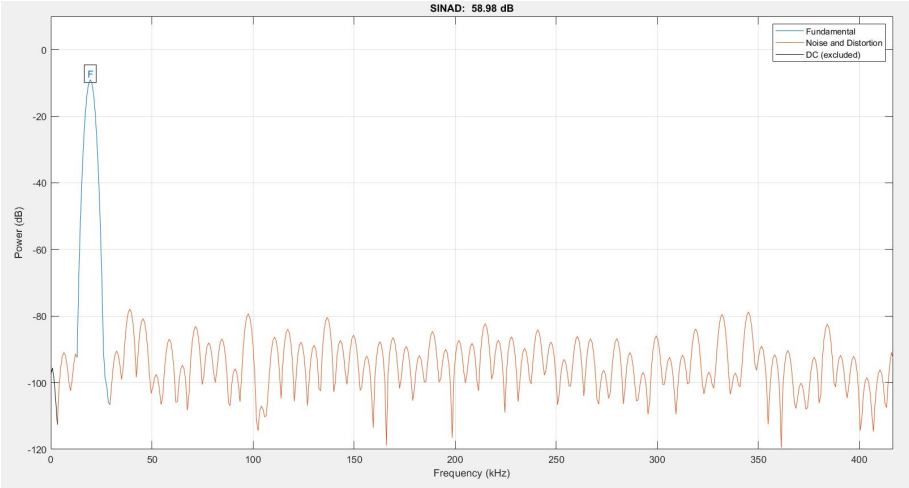


Figura 5.10: SINAD de la FFT

Para una SINAD de 58.98dB, aplicando la ecuación 2.11, se obtiene una **ENOB = 9.505 bits**. Obteniendo así, un sistema con más de 9 bits efectivos.

5.5.4. SFDR

SFDR = 68.91dB

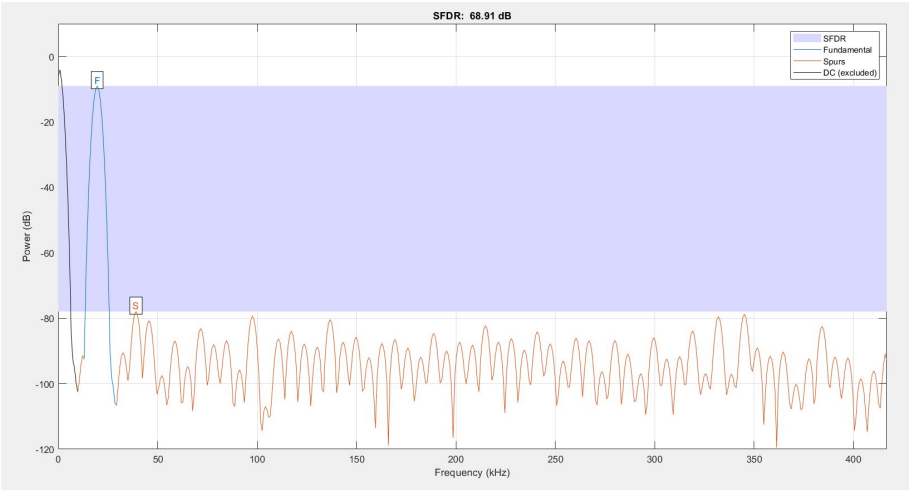


Figura 5.11: SFDR de la FFT

5.6. Resumen

La tabla 5.1 muestra a modo de resumen las características principales del SAR ADC diseñado:

Característica	**Valor**	**Unidad**
Tecnología	180	nm
Alimentación	1.8	V
Fondo de escala	1	V
Resolución	10	bits
Frecuencia de muestreo	834	kHz
Consumo	1.26	μW
ENOB	9.505	bits

Tabla 5.1: Características del SAR ADC diseñado

Capítulo 6

Conclusiones y Trabajo futuro

6.1. Conclusión

Este trabajo implementa un SAR ADC con tecnología CMOS de 180nm y alimentación de 1V con un buen balance precisión-consumo-velocidad-tamaño. Para lograrlo combina un DAC capacitivo con el muestreo, usa la estructura BWC para el array de condensadores del DAC y utiliza un comparador latch dinámico.

Para la elección de arquitecturas usadas se ha hecho un estudio de los convertidores ADC, de sus tipos más usados y después una vez elegido, de las formas en que se puede implementar. Todo esto pensando en qué partes aportan el mayor beneficio para la aplicación para la que se está diseñando.

Durante el diseño se ha seguido una metodología "top-down approach" en el que se diseña el circuito con bloques ideales y poco a poco se sustituyen por los reales.

Los bloques que han sido más críticos para implementar han sido los interruptores y el comparador. Esto es debido a las capacidades parásitas que introducen a las líneas, la inyección de carga y el ruido "kickback" del comparador. Para combatir estos problemas se les ha realizado un estudio, implementado varias soluciones y ajustado valores con el fin de obtener una buena solución de compromiso de precisión-velocidad.

A la vista de los resultados, podemos concluir con que se ha obtenido un SAR ADC óptimo para apliaciones que requieran un buen trade-off entre sus especificaciones.

6.2. Propuesta de trabajo futuro

El SAR ADC diseñado ha obtenido buenas prestaciones pero todavía queda lejos de poder integrarse en un chip y convertirse en un producto. El circuito está implementado a nivel de transistor por lo que habría que diseñar y simular su layout en silicio para que pueda ser manufacturado.

Además, el bloque digital de aproximaciones sucesivas se tendría primero que sintetizar para poder integrarlo en el layout. Como es una lógica sencilla, la sintetización e integración de esta arquitectura será rápida.

En este trabajo no se han aplicado técnicas de cancelación de offset para el comparador, por lo que en un chip real podrían aparecer errores importantes de offset que afecten la precisión del ADC.

En base a cómo han ido deteriorándose las prestaciones del circuito al introducir bloques reales, el bloque que más impacto ha tenido ha sido el del comparador. En este trabajo se han buscado conseguir al menos 9 bits efectivos, pero se podría intentar sacar el máximo jugo del SAR con resolución de 10 bits. La forma en que afronto este problema es ajustando las dimensiones del par diferencial para que haya un buen balance entre su offset (precisión) y el ruido que añade a la línea. Pero se podrían estudiar y aplicar en detalle soluciones para combatirlo de una forma más efectiva aumentando así la precisión del ADC.

Otra mejora que se podría hacer es diseñar buffers o protecciones a las entradas del circuito ya que en el diseño he empleado fuentes ideales de tensión pero en el mundo real no se sabe que voltaje podría aparecer a la entrada del ADC. Por lo tanto podría darse el caso de que se rompiera o se viera afectado su funcionamiento.

Bibliografía

[1] Analog Devices. “Mixed Signal and DSP Design Techniques”. En: 2002. URL: https://www.analog.com/en/education/education-library/mixed_signal_dsp_design_book.html.

[2] Rodrigo Lorente Sanjurjo. “ANALYSIS AND DESIGN OF CONVERTERS IN MATLAB”. En: 2010. URL: https://e-archivo.uc3m.es/handle/10016/11150.

[3] R. Jacob Baker. *CMOS Circuit Design, Layout, and Simulation*. 3rd. Wiley-IEEE Press, 2010. ISBN: 0470881321. URL: https://www.researchgate.net/publication/295256070_CMOS_Circuit_Design_Layout_and_Simulation_Third_Edition.

[4] Ali Hajimiri. “Analog Circuit Design by Ali Hajimiri”. En: 2019. URL: https://www.youtube.com/watch?v=ljlDriLo_7U&list=PLc7Gz02Znph-c2-ssFpRrzYwbzplXfXUT&index=82&t=616s.

[5] Microchip. “microchip.com”. En: 2020. URL: https://microchipdeveloper.com/adc:adc-differential-nonlinearity.

[6] National Instruments. “NationalInstruments”. En: 2020. URL: https://www.ni.com/es-es/support/documentation/supplemental/06/understanding-frequency-performance-specifications.html.

[7] Rafael Ramos Lara. “Sistemas Digitales de Instrumentación y Control”. En: 2007. URL: https://upcommons.upc.edu/bitstream/handle/2117/6122/TEMA2.pdf.

[8] DigiKey. “Asociar el ADC correcto con la aplicación”. En: 2018. URL: https://www.digikey.es/es/articles/match-the-right-adc-to-the-application.

[9] Michiel van Elzakker. “Low-power analog-todigital conversion”. En: 2006. URL: https://pdfs.semanticscholar.org/bd36/e8548e401079fa78c748e7a2558f6cfab514.pdf.

[10] Walt Kester. “Which ADC Architecture Is Right for Your Application?” En: 2005. URL: https://www.analog.com/en/analog-dialogue/articles/the-right-adc-architecture.html.

[11] Dai Zhang. “Design and Evaluation of an Ultra-Low Power Successive Approximation ADC Master thesis performed in Electronic Devices”. En: 2009. URL: http://www.diva-portal.org/smash/get/diva2:216811/FULLTEXT01.pdf.

[12] Raheleh Hedayati. “A Study of Successive Approximation Registers and Implementation of an Ultra-Low Power 10-bit SAR ADC in 65nm CMOS Technology”. Tesis doct. Sep. de 2011.

[13] Long Chen. “Design Techniques for Low-power SAR ADCs in Nano-scale CMOS Technologies”. En: 2016. URL: https://repositories.lib.utexas.edu/handle/2152/40286.

[14] B.Divya J.Pravalika S.Mamatha P.Vineesha. “DESIGN AND SIMULATION OF SUCCESSIVE APPROXIMATION ADC IN VERILOG USING MODELSIM”. En: 2013. URL: http://docshare02.docshare.tips/files/17055/170555507.pdf.

[15] https://electronics.stackexchange.com/. “https://electronics.stackexchange.com/”. En: 2020. URL: https://electronics.stackexchange.com/questions/423487/charge-injection-confusion.

[16] Vicente Torres Carot. “apuntes de Procesado de Señal en Sistemas Electrónicos”. En: 2020.