1. **introducción y estado del arte**

Este proyecto se incluye en el marco de los VCO-ADCs. Este tipo de convertidores analógico-digitales están empezando a ser utilizados tanto en áreas de investigación como comerciales, por su bajo consumo y alta precisión en las conversiones. Estos convertidores están basados en los convertidores sigma-delta, ampliamente utilizados en la industria desde hace muchos años.

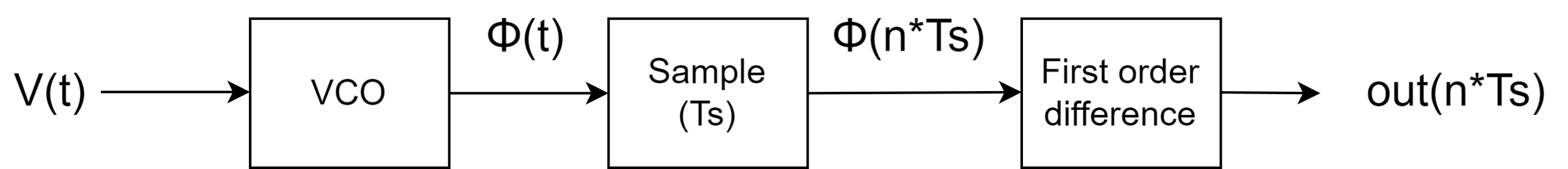
Frente a otros tipos de convertidores, los VCO-ADCs presentan varias ventajas y desventajas. Entre sus ventajas se encuentran un bajo consumo energético, área y relación señal a ruido. Por otro lado, los VCO-ADCs, al igual que los convertidores sigma-delta, presentan el inconveniente de no realizar conversiones instantáneas, produciendo un retraso significativo entre el cambio en la señal de entrada analógica a la señal de salida digital. Además, no son aptos para alta frecuencia, por ser convertidores de sobre muestreo.

Aunque estos inconvenientes son detrimentales en ciertas aplicaciones, hay otras en las que no tienen apenas impacto. Tal es el caso del procesamiento de señales de audio, dado que son de baja frecuencia y no se generan problemas por retrasos de algunas decenas de microsegundos en las conversiones.

A lo largo de este Trabajo de Fin de Máster se estudia el diseño conceptual y a nivel de sistema de un VCO-ADC de muy bajo consumo basado en múltiples osciladores utilizado para la digitalización de señales de audio procedentes de un micrófono MEMS. También se detalla la implementación a nivel de transistor de algunos bloques de este.

* 1. Que es un VCO-ADC

Un VCO-ADC es un convertidor analógico digital basado en un oscilador controlado por voltaje. De manera intuitiva, puede entenderse que un cambio de voltaje en una señal de entrada produce una variación de frecuencia en el oscilador controlado por voltaje (VCO). El VCO se encuentra conectado a un circuito electrónico con la capacidad de transformar una frecuencia de entrada en un número binario a la salida (Frequency to Digital). De esta forma, variaciones en el voltaje de entrada producen variaciones en la frecuencia del oscilador, que producen variaciones digitales a la salida. Si bien puede parecer que esto no es una tarea sencilla, y no lo es desde un punto de vista matemático, los circuitos electrónicos resultantes no son complejos de implementar y son relativamente pequeños en área. Esta es precisamente la virtud de los VCO-ADCs. A continuación, se explica matemáticamente el funcionamiento de un VCO-ADC básico, que tiene una descripción de sistema como la siguiente:



La ecuación fundamental de un VCO lineal con frecuencia en reposo igual a 0 es la siguiente:

F(t) = K \* V(t), donde K es la ganancia del oscilador en Hz / V

Integrando ambas partes de la ecuación obtenemos

**∫** F(t) dt = **∫** K \* V(t) dt

Como la frecuencia es la derivada de la fase con respecto al tiempo, podemos reescribir la ecuación anterior como

Φ(t) = **∫** K \* V(t) dt

Suponiendo que existe un dispositivo capaz de muestrear la fase del oscilador de forma digital, la salida de este dispositivo con periodo de muestreo Ts es:

Φ(n\*Ts) =  **∫** K \* V(n\*Ts) + E(n\*Ts)

Se ha añadido una función E(t), que representa el error cometido por muestrear digitalmente la fase, al no disponer de infinitos niveles de precisión.

Finalmente, si realizamos la primera diferencia, es decir, Φ(n\*Ts) - Φ((n – 1)\*Ts), obtenemos

Δ Φ (n\*Ts) =**∫** K \* V(n\*Ts) - **∫** K \* V((n-1)\*Ts) + E(n\*Ts) - E((n-1)\*Ts)

Y, si Ts es lo suficientemente menor que el periodo de la señal muestreada, podemos realizar una aproximación lineal de la diferencia de integrales:

Δ Φ (n\*Ts) =K \* V(n\*Ts) \* Ts + E(n\*Ts) - E((n-1)\*Ts)

Esta es, precisamente, la salida digital del sistema

Out(n\*Ts) = K \* V(n\*Ts) \* Ts + E(n\*Ts) - E((n-1)\*Ts)

El error de salida es la diferencia de los dos últimos errores, de forma equivalente a un convertidor sigma-delta de primer orden. Como se ve en las siguientes ecuaciones, realizar esta operación con el error equivale a reducir el ruido de baja frecuencia.

Z(E(n\*Ts) - E((n-1)\*Ts)) =

E(z) – z^-1 + E(z) =

E(z) \* (1 – z^-1)

Vista como función de transferencia en el plano Z, la ecuación XXX, tiene la siguiente respuesta:



Esta respuesta puede ser interpretada como un filtro de paso alto con respuesta lineal en fase.

Por otro lado, el valor digital de salida depende de K y Ts. Fundamentalmente, un sistema equivalente al descrito matemáticamente en este apartado obtiene un valor digital de salida proporcional al voltaje de entrada, y realiza noise-shaping de primer orden del error de cuantización.

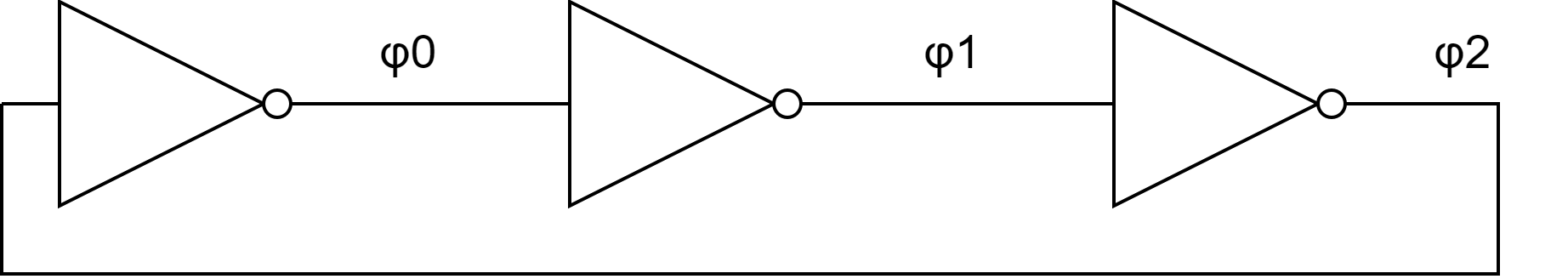
1.1.2 Como implementar un VCO-ADC

Como se ha visto, un VCO-ADC consta de tres partes fundamentales: Un VCO, un muestreador de la fase y la realización de la primera diferencia.

1.1.2.1 Como implementar un VCO

Un VCO puede implementarse de muchas maneras. Para este trabajo, se analiza una implementación con un oscilador en anillo, por ser fácilmente integrable y tener un bajo consumo y área. Sin embargo, esta elección no está exenta de dificultades, pues un oscilador en anillo es sistema altamente no lineal que se ve alterado fácilmente por las condiciones externas.

El oscilador en anillo básico consiste en un número impar de inversores (puertas NOT) conectadas en bucle, como en la siguiente imagen:



Este sistema oscila, ya que no es posible encontrar una configuración estable para el mismo. El periodo total de oscilación de un oscilador en anillo puede calcularse a partir del tiempo de respuesta de los inversores que lo componen. Si un inversor tiene un tiempo de respuesta Ti, y un oscilador en anillo está compuesto por 3 inversores, su periodo de oscilación será 2\*3\*Ti. La siguiente tabla muestra el estado del oscilador en tiempo discreto a lo largo de un periodo de oscilación:



Donde en rojo se muestra la fase que se encuentra en un estado inestable. Puede observarse como, efectivamente, son necesarios 6 periodos (6 tiempos de reacción de un inversor individual) para que el oscilador vuelva a su estado inicial. De forma genérica, podemos establecer que el periodo del oscilador completo es 2 \* Ninv \* Ti.

Para variar la frecuencia de un oscilador en anillo, es necesario aumentar o reducir la corriente de alimentación de los inversores. Una mayor corriente implica un menor tiempo de respuesta, por tanto, una mayor frecuencia de oscilación y viceversa.

Finalmente, para construir un VCO a partir de un oscilador en anillo, basta con añadir un transconductor a la entrada del oscilador, que transforma un voltaje de control en una corriente de control.

No obstante, una característica fundamental del VCO resultante para que pueda ser utilizado en un VCO-ADC es que cumpla la ecuación de funcionamiento XXX, que básicamente indica un comportamiento lineal.

Ciertamente, el resultado de duplicar la corriente de control de un oscilador en anillo no resulta en un aumento del doble de su frecuencia. Sin embargo, es posible establecer un régimen lineal de operación entre ciertos valores. Cuando se superan estos valores, el oscilador en anillo entra en una zona no lineal, lo que provoca un mal funcionamiento del VCO-ADC. Por tanto, esta zona de operación determina el rango dinámico del convertidor. Un oscilador en anillo tiene una función de transferencia con una forma similar a la siguiente:

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Puede apreciarse en la figura la zona de operación lineal.

1.1.2.2 Como implementar un muestreador de fase

Gracias a que el estado de un oscilador en anillo puede describirse digitalmente, la fase del oscilador es sencilla de muestrear. De hecho, en la figura XX se han numerado diferentes estados de un oscilador en anillo. De forma efectiva, cada uno de estos números representa, de forma discreta, la fase del oscilador. Realmente, la operación realizada al asignar cada número a un estado en el caso de la figura es:

R(t) = floor(Φ(t) \* (6 / (2 \* pi))) mod 6

Ya que la fase va desde los valores 0..5 en vez de 0…2\*pi y se repite.

R(t) puede realizarse en hardware con un contador que aumente su valor cada vez que cambia el estado del oscilador en anillo. Este contador puede ser un contador binario normal, conectando su reloj a la salida de la operación XOR de todas las fases del oscilador. De hecho, es posible utilizar un contador de N bits, tal que su salida sea

R(t) = floor(Φ(t) \* (6 / (2 \* pi))) mod 2^N

R(t) es ahora fácilmente muestreable mediante un simple registro con un reloj a la frecuencia Fs. Un ejemplo de este sistema, con un contador de 3 bits, se muestra en la siguiente figura:



Podría parecer un problema que R(t) se de la vuelta cada N estados. Sin embargo, no es tal el caso, ya que lo realmente interesante es la diferencia de fase Δ Φ (Eq XXXX), por lo que siempre que el muestreo de R(t) se realice de forma suficientemente rápida para que el contador no haya dado una vuelta completa entre dos muestras, se obtendrá un resultado Δ Φ correcto.

Matemáticamente, esto puede analizarse de la siguiente manera:

La salida del contador es del tipo

R2(n \* Ti) = n mod 2^N, donde Ti es el tiempo que tarda en aumentar en una unidad el contador (que es igual al tiempo de respuesta de un inversor del oscilador).

Así,

R2(n2 \* Ti) = n2 mod 2 ^ N

R2(n1 \* Ti) = n1 mod 2 ^ N

Por lo que

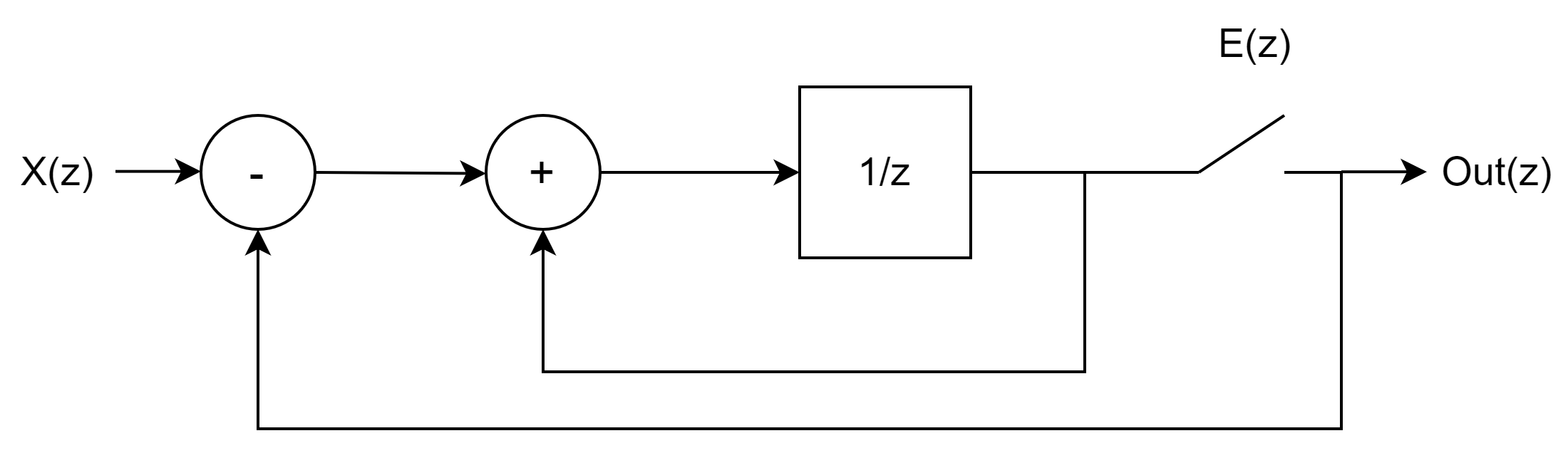
R2(n2 \* Ti) – R2(n1 \* Ti) = (n2 – n1) mod 2^N.

Mientras que el valor absoluto de (n2 – n1) sea menor que 2^N, el resultado de la primera diferencia será correcto. Este es el límite absoluto para la frecuencia de muestreo, teniendo en cuenta la frecuencia máxima del oscilador.

**1.1.2.3 Como implementar un noise-shaper**

Como se ha visto en las ecuaciones XXXX, con simplemente utilizar la primera diferencia, ya se consigue noise-shaping de primer orden. Es por esto por lo que se considera que los VCO-ADCs tienen noise-shaping de primer orden de forma “inherente”.

Sin embargo, para lograr una buena SNR en el convertidor, es necesario utilizar una frecuencia de muestreo muy alta, lo que puede resultar impráctico para el tratamiento de la señal de salida. En la industria, el estándar de transmisión de convertidores AD para audio es 3.2 MHz. En el caso de un VCO-ADC, esto supone un problema, ya que la tasa de muestreo puede situarse en decenas o centenas de MHz. Por tanto, es necesario realizar un diezmado, lo que reduciría enormemente la SNR del convertidor si no se realizase de manera adecuada. Una forma de reducir este problema es utilizar un modulador sigma-delta en el proceso de diezmado. Este modulador añade noise-shaping al error introducido por el remuestreo:



Puede interpretarse el remuestreo como un error que se introduce en la señal para cada n, tal que, si no hubiera un modulador entre medias, Out(n) = X(n) + E(n).

Si analizamos el diagrama de bloques de la figura:

Out(z) = (X(z) – Out(z)) \* (z^-1 / (1 – z^-1)) + E(z)

Out(z) \* (1 + (z^-1 / (1 – z^-1))) = X(z) \* (z^-1 / (1 – z^-1)) + E(z)

Out(z) = X(z) \* z^-1 + E(z) \* (1 – z^-1)

Como se ve, se ha conseguido el mismo resultado que en XXXX: Mantener la señal de entrada intacta y realizar noise-shaping del ruido introducido.

Este modulador es directamente implementable en hardware utilizando un registro en el lugar del bloque z^-1

Con los bloques de esta sección, es posible implementar un VCO-ADC con el estado del arte actual. Sin embargo, como se verá más adelante, en este proyecto se utilizan ligeras variaciones de las implementaciones descritas anteriormente.

**1.1    Procesamiento en el Borde de señales de audio**

El procesamiento en el borde (Edge-computing) para señales de audio es un campo de mucha importancia y en el que se ha realizado mucha investigación, ya que el procesamiento de audio es una tarea muy importante y que debe realizarse con poca latencia. Además, el proyecto consiste precisamente en utilizar estas técnicas para seleccionar **1.2    Parametros de audio de un microfono inteligente**

// Tengo que preguntar que pongo aquí jajaja ¿SNR? ¿Rango dinámico? ¿SNDR? ¿Awaiting? ¿SQNR

Carlos andrez perez cruz – Ruben garvi – Sensors. **1.3    Planteamiento del proyecto: diseño de un microfono con   
extension de rango dinamico mediante procesamiento en el borde**

Como se ha expuesto en apartados anteriores, tanto un rango dinámico amplio como una relación señal a ruido alta son características deseables en los convertidores A/D para micrófonos inteligentes. En el caso concreto de un VCO-ADC que emplea un oscilador en anillo esto supone un problema, ya que para aumentar su relación señal a ruido es necesario aumentar la ganancia del control de su oscilador interno, mientras que aumentar esta ganancia implica reducir el rango dinámico. Esto es debido a que un oscilador en anillo solamente tiene un comportamiento lineal bajo ciertas condiciones de corriente/voltaje y temperatura. Obviando la temperatura, la frecuencia producida por una corriente de control I tiene una relación similar a la siguiente:

Gráfico, Gráfico de líneas

Descripción generada automáticamente

En la parte lineal de la función, podemos aproximar el valor de F con:

F(I) = K \* I + F0, donde F0 es la frecuencia en reposo del oscilador

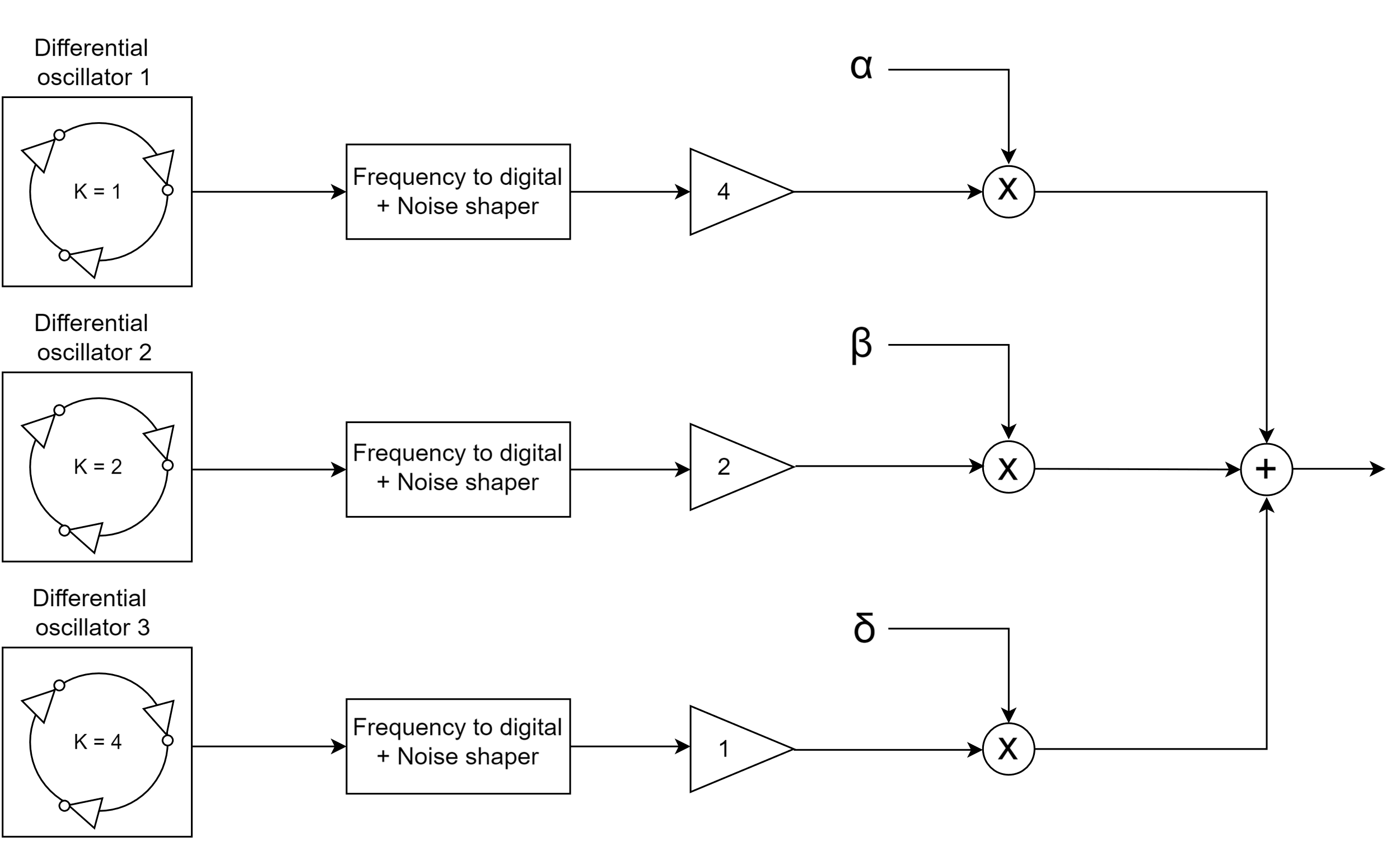
Si asumimos que I es una variable que depende del tiempo, entones

F(t) = K \* I(t) + F0.

Por tanto, si para una ganancia de control K=1 el régimen linear del oscilador del ejemplo se encuentra aproximadamente en el intervalo [-2,2] (mA), para una ganancia K cualquiera este intervalo se reduce a [-2/K , 2/K] (mA), habiendo reducido el rango dinámico de forma proporcional al aumento de ganancia.

La solución propuesta para este problema es utilizar varios osciladores en anillo en un mismo convertidor, y combinar sus salidas para obtener un resultado que tiene tanto una alta ganancia para señales pequeñas, y por tanto muy buena relación señal a ruido en la conversión de estas, como un alto rango dinámico que permite la conversión de señales grandes.

La siguiente figura muestra un diagrama de bloques conceptual que ejemplifica esta solución:



Antes de continuar con la explicación, es necesario hacer un inciso para comentar el carácter diferencial del sistema implementado, pues es fundamental para la siguiente sección.

Es bien sabido que en circuitos electrónicos sensibles al ruido es recomendable utilizar circuitos diferenciales. Tal es el caso para este convertidor analógico digital. Además, en este caso, ofrece una ventaja extra. Utilizando dos osciladores como los del ejemplo anterior, donde K1 = -K2:

F1(t) = K1 \* I(t) + F0

F2(t) = -K1 \* I(t) + F0

Si restamos F1(t) – F2(t), el resultado es

F(t) = 2 \* K1 \* I(t)

Puede observarse que el resultado equivale a un nuevo oscilador cuya frecuencia en reposo es 0.

Para lograr combinar las salidas de varios osciladores con diferente ganancia, es necesario ajustar la salida de cada oscilador antes de la combinación. De esta forma, para el circuito “combinador”, no habrá ninguna diferencia más allá de la SNR de cada uno. Por supuesto, este ajuste se realiza después de haber digitalizado la señal, ya que de hacerlo antes empeoraría la SNR de los de mayor ganancia. En el diagrama de bloques conceptual (Figura XX), este ajuste está representado como un bloque de ganancia.

En las siguientes figuras se observan las funciones de transferencia de tres osciladores en anillo, cada uno con ganancias distintas, y el ajuste realizado digitalmente:

Gráfico, Gráfico de líneas

Descripción generada automáticamente

F1(I) = K \* I(t)

F2(I) = 2\* K \* I(t)

F3(I) = 4\* K \* I(t)

Después del ajuste:

Gráfico, Gráfico de líneas

Descripción generada automáticamente

F1(I) = 4\*K \* I(t)

F2(I) = 4\* K \* I(t)

F3(I) = 4\* K \* I(t)

Dentro del régimen lineal, F1, F2 y F3 son indistinguibles.

Como puede observarse, después del ajuste digital de ganancia, es perfectamente posible mantenerse siempre dentro del régimen lineal de alguno de los osciladores a lo largo de todo el rango dinámico, pudiendo aprovechar al máximo la SNR de los de mayor ganancia cuando la señal de entrada es pequeña. Al hacer K3 = 2\*K2 = 2\*K1, no es necesario ningún elemento de hardware extra para realizar el ajuste digital, pues basta con hacer desplazamientos.

Ahora bien, para lograr un comportamiento óptimo del convertidor, es necesario generar las señales α, β, δ (ver Figura XX) de tal forma que α + β + δ = 1, y asegurarse de que los osciladores que se encuentran fuera de su régimen lineal tengan una ponderación casi nula en cualquier situación.

La siguiente sección analiza dos posibles métodos para lograrlo. **1.3.2 Combinación de caminos con redes neuronales**

Una prometedora opción a explorar es el uso de redes neuronales, ya que se cree, en base a su buen funcionamiento en problemas similares (referencias), podrían conseguir un muy buen resultado.

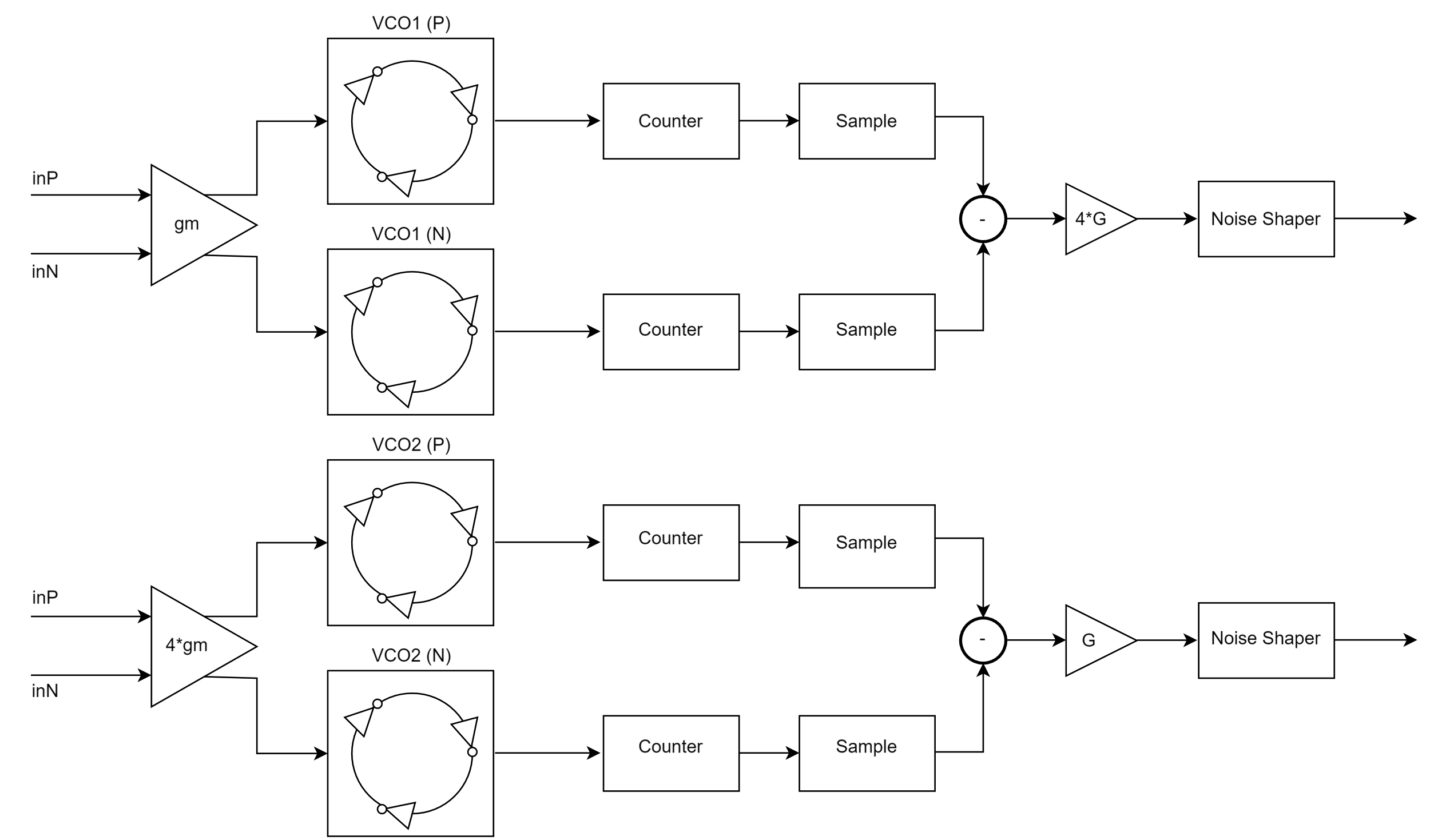
Una red neuronal simple con tres entradas, una para el valor de cada oscilador, y tres salidas utilizando la función softmax en la capa de salida podría realizar esta labor convenientemente. Se cree que la red neuronal no necesitaría una gran cantidad de capas internas, al tratarse de un problema relativamente simple.

Sin embargo, la red neuronal presenta varias desventajas. En primer lugar, debe ser entrenada, posiblemente con datos de usuarios y sus percepciones, lo que supone un coste muy grande. Además, y posiblemente más importante, la implementación de una red neuronal es muy costosa en hardware, disparando el área y el consumo de un convertidor que debe ser de muy baja potencia.  **1.3.3 Combinación de caminos con estimadores estadísticos de   
potencia**

Una opción más clásica para la generación de α y β es utilizar la potencia de la señal de entrada para ponderar cada salida. Dicha potencia puede ser comparada con valores preestablecidos, que junto a un algoritmo clásico establecen la ponderación de cada camino. Esta forma de abordar el problema es más sencilla en coste, consumo y área, pero presenta también varios inconvenientes. En primer lugar, para aprovechar al máximo el rango dinámico de cada oscilador, los valores preestablecidos deben ser cuidadosamente seleccionados. Además, el algoritmo que establece las ponderaciones debe ser simple y a la vez funcional. **2. Diseño del sistema implementado**

La siguiente sección analiza el sistema implementado a nivel de sistema en detalle. La implementación hardware concreta de cada componente se describe en la sección 3: Diseño hardware del sistema  **2.1 Diseño de alto nivel de un sistema de adquisición de audio   
con extensión de rango y 2 VCO**

El sistema planteado utiliza 2 VCO para probar el concepto de la extensión de rango de manera sencilla. Sin embargo, el diseño es fácilmente escalable para utilizar más osciladores. Para cada VCO se dispone de un elemento Noise-Shaper, que permite alcanzar una mejor relación señal a ruido en cada canal. Se dispone pues de dos caminos, de ahora en adelante HDR (High Dynamic Range) y HSNR ( High Signal-to-Noise Ratio). La única diferencia efectiva es que la ganancia del oscilador del segundo es cuatro veces la del primero, que es corregida digitalmente como se ha explicado en el apartado XXX. El diagrama a nivel de sistema para cada camino es el siguiente:



Para comprobar el desempeño del sistema, se ha creado en Simulink un modelo, adjunto como integer\_true.slx. En este modelo se simulan los VCO mediante bloques como los siguientes:

Diagrama, Esquemático

Descripción generada automáticamente

El bloque roX es el oscilador propiamente dicho, y contiene los inversores. Se realiza un tratamiento previo a la señal de entrada donde se le añade el ruido correspondiente al oscilador, así como la ganancia. Utilizar una ganancia inversa en cada uno de los VCOs proporciona el comportamiento diferencial buscado.

El contador, muestreo y noise-shaper son fácilmente implementables con acumuladores.

Finalmente, se realiza la corrección digital de ganancia

Al ejecutar simulación funcional del modelo para una señal de entrada pequeña (-18 dB con respecto al fondo de escala), se comprueba que el resultado de salida es el esperado. Se utiliza una señal pequeña y no de fondo de escala para la comparación para evitar que el canal HSNR entre en saturación

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Puede observarse que el canal HDR tiene una forma de onda con mucho más ruido, pero ambas representan una señal con la misma amplitud, habiendo conseguido el propósito del sistema. Es fácilmente comprobable en el plano de la frecuencia que la SNR del canal HDR es inferior a la del HSNR, en el caso concreto de esta señal, respectivamente, 74.49 dB y 85.63dB.

Gráfico, Gráfico de superficie

Descripción generada automáticamente

Una vez comprobado que es posible implementar a nivel de sistema ambos caminos, se procede a decidir el método por el que se va a implementar la extensión de rango dinámico. Se contemplan a continuación las dos opciones planteadas en el apartado XXXX.

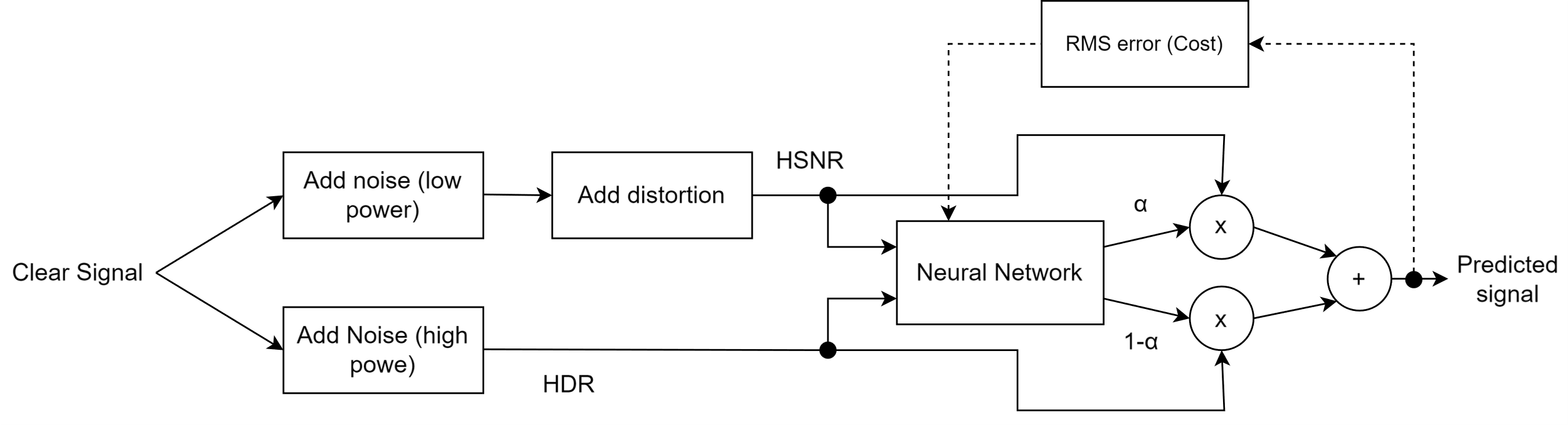
**2.2 Opción 1: optimización del rango dinámico por comparadores   
implementados con redes neuronales**

Con el fin de estudiar el uso de redes neuronales para la selección de caminos, se ha desarrollado un programa en Python, utilizando Keras y Tensorflow, que entrena una red neuronal para asignar un peso a cada uno de los dos caminos. La red neuronal diseñada tiene dos entradas, correspondientes al valor en tiempo real de los caminos HDR y HSRN, dos capas ocultas, una de 32 y otra de 16 neuronas activadas mediante ReLU, y una capa de salida que utiliza la función Softmax, produciendo los valores α y 1 – α.



La arquitectura de esta red se ha diseñado de esta manera por varios motivos. En primer lugar, se proporcionan suficientes capas y neuronas intermedias para que esto no suponga una limitación, dado que el objetivo es comprobar la viabilidad del uso de redes neuronales para resolver el problema. En segundo lugar, la capa de salida proporciona dos pesos que han de sumar uno. Por ello, se utiliza la función softmax, que es típicamente utilizada en labores de clasificación por esta propiedad. En las demás capas se utiliza ReLU (Rectified Linear Unit) por su eficiencia computacional.

La estructura completa del sistema de simulación y entrenamiento de la red neuronal es la siguiente:



Como entradas a la red, se van a utilizar dos señales que representan el canal HDR y el canal HSNR respectivamente. El canal HDR debe tener una distorsión mínima, pero una cantidad de ruido alta. Por el contrario, el canal HSNR debe tener mucha distorsión para una señal de entrada grande, pero una cantidad reducida de ruido.

Para simular el efecto del ruido y la distorsión de los VCOs, se aplica una distorsión como la siguiente al canal HSNR:

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Esta función, definida como (1 / (1 + np.exp(-4\*x)) – 0.5) (sigmoidal ajustada para pendiente 1 en el origen y asíntotas en 0.5,-0.5), produce un comportamiento totalmente lineal en el origen y una saturación progresiva hacia los extremos, de forma similar al comportamiento del oscilador en anillo del canal HSNR. Además de esta distorsión, se añade una pequeña cantidad de ruido uniformemente distribuido a la señal.  
Para el canal HDR, no se añade ningún tipo de distorsión, pero se introduce una cantidad de ruido 10 veces superior a la del canal HSNR.

Gráfico, Gráfico de líneas

Descripción generada automáticamentePara una entrada senoidal de amplitud 1, la entrada a la red neuronal de ambos canales es la siguiente, respectivamente:

Las dos señales generadas mediante los procedimientos descritos anteriormente son las entradas a la red neuronal. La salida de la red son los pesos por los que deben ser multiplicadas cada una de las entradas, por lo que, tras realizar una ponderación y suma, se obtiene la señal predicha por la red, que es la salida del sistema. Si se está realizando un entrenamiento de la red, se calcula el error RMS entre esta salida y la señal de entrada (Clear signal). Este resultado es utilizado como función de coste por el optimizador de la red neuronal en el entrenamiento.

Tras entrenar la red utilizando como señal de entrada una rampa senoidal ascendente, se ha probado el desempeño de la misma con diferentes señales, obteniendo con todas ellas un comportamiento similar. A continuación, se presenta el resultado de aplicar una señal senoidal modulada a la entrada, por considerarse una señal especialmente representativa del comportamiento de la red

Gráfico

Descripción generada automáticamente

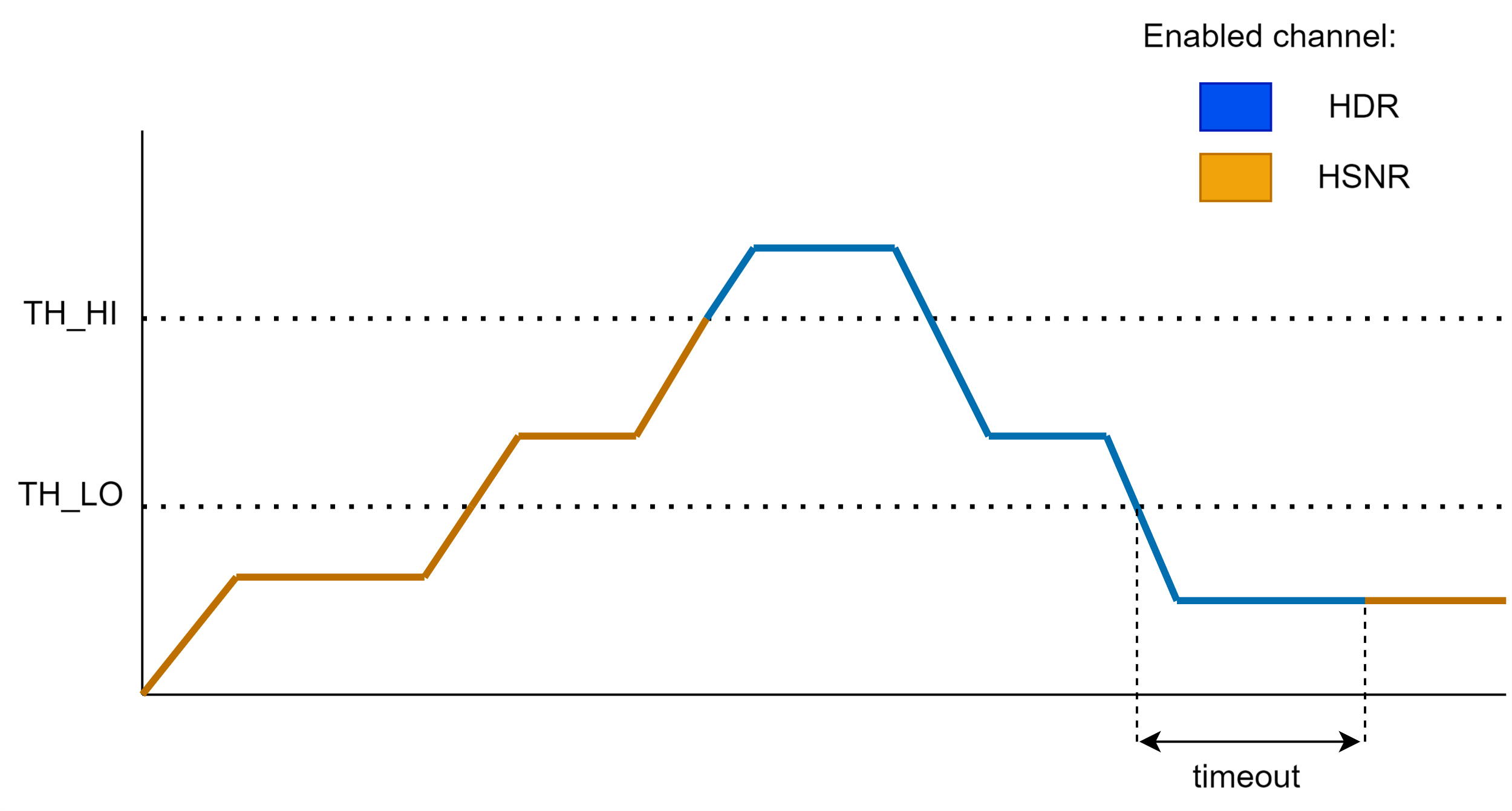
nota: el canal HSNR tiene una escala diferente.

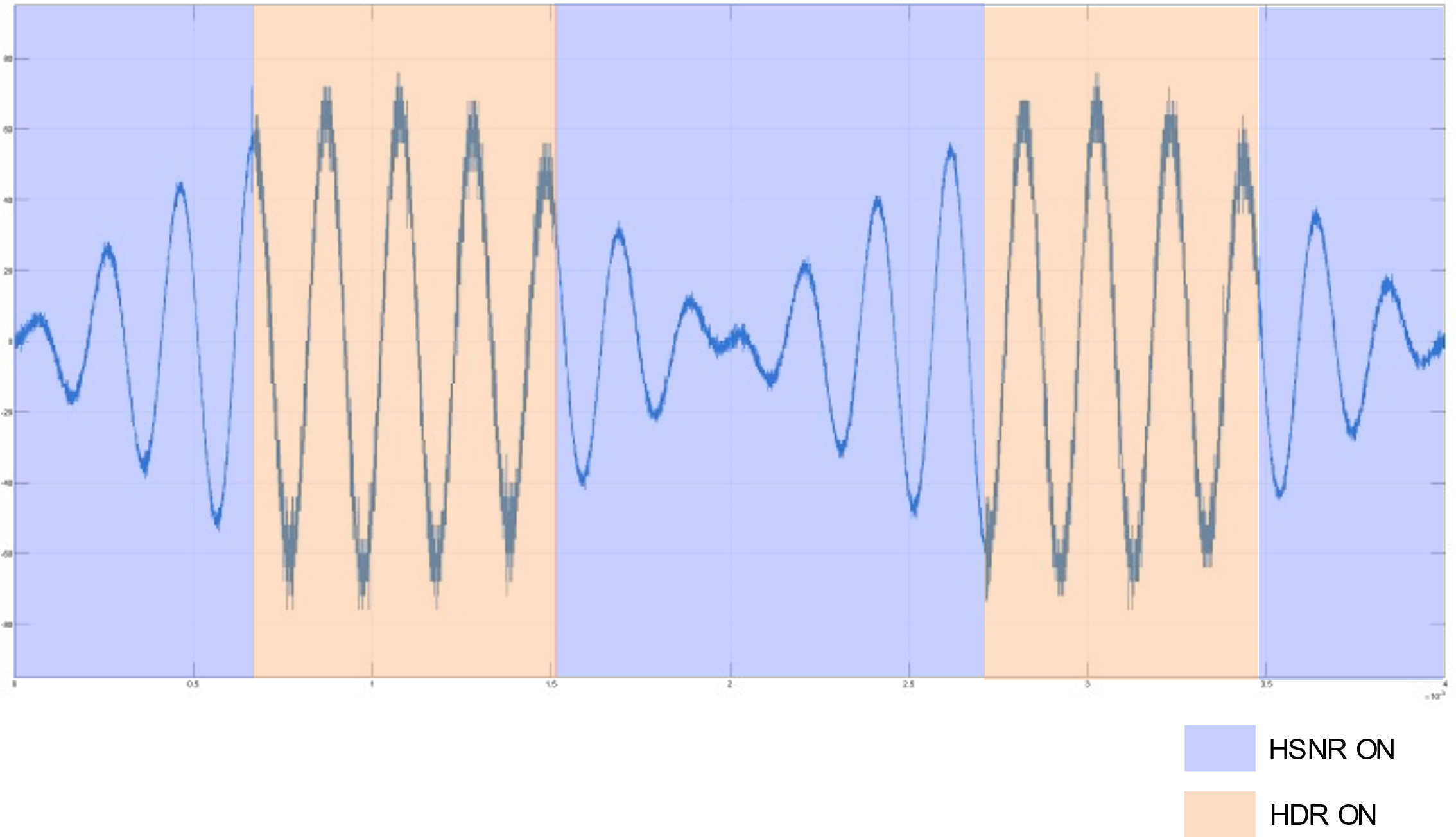
La figura consta de varias señales: la señal de entrada (Clear signal), el resultado de aplicar distorsión y ruido a la misma (HDR y HSNR), los pesos establecidos por la red para cada canal en cada instante de tiempo (Weight HDR y Weight HSNR), y la salida del sistema (Predicted Signal)

Como puede observarse, la red ha aprendido a generar los pesos correctamente, pues la señal predicha es prácticamente igual a la de entrada (Error RMS = 0.03). Cuando la señal de entrada es pequeña, el peso del canal HSNR se ve incrementado, mientras que el del canal HDR se ve disminuido. Cuando la señal de entrada es grande, ocurre lo contrario. Sin embargo, la propia red neuronal introduce ruido al variar los pesos con cierta aleatoriedad para valores cercanos, lo cual puede suponer un problema para una implementación real. Esto, posiblemente, se puede solucionar aplicando algún tipo de filtro a la salida de la red para evitar cambios bruscos. Es interesante comprobar como la red neuronal ha aprendido que, alrededor de una amplitud de 0.2, es deseable ponderar más el canal HDR frente al HSNR, pues es donde la distorsión comienza a ser más notoria (ver figura xx). Es fácilmente observable que la red neuronal a aprendido a detectar un umbral alrededor de ese valor, y que un cambio de canal ocurre muy rápidamente una vez se atraviesa, teniendo un comportamiento casi de selección binaria a la salida.

La red, a pesar de su correcto desempeño, presenta todos los problemas establecidos en el apartado XXXX para una implementación real. Además, se ha comprobado que la red neuronal ha estimado que el comportamiento de selección de camino óptimo es prácticamente una selección binaria dependiente de un umbral, de forma muy similar a como se realizaría mediante estimadores de potencia, que presentan una implementación en hardware mucho más sencilla. Por tanto, se ha decidido utilizar este segundo método. **2.3 opción 2: Optimización del rango dinámico con estimadores   
estadísticos de potencia**

Para desarrollar un comportamiento similar al de la red neuronal del apartado anterior, implementando el hardware con comparadores digitales, se ha implementado un comportamiento como el siguiente. Para simplificar aún más la implementación y reducir área y consumo, se realiza una selección binaria de camino, es decir, que α solo puede tomar los valores 1 o 0:

  
  
El canal HDR es seleccionado inmediatamente cuando se sobrepasa un límite TH\_HI. Este límite debe corresponderse con el valor en que la distorsión del canal HSNR comienza a ser notable. Por otro lado, para que se retorne al canal HSNR una vez se ha activado el HDR, debe atravesarse el umbral TH\_LO, y se debe permanecer bajo ese umbral un tiempo suficiente (timeout en la imagen). El propósito de este tiempo de espera es evitar cambios constantes entre el canal HDR y el HSNR, algo que podría llegar a causar una distorsión audible.

En una simulación en Matlab, utilizando los dos caminos descritos anteriormente y este algoritmo de selección, el resultado es el siguiente: ****El modelo de Simulink utilizado para generar esta simulación se encuentra adjunto con el nombre integer\_true.slx

**3. Diseño de hardware del sistema**El objetivo de esta sección es analizar el sistema implementado, con el fin de lograr un comportamiento como el del sistema descrito en 2.1, “Diseño de alto nivel de un sistema de adquisición de audio con extensión de rango y 2 VCO”, utilizando el método de selección de caminos por estimadores de potencia descrito en el mismo apartado. Como se describe en la figura XX de este apartado, el sistema consta de dos caminos, cada uno con una serie de bloques. (Osciladores (VCOs), Contador, Muestreo, Noise-Shaper). La implementación hardware de cada uno de estos bloques se encuentra descrita por apartados en esta sección.

**3.1 Etapa analógica de entrada**

La entrada al sistema real procede de un micrófono MEMS, que debe ser adaptada para poder utilizarse en el convertidor. Dado que un oscilador en anillo se controla mediante corriente, es necesario añadir un transconductor a su entrada, de forma que las variaciones de voltaje sean transformadas en variaciones de corriente. Para el canal HDR se utiliza un transconductor con 4 veces menos ganancia que para el canal HSRN, para lograr un comportamiento como él de la figura XX (Gráfica corriente vs frecuencia de osciladores con diferente ganancia). La salida de estos transconductores se conectan a un oscilador en anillo diferencial de tipo “feed forward” de 16 etapas. Dado que en la introducción y estado del arte no se ha especificado sobre los diferentes tipos de oscilador en anillo, porque se pretendía dar una visión de conjunto e introductoria en tal sección, se les dedica aquí un pequeño apartado.

Existen varios tipos de oscilador en anillo, siendo el más simple de todos el mostrado en la figura XX (oscilador en anillo no diferencial). Este presenta la ventaja de ocupar un área reducida, pero no puede tener un número de fases par por la naturaleza de su propio diseño, algo que resulta muy inconveniente para el diseño de este VCO-ADC en concreto por razones que se detallarán más adelante. Por otro lado, encontramos los osciladores en anillo de tipo “Direct Cross Coupled” (CC) y de tipo “Feed Forward Cross Coupled” (FFCC). Estos osciladores son ambos diferenciales, pero presentan estructuras diferentes. Por diferencial se entiende, en este contexto, que cada celda del oscilador proporciona dos señales de salida, tal que una es la inversa lógica de la otra, ocurriendo lo mismo con la entrada. A continuación, se muestra una figura con las dos estructuras, extraída del paper “The Analog Behavior of Pseudo-Ring oscillators used in VCO-ADCs”:

Diagrama, Esquemático

Descripción generada automáticamente

El principio de funcionamiento, intuitivamente, consiste en dos osciladores en anillo simples en los que cada fase esta conectada con su fase pareja en el otro oscilador mediante un inversor, de tal forma que ambos osciladores deben mantenerse sincronizados y con señales inversas. Esto es muy fácilmente observable en el oscilador CC (izquierda). En el oscilador FFCC, un principio de funcionamiento similar se utiliza, salvo que, en este caso, cada fase se conecta con la siguiente en lugar de con su pareja (de ahí el nombre feed forward). Para diseñar un oscilador con un número de fases par, basta con invertir una de las conexiones entre celdas, conectando la parte positiva con la entrada positiva de la siguiente celda, y la negativa con la negativa, permitiendo ambos esta posibilidad.

El oscilador de la izquierda (CC) tiene la ventaja de ocupar un área más reducida que el de la derecha (FFCC), además de un layout analógico más sencillo de realizar. Sin embargo, el FFCC presenta ventajas en cuanto a ganancia, motivo por el que se ha utilizado en el diseño de este chip.

No debe confundirse el carácter diferencial de los osciladores con la idea de que el sistema utiliza una señal diferencial. Cada uno de los osciladores de la figura XX, donde se especifica el diseño a nivel de sistema se implementa como un oscilador diferencial FFCC, pero ambos forman parte de un sistema diferencial mayor, que utiliza como entrada una señal diferencial.

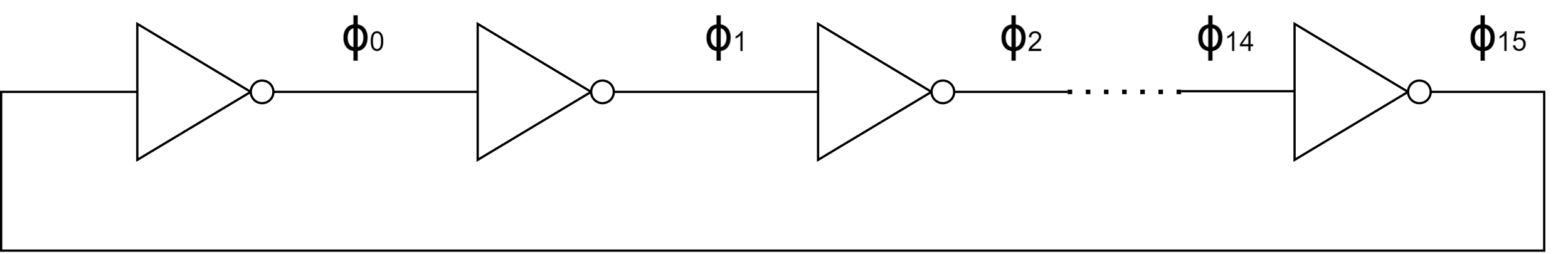
**3.2 Contador**

Un requisito fundamental para diseñar un VCO-ADC es muestrear la fase del oscilador. En el caso de este chip, la fase del VCO se muestrea y digitaliza mediante código Gray. El código gray se utiliza en lugar de binario convencional por su interesante propiedad de que solamente es necesario cambiar de estado un bit al sumar o restar uno a cualquier número. Como es posible que se realice un muestreo en un cambio de estado del contador, y por tanto muestrear un valor indefinido, con binario convencional se podría cometer un error muy grande. Por ejemplo, si el contador está cambiando del estado 0111 al estado 1000 (7 a 8), todos los bits muestreados tendrían un valor indefinido, pudiendo producirse grandes errores. Utilizando código Gray, por la propiedad anteriormente mencionada, solamente se puede cometerse un error de +-1. Para el ejemplo anterior, 7 es 1100 en código Gray, y 8 es 0100, por lo que únicamente el primer bit podría haberse muestreado de forma incorrecta. En cualquiera de los casos, el valor tras muestrear sería 7 u 8, habiendo cometido un error pequeño, de un LSB. Si bien esta codificación no es apta para realizar operaciones aritméticas, basta con añadir un decodificador de código Gray a binario convencional para disfrutar de ambas propiedades.

Diagrama, Esquemático

Descripción generada automáticamente

Es posible que el lector se pregunte como se implementa un contador Gray partiendo de un oscilador en anillo. Para clarificar esto, nombremos cada una de las fases de un oscilador en anillo de 16 etapas como ϕ0 … ϕ15. Si bien en el siguiente diagrama cada celda se dibuja como un oscilador, en el diseño real se trata de las fases positivas de un oscilador en anillo FFCC como el descrito en el apartado anterior. No obstante, esto no tiene ninguna relevancia en la explicación, al margen de solucionar una posible duda sobre la paridad de las fases que pueda surgir.



Para obtener un contador Gray ascendente basta con establecer:

G0 = ϕ1 ⊕ ϕ3 ⊕ ϕ5 ⊕ ϕ7 ⊕ ϕ9 ⊕ ϕ11 ⊕ ϕ13 ⊕ ϕ15

G1 = ϕ2 ⊕ ϕ6 ⊕ ϕ10 ⊕ ϕ14

G2 = ϕ4 ⊕ ϕ12

G3 = ϕ0 ⊕ ϕ8

Para obtener un contador de salida ascendente con bits Counter[3..0] = [G3,G2,G1,G0]. Para entender el porqué de estas operaciones, se puede pensar en un oscilador en anillo como una “inestabilidad” que se propaga a lo largo del círculo de inversores. Solamente una de las fases ϕn cambia de estado al mismo tiempo, la que se encuentra en dicho estado inestable. Con ello en mente, el operador XOR se utilizan debido a una interesante propiedad. Si se cambia el valor de entrada de una y solamente una de variables en una operación XOR como (a ⊕ b ⊕ c … ⊕ n), el resultado siempre se invierte. Dicho de otra manera, la operación XOR de un vector binario [a , b , c … , n] es la paridad de dicho vector.

Es posible utilizar esta propiedad para generar cualquier secuencia binaria a partir de un oscilador en anillo. Para hacerlo, es necesario realizar una operación del tipo ϕa ⊕ ϕb … ⊕ ϕn, donde a,b .. n son las fases que corresponden con un cambio de estado en la secuencia binaria a generar. En el caso del código gray, aparecen marcadas en verde en la siguiente tabla, y por ello son las fases seleccionadas en la operación anterior. (Nota: Es posible que el resultado de aplicar esta técnica sea el inverso bit a bit a la secuencia esperada. En tal caso, basta con añadir un inversor a la salida)



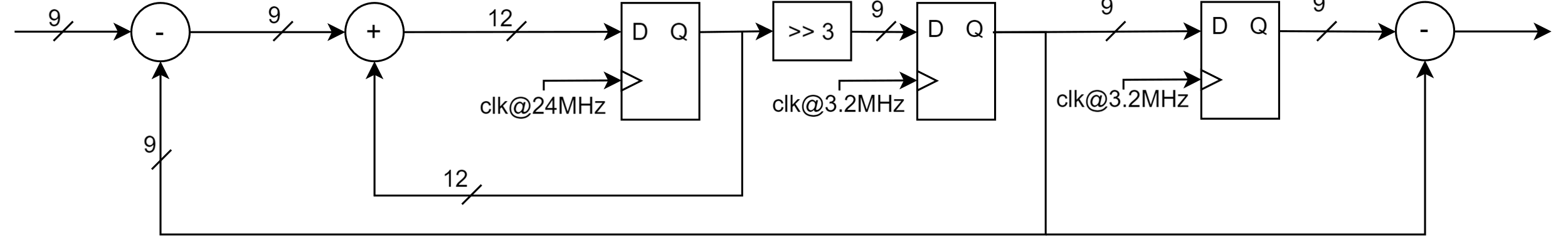
En el caso del código Gray, es muy sencillo conseguir un bit extra para el contador utilizando para ello una conexión directa con la fase ϕ0 del oscilador. Para entender esto correctamente, es recomendable estudiar la figura XX del anexo, donde se muestra la evolución de un oscilador en anillo de 16 etapas tal y como se hace en la introducción con uno de 3.

Una vez realizado este proceso, se dispone de la fase global del oscilador en anillo digitalizada en un número de 5 bits, y tras utilizar un conversor de Gray a binario convencional como el de la figura XX, se dispone de este número en el formato clásico requerido por los circuitos aritméticos eficientes. No obstante, falta por definir la frecuencia del reloj utilizado para muestrear el estado del contador . Esta frecuencia debe ser tal que el contador gray de 5 bits no pueda darse la vuelta entre dos muestras, pues esto induciría a tratar dos datos diferentes como iguales, produciendo un error.

La frecuencia de muestreo debe ser, por tanto, como mínimo la frecuencia del oscilador en anillo. Sin embargo, por razones que veremos más adelante, en este proyecto se utiliza el doble de la frecuencia mínima. **3.3 Noise shaper**

Como se ha especificado anteriormente, la frecuencia de salida del convertidor ha de ser de 3.2 MHz, y para reducir el ruido de cuantificación por el remuestreo, se utiliza un elemento de noise-shaping completamente digital similar al utilizado en los convertidores sigma-delta.

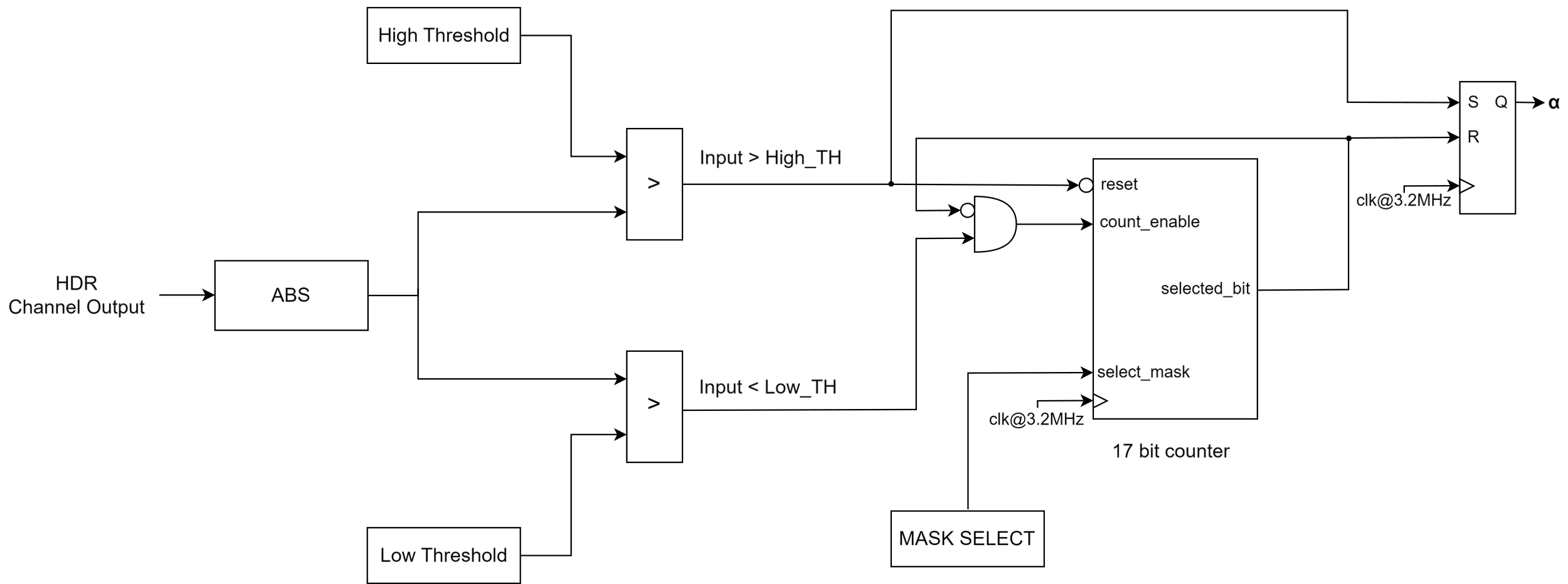
La implementación del noise-shaper se realiza de la siguiente manera, que es una traducción elemento por elemento del sistema descrito en la figura XX de la introducción a hardware:

****

La única diferencia entre ese sistema y esta implementación es la precisión. Es imposible utilizar una cantidad ilimitada de bits en el acumulador interno, por lo que se utiliza la cantidad de bits mínima para que no se produzcan overflows entre dos muestreos a 3.2 MHz.

Esta cantidad resulta en 3 bits más que los circuitos que funcionan a 3.2Mhz, ya que 3 \* 8 = 24. Utilizar esta cantidad de bits, no obstante, reduce la precisión, ya que en la realimentación solo se utilizan los 9 MSB de los 12 bits del acumulador. **3.4 Comparador de umbral y selector de camino**

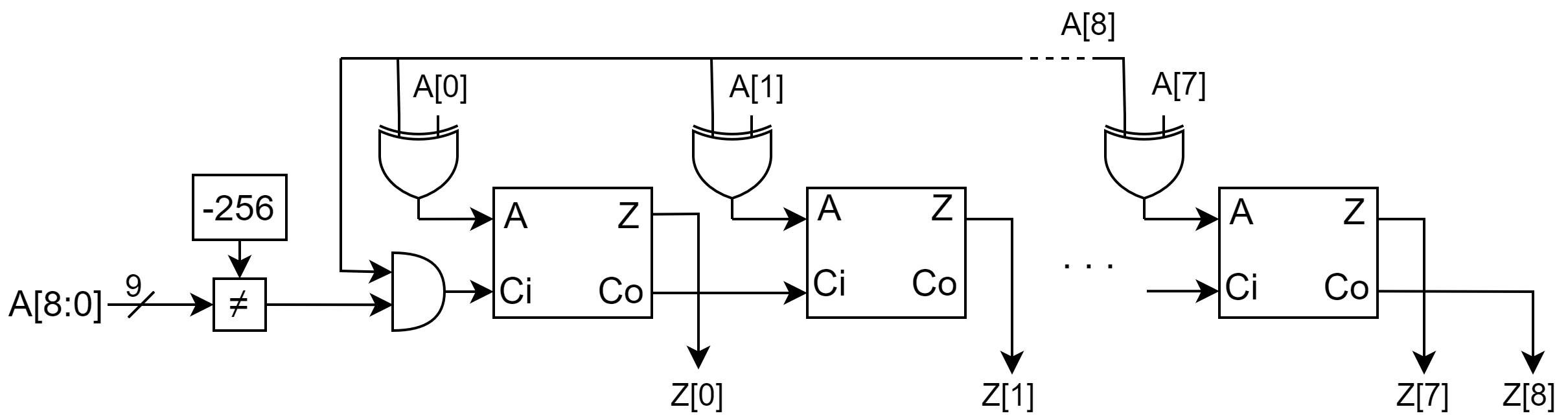
Para conseguir el comportamiento descrito en el apartado XXX, se propone el siguiente circuito:

****

El circuito compara el valor absoluto de la entrada con los dos umbrales. En caso de que dicho valor se encuentre por encima del umbral superior, el flip-flop de salida es se pone a ‘1’ incondicionalmente. Además, se realiza un reset del contador de forma asíncrona. En caso de que el valor caiga por debajo del umbral inferior, el contador cuenta hasta alcanzar un valor programable mediante el registro MASK\_SELECT. Este valor puede ser desde 2^13 hasta 2^17. Estos valores han sido elegidos para que, utilizando un reloj de 3.2 MHz, el “timeout” pueda encontrarse entre 0.04 s (24 Hz) y 0.0025 s (390 Hz). Cuando se alcanza este valor, el contador deja de contar y se activa el canal HSNR. En caso de que el valor se encuentre entre ambos umbrales, el circuito simplemente mantiene su estado.

Para implementar este circuito de forma eficiente, ha sido necesario diseñar el bloque de que realiza la operación de valor absoluto, los comparadores y el contador.

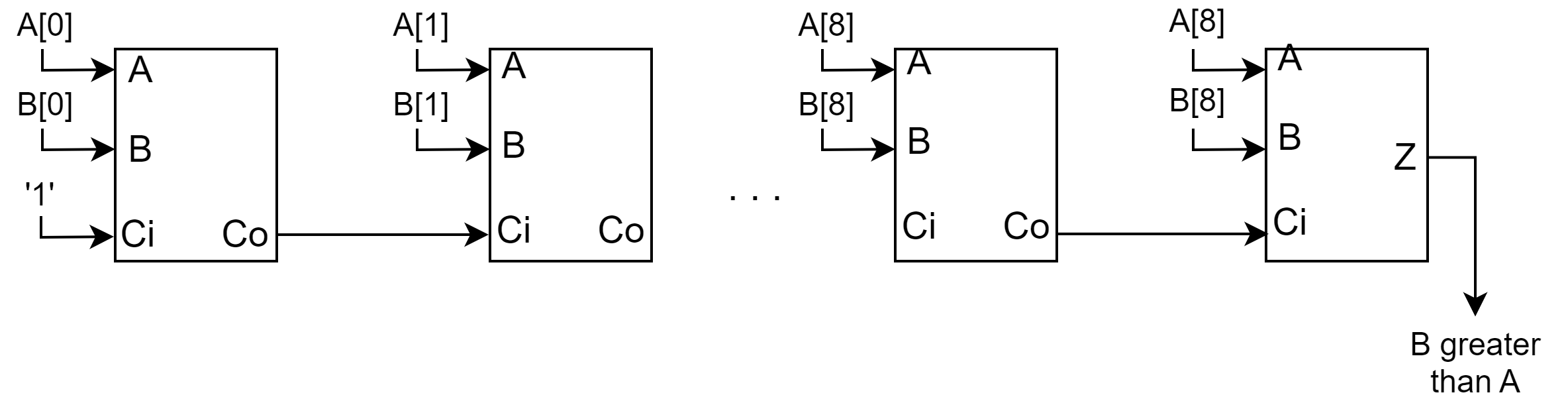
El bloque para calcular el valor absoluto sigue la siguiente estructura:

****

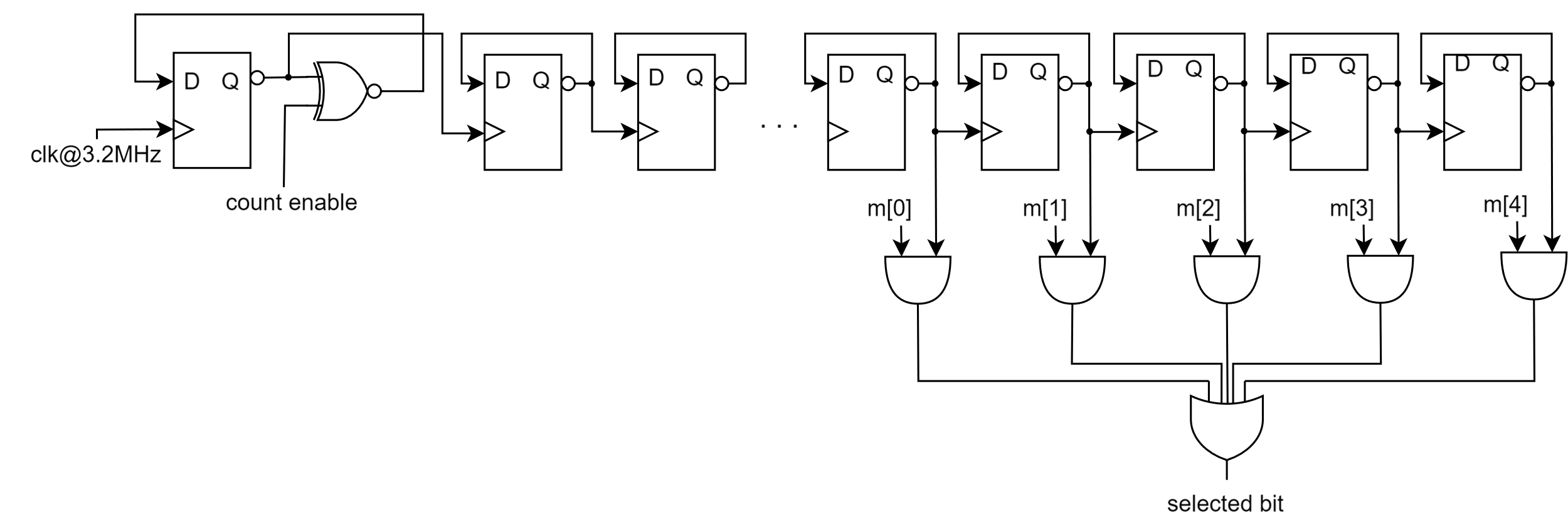
**L**as puertas XOR actúan como NOT controlada. De esta forma, si A[8] es igual a 1 se invierte todo A. En ese caso, y siempre que el valor de entrada sea diferente a -256, se suma uno al valor obtenido, habiendo completado la operación de valor absoluto para complemento a dos.

El output de este circuito es el valor absoluto de la entrada excepto cuando esta tiene el valor de -256. En este caso, la salida es 255. Esto se hace para poder mantener una salida de 9 bits, ya que para representar +256 sería necesario otro bit extra. Añadir este bit significaría tener que complicar el hardware de los comparadores, y no existe apenas ninguna diferencia a efectos prácticos entre utilizar 255 y 256 en este caso.

Los comparadores son simplemente restadores de 10 bits en complemento a dos que solamente producen como resultado el bit de signo. Es necesario realizar una extensión en signo de 1 bit, ya que el valor de la resta de dos números de 9 bits necesita 10 bits para ser representado.



Por último, el contador se realiza con la siguiente estructura:

La función “count enable” se implementa con una simple puerta XNOR a la entrada del primer flip-flop. Otra opción de implementación hubiera sido utilizar “clock gating” en el primer flip-flop, lo que resultaría más eficiente en términos de consumo. Sin embargo, podría ocasionar problemas en caso de que el flanco de reloj llegase antes que la señal de control a la entrada. Para evitar este problema, se ha decidido optar por la opción más segura, a pesar del aumento de consumo.

Por otro lado, la lógica de selección elige la salida del contador, asignando el bit 13,14,15,16 o 17 en función de la máscara.

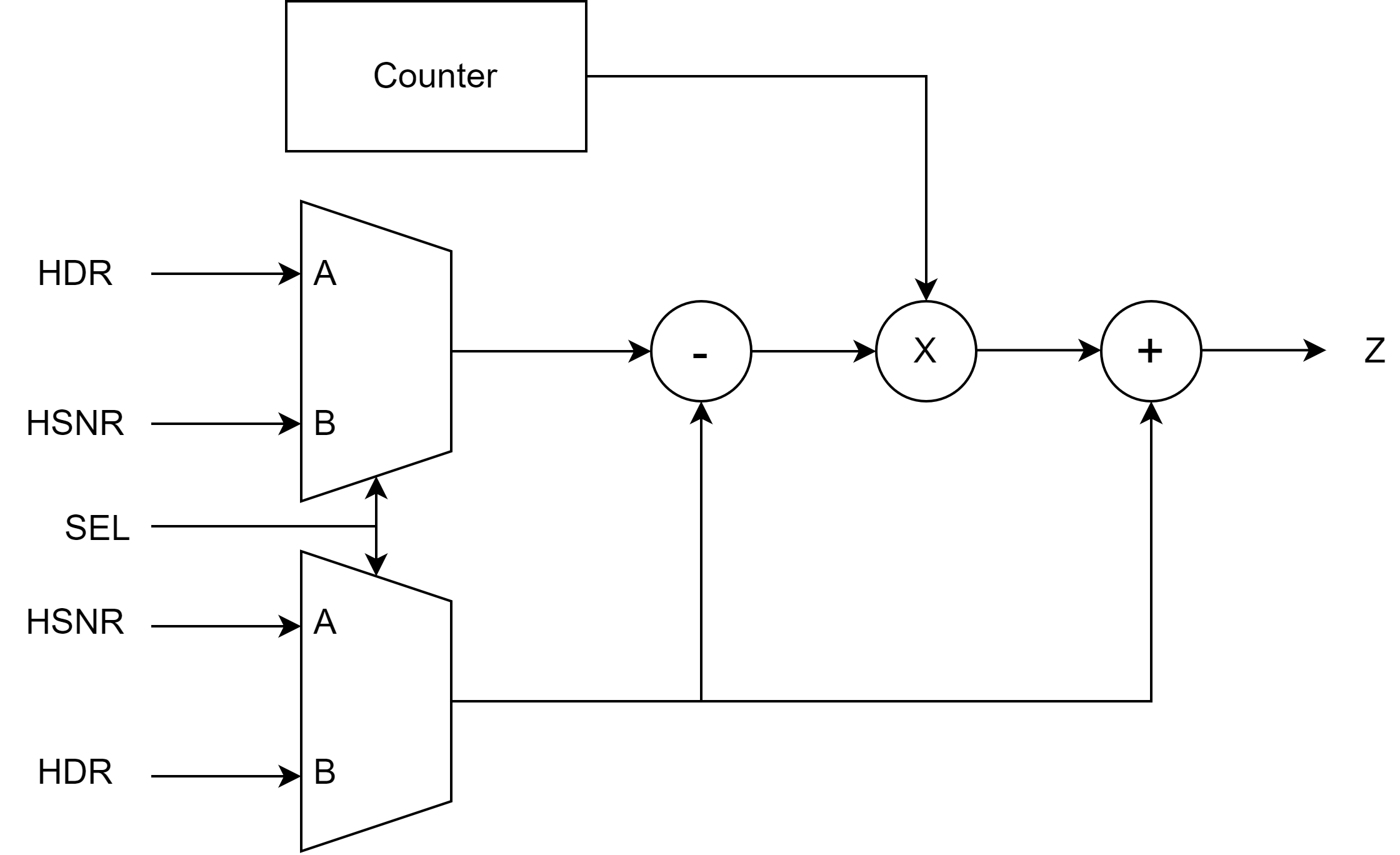
El reset no se muestra en el diagrama, pero es activo a nivel bajo y asíncrono. **3.5 Combinador de salida progresivo**

Para que el cambio del canal HDR al canal HSNR se produzca de forma progresiva, hay que hacer que Z(n) = α(n) \* HSNR + (1- α(n)) \* HDR, donde α(n) es una progresión que va desde 0 hasta 1 de forma más o menos lineal. Como el bloque produce un α de ‘1’ o ‘0’, es necesario desarrollar un circuito que construya una progresión cuando una entrada binaria cambia su valor. La respuesta inmediata es un contador de N bits, similar al anterior produciendo 2^N valores en la progresión y deteniendo el conteo cuando llega a su valor máximo. Si estos valores son interpretados como números decimales, se consigue una progresión desde 0 hasta 1.

Para simplificar la implementación del combinador, es posible reescribir Z(n) = α(n) \* HSNR + (1- α(n)) \* HDR como:

Z(n) = α(n) \* (HSNR - HDR) + HDR

De esta forma, una posible implementación es la siguiente:



SEL debe tomar el valor de 1 cuando se está realizando un cambio del canal HDR al canal HSNR, y 0 cuando sea al revés.

Si se quiere realizar una implementación más sencilla en hardware, es posible sustituir el multiplicador por un barrel shifter, utilizando potencias de dos generadas por un registro de desplazamiento como secuencia en lugar de un contador. En este caso, α toma los valores 0.125, 0.25, 0.5, 1 (para un registro de desplazamiento de 4 bits)

Se ha observado mediante simulación que la diferencia entre realizar un cambio progresivo entre los canales a utilizar un valor binario de α es prácticamente inexistente cuando las ganancias de los osciladores mantienen una relación correcta. El combinador progresivo no supondría pues una gran ventaja, pero consumiría energía y área de manera innecesaria, por lo que no se implementa en este chip. **3.6 Modelo bit-true completo - simulación del sistema bit-true**

Con el fin de servir como referencia para la implementación de la parte digital del chip en Cadence Virtuoso, se ha desarrollado un modelo a nivel de puerta lógica del mismo. Este modelo se encuentra adjunto bajo el nombre bit-true.slx .

Gracias a este modelo, se puede comprobar el comportamiento del chip, tanto SNR de los dos canales como el cambio de canal y el funcionamiento del bloque generador de α.

// Introducir figura con la simulación del modelo bit-true

**3.7 simulaciones a nivel de transistor del bloque   
implementado (consumos,simulacion temporal en cadence)**

Como objeto del TFM, además de participar en la elaboración de modelos a nivel de sistema, se ha desarrollado la implementación del bloque α en Cadence Virtuoso. Los circuitos implementados son prácticamente iguales a los expuestos en la parte de diseño hardware.

Se ha realizado una comprobación a nivel funcional que ha tenido el siguiente resultado:

// Figura con el resultado de la comprobación funcional

En ella se observa que el bloque α tiene un desempeño correcto a nivel funcional

Por otro lado, se ha analizado el consumo del bloque, resultando en XXXX, lo que representa un XXX% del total del consumo del chip

**4. Layout  
  
             4.1 Layout del chip completo**

Imagen de la pantalla de un video juego

Descripción generada automáticamente con confianza baja

El chip completo desarrollado del que forman parte los componentes de este trabajo de fin de master se muestran en la imagen superior.  **4.2 Layout del circuito objeto del TFM**

La implementación del bloque α se ha realizado mediante el uso de celdas estándar, pero, al no disponer de una herramienta de síntesis y place & route, este se ha realizado completamente a mano. Para implementar este layout se dispone de un paquete de celdas estándar y cuatro niveles de metal, siendo preferible utilizar los de nivel más bajo siempre que sea posible. El espacio que ocupa el layout del bloque α es de 106 x 16.28 um, resultando en un área bastante compacta para la cantidad de conexiones que se han tenido que realizar.

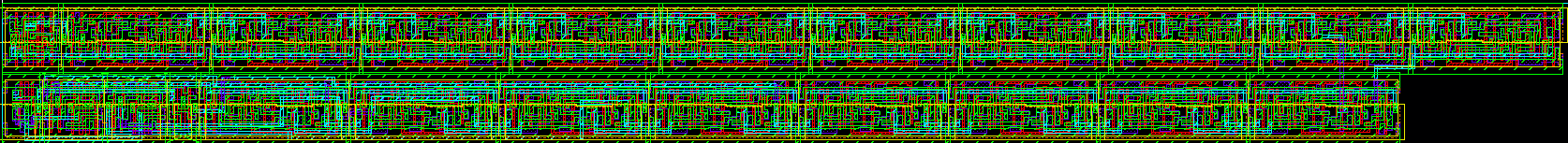
En los diagramas de implementación hardware puede verse que muchos circuitos tienen lazos (por ejemplo, el componente de valor absoluto debe calcular el primer carry a partir de todas las entradas). Estos lazos producen dificultades a la hora de hacer un layout compacto, lo que ha supuesto un reto y ha complicado la implementación del mismo.

En primer lugar, se ha tenido que desarrollar el layout de los comparadores. Este es posiblemente el mas sencillo de los cuatro componentes necesarios para el bloque α, que gracias a su estructura regular se ha podido diseñar con el siguiente patrón:

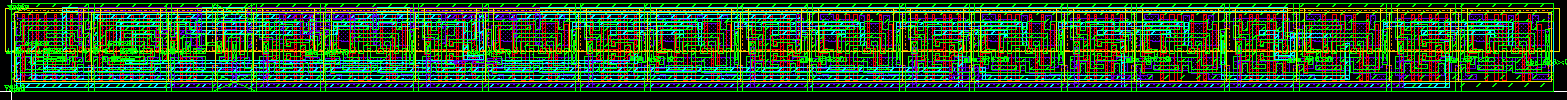
Un par de semáforos en un poste

Descripción generada automáticamente con confianza baja

A continuación se ha implementado el layout del contador. Este layout se ha dividido en dos líneas de celdas estándar para evitar que sea demasiado largo. El resultado mide 100 x 9 uM, que se encuentra dentro de las especificaciones:



Por último, se ha implementado el bloque de cálculo del valor absoluto, que es el que ha supuesto un reto mayor, a pesar de ocupar un área mas pequeña:



Finalmente, se ha realizado la interconexión de todos los componentes para disponer del bloque Alpha completamente implementado y funcional:

Texto

Descripción generada automáticamente con confianza media

Todas las imágenes de los layout se encuentran adjuntas en alta resolución.

**4.3 Verificación del bloque implementado y estimacion de   
potencias**

La verificación es un proceso fundamental en el desarrollo de circuitos integrados para aplicaciones específicas. En el caso de Cadence, se dispone del lenguaje de descripción hardware verilog-A, que utiliza una sintaxis muy similar a Verilog, pero puede ser utilizado para generar señales analógicas.Para comprobar el correcto funcionamiento del bloque α, se ha desarrollado un test para cada componente, así como un test de funcionamiento general del bloque.

Para el caso del valor absoluto, simplemente se ha realizado una prueba con todos los valores posibles de entrada. En el caso de los comparadores, se han generado dos señales de conteo a diferente frecuencia, de forma que una vez desincronizadas, se ha podido observar el comportamiento del circuito.

// Introducir test del valor absoluto

// Verificacion del contador (No me acuerdo bien)

La verificación del comportamiento completo del bloque α completo es más compleja. A diferencia de Verilog, Verilog-A no permite generar señales digitales, y tampoco la sintaxis #n para especificar un delay de n segundos en una señal. Por tanto, ha sido necesario desarrollar un script en Verilog-A que permita leer señales especificadas en un archivo. Dicho archivo contiene un numero de N bits en cada línea, seguidos por un delay especificado en segundos. El script de Verilog-A lee línea por línea el archivo especificado, y genera un bus de N bits de salida con la palabra especificada. De esta forma, es posible generar señales digitales de una manera mucho más conveniente y sencilla. Un ejemplo del archivo de datos a partir del cual el script podría generar una señal digital es el siguiente:

Imagen en blanco y negro de un teclado

Descripción generada automáticamente con confianza media

El archivo se encuentra adjunto con el nombre de va\_signals\_from\_file. Para facilitar aún más el proceso, se ha desarrollado un script en Python que convierte un archivo similar al anterior pero con números en base 10 en el equivalente que puede leer el archivo de Verilog-A, añadiendo delays por defecto si no se especifican. Otra versión del script de Verilog-A permite ignorar los delays y disparar un cambio con una señal de reloj externa. Este script se encuentra adjunto bajo el nombre va\_signals\_from\_file\_clocked.

Disponiendo de estas herramientas, se ha podido realizar una verificación del bloque α. Para ello, se generan señales de entrada por arriba y por debajo de los valores umbral, lo que permite observar el comportamiento del contador. También se prueban todos los valores de máscara para observar el punto de parada del contador, así como el reset. El resultado (satisfactorio) de esta simulación es el siguiente:

Pantalla de computadora con letras

Descripción generada automáticamente con confianza media

En la simulación se aprecia el cambio del valor de Alpha y los momentos en los que el contador esta contando y los que no frente a todos los valores de la máscara de selección y diferentes valores de entrada. Con esta información puede comprobarse que el desempeño funcional es correcto.

Aunque simulaciones a nivel de comportamiento son necesarias, es necesario realizar también simulaciones post-layout. Estas simulaciones, además, permiten estimar el consumo del bloque implementado.

En primer lugar, se ha comprobado que el resultado de la simulación post-layout es idéntica a la de nivel de comportamiento (transistor). Si bien los retardos aumentan ligeramente, estos se encuentran muy lejos de suponer un problema.

La potencia varía mucho en este circuito según la señal de entrada, ya que el contador es el elemento que más consume, y si la señal de entrada no varía mucho en amplitud su consumo es mínimo. Sin embargo, para una señal de entrada que este constantemente saltando del canal HDR al HSNR, el consumo será mayor.

Como estimación, la potencia utilizada en la simulación de la figura XXXX es []. Este consumo puede entenderse como una cota superior, ya que el contador se encuentra prácticamente siempre activo.

**5.Conclusiones  
  
6. Planificacion del trabajo y presupuesto (? Ver que hay que poner de esto)  
  
7. Bibliografia**