**1. Introduccion y estado del arte  
  
     1.1    Procesamiento en el Borde de señales de audio  
  
     1.2    Parametros de audio de un microfono inteligente  
  
     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

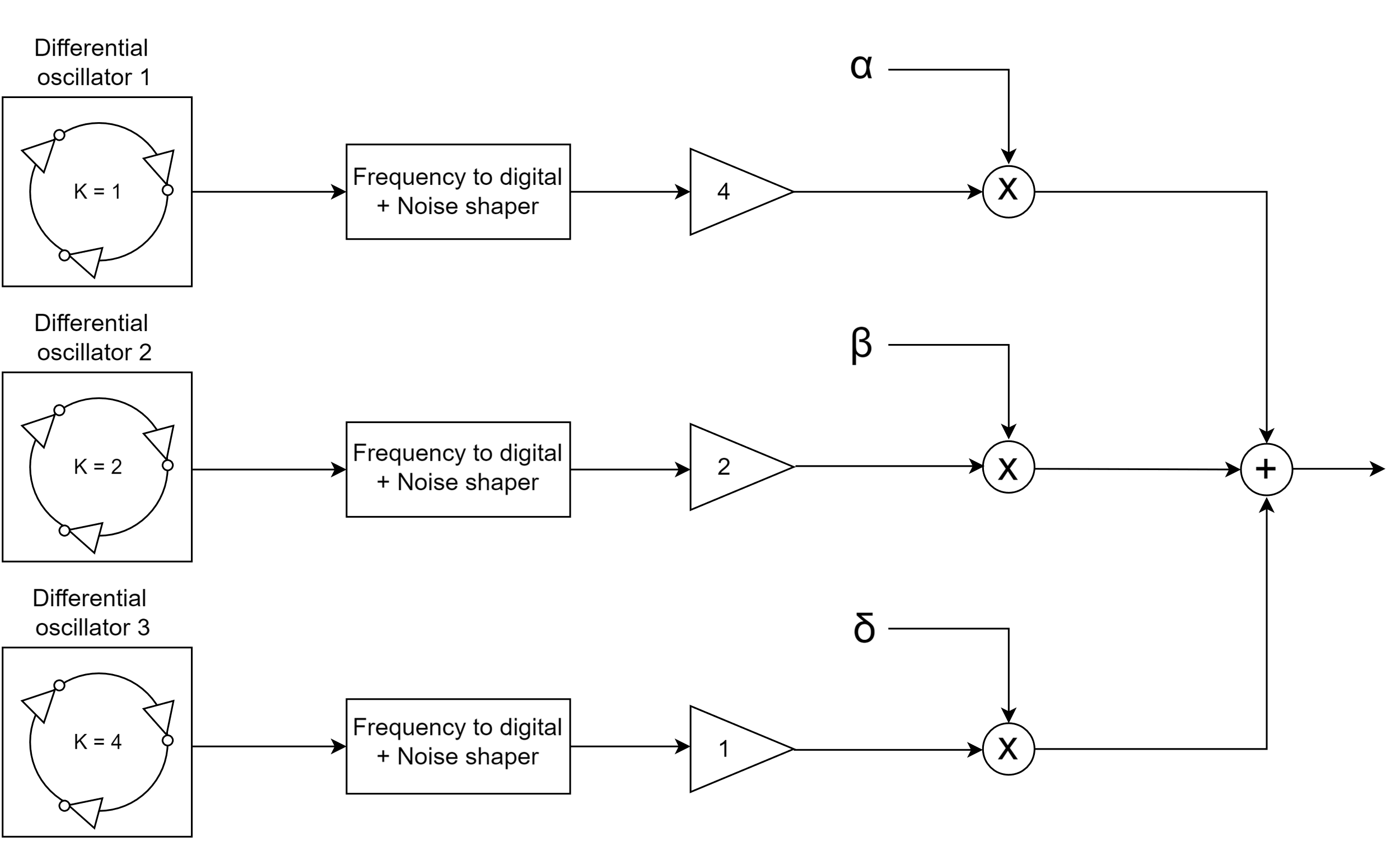
Si asumimos que I es una variable que depende del tiempo, entones

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 esta representado con 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 Combinacion 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.

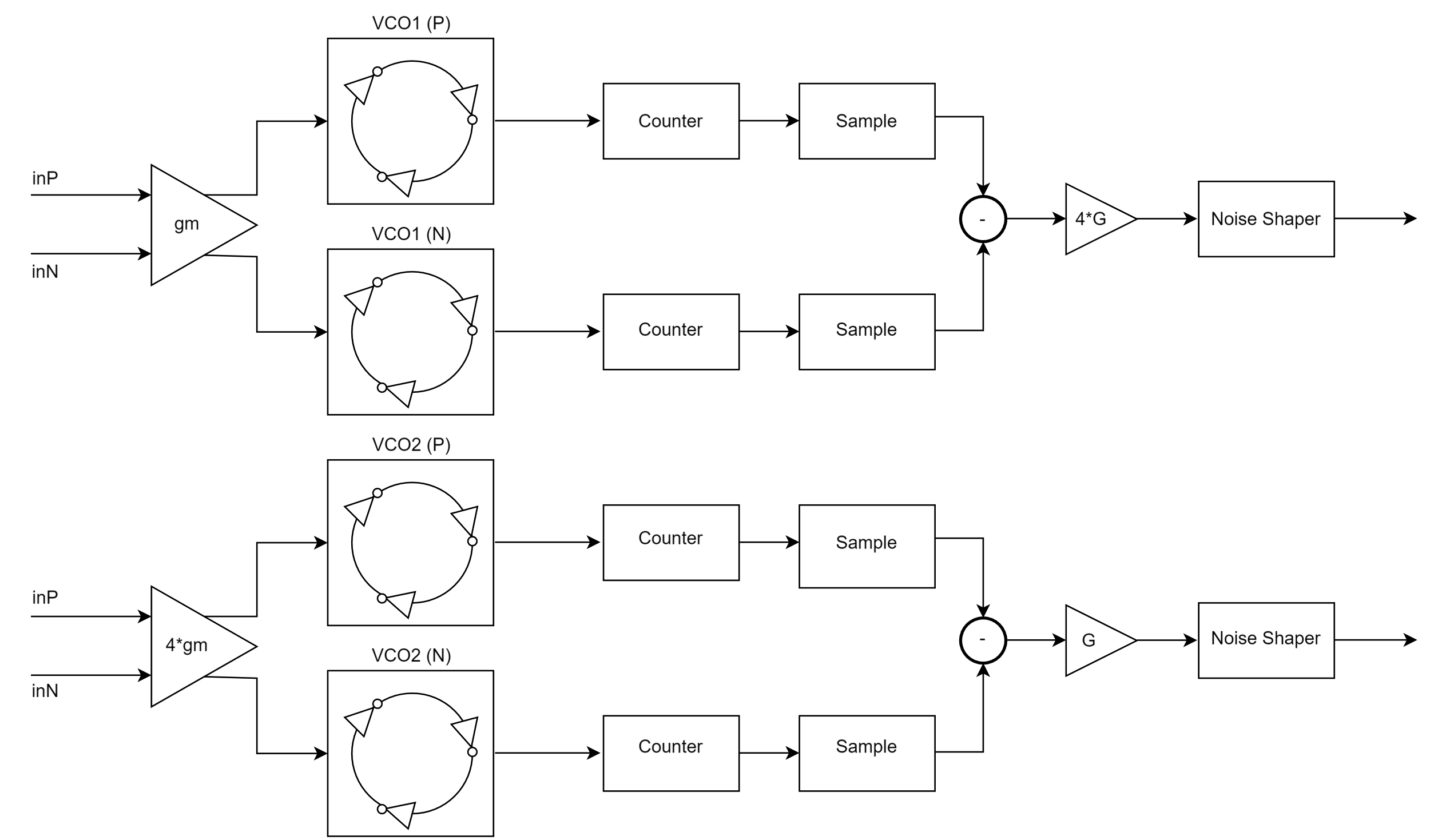
Una red neuronal simple con tres entradas, una para el valor de cada oscilador, y tres salidas utilizando la función softmax en la capa de salida podría realizar esta labor convenientemente. Se cree que la red neuronal no necesitaría una gran cantidad de capas internas, al tratarse de un problema relativamente 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 mas 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 Combinacion de caminos con estimadores estadisticos 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 adquisicion de audio   
con extension 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 entender mejor el funcionamiento de este sistema, se recomienda la lectura de [paper colorines], que explica a nivel de sistema el funcionamiento de los VCO-ADCs.

Tras una simulación funcional en Matlab 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 mas ruido, pero ambas representan una señal con la misma amplitud, habiendo conseguido el propósito funcional 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 Opcion 1: Optimizacion del rango dinamico por comparadores   
inmplementados 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 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 activadas mediante ReLU y una capa de salida que utiliza la funcion Softmax, produciendo los valores α y 1 – α.

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.

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

Gráfico, Gráfico de líneas

Descripción generada automáticamenteTras entrenar la red neuronal tratando de minimizar el error medio al cuadrado entre una señal de entrada de tipo rampa senoidal ascendente, se obtiene el siguiente resultado, para una señal de entrada no utilizada en el entrenamiento:

Gráfico

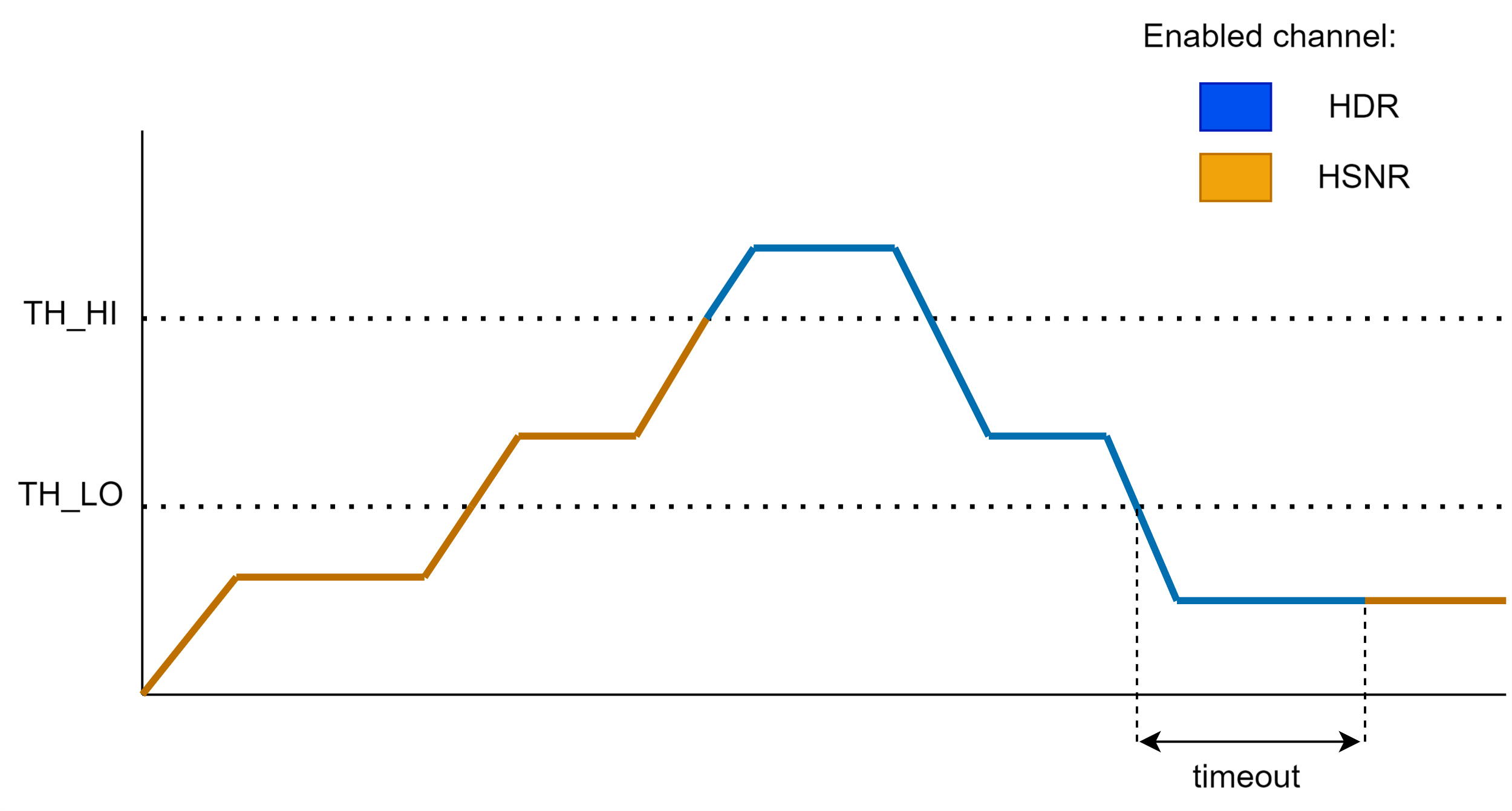
Descripción generada automáticamente

nota: el canal HSNR tiene una escala diferente.

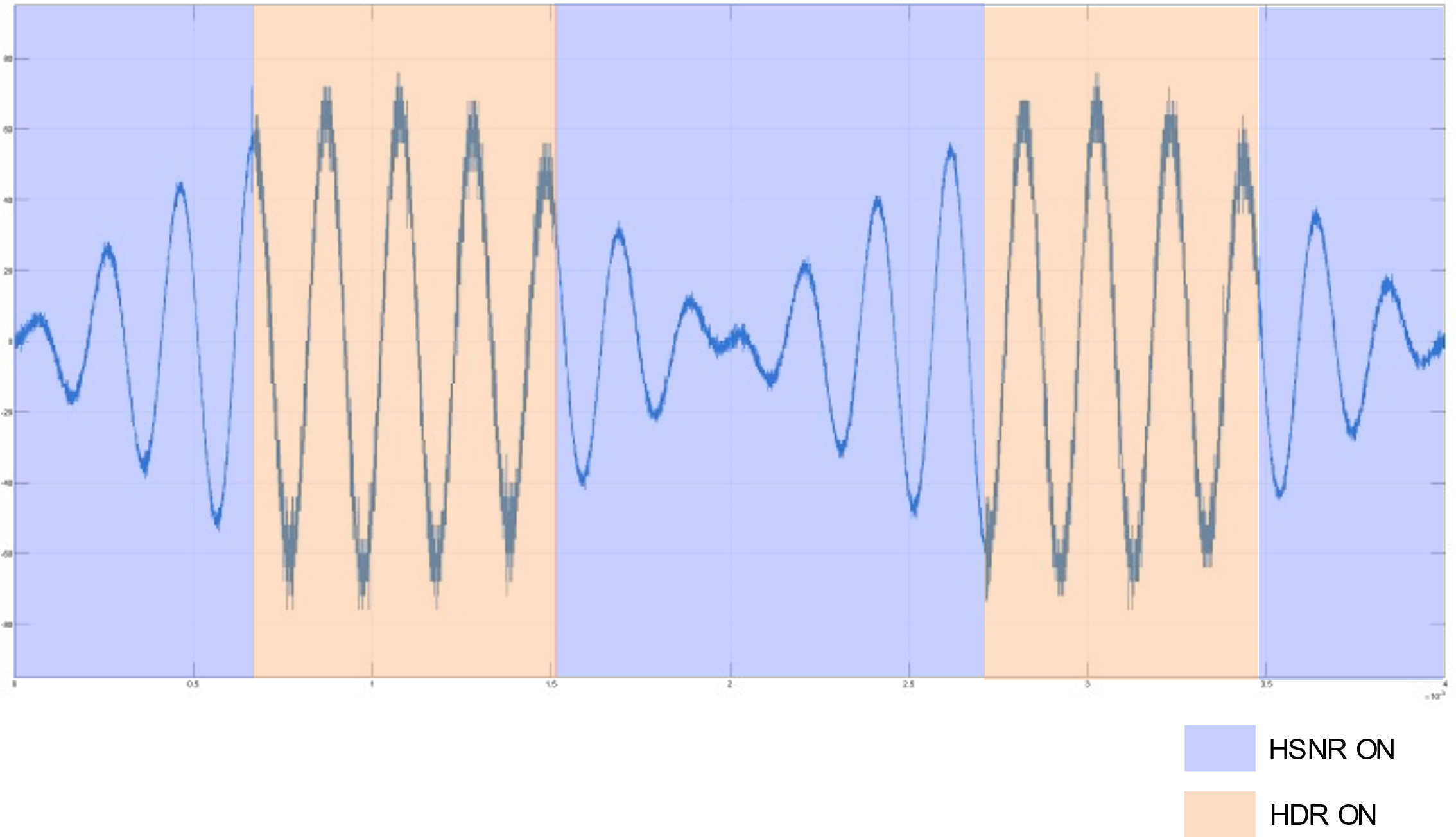
Como se observa en la figura, la red ha aprendido a generar los pesos correctamente. Sin embargo, la propia red neuronal introduce ruido al variar los pesos con cierta aleatoriedad para valores cercanos, lo cual puede suponer un problema para una implementación real. Esto, posiblemente, se puede solucionar 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 “threshold” alrededor de ese valor, y que un cambio de canal ocurre muy rápidamente una vez se atraviesa.

La red neuronal, a pesar de su aparentemente 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 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   
  
             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. La salida de estos transconductores se conectan a un oscilador en anillo diferencial de tipo “feed forward” de 16 etapas [ Introducir diagrama de oscilador en anillo diferencial]. Este oscilador se utiliza porque permite un número de fases par, algo muy importante para la implementación del contador que se utiliza para la digitalización de la fase.

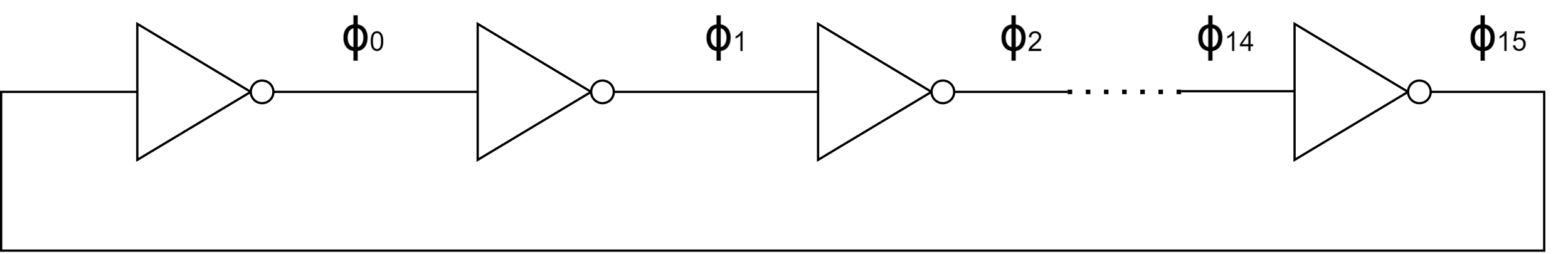
**3.2 Contador**

La fase del VCO se 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, todos los bits muestreados tendrían un valor indefinido. Utilizando código Gray, por la propiedad anteriormente mencionada, solamente se puede cometerse un error de +-1. 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



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 entender el porqué de estas operaciones, podemos 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.

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.

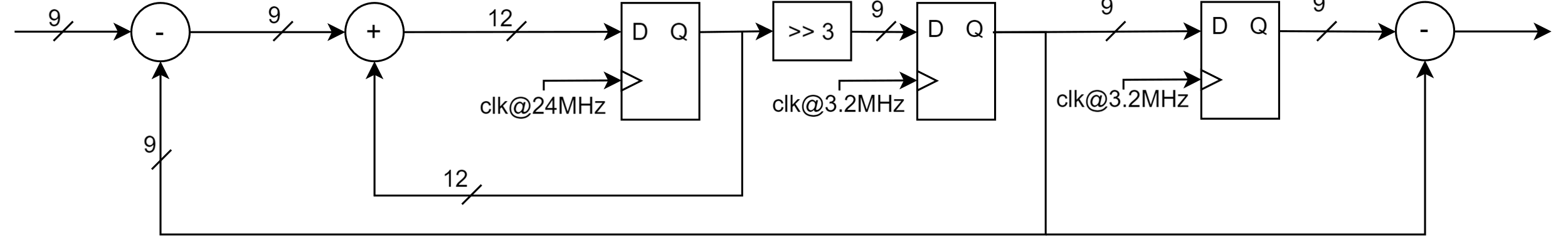
Es posible utilizar esta propiedad para generar cualquier secuencia binaria a partir de un oscilador en anillo. Para hacerlo, es necesario realizar una operación del tipo ϕa ⊕ ϕb … ⊕ ϕn, donde a,b .. n son las fases que corresponden con un cambio de estado en la secuencia binaria a generar. En el caso del código gray, aparecen marcadas en verde en la siguiente tabla.



Una vez realizado este proceso, se dispone de la fase global del oscilador en anillo digitalizada, habiendo realizado la primera parte de la conversión analógico-digital. No obstante, falta por definir la frecuencia del reloj de muestreo. Esta frecuencia debe ser tal que el contador gray de 4 bits no pueda darse la vuelta entre dos muestras, pues esto induciría a tratar dos datos diferentes como iguales, produciendo un error.

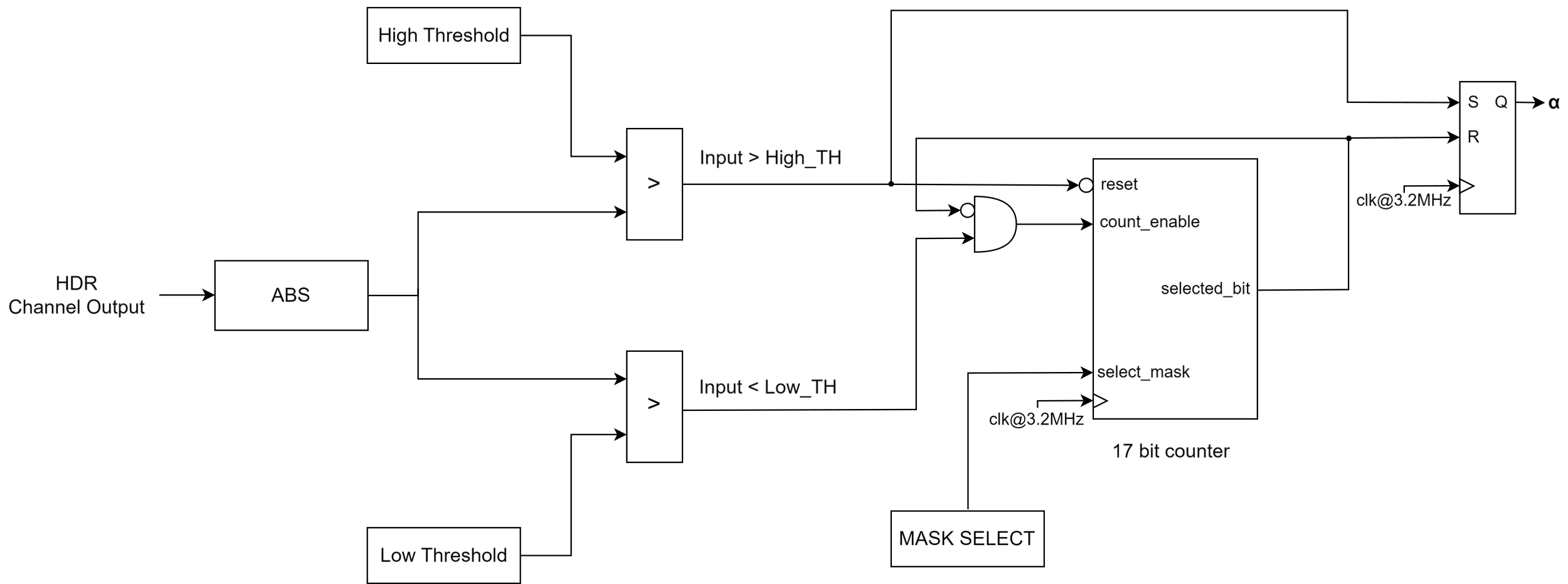
La frecuencia de muestreo debe ser, por tanto, como mínimo la frecuencia del oscilador en anillo. Sin embargo, por razones que veremos más adelante, en este proyecto se utiliza el doble de la frecuencia mínima. **3.3 Noise shaper**

La implementación del noise-shaper se realiza de la siguiente manera:

****

Para evitar que el acumulador interno sufra overflows, es necesario que tenga 3 bits más que los circuitos que funcionan a 3Mhz. (3 \* 8 = 24). La implementación del noise-shaper es prácticamente igual que su visión a nivel de sistema **3.4 Comparador de umbral y selector de camino**

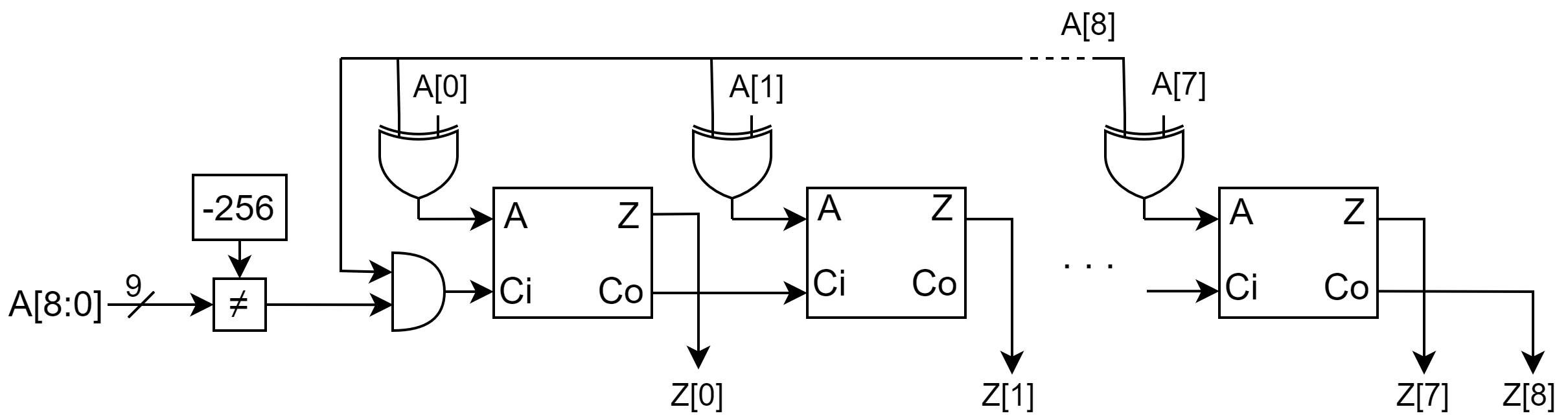
Para conseguir el comportamiento descrito en el apartado XXX, se propone el siguiente circuito:

****

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.

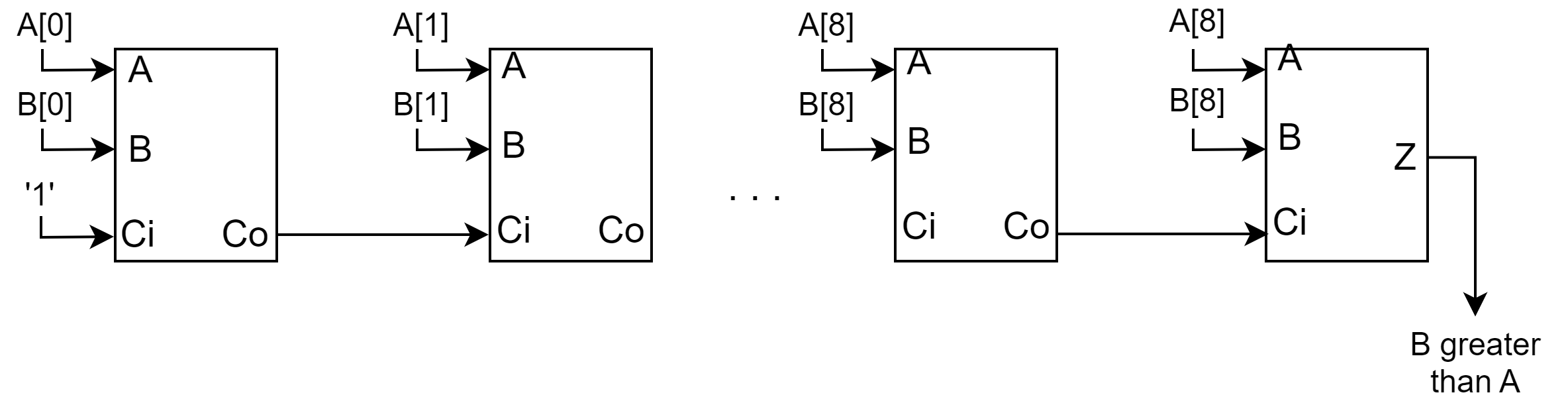
El bloque para calcular el valor absoluto sigue la siguiente estructura:

****

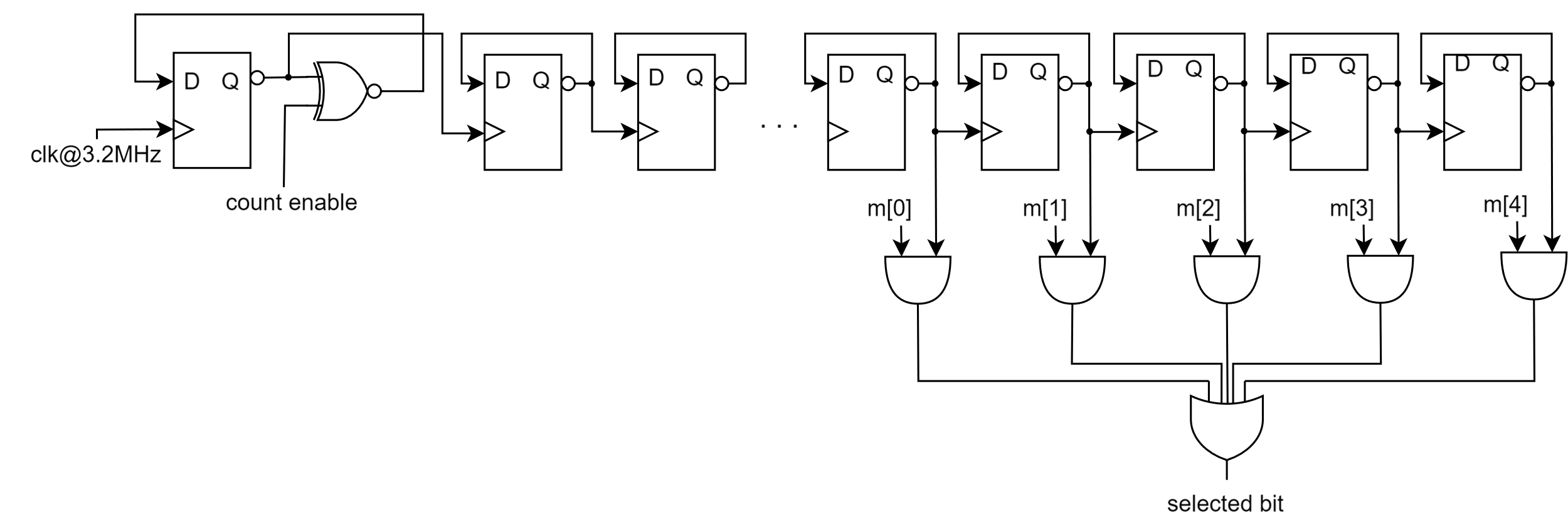
**L**as puertas XOR actúan como NOT controlada. De esta forma, si A[8] es igual a 1 se invierte todo A. En ese caso, y siempre que el valor de entrada sea diferente a -256, se suma uno al valor obtenido, habiendo completado la operación de valor absoluto para complemento a dos.

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.

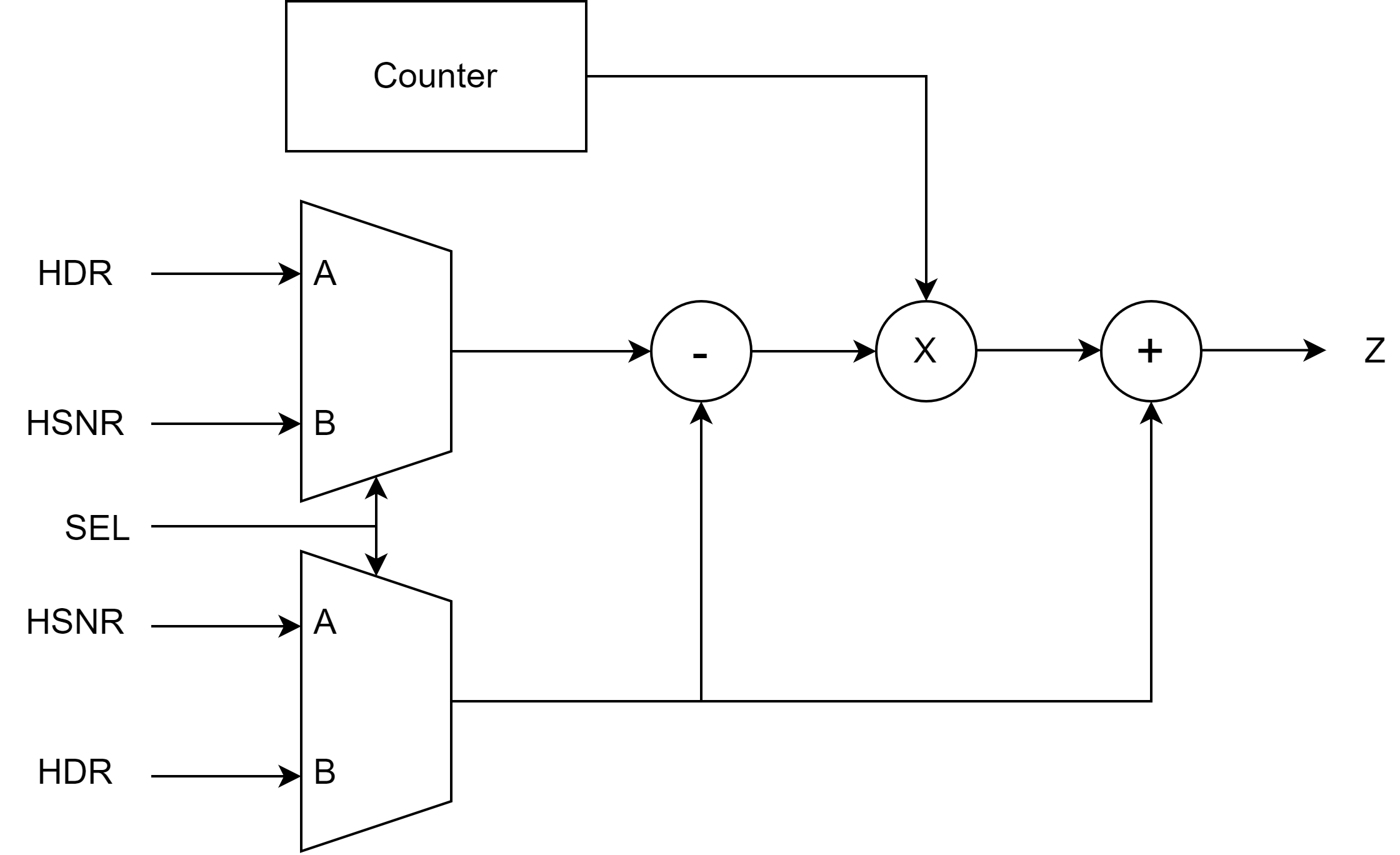
El reset no se muestra en el diagrama, pero es activo a nivel bajo y asíncrono. **3.5 Combinador de salida progresivo**

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 contador produce un α binario, 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 al máximo valor. 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 que 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.

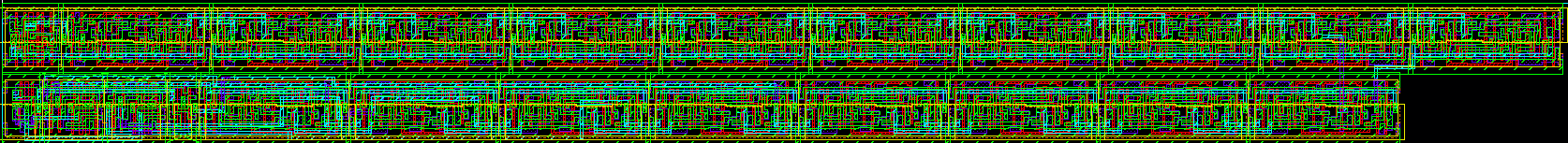
En los diagramas de implementación hardware puede verse que muchos circuitos tienen lazos (por ejemplo, el componente de valor absoluto debe calcular el primer carry a partir de todas las entradas). Estos lazos producen dificultades a la hora de hacer un layout compacto, lo que ha supuesto un reto y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

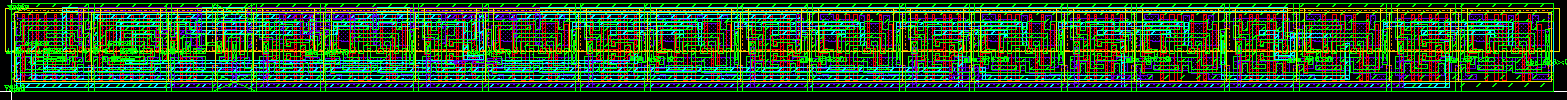
Un par de semáforos en un poste

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**