1 – Introduccion y estado del arte

* 1. – Parametros de audio de un microfono inteligente
  2. - Diseño de un micrófono con extensión de rango dinámico

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

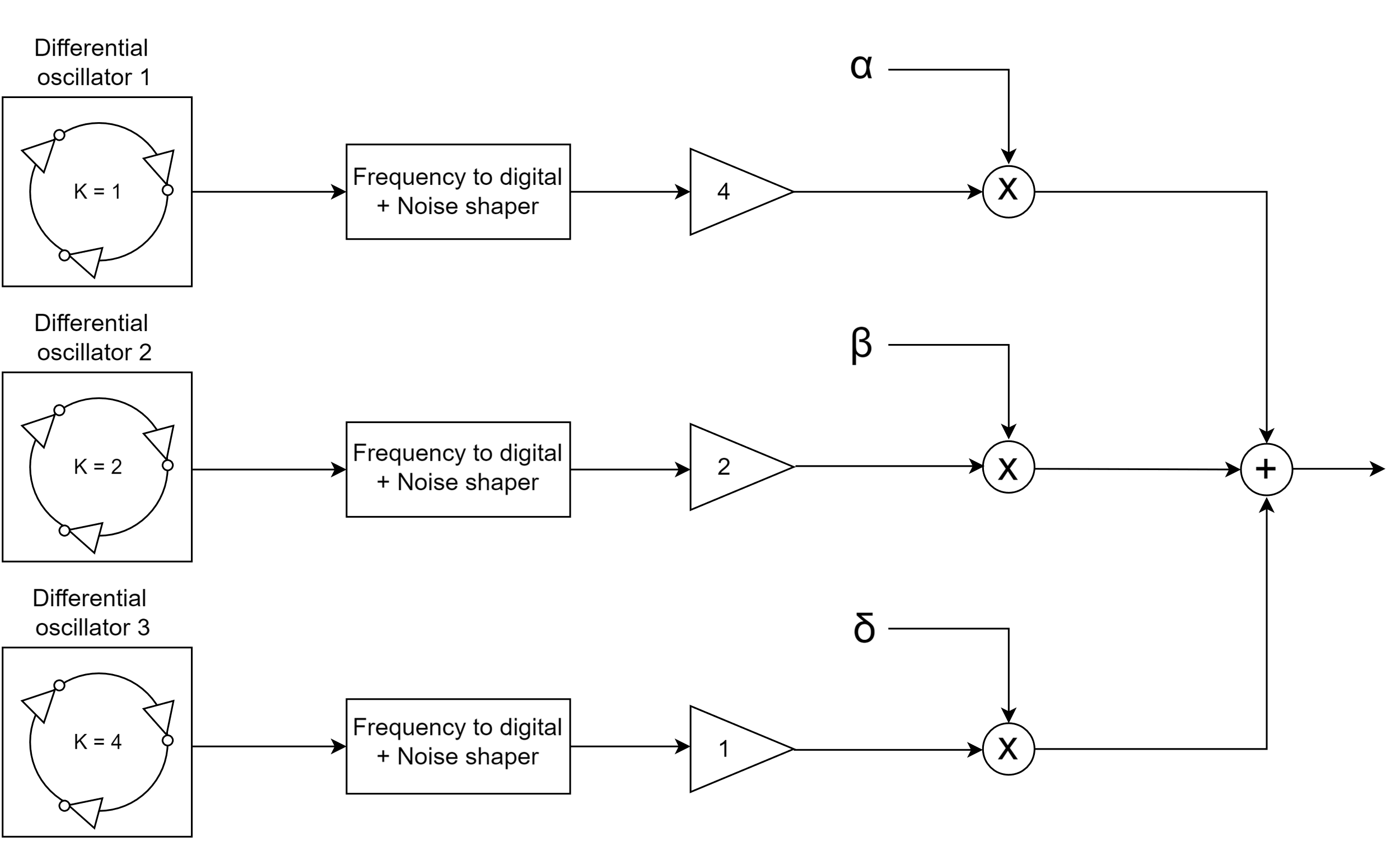
Si asumimos que I es una variable que depende del tiempo, entones

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:

Antes del ajuste:

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Utilizando la aproximación lineal para cada una de las funciones:

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 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.

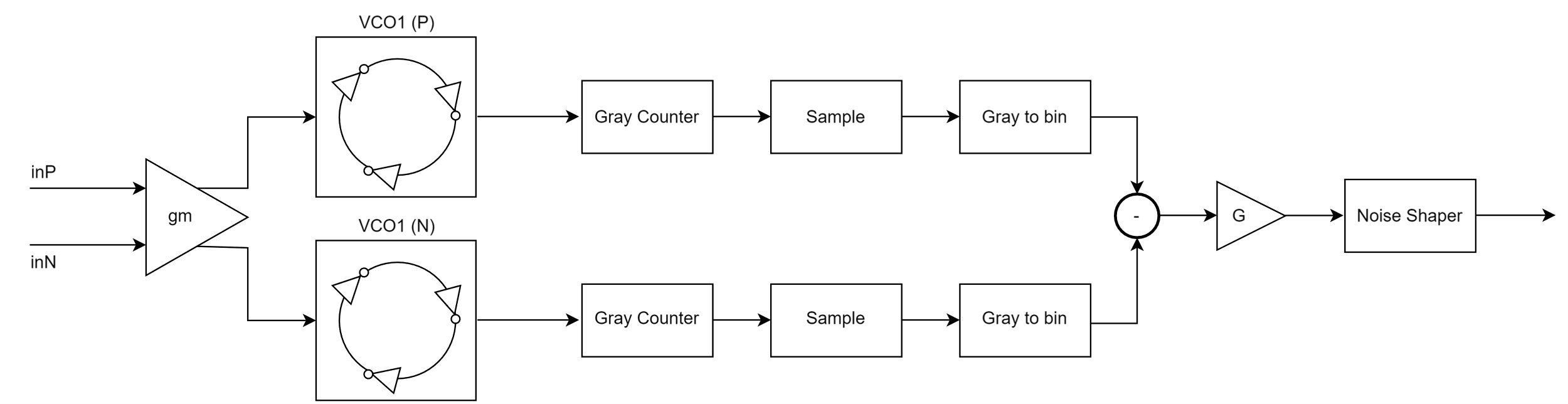
1.3.3 Combinación de caminos mediante estimadores estadísticos de potencia

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.

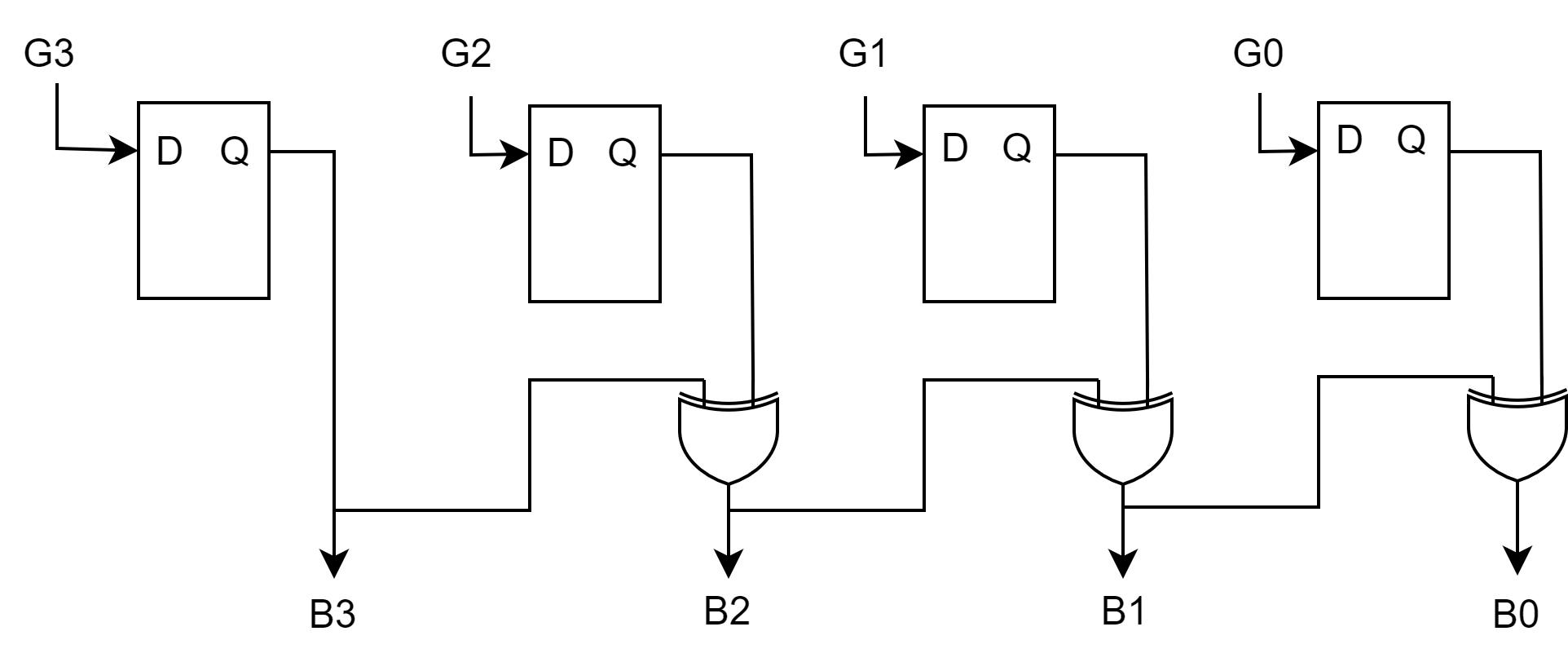
2. Diseño del sistema implementado

El diseño implementado utiliza dos caminos con diferente rango dinámico. Uno de los caminos presenta un oscilador con ganancia K, y el otro con ganancia K2 = 4K.

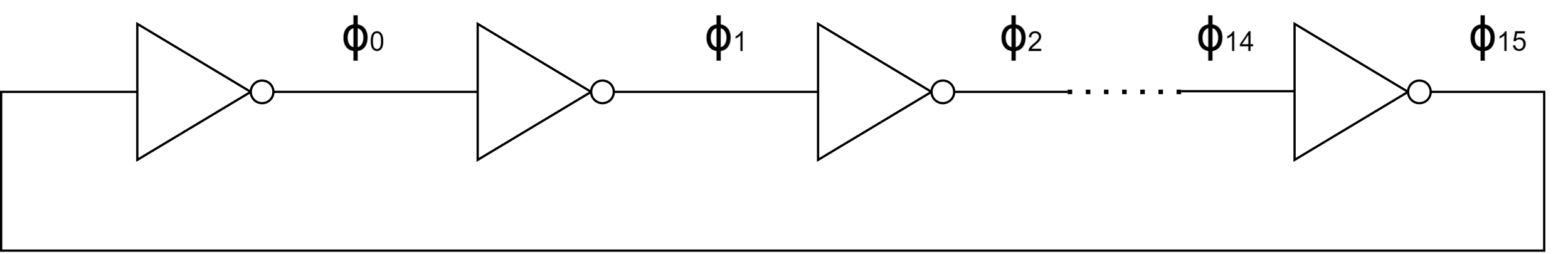
La siguiente sección contiene información detallada de como esta implementado cada uno de los caminos, que son idénticos a excepción de las ganancias de los osciladores y ajustes de corrección de ganancia digitales.



El camino comienza con un transconductor diferencial que transforma un voltaje de entrada procedente de un micrófono MEMS en una corriente de control para los VCO, que consisten en un oscilador en anillo de 16 etapas. La fase del VCO se digitaliza a continuación en 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 muestree 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, como el mostrado en la siguiente figura, para disfrutar de ambas propiedades.



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.

---------------------------------------------------------------------------------------------------

Después de digitalizar la fase del oscilador, es necesario extender el número de bits disponible, por razones que se explicarán después. Para ello, se hace uso de un contador binario asíncrono clásico. (introducir figura graytobin+ contador asíncrono).

Es preferible utilizar un contador asíncrono a uno síncrono por dos motivos. En primer lugar, el área que ocupa un contador asíncrono es notablemente menor al de uno síncrono. Pero, principalmente, el consumo energético de un contador asíncrono es mucho menor al de un contador síncrono. La ventaja de utilizar contadores síncronos es su velocidad, innecesaria en el caso de este proyecto. Todos los contadores del chip implementado son, por tanto, asíncronos.

-------------------------------------------------------------------------------------------------------

Parte 2: Noise shaper

Como se establece en el paper “colorines” [referencia], simplemente con realizar la primera diferencia de dos frases muestreadas consecutivas, se obtiene una señal de salida con noise-shaping de primer orden, ya que:

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

Integral(F(t) dt) = Integral(K \* I(t) dt)

Φ(t) = Integral(K \* I(t) dt)

Muestreo:

Φ(n \* Ts) = sum(K \* I(n \* Ts)) + e(n \* Ts)

Φ(n-1 \* Ts) = sum(K \* I(n-1) \* Ts) + e(n-1 \* Ts)

Φ(n \* Ts) - Φ(n-1 \* Ts) = K\*(sum( I(n \* Ts)) \* Ts – sum(I(n-1 \* Ts)) \* Ts) + e(n \* Ts) - e(n-1 \* Ts)

Φ(n) - Φ(n-1) = K \* Ts \* I(n \* Ts) + e(n \* Ts) - e(n-1 \* Ts)

// Establecer valores del fondo de escala, Ts y K para que todo cuadre.

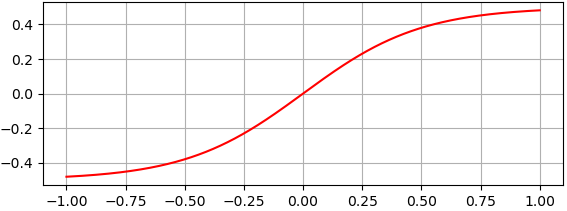
Para mejorar la SNR del convertidor, se propone no solamente hacer este procedimiento, sino incluir también un elemento noise-shaper.

// Explicar el noise shaper

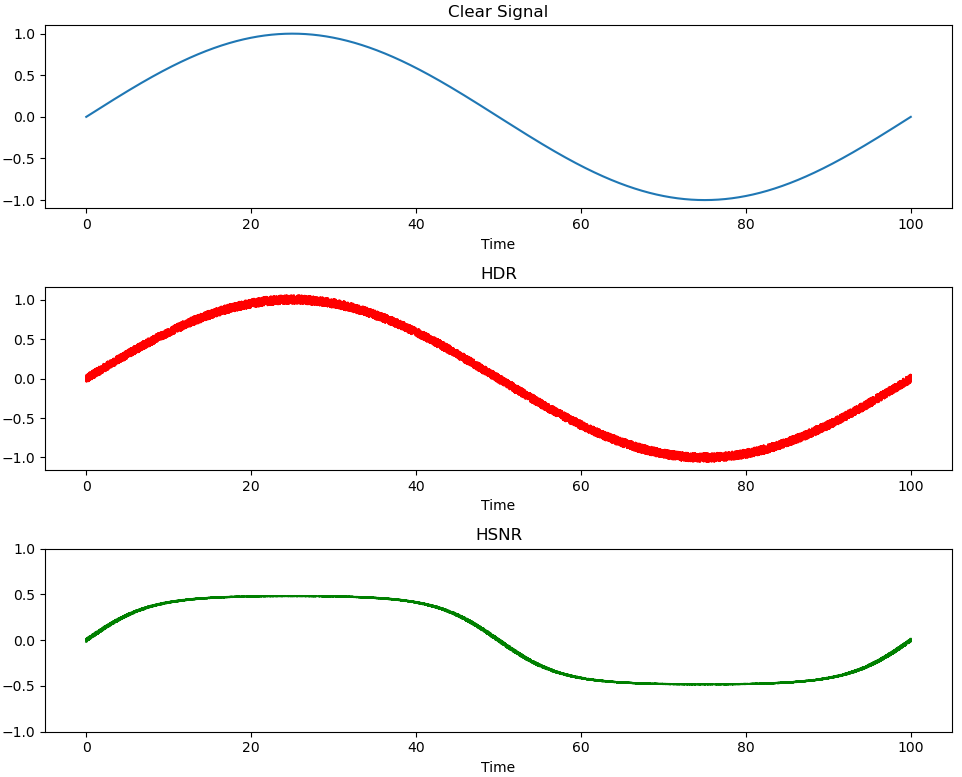
1.3.2 Selección de caminos mediante redes neuronales

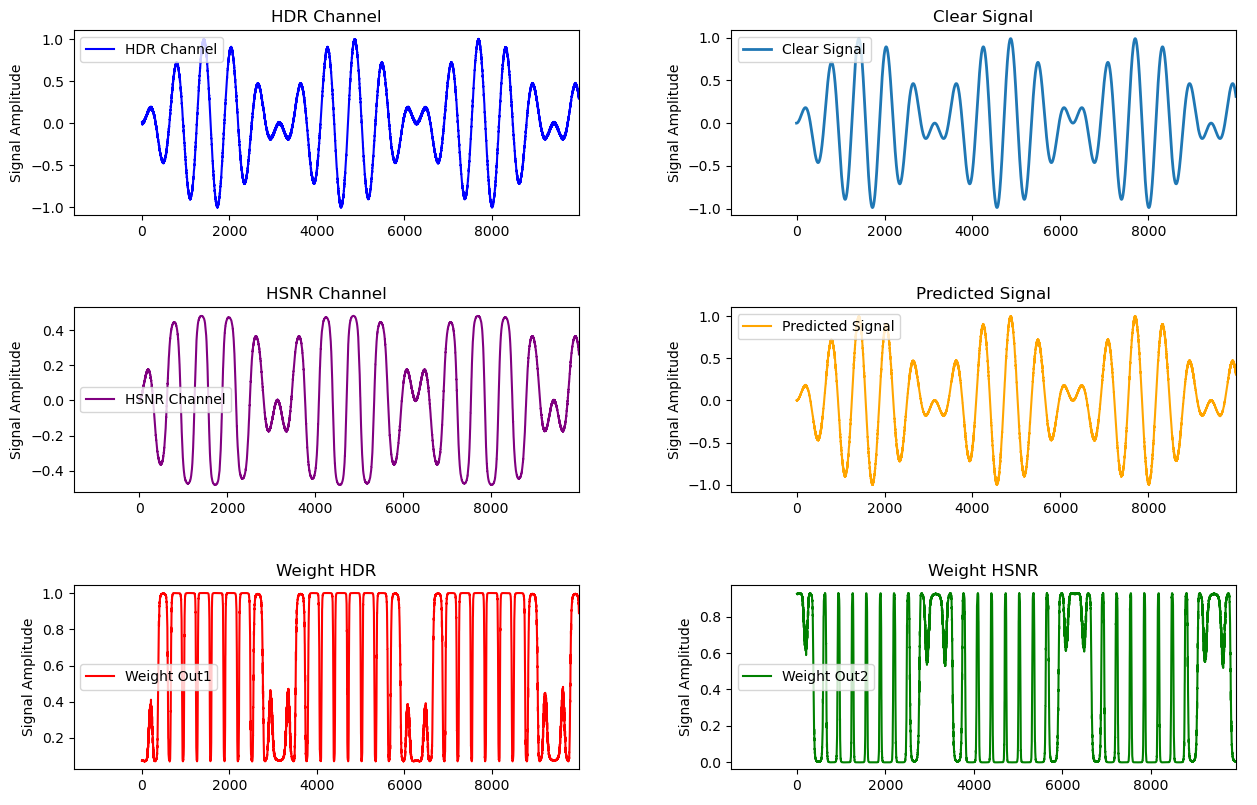
Con el fin de estudiar el uso de redes neuronales para la selección de caminos, se ha desarrollado un programa en Python utilizando Keras y Tensorflow que entrena una red neuronal para asignar un peso a los dos caminos. La red neuronal diseñada tiene dos entradas, correspondientes al valor en tiempo 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:

 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:

Tras 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:



nota: el canal HSNR tiene una escala diferente.

Como se observa en la fugura, 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 más deseable ponderar más el canal HDR frente al HSNR, pues es donde la distorsión comienza a ser mas 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.