Mi nombre es Sara García Sánchez con nº de colegiado en trámites y solicito que el proyecto fin de carrera **Diseño y Simulación de los Algoritmos de Recepción para WLAN Basada en OFDM**, sea admitido como candidato al **Premio NOKIA**, adjuntando a continuación, los datos del participante y del proyecto así como un resumen del mismo,

DATOS DEL PARTICIPANTE

- Nombre: **Sara**
- Apellidos: **García Sánchez**

DATOS DEL PROYECTO

- Nombre y Apellidos del tutor: **Dra. Ana García Armada**
- Departamento: **Departamento de Teoría de la Señal y Comunicaciones**
- Fecha de lectura: **23-Enero-2004**
- Calificación: **Matrícula de Honor**

RESUMEN

1. INTRODUCCIÓN

Este proyecto surgió del deseo de probar en un escenario real, como es la Plataforma PRAGA, el receptor de una WLAN basada en OFDM. Por ello, a lo largo de este trabajo se van a desarrollar, implementar y simular técnicas imprescindibles en un receptor WLAN basado en el estándar IEEE 802.11a con el objetivo de lograr un compromiso entre complejidad y prestaciones.

En concreto, este proyecto se enmarca en el paso intermedio entre la programación en Matlab de las funciones necesarias para representar el funcionamiento de la WLAN, y el desarrollo en bajo nivel, realizado en Verilog como paso previo a implementar dichas funciones en una FPGA.

En este paso intermedio se intenta introducir las condiciones reales que existen en la implementación física y que no han sido tenidas en cuenta en el desarrollo en Matlab, especialmente la representación de las señales con un número finito de bits. Para ello se empleará la herramienta de simulación llamada Simulink. Además, por motivos de eficiencia, se intentará aprovechar al máximo los códigos de partida en Matlab, adecuándolos a la estrategia de simulación que sigue Simulink.

Una vez que se tenga realizado el diseño del receptor de la WLAN, así como otros elementos necesarios para su funcionamiento (transmisor, canal, etc...) se procederá a evaluar una serie de técnicas de sincronismo, tanto en tiempo como en frecuencia, y de estimación y corrección de canal, previamente estudiadas, con el objetivo de intentar mejorar las prestaciones que ofrece el sistema. Previamente, la red habrá de ser validada frente a la ya diseñada en Matlab y cuyo correcto funcionamiento ya está comprobado.

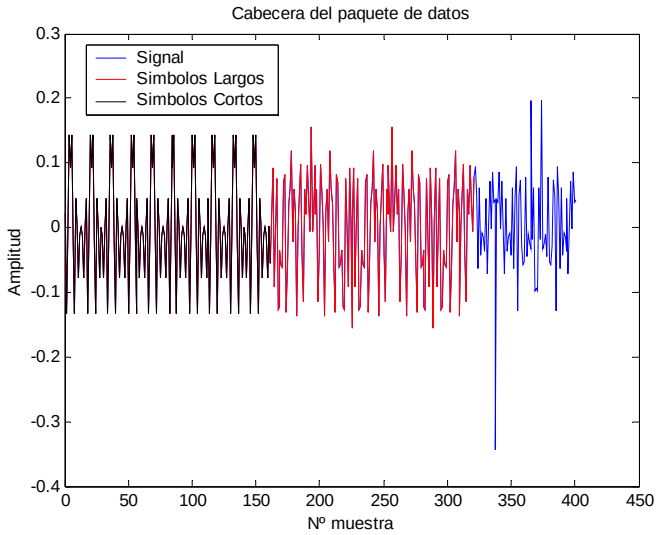
Por último comentar que, con el diseño del receptor WLAN en Simulink, se busca una realimentación entre la programación de alto nivel de Matlab y la de bajo nivel de Verilog que permita simular de manera más real el funcionamiento del sistema para obtener unos resultados mas fieles con la realidad.

2. TÉCNICAS IMPLEMENTADAS Y EVALUADAS

A continuación se desarrollarán teóricamente las técnicas que se han implementado en el receptor de la WLAN para, posteriormente, evaluarlas. Concretamente se estudiarán las técnicas que permitan:

- Sincronizarse en tiempo y frecuencia con la trama OFDM.
- Estimar y corregir el canal.

Pero antes de entrar en la explicación de dichas técnicas se introducirá brevemente el formato de la trama y de la cabecera empleada. En la siguiente figura se muestran las distintas partes para ir familiarizándose con ella.



Cabecera de una trama OFDM.

Los símbolos cortos se emplearán para el sincronismo temporal con la trama y la estimación gruesa en frecuencia. Posteriormente con los largos se estimará el canal, aunque ya se verá que es posible que se les dé un nuevo uso a los símbolos largos. También recordar que los símbolos cortos son 10 repeticiones de 16 muestras y los símbolos largos son 2 símbolos iguales de 64 muestras cada uno con un prefijo cíclico de 16 muestras cada uno. Al ser símbolos especiales, los dos prefijos cíclicos se juntan al comienzo teniendo 32 muestras de prefijo cíclico y luego las 128 muestras correspondientes a los dos símbolos juntos.

2.1 TÉCNICAS DE SINCRONISMO

En un sistema OFDM, como el que aborda este proyecto, la sincronización en el receptor es un paso a diseñar muy importante porque los sucesivos bloques y sus prestaciones dependerán de lo bien o mal que se haya sincronizado el sistema tanto en tiempo como en frecuencia.

El receptor está continuamente escuchando y debe determinar dónde comienza el paquete y cual es el instante óptimo de tiempo en el que se minimizan los efectos de las interferencias entre símbolos (ISI) y entre portadoras (ICI). Esto es la sincronización en tiempo. Además debe estimar lo más precisamente posible la desviación en frecuencia que ha causado el canal, porque esto provocaría la pérdida de ortogonalidad entre las portadoras y por lo tanto la interferencia entre ellas (ICI).

En este proyecto, las técnicas de sincronismo (temporal y frecuencial) elegidas para ser evaluadas e implantadas se corresponden con la técnica PSA (Pilot Symbol Aided). Se debe intentar que los mecanismos sean lo más eficientes posibles para que no sea necesario insertar demasiadas muestras de cabecera lo que provocaría una pérdida en la tasa de datos. Por otro lado el rango de adquisición, que quedará probado en los resultados obtenidos, es mayor que en las técnicas del primer grupo.

Estas técnicas van a emplear los símbolos cortos y largos de la cabecera explicada en el apartado anterior. Sin embargo dicha cabecera puede no llegar completa al bloque de sincronismo porque en el bloque anterior (control automático de ganancia) se hayan necesitado varios símbolos cortos para conseguir estabilizarse. En principio se desarrollarán las técnicas suponiendo que van a llegar los 10 símbolos cortos y los 2 símbolos largos, pero en los últimos algoritmos se planteará la posibilidad de que esto no se cumpla y, de hecho, algunos de los escenarios simulados se comportarán de esta manera.

2.1.1 Sincronismo en Tiempo

Los sistemas OFDM son relativamente poco sensibles a los errores cometidos en la sincronización temporal gracias a la existencia del prefijo cíclico. Esta preguarda permite que el margen de error que se pueda cometer en la sincronización en tiempo sea bastante amplio. Se van a tratar varios mecanismos de sincronización, todos ellos basados en el método de Schmidl y Cox [11].

Primer algoritmo:

El algoritmo propuesto por Schmidl y Cox es el siguiente:

1. Se realiza una correlación entre las dos primeras mitades del símbolo:

$$P[d] = \sum_{m=0}^{L-1} \left(r^*[d+m] r[d+m+L] \right)$$

2. Se calcula la potencia promediada en la segunda mitad del símbolo:

$$R[d] = \sum_{m=0}^{L-1} |r[d+m+L]|^2$$

3. Se obtiene la métrica temporal como:

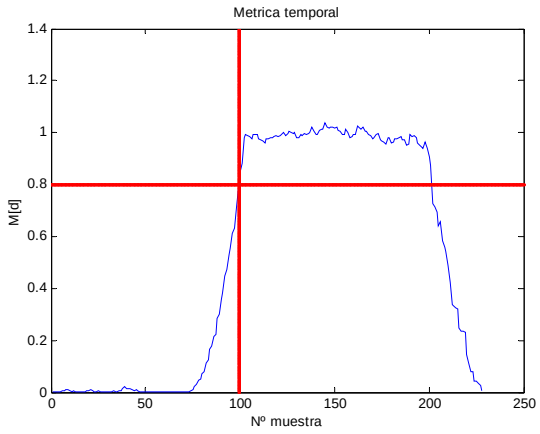
$$M[d] = \frac{|P[d]|^2}{(R[d])^2}$$

4. Para saber dónde empieza la trama se compara la métrica anterior con un umbral:

$$M[d] > umbral\_amplitud$$

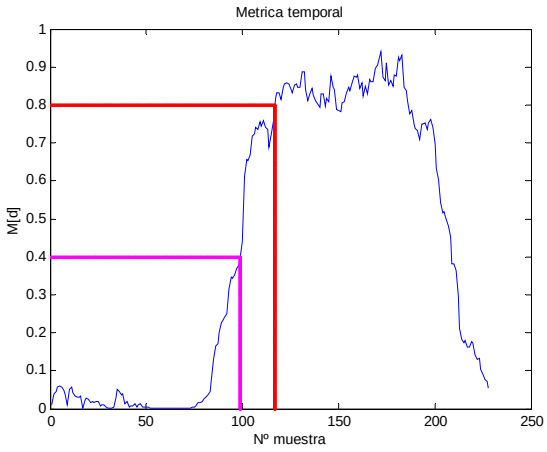
Como en este proyecto se están siguiendo las recomendaciones del estándar IEEE 802.11a, habrá que tener en cuenta que para el sincronismo temporal se deben emplear los símbolos cortos de los que consta la cabecera. De manera que las correlaciones se realizarán sobre las 161 primeras muestras de la trama que son las correspondientes a los 10 símbolos cortos.

En la siguiente figura, se puede ver la gráfica con la que se decidirá en qué instante comienza la trama. La línea de puntos horizontal se corresponde con el umbral óptimo que se cruza con la métrica en el punto marcado por la línea vertical. Con este umbral se decidiría que la trama comienza en la muestra 100 y a partir de ahí se comenzaría a recibir los símbolos cortos y largos con los que se estimaría el offset en frecuencia y los efectos del canal.



Métrica temporal en un caso bueno.

Estas pruebas han sido realizadas para un escenario relativamente bueno. En el que la señal enviada ha pasado por un canal gaussiano, ha sufrido una desviación en frecuencia pequeña y tiene una relación señal a ruido de 20 dB. Por eso en las gráficas se ven claramente las subidas y bajadas y, con el umbral adecuado no habría error en la estimación temporal. Sin embargo, a lo largo de este proyecto, se van a presentar escenarios que no son tan agradables y en los que este algoritmo puede llevar a confusión y detectar un paquete cuando sólo era ruido. Esto es lo que muestra la siguiente figura, que se corresponde con una simulación que ha sido realizada para un canal HiperLAN 2 tipo B, que más adelante se detallará cómo es, con una relación señal a ruido de 10dB y una desviación en frecuencia el doble que en el anterior caso.



Métrica temporal en un caso malo.

En este caso con el umbral anterior el sistema se habría sincronizado unas 15 muestras tarde, aproximadamente, con lo cual se habría perdido toda la información. Ahora el umbral bueno sería la mitad del anterior, pero esto es un parámetro que debe ser fijo y aunque se supiese su variación con la relación señal a ruido del canal, en principio se supondrá que no se puede saber la snr. Además existe otro inconveniente y es la posibilidad de que, sin existir trama, el sistema se sincronice. Esto sería debido a que como el receptor está continuamente escuchando, puede llevar a error en alguna ocasión en la que, por motivos del ruido, la señal supere el umbral en una muestra y después todo lo que venga a continuación no sea un paquete sino ruido. Esto provocaría una falsa alarma que haría sincronizarse al sistema con un supuesto paquete que no se corresponde con un paquete con símbolos OFDM. El segundo algoritmo trata de evitar este inconveniente.

Segundo algoritmo:

Como resumen del algoritmo anterior se podría decir que se realizaba la correlación cruzada de la señal que se recibía consigo misma, se normalizaba a su potencia y se decidía que había llegado un paquete cuando una de las muestras de esa correlación normalizada superaba el umbral de detección.

En este segundo algoritmo se plantea la posibilidad de emplear más de una muestra de la métrica temporal (que es la correlación cruzada y normalizada de la señal recibida) para decidir si lo que se está recibiendo es un paquete o no. De modo que si varias muestras consecutivas superan el umbral de detección, se considerará que está llegando una trama OFDM.

Tercer algoritmo:

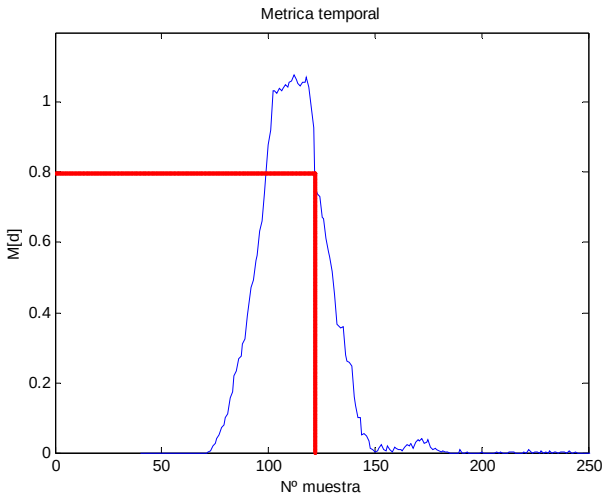
En este tercer algoritmo se va a contemplar, por primera vez, la posibilidad de la cabecera que no llegue completa. Realmente no es una posibilidad sino un hecho, puesto que el primer bloque que compone el receptor de un sistema OFDM es un control automático de ganancia, el cual emplea un algoritmo que necesita un cierto número de muestras de la cabecera para estabilizarse y hasta que este bloque no se estabiliza no devuelve la señal recibida al bloque en el que se encuentran estos algoritmos de sincronización.

Además el número de muestras que necesita el algoritmo del CAG para estabilizarse no es necesariamente el mismo en todos los paquetes, por ello, de los 10 símbolos cortos que forman la cabecera llegarán al bloque de sincronismo, un número de símbolos variables.

Con los algoritmos anteriores, una vez que se había estimado dónde comenzaba el paquete, se realizaba una primera estimación de la desviación en frecuencia, se esperaban 10 símbolos cortos, cada uno de 16 muestras, y entonces se empezaba a trabajar con los símbolos largos. Ahora no se sabe cuántos símbolos cortos han llegado y por lo tanto lo que se tendrá que estimar es cuándo se han acabado los símbolos cortos y por lo tanto cuándo empiezan los largos. Para ello se seguirá empleando la misma métrica temporal que en los algoritmos anteriores pero en vez de comprobar cuándo dicha métrica supera el umbral de detección, se observará cuándo desciende de dicho umbral.

En todos los algoritmos vistos hasta ahora y en el que queda por explicar, se emplean unos umbrales de detección que deberán ser simulados con el objetivo de encontrar cuáles son los óptimos para cada uno de ellos.

En los siguientes ejemplos se va a suponer que han llegado sólo 5 símbolos cortos y que, como en los casos anteriores, antes de la primera cabecera se habían escuchado 100 muestras de ruido. La siguiente figura muestra cómo quedaría la métrica temporal para un caso bueno.



Métrica temporal con sólo 5 símbolos cortos.

Como se observa, la anchura de la zona plana es ahora mucho menor. Según se ha ido viendo a lo largo de estos algoritmos, todos basados en el método propuesto por Schimdl y Cox, la forma que adquiere la métrica temporal es la de una montaña que empieza a subir *L* muestras antes de que comience la cabecera, permanece plana hasta *2*L* muestras antes de que se acaben los símbolos cortos y finalmente desciende durante *L* muestras, donde *L* es la anchura de la ventana de correlación, en este caso *L=32*.

Por lo tanto, en este algoritmo, una vez que se ha detectado que la métrica está comenzando a descender, habría que esperar *2*L (64)* muestras hasta saber que ya han acabado los símbolos cortos y llegan los largos.

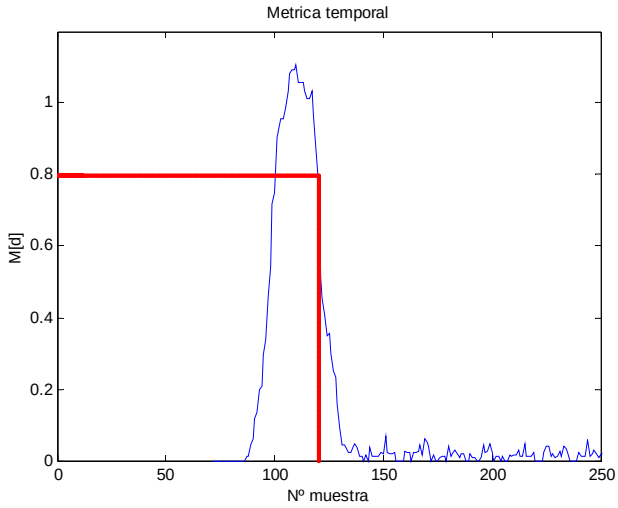
Sin embargo, este tercer algoritmo tiene un inconveniente y es que, como se ha visto, a medida que llegan menos símbolos cortos, la zona plana de la métrica temporal se va haciendo más pequeña hasta que llegue un momento en el que no llegue a subir del todo. Esto ocurre cuando el número de muestras sobre el que se hace la correlación es menor que dos veces el tamaño de la ventana, es decir cuando llegan menos de cuatro símbolos.

Cuarto algoritmo:

En este último algoritmo se intenta resolver el problema planteado anteriormente. Como se comentó el límite para que la métrica temporal alcance la forma deseada es que lleguen al menos tantas muestras como el doble de la ventana de correlación.

En este algoritmo se planteará la posibilidad de reducir la ventana de correlación, observando tanto sus ventajas como sus inconvenientes. Si la ventana de correlación es más pequeña, la métrica será más inestable al contar con menos muestras, pero permitirá trabajar con menos símbolos cortos.

La siguiente figura muestra como quedaría la métrica temporal si las correlaciones se hiciesen con una ventana de 16 muestras. En el caso representado se han recibido tres símbolos y, desde que se detecta que la métrica ha descendido por debajo del umbral de detección, habría que esperar el doble de la ventana de correlación, ahora 2*16 muestras, hasta empezar a recibir los símbolos largos.



Métrica temporal con una ventana de 16 muestras.

Los cuatro algoritmos explicados en este apartado son los estudiados en este proyecto y en la parte de resultados dentro de la memoria completa del proyecto se muestran el comportamiento y las prestaciones que ofrece cada uno de ellos en distintos escenarios.

2.1.2 Sincronismo en Frecuencia

En el caso de sistemas de una sóla portadora, la desviación en frecuencia que pueda sufrir dicha portadora se traduce en una degradación, más o menos grave, de la relación señal a ruido. Sin embargo, en el caso de modulaciones multiportadora, además de esta degradación se producen interferencias entre las portadoras. Esto ha supuesto siempre una gran desventaja de estos sistemas frente a los sistemas monoportadora, pero en este apartado se verá que existen técnicas que pueden reducir considerablemente los efectos producidos por la desviación en frecuencia.

Como este punto es de los más críticos en un receptor OFDM, se han estudiado y evaluado numerosas técnicas, cuyos algoritmos se explican a continuación. La mayoría de ellos están basados en el método de Schmidl y Cox, pero han sufrido alguna modificación con el objeto de mejorar sus prestaciones. Finalmente se verán dos algoritmos, basados en un método distinto, donde el último de ellos empleará las subportadoras pilotos para reestimar la desviación en frecuencia. Los resultados de estos algoritmos se pueden ver en la memoria completa del proyecto.

Para el sincronismo en frecuencia el método de Schimdl y Cox propone el siguiente algoritmo:

1. Se parte del primer paso del algoritmo para la sincronización temporal, donde se calculó la correlación entre las dos mitades del símbolo P[d] de forma que se han cancelado los efectos del canal.
2. Se calcula el desplazamiento en frecuencia de la siguiente forma:

$$\phi = \angle\left(P\left[d\right]\right)$$

$$\Delta f = \frac{\phi}{\pi T}$$

Partiendo de esta base, los diferentes algoritmos que se proponen lo único que modifican es el punto en el que se toma el valor de la desviación en frecuencia, es decir, tanto ϕ como Δf son vectores con tantos valores como muestras tenga P[d]. Lo que hay que decidir es en qué instante se toma el valor óptimo de Δf que luego habrá que corregir.

Llegados a este punto, y para que sea de más fácil comprensión los siguientes algoritmos, se va a hacer una pequeña aclaración en cuanto a la notación:

- ϕ: es la fase que forma la correlación P[d].
- Δf: es la desviación en frecuencia que ha sufrido la señal y que habrá que corregir.
- ε: es Δf pero normalizada a la separación entre portadoras.

$$\varepsilon = \frac{\Delta f}{B/N} = \frac{\phi/\pi T}{1/T} = \frac{\phi}{\pi}$$

Primer algoritmo:

El primer algoritmo que se propone va ligado al primero que se propuso para la sincronización en tiempo y consiste en, una vez que se ha detectado que ha llegado un paquete, tomar el offset de frecuencia en ese mismo instante. Sin embargo, los umbrales en ambos casos no tienen por qué ser los mismos.

El umbral de detección, se simulará buscando su valor óptimo para que el error cometido al estimar en qué momento comienza la trama sea el menor posible. En el caso de la estimación del offset de frecuencia, el umbral, que por similitud se denominará de estimación, se buscará con la finalidad de que el error cometido al estimar el offset de frecuencia sea menor.

De esta forma cuando la métrica temporal supere el umbral de detección se habrá estimado que la trama acaba de empezar y, cuando supere el de estimación, se tomará en ese instante el valor correspondiente de **ε** y ése será el offset de frecuencia a corregir en el resto de la trama.

La forma de corregir la desviación en frecuencia es tan simple como desplazar la señal en frecuencia lo que implica multiplicar la señal por la exponencial de la siguiente manera:

$$s_{\text{corregida}}[n] = s_{\text{sin corregir}}[n] \cdot e^{-j\frac{2\pi\varepsilon \cdot n}{N}}$$

donde:

- ε, es el offset estimado.
- N, es el número de portadoras (64).

Segundo algoritmo:

También va emparejado al segundo algoritmo del sincronismo temporal y consiste en comprobar que más de una muestra consecutiva supera el umbral, en este caso de estimación. Si es así se tomará ese valor de offset.

Tercer algoritmo:

El tercer algoritmo, también equiparable a su homólogo temporal, consiste en realizar la comprobación con el umbral de estimación en la bajada de la métrica temporal y, cuando sea rebasado, tomar ese valor del offset. Este algoritmo pone solución al hecho de que la cabecera no llegue entera.

Hasta ahora se ha estimado el offset aplicando el método de Schmidl y Cox sobre los símbolos cortos. Se van a plantear ahora tres algoritmos más que trabajan sobre los símbolos largos, dos de ellos con un método distinto.

Cuarto algoritmo:

Como se ha comentado los símbolos cortos se empleaban para el sincronismo temporal y para la estimación gruesa del offset de frecuencia. Así mismo, los símbolos largos se empleaban para la estimación fina del offset y la estimación de los efectos del canal.

Se verá más adelante, que el error cometido por los algoritmos anteriores al estimar el offset no es lo suficientemente pequeño como para que no afecte considerablemente a la probabilidad de error, especialmente si los paquetes son largos. Para intentar mejorar la estimación gruesa realizada con los símbolos cortos se plantea la reestimación con los símbolos largos. El método a seguir es el mismo, se calcula la métrica temporal sobre los símbolos largos, esta vez con una ventana de correlación de 64 muestras. El hecho de que la longitud de la ventana sea mayor es una ventaja pues como ya se ha comentado, las correlaciones son más estables.

Una vez que se tiene la métrica temporal, se vuelve a aplicar la comprobación con el umbral sobre dicha métrica para reestimar el valor del offset.

Queda mencionar que si lo que se desea es reestimar el offset, los símbolos largos deben ser previamente corregidos con la desviación en frecuencia estimada con los cortos.

En este algoritmo, el principal problema surge en la precisión. Una vez que se ha realizado una primera estimación del offset con los símbolos cortos se deben estimar valores muy pequeños de offset y, en ocasiones, debido a que se está trabajando con valores en punto fijo, con precisión limitada, se puede empeorar la primera estimación.

Quinto algoritmo:

Tratando de dar solución al problema surgido en el algoritmo anterior, se intentó estimar el offset sólo con los símbolos largos, es decir, los símbolos cortos se emplearían únicamente para el sincronismo en tiempo y los largos para el sincronismo en frecuencia y la estimación del canal.

Trabajando únicamente con los símbolos largos surgen dos algoritmos, el quinto y el sexto, según la notación seguida hasta ahora.

En este quinto algoritmo, se aplicaría el método que se lleva siguiendo a lo largo de todo este capítulo, pero sólo sobre los símbolos largos, con una ventana de L=64 muestras lo cual permite una estimación del offset en frecuencia mucho más estable y por lo tanto fiable.

Sexto algoritmo:

En este caso se va a aplicar sobre los símbolos largos un método distinto al de Schmidl y Cox. Este método basado en el artículo de Paul H. Moose [12] y los pasos a seguir según este autor serían:

1. Multiplicar las muestras del segundo símbolo largo recibido por las del primero conjugado y sumarlas, cancelando de esta manera las contribuciones del canal:

$$P' = \sum_{m=0}^{L-1} \left(r[m+L] \cdot r^*[m] \right)$$

2. Finalmente estimar el offset a partir de este valor del mismo modo que se hizo anteriormente:

$$\phi = \angle(P')$$

$$\Delta f = \frac{\phi}{\pi T}$$

$$\varepsilon = \frac{\phi}{\pi}$$

Séptimo algoritmo:

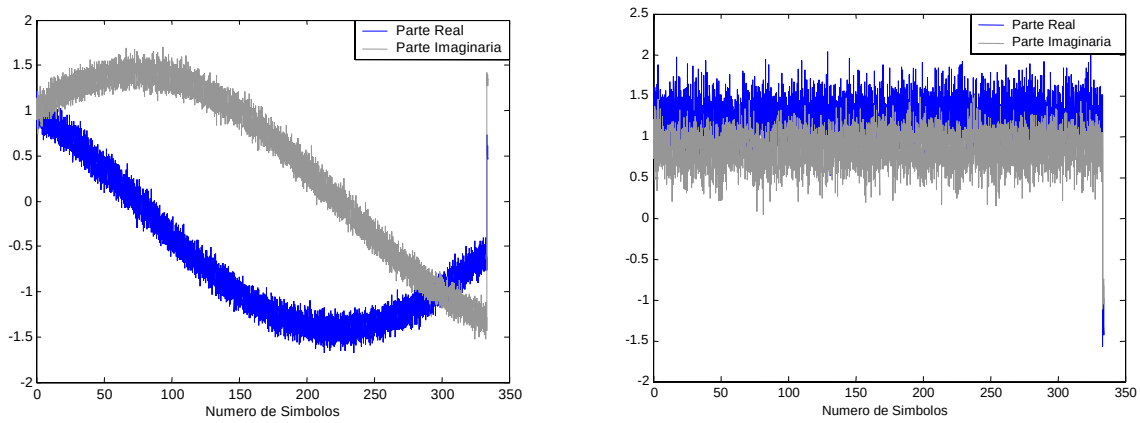
Por último se va a abordar el único algoritmo que emplea las subportadoras piloto para reestimar el offset.

Este paso se lleva a cabo en el último bloque, una vez que el paquete ha sido sincronizado en tiempo y en frecuencia con uno de los algoritmos anteriores y corregida dicha desviación en frecuencia y además después de que el canal ha sido estimado y corregido. En ese momento se puede acceder a las subportadoras piloto que van en cada símbolo y reestimar el offset.

Para reestimar el offset se emplea el algoritmo anterior (el sexto) sobre las cuatro portadoras piloto que hay en cada símbolo OFDM. Esta estimación puede ser bastante inexacta puesto que en los símbolos largos se realiza sobre 64 muestras y en este caso sólo sobre cuatro. Por ello se decidió realizar un promedio sobre varios símbolos OFDM para reducir las fluctuaciones.

En los resultados mostrados en el proyecto se ve que un error de offset pequeño durante 50 símbolos es tolerable, por lo tanto se decidió tomar este dato como límite superior, es decir, durante 50 símbolos se aplica el sexto algoritmo sobre las cuatro muestras correspondientes a las subportadoras piloto y posteriormente se realiza un promedio.

A modo de ejemplo, se presenta la siguiente figura que muestra los beneficios que se pueden conseguir en paquetes largos si se reestima el offset (a partir de ahora denominado tracking de pilotos).



Beneficios de la reestimación del offset con los pilotos.

En este ejemplo, mencionado anteriormente, se mandan más de 30.000 datos, todos de valor '1'. Por ello los símbolos recibidos, si se ha modulado con una QPSK, deberían ser '1+j'. En la figura de la izquierda se ve cómo afecta el error de estimación en frecuencia y en la figura de la derecha se ve cómo el tracking de pilotos ayuda a su corrección obteniendo los símbolos esperados.

3. ESTIMACIÓN Y CORRECCIÓN DEL CANAL

En el apartado anterior se han comentado técnicas de sincronismo temporal y frecuencial. Ahora, una vez que se tiene claro dónde comienzan los símbolos largos y se ha corregido la desviación en frecuencia que habían sufrido al pasar por el canal, el siguiente paso es la estimación y corrección del canal.

3.1 ESTIMACIÓN DEL CANAL

En el proceso de estimación del canal existen dos pasos a seguir:

- Extracción de la información sobre el canal.
- Estimación del comportamiento del canal a partir de la información extraída en el paso anterior.

En este proyecto se emplearán las técnicas PSA, basadas en datos conocidos que, aunque suponen una pérdida en la tasa de datos, proporcionan una estimación del canal más fidedigna. Los métodos PSA se basan en el concepto de PSAM (Pilot Symbol Assisted Modulation) que primeramente fueron pensados para sistemas de una sóla portadora. Consisten en transmitir símbolos conocidos dentro de la secuencia de datos para estimar el canal y poder emplear así modulaciones de varias amplitudes. Si se aplica esta idea a los sistemas multiportadora, en concreto OFDM, hay que tener en cuenta que el canal debe ser estimado en las dos dimensiones, tiempo y frecuencia, y por lo tanto deben insertarse símbolos pilotos a lo largo de toda la rejilla tiempo-frecuencia.

En este caso y basándose en el estándar, se cuenta con los símbolos largos y con portadoras pilotos distribuidas a lo largo de la trama OFDM para llevar a cabo la estimación del canal. Finalmente, en este proyecto se estudiará e implementará una técnica que extrae la información de los dos símbolos largos que existen en la cabecera. Debido a que las longitudes de paquetes que se van a emplear no son los suficientemente grandes como para que el canal tenga una variación temporal significativa, no será necesario realizar una reestimación a lo largo de la trama. Aunque si en algún momento se considerase necesario se podrían emplear las subportadoras piloto para realizar dicha reestimación.

A continuación se va a explicar cómo es la extracción de la información del canal mediante los símbolos largos. Si se denota H(k) a las atenuaciones complejas que posee el canal en la k-ésima portadora, la expresión de un símbolo recibido en el dominio del tiempo sería:

$$y[n]=\frac{1}{\sqrt{N}}\sum_{k=0}^{N-1}X(k)H(k)e^{j\frac{2\pi nk}{N}}+w[n]$$

donde X(k) es la señal enviada en el dominio de la frecuencia y w[n] es un ruido complejo, aditivo blanco Gaussiano. Después de realizar la transformada discreta de Fourier, el símbolo recibido en el dominio de la frecuencia está formado por dos sumandos, uno correspondiente a las muestras de la DFT aplicada sobre el ruido y otro correspondiente a la señal enviada en el dominio de la frecuencia, multiplicada por las atenuaciones complejas del canal.

$$Y[k]=X[k]\cdot H[k]+W[k]$$

Si ahora este símbolo se corresponde con uno de los símbolos largos cuyos valores son conocidos (P[k]), se podrán estimar las atenuaciones del canal con un margen de error debido al ruido.

$$\hat{H}[k]=\frac{Y[k]}{P[k]}=\frac{P[k]H[k]}{P[k]}+\frac{W[k]}{P[k]}=H[k]+\Delta H[k]$$

Como el símbolo largo tiene muestras en todas las frecuencias, no hace falta estimar cómo sería el canal en el resto de las frecuencias, con lo que el último de los dos pasos comentados al comienzo de este apartado se ahorra si se trabaja con los símbolos largos.

Además, se puede aprovechar el hecho de tener dos símbolos largos. Se realizará el cálculo de las atenuaciones complejas del canal con cada uno de los símbolos y posteriormente se realizará un promedio entre los dos conjuntos de valores obtenidos

3.2 CORRECCIÓN DEL CANAL

Las técnicas de corrección de canal también se conocen como técnicas de igualación pues tratan de compensar la atenuaciones que ha producido el canal y que ya han sido estimadas.

El símbolo corregido, en el dominio de la frecuencia, resultará de la multiplicación del símbolo recibido por el coeficiente de igualación I[k].

$$Y_C[k]=Y[k]\cdot I[k]$$

Existen varios métodos de obtener el coeficiente de igualación a partir de las atenuaciones estimadas del canal. Los tres que fueron propuestos para ser estudiados son:

- Zero Forcing (ZF): $I[k]=\frac{1}{\hat{H}[k]}$
- Maximum Ratio Combining (MRC): $I[k]=\hat{H}^*[k]$
- Equal Gain Combining (EGC): $I[k]=\frac{\hat{H}^*[k]}{|\hat{H}[k]|}$

De los tres el que siempre ha tenido mejores resultados ha sido el primero de ellos pero tiene una desventaja a la hora de ser implementado físicamente en la FPGA y es que cualquier operación que conlleve una división implica un coste computacional muy elevado que, siempre que se pueda, ha de ser evitado.

4. DISEÑO DEL RECEPTOR EN SIMULINK

En este capítulo se van a comentar los pasos que hubo que seguir para llegar del diseño de la WLAN en Matlab con coma flotante, al diseño final en Simulink con punto fijo. A continuación e describirán los bloques en los que se ha estructurado el receptor.

4.1 BLOQUE SINCRO T/F

En este bloque se realizan las funciones relacionadas con el sincronismo en tiempo y frecuencia. Previo a este bloque se encuentra el bloque correspondiente al CAG y el atenuador, que devuelve la señal atenuada por la ganancia indicada por el CAG asegurando que sus niveles de amplitud no puedan saturar los conversores A/D y D/A. Posterior al bloque de sincronismo, se encuentra el bloque que debe corregir el offset que aquí se estime.

Funciones:

Como ya se comentó en este bloque se llevan a cabo las técnicas de sincronismo temporal y frecuencial. Para ello se emplean los símbolos cortos y/o los símbolos largos.

Se tratará de estimar lo más fielmente posible dónde comienza el paquete o, en el caso de que no llegue completo, dónde terminan los símbolos cortos y comienzan los largos. También se estimará la desviación en frecuencia que ha sufrido el paquete y se devolverá este dato al siguiente bloque para que pueda corregirlo.

4.2 CORRIGE OFFSET

Funciones:

Del bloque anterior se recibirá un valor de offset de frecuencia que debe ser corregido en este bloque. De esta forma se le devolverá al bloque siguiente la señal corregida en frecuencia para estimar el canal adecuadamente.

4.3 ESTIMA / CORRIGE CANAL

Este es el último bloque simulado en este proyecto. En él, se trata de estimar los efectos que el canal ha provocado sobre la señal recibida y corregirlos, posteriormente, para devolver una señal IQ totalmente corregida al demodulador que poseen las placas de la Plataforma Praga y que, tras demodular y decodificar, obtendrán los datos recibidos. Estos dos últimos pasos no serán estudiados en este proyecto.

Funciones:

En este bloque se realizan dos funciones básicas: se estima el canal y se corrige sobre los datos. Para ello en principio se emplean los símbolos largos que también han sido corregidos en frecuencia por el bloque anterior. Pero además existe la posibilidad de reestimar los efectos del canal en el último estado con las subportadoras piloto.

En principio, este bloque sólo devolvía los símbolos correspondientes a la constelación elegida para que, posteriormente con el demodulador que tiene integrada la Plataforma PRAGA, se obtuviesen los datos recibidos. Sin embargo se incluyó dentro de este último bloque un demodulador para que se pudiese comprobar que los bloques anteriores funcionaban correctamente antes de integrarlos en la plataforma.

4.4 TRANSMISOR OFDM

Los tres bloques anteriores son los característicos de un receptor OFDM y, junto con el bloque del CAG, conforman el receptor que se simuló para ser integrado en la Plataforma PRAGA. Pero para simular completamente su funcionamiento fue necesario diseñar un bloque más, correspondiente al transmisor OFDM.

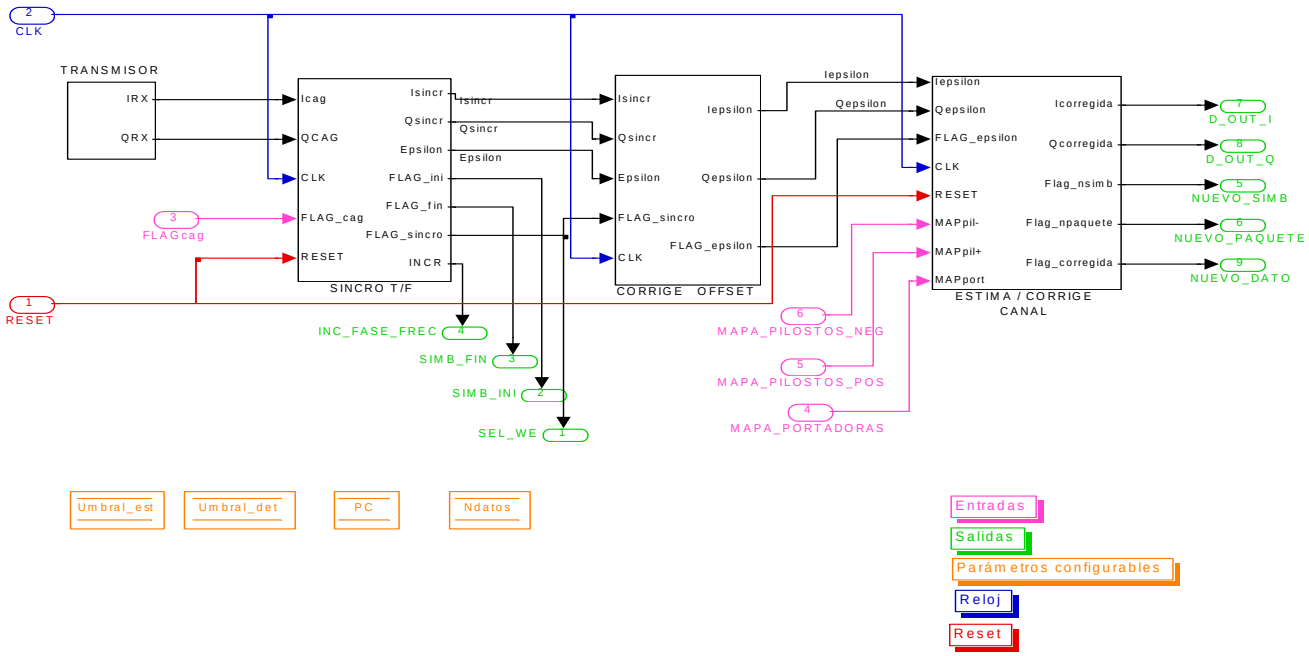
Si se quería observar la probabilidad de error en los datos, se debía conocer, además de los datos recibidos, los datos enviados. Además debían simularse los efectos que provocaba el canal. Por todas estas razones hubo que diseñar un bloque con las siguientes características.

Funciones:

Aunque este módulo se ha denominado transmisor OFDM, en él se llevan a cabo más funciones que las propias de un transmisor:

- Se generan aleatoriamente los datos.
- Se codifican los datos con una de las siguientes modulaciones: BPSK, QPSK, 16-QAM y 64-QAM.
- Se generan los símbolos OFDM a partir de los símbolos anteriores.
- Se genera la cabecera y, junto con los símbolos OFDM, el paquete.
- Se le hace pasar por el canal que en ese momento se esté simulando, a elegir entre: HiperLAN 2 tipo B o UMTS exterior tipo A.
- Se le añade el ruido Gaussiano, así como muestras de ruido delante del paquete para simular la posibilidad de que el paquete llegue en cualquier momento.

La siguiente figura muestra la relación entre los bloques expuestos en los apartados anteriores.



Esquema de conexión de bloques en el receptor.

5. SIMULACIONES REALIZADAS EN LA WLAN

Por último, se van a llevar a cabo las simulaciones y extraer de ellas los resultados oportunos que permitan conocer cómo va a ser el funcionamiento de la WLAN, qué prestaciones, principalmente en lo relativo a probabilidad de error, va a tener y que permitan escoger, entre las numerosas técnicas presentadas anteriormente, las que ofrezcan un mejor comportamiento global teniendo en cuenta los escenarios en los que van a ser probadas.

Con simulaciones previas sobre todos los canales, se dedujeron que los umbrales óptimos de decisión y estimación habían cambiado respecto a los equivalentes en Matlab.

Canal	Umbral de Detección		Umbral de Estimación	
	Matlab	Simulink	Matlab	Simulink
Gauss	0.8-0.9	0.75-0.8	0.6	0.5
HiperLAN	0.7-0.8	0.65	0.4	0.55-0.6
UMTS	0.7-0.8	0.7-0.75	0.3-0.4	0.5

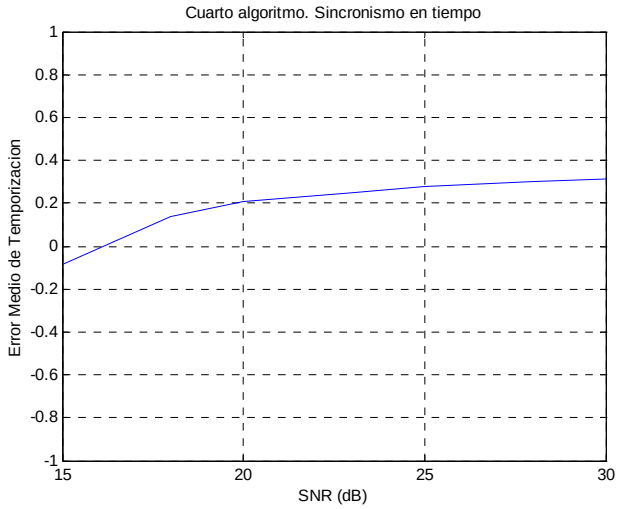
Comparación de los umbrales de Matlab y Simulink.

Se realizaron simulaciones con modulaciones QPSK, 16QAM y 64QAM y en tres tipos de canales:

- Canal Gauss
- Canal Hiperlan 2 tipo B
- Canal UMTS Exterior tipo A

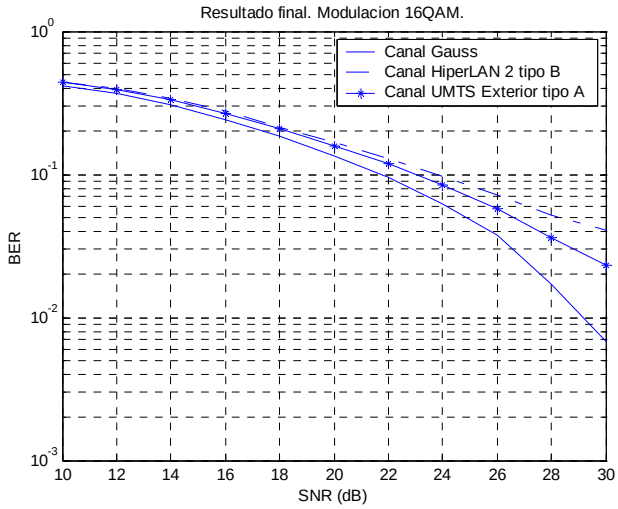
Las siguientes figuras muestran los resultados obtenidos con los mejores algoritmos para sincronismo en tiempo, frecuencia y corrección de canal (para más resultados, por favor referirse a la memoria completa del proyecto).

Para el caso del algoritmo de sincronización temporal, se muestra el error medio de temporización respecto de la relación señal a ruido. Este error representa el número de muestras que el receptor falla al indicar el comienzo de una trama.



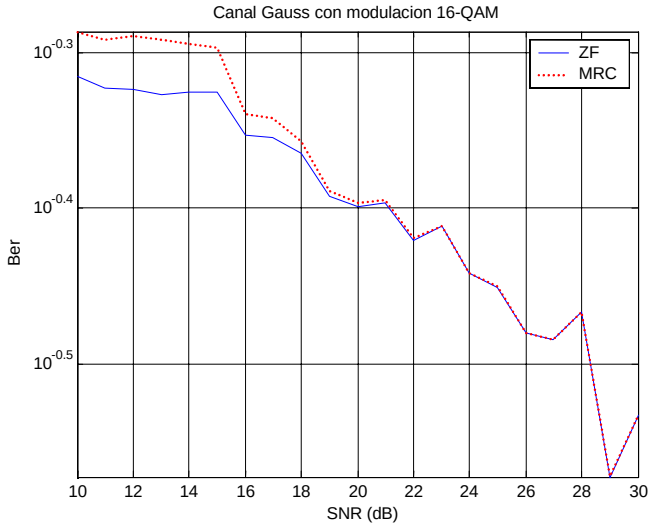
Cuarto algoritmo. Error medio de temporización.

Para el caso del algoritmo de sincronización en frecuencia, se muestra la probabilidad de error en bit respecto de la relación señal a ruido para una modulación 16QAM.



Resultados finales. Modulación 16-QAM.

Para el caso del algoritmo de corrección de canal, se muestra la probabilidad de error en bit respecto de la relación señal a ruido para una modulación 16QAM y canal Gauss.



Corrección de canal: canal Gauss, modulación 16-QAM y paquetes largos.

6. CONCLUSIONES

El primer paso de este proyecto fue desarrollar el receptor, así como otros elementos necesarios (transmisor, varios canales, demodulador, etc...), de la WLAN en la herramienta Simulink para reflejar las condiciones reales en las que se va a implementar finalmente la red. Aquí se descubrió lo eficaz que resulta esta herramienta para introducir el punto fijo en las simulaciones.

Sin embargo, como lo que se quería era aprovechar los códigos programados en Matlab, en vez de llevar a cabo una programación a más bajo nivel, se emplearon unos bloques llamados 'S-Function' que permiten adaptar el código proveniente de diferentes lenguajes de programación, entre ellos el de Matlab, a la simulación llevada a cabo en Simulink. Este paso fue uno de los más difíciles de ejecutar pues, como se comentó en las principales diferencias entre Matlab y Simulink, tienen estrategias de simulación muy distintas, que hacen que la labor de adaptación entre una herramienta y la otra sea muy complicada especialmente para el caso en el que los programas que se quieran adaptar sean algo complejos.

También por este motivo se puso mucho énfasis en la validación pues, aunque el número de líneas programadas para Simulink no era elevado, jugaba en numerosas ocasiones con las muestras correspondientes a las señales, almacenándolas y devolviéndolas, por lo que hubo que asegurarse que se hacía en el orden correcto y que no se perdía ninguna.

Otra ventaja que se obtuvo con el diseño de la WLAN con Simulink fue la realimentación que se estaba buscando y que permite encontrar puntos a mejorar en los programas de Matlab, tales como funciones que no sean de fácil implementación en una FPGA o algoritmos que, debido al punto fijo, no se comporten todo lo bien que se esperaba y a los que se les debe buscar una alternativa mejor.

En este punto hay que destacar que se estudiaron varias posibilidades para sustituir a la función arcotangente, e incluso combinaciones de dichas posibilidades. Todas fueron evaluadas buscando un compromiso entre la complejidad y las prestaciones que aportaban, aunque finalmente se añadió al Proyecto Praga un nuevo módulo, denominado cordic, que permitía realizar la función de la arcotangente en la FPGA.

Como punto negativo, hay que comentar que las simulaciones realizadas sobre Simulink son mucho más lentas que sobre Matlab, especialmente cuando se querían comparar ciertos métodos de sincronismo en frecuencia y había que simular muchos paquetes muy grandes.

En cuanto a los algoritmos estudiados a continuación se comentará un breve resumen para resaltar las principales diferencias.

En principio comentar que los dos primeros algoritmos de sincronismo, tanto temporal como en frecuencia, fueron rechazados debido a la alta posibilidad de que la cabecera de los paquetes no llegaran enteras. De esta forma quedan, por ahora, dos posibles algoritmos para la sincronización temporal y cinco para la frecuencial.

De los dos restantes para el sincronismo en tiempo, la única diferencia es la longitud de la ventana de correlación, que hace que el primero de ellos sea más estable pero necesite más muestras que el segundo de cabecera recibida.

Finalmente se optó por el último de ellos pues se supo que en el caso peor el número de símbolos cortos que llegarían provenientes del CAG eran solamente tres con lo cual el primero de ellos no funcionaría.

En cuanto a los cinco algoritmos restantes de sincronismo frecuencial se optó por emplear conjuntamente los dos últimos, pues el penúltimo era el que menor error estimando el offset de frecuencia daba, siendo además el más simple de implementar y el último era el que mejoraba el comportamiento cuado los paquetes son muy largos.

De los métodos de corrección de canal, se comprobó lo que ya se esperaba, que el método del Zero Forcing, a pesar de implicar una división, que es bastante compleja de implementar en una FPGA, era el que mejor se comportaba con mucha diferencia, por lo que se desecharon el resto.

Concluir que este proyecto ha sido implementado en Verilog, ver referencia [19] de la bibliografía.

Por último una pequeña reflexión, parece increíble la cantidad de opciones y distintos subalgoritmos que surgen de un único método [11] cuando se trata de implementar. Todos ellos han de ser evaluados para ver cual es la mejor solución manteniendo el compromiso entre complejidad y prestaciones.