UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

INGENIERÍA DE TELECOMUNICACIÓN

Departamento de Teoría de la Señal y Comunicaciones

PROYECTO FIN DE CARRERA

ESTUDIO DE LA MODULACIÓN SC-FDMA E IMPLEMENTACIÓN EN UN DISPOSITIVO SFF SDR

Autor: Kun Chen

Tutora: Ana García Armada

Leganés, Junio de 2012

TÍTULO: **Estudio de la Modulación SC-FDMA e implementación en un dispositivo SFF SDR.**

AUTOR: **Kun Chen.**

TUTORA: **Ana García Armada.**

EL TRIBUNAL

PRESIDENTA: **Matilde Sánchez Fernández.**

SECRETARIO: **Víctor Pedro Gil Jiménez.**

VOCAL: **Celia López Ongil**

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día 13 de Junio de 2012 en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

Presidente **Vocal** **Secretario**

AGRADECIMIENTOS

Durante casi 6 años de esfuerzo y sacrificio, mi familia ha sido el pilar de apoyo para mí. Ya que familia siempre ha sido para mí una imagen de constancia y perseverancia ante la vida. Por tanto siento una gran felicidad y placer por el apoyo incondicional que me han brindado.

En este periodo de tiempo, mis amigos y compañeros de la carrera han sido muy importantes para mí, son los que han estado conmigo, compartiendo la felicidad y tristeza del día a día. Sin ellos quizás no habría sido lo mismo, y el camino no hubiese sido tan fácil y tan cómodo.

Por último tengo que agradecer a todos los profesores que me han ayudado con el presente proyecto, en especial a mi tutora, ya que han hecho un gran esfuerzo por ayudarme en todas las complicaciones que he tenido durante el desarrollo del proyecto.

RESUMEN

El ser humano tiene el afán por superarse a sí mismo una y otra vez, y siempre se está marcando nuevos retos. Las comunicaciones móviles han crecido vertiginosamente en todo el mundo, y sobre todo en España. Por tanto, ya sea por necesidad o por ese afán de superación, se debe desarrollar nuevos mecanismos o sistemas de comunicación más avanzados.

Unos de los grandes objetivos de las empresas de telecomunicaciones, es reducir el coste por bit en la transmisión de la misma, para ello se deben rediseñar constantemente nuevas tecnologías. En la cuarta generación en telefonía móvil se vislumbra LTE (Long Term Evolution) donde se desea conseguir una velocidad de descarga hasta 100Mbps mediante una modulación OFDM. Este nuevo estándar no sólo quiere alcanzar tasas elevadas de transmisión, sino rediseña por completo la nueva arquitectura de red, por ejemplo se desea eliminar toda la red de conmutación de circuitos, dejando exclusivamente una única red de paquetes IP. De este modo estaríamos reduciendo costes tanto en la red, como el enlace radio.

Los que nos concierne a nosotros, es el estudio de la modulación de LTE en el uplink, la nueva modulación SC-FDMA (Single Carrier – FDMA). Dicha modulación pretende reducir elevados y frecuentes picos que se producen en OFDM, aumentando así el rendimiento y eficiencia de los terminales móviles de los usuarios. Para realizar dicho estudio se realizará dos etapas: primero se simulará en Matlab y posteriormente se implementará en un dispositivo hardware.

En la primera etapa construiremos las los canales físicos según el 3GGP mediante el lenguaje de Matlab y realizaremos las simulaciones pertinentes para obtener los resultados de la PAPR (Peak to Average Power Ratio), y además estudiaremos la probabilidad de error teniendo en cuenta el canal físico. En la segunda etapa, implementaremos dichos canales en un dispositivo hardware SFF SDR (Small Form Factor Software Definef Radio), donde se codificará en lenguaje VHDL todo lo que se hizo en Matlab. De este modo en la segunda etapa, podremos recoger una señal real, y podremos medir la PAPR en una situación más práctica y más realista, tanto en banda base como en radiofrecuencia. Y por último, realizaremos unas pequeñas comparaciones entre las dos etapas.

Tabla de contenido

ESTUDIO DE LA MODULACIÓN SC-FDMA E IMPLEMENTACIÓN EN UN DISPOSITIVO SFF SDR 1
EL TRIBUNAL 3
AGRADECIMIENTOS 5
RESUMEN 7
ÍNDICE DE ILUSTRACIONES 11
ÍNDICE DE TABLAS 13
1. Introducción. 15
1. Motivación. 15
2. Experiencias previas. 16
3. Objetivos. 16
4. Contenido del proyecto. 16
2. Análisis del estándar 3GGP. 19
1. Parámetros generales. 19
2. PUSCH. 21
3. PUCCH. 23
PUCCH1. 24
PUCCH2. 25
PUCCH3. 26
4. DRS. 26
DRS PUSCH. 28
DRS PUCCH. 29
5. SRS. 31
6. Mapeado en el dominio de la frecuencia. 31
7. Canal. 33
8. Receptor y BER. 35
3. Simulación en Matlab. 38
1. Simulación de la PAPR. 38
- Simulación 1. 38
- Simulación 2. 39
- Simulación 3. 40
- Simulación 4. 41
- Simulación 5. 42
- Simulación 6. 43
2. Simulación de la Probabilidad de error. 44

- Simulación 1. 44
- Simulación 2. 45
- Simulación 3. 46
- Simulación 4. 47
- Simulación 5. 48

4. Descripción técnica de bloques VHDL. 50
   1. Generador de bits aleatorios. 50
   2. Modulador de bits. 51
   3. Pre codificador. 54
   4. Generador de datos. 57
   5. Pucch. 59
   6. Generador de control. 62
   7. Mapeador. 63
   8. SC-FDMA 65
   9. Transmisor. 68
   10. Ampliaciones. 70
5. Entorno de trabajo y resultados. 72
   1. Entorno de trabajo. 72
   2. Caso ideal. 73
   3. Caso real 256 puntos. 78
   4. Caso real 256 puntos con SRS. 83
   5. Caso real 256 puntos con diferentes PUCCHs. 84
   6. Caso real 256 puntos con diferentes modulaciones. 86
   7. Caso real 2048 puntos. 88
6. Principales dificultades. 94
7. Conclusión y líneas de trabajo futuras. 96

Apéndice A: Algoritmo de la mediana. 101

Apéndice B: Resultados de la síntesis. 103

Apéndice C: Presupuesto 105

Acrónimos. 106

Referencias. 107

ÍNDICE DE ILUSTRACIONES

Ilustración 1: Comparativa OFDM vs SC-FDMA 15
Ilustración 2: Trama FDD 19
Ilustración 3: Trama TDD 19
Ilustración 4: Un slot 20
Ilustración 5: Diagrama de bloques 21
Ilustración 6: Ejemplo de ordenación: prefijo cíclico extendido y 2 bloques para la transmisión 23
Ilustración 7: Ejemplo de un slot para un usuario, datos y DRS 26
Ilustración 8: Datos y DRS 29
Ilustración 9: PUCCH1 y DRS 30
Ilustración 10: PUCCH2 y DRS 30
Ilustración 11: PUCCH3 y DRS 30
Ilustración 12: Mapeado de los datos en la frecuencia 32
Ilustración 13: Interpolado h(t) 33
Ilustración 14: Frecuencia de muestreo 33
Ilustración 15: 2 canales 2 usuarios 34
Ilustración 16: 1 canal N usuarios 35
Ilustración 17: Simulación diferentes PUCCHs 38
Ilustración 18: Simulación del caso ideal 39
Ilustración 19: Simulación para diferentes BW 40
Ilustración 20: Simulación datos con SRS 41
Ilustración 21: Simulación para diferentes modulaciones 42
Ilustración 22: Simulación para diferentes longitudes de CP 43
Ilustración 23: Estimación del canal en frecuencia 44
Ilustración 24: Constelación QPSK txón y rxón 44
Ilustración 25: Simulación para dos BW extremos 45
Ilustración 26: Simulación para los tres canales 46
Ilustración 27: Simulación para diferentes modulaciones 47
Ilustración 28: Simulación para diferentes prefijos cíclicos 48
Ilustración 29: Generador de bits aleatorios 50
Ilustración 30: ModelSim Generador de bits aleatorios 51
Ilustración 31: Modulador de símbolos complejos 51
Ilustración 32: ModelSim Modulador de símbolos complejos 53
Ilustración 33: ModelSim Modulador visión global 53
Ilustración 34: Pre codificador 54
Ilustración 35: ModelSim FFT carga y cálculo 56
Ilustración 36: ModelSim FFT descarga y almacenamiento 56
Ilustración 37: ModelSim Pre codificador visión global 57
Ilustración 38: Generador de datos 58
Ilustración 39: Pucch1, Pucch2 y Pucch3 59
Ilustración 40: ModelSim Pucch1 generación de un simbolo SC-FDMA 61
Ilustración 41: ModelSim Pucch1 visión global 62
Ilustración 42: Generador de control 63
Ilustración 43: Mapeador en frecuencia de los datos 64
Ilustración 44: Modulación SC-FDMA 65
Ilustración 45: ModelSim Multiplexación y carga de datos 66
Ilustración 46: ModelSim SC-FDMA visión general 67

Ilustración 47: ModelSim simulación global 67
Ilustración 48: Diagrama de bloques 68
Ilustración 49: ModelSim Transmisor 68
Ilustración 50: Transmisor 69
Ilustración 51: Lyrtech SFF SDR 72
Ilustración 52: Diagrama de bloques del caso ideal 73
Ilustración 53: Un slot en banda base 73
Ilustración 54: Varios slots en banda base 74
Ilustración 55: Ancho de banda en banda base 74
Ilustración 56: Un slot en radiofrecuencia 75
Ilustración 57: Varios slots en radiofrecuencia 76
Ilustración 58: Ancho de banda en radiofrecuencia 76
Ilustración 59: Ancho de banda en radiofrecuencia ampliado 77
Ilustración 60: Comparativa caso ideal de Matlab vs caso ideal implementado 78
Ilustración 61: Un slot en banda base 79
Ilustración 62: Varios slots en banda base 80
Ilustración 63: Ancho de banda en banda base 80
Ilustración 64: Un slot en radiofrecuencia 81
Ilustración 65: Varios slots en radiofrecuencia 81
Ilustración 66: Ancho de banda en radiofrecuencia 82
Ilustración 67: Ancho de banda en radiofrecuencia ampliado 82
Ilustración 68: Un slot SRS 83
Ilustración 69: Comparativa SRS 84
Ilustración 70: Un slot PUCCH2 en banda base 85
Ilustración 71: Un slot PUCCH3 en banda base 85
Ilustración 72: Comparativa diferentes PUCCH 86
Ilustración 73: Un slot 16QAM 87
Ilustración 74: Un slot 64QAM 87
Ilustración 75: Comparativa modulaciones 88
Ilustración 76: Un slot en banda base 89
Ilustración 77: Varios slots en banda base 89
Ilustración 78: Ancho de banda en banda base 90
Ilustración 79: Un slot en radiofrecuencia 90
Ilustración 80: Varios slots en radiofrecuencia 91
Ilustración 81: Ancho de banda en radiofrecuencia 91
Ilustración 82: Comparativa anchos de banda 92
Ilustración 83: PAPR de OFDM [8], N=256 K=64portadoras 96
Ilustración 84: Comparativa OFDM vs SC-FDMA [8] 97
Ilustración 85: BPSK: comparación media y mediana 101
Ilustración 86: Pulso normal y pulso DAC 102

ÍNDICE DE TABLAS

Tabla 1: Comparativa BW....20
Tabla 2: Prefijos cíclicos....21
Tabla 3: Modulaciones....22
Tabla 4: Resumen PUCCH....24
Tabla 5: Modulación PUCCH1....24
Tabla 6: Resumen de los RB....32
Tabla 7: Operaciones sustitutivas de la multiplicación....60
Tabla 8: Comparativa del caso ideal....98
Tabla 9: Comparativa del caso real....98
Tabla 10: Resultado síntesis 256 puntos + PUCCH3....103
Tabla 11: Resultado síntesis 2048 puntos....104
Tabla 12: Costes de personal....105
Tabla 13: Costes de material....105
Tabla 14: Costes totales....105

1. Introducción.

1. Motivación.

El mundo está cambiando, las comunicaciones móviles están revolucionado el mundo de tal manera que ahora no podemos vivir separados de nuestro dispositivo móvil. Vivimos en una sociedad en la cual siempre tenemos que estar conectados y siempre comunicados, hasta el punto que invertimos todo nuestro tiempo libre en las comunicaciones.

De aquí surge una de las razones, entre tantas, para seguir innovando e investigando en nuevas tecnologías para la comunicación inalámbrica. En nuestro caso, se va a abordar el diseño del sistema de cuarta generación: LTE. Este nuevo sistema de telecomunicación se caracteriza principalmente en alcanzar una velocidad máxima de bajada de 100Mbps, para ello emplea un sistema de modulación multi portadora, comúnmente conocido como OFDM, y en el uplink se usa la nueva modulación SC-FDMA, que se estudiará más a fondo en el presente proyecto.

La nueva modulación SC-FDMA (Single Carrier-FDMA), por sí sola es una gran novedad, ya que la industria de las telecomunicaciones no suele tomarse la molestia en inventarse una nueva modulación. Hasta hoy, casi todas las técnicas en comunicaciones civiles son heredadas del mundo militar. Sin embargo, esta vez se han inventado una nueva modulación, con ciertas semejanzas a OFDM (Orthogonal Frequency Division Multiple Access), que tiene como objetivo principal la reducción de la PAPR, que es la relación que hay entre los picos de señal con respecto a la media. Como ya se sabe, la modulación OFDM presenta grandes y frecuentes picos, que reducen el rendimiento y la eficiencia de los amplificadores. Estos picos son directamente proporcionales a la cantidad de sub portadoras empleadas, por tanto con la modulación SC-FDMA se intenta eliminar estos picos, reduciendo su frecuencia de aparición y amplitud de la misma.

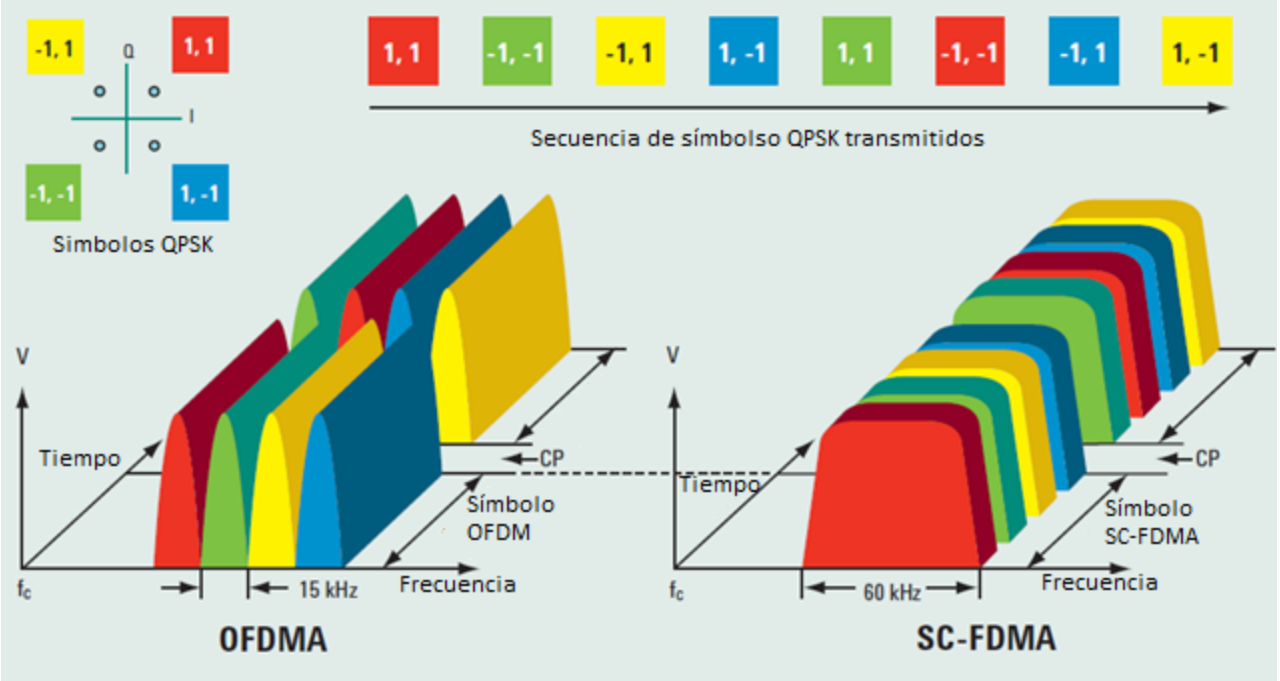


Ilustración 1: Comparativa OFDM vs SC-FDMA

El nombre de “Single Carrier” viene de transmitir un símbolo único en todas las sub portadoras (en paralelo), pero en un tiempo más reducido. Como podemos ver en la ilustración 1, en OFDM se transmite un símbolo en cada sub portadora durante un periodo de símbolo completo. Sin embargo en SC-FDMA, estamos transmitiendo un único símbolo en todas las sub portadoras disponibles, en un mini periodo de símbolo, con esto se consigue una redundancia en la frecuencia.

Para realizar dicho estudio, se va a tomar como referencia el estándar documento 3GGP TS 36.211 V10.3.0 (2011-09), implementándolo en Matlab, para realizar una simulación a priori de comportamiento de la misma, caracterizando la PAPR y la probabilidad de error de bit. En un segundo paso, se va a implementar en un dispositivo SFF SDR el transmisor completo, para poder medir la señal en banda base y en radiofrecuencia, de este modo podremos realizar una comparación entre el caso ideal y el caso real.

2. Experiencias previas.

El Departamento de Teoría de la Señal y Comunicaciones de la Universidad Carlos III de Madrid, adquirió un dispositivo SFF SDR de la marca Lyrtech. En dicho dispositivo se ha realizaron pruebas para testear su funcionalidad y capacidad. Además, se ha llevado a cabo en VHDL un transmisor y un receptor OFDM en la dicha plataforma.

3. Objetivos.

El primer objetivo consiste en realizar una serie de simulaciones en Matlab, caracterizando la PAPR para los diferentes parámetros que nos brinda el sistema: CP, pilotos, modulaciones, PUCCH, etc. Para ello se va a tomar el estándar 3GGP TS 36.211 V10.3.0 (2011-09), y se va a implementar paso por paso todo lo que se indique en dicho documento, pasándolo a un lenguaje propio de Matlab.

El segundo objetivo consiste en implementar todo lo anterior, en un dispositivo SFF SDR, para ello se va a realizar en el lenguaje VHDL. Se va a tomar todas las configuraciones, archivos heredados de las experiencias previas y ejemplos de Lyrtech, para intentar realizar dicha tarea en el menor tiempo posible. De este modo, podemos medir en un caso real la señal en banda base y en radiofrecuencia, para posteriormente tratarla en Matlab y volver a caracterizar la PAPR. Así podremos realizar una comparación exhaustiva entre el caso real y el caso ideal.

4. Contenido del proyecto.

Capítulo 1: Se realizará una breve introducción de la nueva modulación SC-FDMA, y sus comparativas con la modulación OFDM.

Capítulo 2: Se analizará el estándar 3GGP TS 36.211 V10.3.0 (2011-09), explicando paso a paso cuáles son los pasos para realizar la modulación del sistema SC-FDMA.

Capítulo 3: Se presentará el resultado de las simulaciones en Matlab. Se realizará una comparación de la PAPR y la probabilidad de error de bit, para diferentes parámetros de configuración del sistema.

Capítulo 4: Se describirá cada componente VHDL, sus interfaces, sus funcionalidades, etc. Además se presentarán los resultados de las simulaciones en ModelSim.

Capítulo 5: Se describirá en un primer lugar el entorno de trabajo, los equipos empleados en el desarrollo. Y posteriormente se presentará el resultado de la PAPR para el caso real, medido con un osciloscopio, en banda base y en radiofrecuencia, realizándose una comparación exhaustiva con el caso ideal.

Capítulo 6: Se describirá las principales dificultades surgidas durante el desarrollo de la misma.

Capítulo 7: Se intentará encaminar un poco cómo deben progresar las investigaciones y desarrollos futuros, y se dará un pequeño resumen y conclusión de todo el proyecto.

2. Análisis del estándar 3GGP.

1. Parámetros generales.

Ya existen algunos dispositivos con LTE, aunque es posible que no cumplan totalmente con el estándar 3GGP, ya que aún existen una gran cantidad de parámetros de diseño e implementación que no están determinados cuantitativamente, y es por tanto quizás es un elemento adicional de dificultad principal de este trabajo.

La trama básica dura 10ms, está constituida por 10 subtramas de 1ms cada una, y a su vez cada subtrama está formada por 2 slots de 0.5ms cada una. Existen dos tipos de acceso, tanto la FDD como TDD, que corresponden con las dos ilustraciones que hay a continuación.

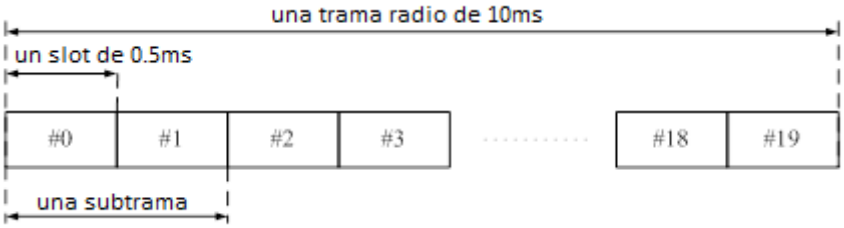


Ilustración 2: Trama FDD

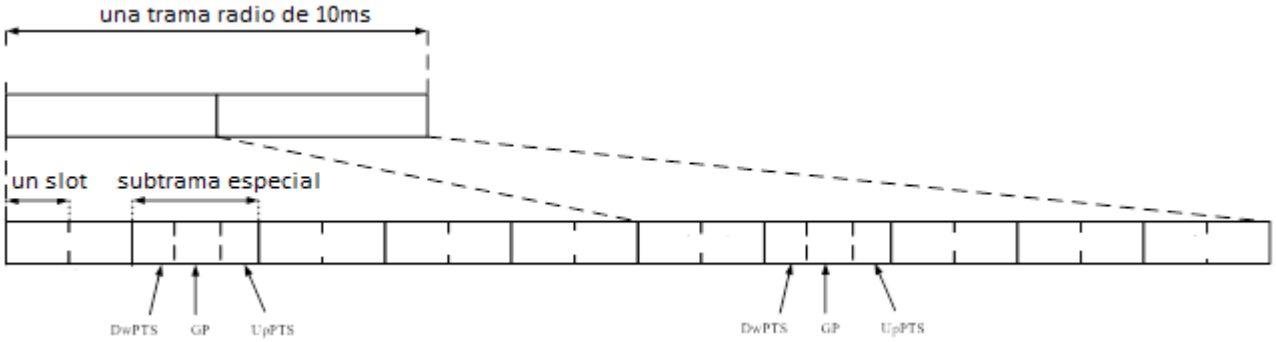


Ilustración 3: Trama TDD

En FDD se envía los 20 slots en la única portadora, sin embargo en TDD es más complejo, posee una mayor libertad de configuración, se puede destinar parte de los slots para subida y el restante para bajada.

Desde nuestro punto de vista, cómo sólo se va a implementar los canales PUSCH (Phisical Uplink Shared Channel) y PUCCH (Phisical Uplink Control Channel), y como trabajamos en banda base, no nos importa si es FDD como TDD. Para nosotros M usuarios transmiten constantemente información, y suponemos que ya se han sincronizado, por lo que nuestro sistema se parece más a FDD que a TDD.

Ahora si hacemos zoom en un slot, que es la unidad básica de la trama, se puede descomponer a su vez en N símbolos SC-FDMA, que cada uno de ellos están formadas por L símbolos mapeados en la frecuencia (véase la ilustración 4). Tenemos que prestar toda nuestra atención en la frecuencia, ya que es donde irán mapeado toda la información.

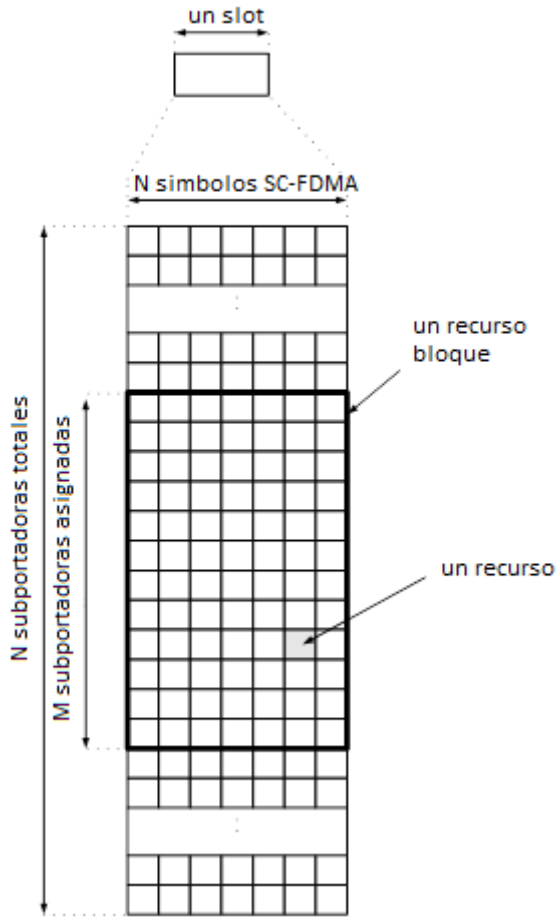


Ilustración 4: Un slot

Sin embargo, diversas fuentes y fabricantes reducen en unos valores concretos para el ancho de banda que se va a emplear, véase en la tabla 1. Dicho valores son todos los anchos de banda propuesto por algunos fabricantes. Se puede calcular el ancho de banda útil del sistema multiplicando el número de sub portadoras por la separación entre sub portadoras. Una vez que tenemos el ancho de banda total y el útil, podemos obtener su eficiencia espectral.

BW_total	3MHz	5MHz	10MHz	15MHz	20MHz
$N_{UL\ RB}$	15 RBs	25 RBs	50 RBs	75 RBs	110 RBs
BW_útil	2.7MHz	4.5MHz	9MHz	13.5MHz	19.8MHz
Eficiencia espectral	90%	90%	90%	90%	99%

Tabla 1: Comparativa BW

Y por último, la cantidad de símbolos SC-FDMA en un slot, puede tomar dos valores diferentes, dependiendo si empleamos una redundancia cíclica normal o extendida, véase en la tabla 2. Es evidente que redundancia extendida nos brinda una mayor protección ante los canales altamente dispersivos, ya que al poseer una mayor longitud de redundancia, evita interferir en el siguiente símbolo.

	Normal	Extendida
Número de símbolos por SLOT	7 símbolos	6 símbolos
Cantidad de prefijo cíclico por símbolo SC-FDMA	1er símbolo: 160 símbolos Resto: 144 símbolos	512 símbolos

Tabla 2: Prefijos cíclicos

2. *PUSCH.*

El canal más importante sin duda es el de datos PUSCH (Physical Uplink Shared Channel), ya que es el canal en la cual los usuarios transmiten sus datos de subida. Como se aprecia en la figura 5.3-1 del estándar [1], existe un conjunto de operaciones que hay que realizar antes de la transmisión, esto es debido a que la modulación SC-FDMA dispone de una gran cantidad de operaciones previas como OFDM, pero con ciertas diferencias significativas que se detallarán a continuación.

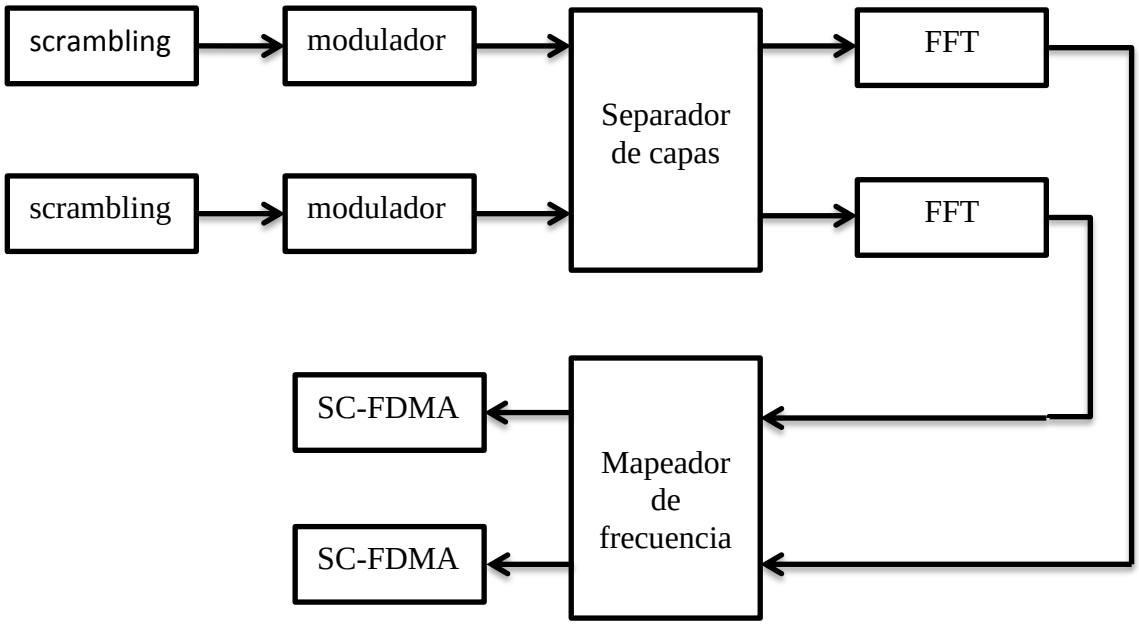


Ilustración 5: Diagrama de bloques

La primera operación es Scrambling, una aleatorización de bits, a través de unos códigos Gold de longitud 31, a la que se suman a los bits a transmitir. La generación de esta secuencia aleatoria a la cual se debe sumar consta de los siguientes pasos:

- Partimos de un número inicial, que depende de unos identificadores que nos pasan de las capas superiores.

$$c_{init} = c_{rnti} * 2^{14} + q * 2^{13} + \left\lfloor \frac{n_s}{2} \right\rfloor * 2^9 + N_{ID}^{cell}$$

  - C_{rnti}: es un identificador temporal de la red radio, posee un rango, y se escoge aleatoriamente uno.

$$0x003D \leq c_{rnti} \leq 0xFFF3$$

- N_s: es el número de slot en la que nos encontramos.

$$0 \leq n_s \leq 19$$

- $N_{cell\ ID}$: es un identificador de la celda. Existen 504 identificadores formado por 3 grupos de 168, y se calcula de la siguiente manera.

$$N_{ID}^{cell} = 3 * N_{1ID} + N_{2ID}, \qquad 0 \leq N_{1ID} \leq 167, \qquad 0 \leq N_{2ID} \leq 2$$

- Una vez que hemos obtenido el número inicial, lo pasamos a binario de 31 bits, y se lo asignamos a x2. Por otro lado nos reservamos otra secuencia de ceros de la misma longitud de x2, y lo llamamos x1.
- Casi siempre, la longitud de nuestros bits a transmitir supera a 31, por tanto debemos aumentar de tamaño estas dos secuencias del siguiente modo.

$$x_1(n+31) = \big(x_1(n+3) + x_1(n)\big) mod 2$$
$$x_2(n+31) = \big(x_2(n+3) + x_2(n+2) + x_2(n+1) + x_2(n)\big) mod 2$$

- Una vez que tenemos estas dos secuencias de una longitud suficiente, generamos la secuencia del código aleatorio a la cual se le debe sumar a nuestros bits.

$$c(n) = \big(x_1(n+16) + x_2(n+16)\big) mod 2$$

Tras aleatorizar los bits, tenemos que modular estos bits por símbolos. En el estándar se soporta hasta tres tipos de modulaciones diferentes que se detalla a continuación. Para entrar en más detalle, véase en la sección 7.1 del estándar [1].

Modulación:	**QPSK**	**16QAM**	**64QAM**

Tabla 3: Modulaciones

Después de la modulación, tenemos un mapeado de capas, este bloque no es más que repartir los símbolos de manera equitativa entre el número de antenas disponibles para la transmisión. El estándar soporta sistemas MIMO, en la cual se puede disponer más de una antena para transmitir y recibir, de este modo estamos creando una redundancia espacial, reduciendo el error y aumentando la velocidad.

En nuestra simulación, se va a simular que nuestro terminal dispone sólo de una antena, ya que por un lado se quiere evitar en primer lugar la complejidad matemática de los sistemas MIMO, y por otro lado es ser realista, ya que en la realidad no existe todavía ningún terminal con dos o tres antenas. Además aumentar el número de antenas implica un aumento de consumo energético, y haría que los terminales perdiesen aún más su autonomía.

Antes de mapear los símbolos en frecuencia para su transmisión se debe hacer antes una DFT, de este modo podemos mapear posteriormente dichos símbolos, ya que es mucho más fácil realizar esta operación en el dominio de la frecuencia que en el tiempo. En el momento de la realización

de esta DFT aprovechamos para ordenar un poco los símbolos, dejando el tamaño de este bloque de datos de la siguiente manera:

- Como un slot caben N símbolos SC-FDMA, tendremos que realizar N veces la operación de la DFT para obtener una columna de símbolos modulados por cada símbolo SC-FDMA.
- En la frecuencia, nos dan la cantidad de bloques de recursos concedidos para la transmisión M, y sabemos que por cada bloque se dispone de 12 portadoras, por tanto podemos mapear 12 símbolos por bloque (un símbolos SC-FDMA). Por tanto la longitud de la DFT es 12*M.

La conclusión de todo lo anterior, consiste en coger un array lineal de símbolos, realizarle la DFT, y posteriormente ordenarlo en un array de dos dimensiones cuyo ancho posee N símbolos SC-FDMA y cuya longitud posee 12xM, para ver con más claridad, ver la ilustración 4.

Una vez que hemos generado los datos, necesitamos generar las señales para la estima de canal para poder demodular los datos, que posteriormente se mapean intercalados los datos y DRS (Demodulation Reference Signal) que son señales pilotos para ayudar en la estimación del canal eliminando los efectos de la misma en el receptor. Por tanto la cantidad de columnas para datos siempre es inferior al máximo disponible. La generación de DRS y el mapeado físico se explicará más tarde.

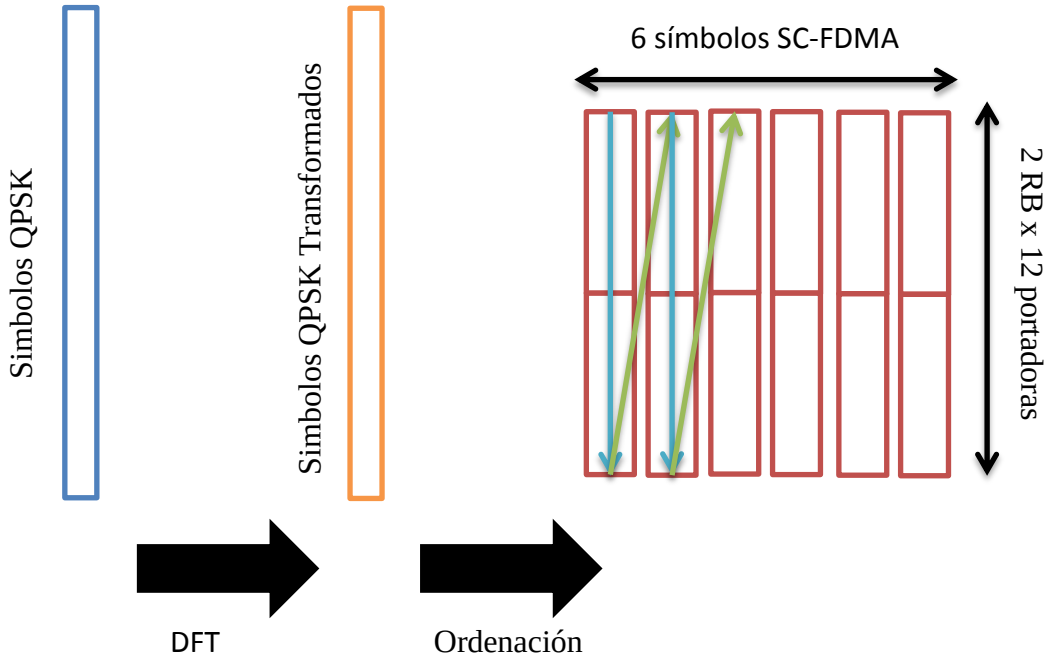


Ilustración 6: Ejemplo de ordenación: prefijo cíclico extendido y 2 bloques para la transmisión

3. PUCCH.

El canal de señalización más importante es el PUCCH (Physical Uplink Control Channel), en la cual el terminal móvil puede comunicarse con la estación base. Existe hasta tres tipos, véase en la tabla 4, cada uno de ellos presenta unas características diferentes, en que se detalla a continuación.

El formato 1 se encarga exclusivamente en asentir la información recibida de la estación base, y el formato 2 se le añade la funcionalidad para enviar información del canal MIMO. Y por último, el

formato 3 sirve para lo mismo que las anteriores, o bien añadir más datos de control, o alargar y dar mayor protección a la información de señalización enviada.

A cada usuario se le concede exclusivamente un único bloque de recursos, o sea 12 portadoras y N símbolos SC-FDMA. Además cada vez que se genera la trama de PUCCH se genera por cada sub trama, o sea por cada dos slots, visto de otra manera, la información generada se divide entre dos slots para ser transmitida.

Formato	Modulación	Núm de Bits	Información
1	N/A	N/A	Petición de recursos
1a	BPSK	1	ACK/NACK
1b	QPSK	2	
2	QPSK	20	CQI
2a	QPSK+BPSK	21	CQI+ACK/NACK
2b	QPSK+QPSK	22	
3	QPSK	48	Versión extendida

Tabla 4: Resumen PUCCH

PUCCH1.

Como ya se ha explicado, en el PUCCH de tipo 1, sólo se envía 1 o 2 bits como mucho por tanto debemos modular estos bits bajo un esquema QPSK, pero un QPSK diferente al PUSCH.

Formato	Bits	d(0)
1a	0	1
	1	-1
1b	00	1
	01	-j
	10	J
	11	-1

Tabla 5: Modulación PUCCH1

Como ya se comentó anteriormente, disponemos únicamente de un bloque de recurso para transmitir un único símbolo, por tanto debemos replicar esta información. En primer lugar debemos replicar estos símbolos en la frecuencia, 12 veces, uno por cada portadora del bloque recurso, de la siguiente manera:

$$y(m) = d(0) * r_{u,v}^{\alpha}(m), \qquad 0 \leq m \leq 11, \qquad 0 \leq \alpha \leq 2\pi$$

Nótese que estamos multiplicando nuestro símbolos por un vector de longitud 12, este vector es un vector base que se explicará en la sección de DRS. Sin embargo el único parámetro que debemos proporcionar es el ángulo α, el valor debe calcularse a partir de las capas superiores, en nuestra simulación hemos generado aleatoriamente dicho ángulo.

Una vez que ya tenemos 12 símbolos, debemos replicarlo a lo ancho, N símbolos SC-FDMA. A igual que en PUSCH, debemos transmitir señales DRS, por tanto la cantidad de columnas generadas debe ser siempre inferior a la de un bloque de recursos. Para la replicar esta información se realiza de la siguiente manera.

$$z(m,n) = S * w_{noc}(n) * y(m), \qquad 0 \leq n \leq 3 \text{ ó } 4$$

- Y: son los 12 símbolos replicados de antes.
- S: es un número complejo que puede ser 1 ó j, que nos sirve para aleatorizar aún más los símbolos. Su selección se va a realizar de manera aleatoria, aunque está determinado por capas superiores.
- W_{noc}: son unos códigos ortogonales cuya longitud es 3 ó 4 en función de las necesidades. Y los estos códigos están detallados en el estándar [1] en las tablas 5.4.1-2 y 5.4.1-3.
- Z: es la salida, los símbolos replicados en frecuencia y tiempo, listos para mezclar con su DRS y ser mapeados.

Nótese que debemos multiplicar nuestro vector de longitud 12 por cada elemento de código ortogonal, de este modo podemos obtener 4 vectores de longitud 12. Esta operación hay que realizar 2 veces, uno por cada slot. Si tenemos información de SRS, en el segundo slot debemos utilizar uno de longitud 3 en vez de 4, así podemos dejar una columna libre para transmitir la señal SRS.

PUCCH2.

En la trama PUCCH del tipo 2, antes de modular los bits, debemos realizar una aleatorización de la misma ya que se envía hasta 22 bits, para ello se vuelve a emplear el código Gold de longitud 31 que ya se ha descrito anteriormente. Lo único que cambia es su secuencia inicial.

$$c_{init} = \left(\left\lfloor \frac{n_s}{2} \right\rfloor + 1\right)\left(2N_{ID}^{cell} + 1\right)2^{16} + c_{rnti}$$

Una vez aleatorizado los bits, se modulan los primeros 20 bits mediante un esquema QPSK, y los bits restantes se modulan de la misma manera que el PUCCH del tipo 1 y se obtiene en total 11 símbolos. Por tanto a diferencia de PUCCH tipo 1, aquí sólo tenemos que replicar los símbolos en la frecuencia, ya que no es necesario volver a replicarlo en el tiempo.

Para replicarlo en la frecuencia, se emplea otra vez el vector base, cuya generación de la misma y el ángulo se realiza de la misma manera que antes.

$$z(m,n) = d(m) * r_{u,v}^{\alpha}(n), \qquad 0 \leq m \leq 11, \qquad 0 \leq n \leq 9$$

Nótese que estamos construyendo directamente un array de dos dimensiones, cuya largo son las 12 portadoras, y de ancho son los 10 símbolos. Este array de 10 columnas se dividirá entre 2, una parte para cada slot, y los lugares que no se han rellenado serán posteriormente utilizados para DRS, que se genera con la ayuda el undécimo símbolo que ha sobrado.

Además, hay que añadir, que el formato 2 no soporta un prefijo cíclico extendido, por tanto en nuestra simulación, si elegimos un prefijo cíclico extendido, se elimina el PUCCH de formato 2.

PUCCH3.

El último tipo de PUCCH mezcla un poco de los dos primeros. En primer se realiza una aleatorización de bits como en el PUCCH de tipo 2, ya que se va a transmitir hasta 48 bits, se emplea un esquema QPSK que genera 12 símbolos.

Una vez que tenemos los 12 símbolos, no es necesaria una réplica en frecuencia, porque tenemos 12 símbolos para 12 sub portadoras, y sólo se replica en el tiempo mediante códigos de ortogonalización detallados en la tabla 5.4.2A-1 del estándar [1]. Cabe mencionar que el procedimiento para la multiplicación y manipulación es casi la misma que la de sus hermanos, sin embargo existe una pequeña diferencia en que consiste en que los códigos son de longitudes diferentes, y que de las 12 portadoras, la mitad de ellas usando un código y la otra mitad usa otras.

La única gran novedad en el PUCCH de tipo 3 con respeto a sus hermanos, es que se realiza una DFT de sus símbolos antes de ser mapeados, al igual que PUSCH. Esto puede ser debido para reducir su PAPR, ya que al disponer de tantos bits, puede aumentar su PAPR.

4. DRS.

Como ya ha comentado en apartados anteriores, la información generada por los canales PUSCH y PUCCH debe intercalarse con las señales DRS, que nos ayudará en la demodulación. Como podemos apreciar, esta información se genera a lo largo de las frecuencias (columnas), por tanto se envía dicha información en varios símbolos SC-FDMA y se replica en todas las frecuencias.

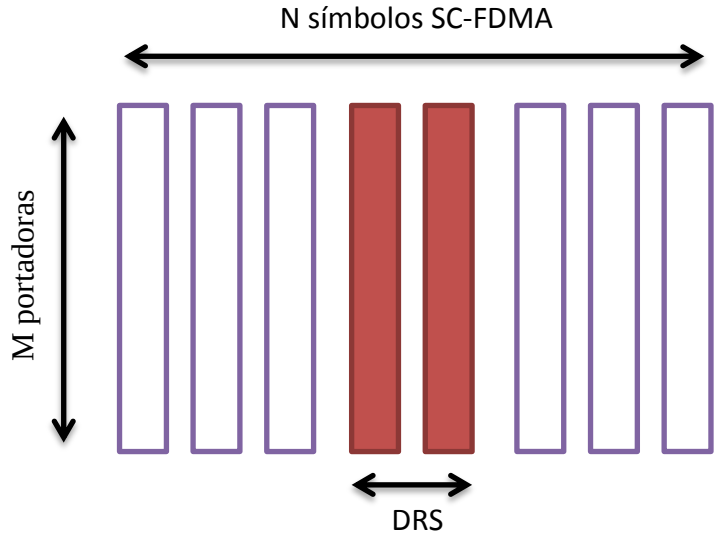


Ilustración 7: Ejemplo de un slot para un usuario, datos y DRS

Como se aprecia en la ilustración 7, es un ejemplo, se transmite la señal DRS en ciertas posiciones, en unos símbolos SC-FDMA a lo largo de todas las frecuencias. Por tanto como se transmiten secuencias conocidas, en el receptor se puede estimar el canal para eliminar y contrarrestar los efectos del canal mezclados con los datos. Por otro lado, se transmiten varias secuencias DRS de manera contigua para reducir los efectos del ruido promediando todas las secuencias.

Para estimar el canal, se opera de la siguiente manera:

$$R_i(jw) = H(jw)T(jw) + N(jw), \qquad H^*(jw) = \frac{R(jw)}{T(jw)}$$

- T: es la secuencia DRS transmitida conocida por el receptor.
- H: es el canal equivalente.
- N: es el ruido gaussiano aditivo.
- R: es la señal recibida por la estación base, donde el coeficiente i es el número de secuencias enviadas de manera contigua.

Por tanto para estimar el canal, se divide la secuencia recibida promediada entre la secuencia conocida.

Para la generación de DRS y para información PUCCH, se necesita en primer lugar generar una secuencia base. La generación de dicha secuencia depende de la cantidad de portadoras empleadas para la transmisión, si es mayor o menor que 3 bloques de recursos.
Sea M, la cantidad de bloques de recursos. Por tanto si sólo empleamos un o dos bloques de recurso, la secuencia base es simplemente un número complejo cuya fase viene dada por la secuencia $\varphi_u(n)$ que está recogida en las tablas 5.5.1.2-1 y 5.5.1.2-2 del estándar [1].

$$longitud < 3 * RB, \qquad r_u(n) = \exp\left(j\varphi_u(n)\frac{\pi}{4}\right), \qquad 0 \le n \le longitud$$

Ahora bien, si la cantidad de recurso bloques es mayor o igual que 3, la secuencia se genera totalmente diferente. En primer lugar necesitamos calcular el número N_{ZC}^{RS} que es el número primo más grande y cercano a M, pero siempre sin superarle. Una vez obtenido este número, se introducen en las fórmulas y se obtiene la secuencia base.

$$M \ge 3 * RB$$
$$\bar{q} = N_{ZC}^{RS}\frac{(u+1)}{3}, \qquad q = \lfloor \bar{q} + 0.5 \rfloor + v(-1)^{\lfloor 2\bar{q} \rfloor}$$
$$x_q(m) = \exp\left(-j\frac{\pi q m(m+1)}{N_{ZC}^{RS}}\right), \qquad 0 \le m \le N_{ZC}^{RS}$$
$$r_{u,v}(n) = x_q(n \bmod N_{ZC}^{RS}), \qquad 0 \le m \le longitud - 1$$

Como ya se comentó anteriormente, la secuencia base $r_{u,v}(n)$ se le puede añadir un ángulo α para introducirle una rotación de fase.

$$r_{u,v}^{\alpha} = \exp(j\alpha n)\, r_{u,v}(n), \qquad 0 \le n \le M$$

Nótese que para obtener la secuencia base, se han empleado dos números, u y v. Donde u se le conoce como Group Hopping y v se le conoce como Sequence Hopping.

Para generar el número u, se obtiene sumando dos números comprendidos entre 0 y 29, proporcionado por capas superiores y parámetros del sistema.

$$u = (f_1 + f_2) mod 30, \qquad f_1 \epsilon [0, 29], \qquad f_2 \epsilon [0, 29]$$

Y para generar el número v, sólo se tiene en cuenta para secuencias mayores e iguales que 6 bloques de recursos. Dicho número es binario, ya que emplea el código de secuencia aleatoria, y cuyo resultado es un solo bit.

$$v = c(n_s) \epsilon [0,1]$$

Ahora bien, el procedimiento para la generación de la secuencia base es la misma, pero su uso y mapeo es totalmente diferente para cada situación. Por tanto tendremos que especificar para cada trama cómo se utiliza.

DRS PUSCH.

Con el procedimiento de generación de la secuencia base, somos capaces de generar un vector, que es una secuencia base, cuya longitud coindice con el número de portadoras empleadas para la transmisión. Por tanto la secuencia base es directamente una columna entera, véase el ejemplo de la ilustración 8, las dos columnas rojas corresponden con la DRS (Demodulation Reference Signal).

Ahora bien hemos generado una secuencia base de 12xM portadoras, siendo M la cantidad de recursos bloques; ahora tenemos que replicar esta información tantas veces como se necesite, en el ejemplo de la ilustración 7, tenemos dos columnas, por tanto a esta secuencia base habrá que realizarle una serie de manipulaciones para conseguir estas dos columnas.

Para el caso de la trama PUSCH, necesitamos en total 2 secuencias DRS para cada slot. Por tanto tenemos que replicar dicha secuencia de la siguiente manera.

$$r_{PUSCH}(n, m) = w(m) r_{u,v}^{\alpha}(n), \qquad 0 \leq m \leq 1, \qquad 0 \leq n \leq 12 * M$$
$$\alpha = 2\pi \frac{n_{cs}}{12}, \qquad n_{cs} \epsilon [0, 11], \qquad w = [1 \quad 1]$$

- α: ángulo de rotación que se le introduce a la secuencia base, como ya se ha dicho anteriormente, para cada situación se emplea un ángulo diferente.
- w: patrón de réplica, este patrón varía en función el número de capas, el número de antenas, como sólo tenemos una capa y una antena, empleamos siempre el mismo patrón. Para ver todos los patrones, véase la tabla 5.5.2.1.1.-1 del estándar [1].

Y por último destacar que estas dos columnas, se intercalan con los datos, y se sitúan siempre en el centro de la misma, para verlo más claridad ver la ilustración siguiente, donde los recuadros con color corresponden con secuencias DRS.

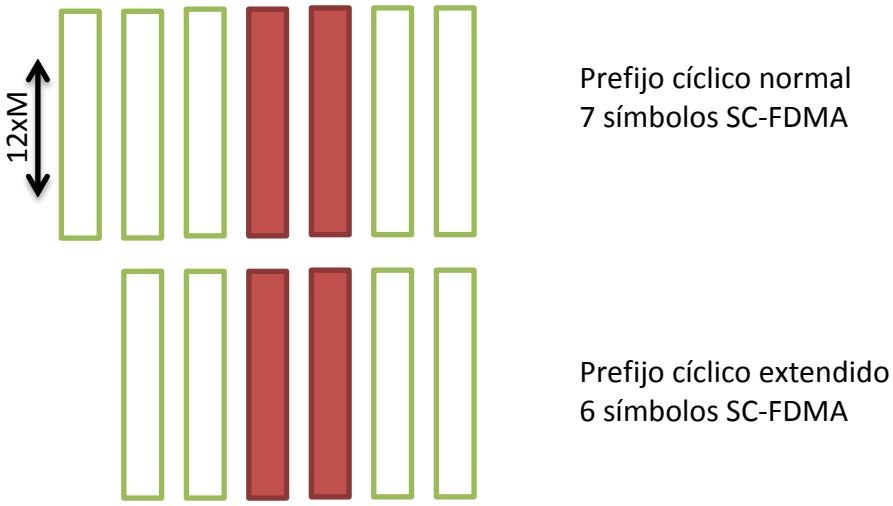


Ilustración 8: Datos y DRS

DRS PUCCH.

Para generar las secuencias de DRS para el PUCCH, se emplea la misma filosofía que para PUSCH, pero con pequeñas diferencias.

$$r_{PUSCH}(n,m) = w(m)r_{u,v}^{\alpha}(n)z, \qquad 0 \leq m \leq 1, \qquad 0 \leq n \leq 12 * M$$
$$\alpha = 2\pi\frac{n_{cs}}{M}, \qquad n_{cs}\epsilon[0,M],$$

- α: ángulo de rotación que se le introduce a la secuencia base, como ya se ha dicho anteriormente, para cada situación se emplea un ángulo diferente.
- w: patrón de réplica, este patrón depende principalmente de los parámetros del sistema, y del número de slot que nos encontremos, para más información ver las tablas 5.5.2.2.1-2, 5.5.2.2.1-3 del estándar [1].
- z: toma el valor 1 cuando estamos en el formado 1 y 3. Cuando estamos en el formato 2, como se tiene más de 20 bits, se generan más de 10 símbolos QPSK, por tanto se asigna a z el valor del último símbolo.

Para cada formato, el lugar donde reside la secuencia de DRS cambia, por tanto en la ilustración siguiente se detalla la posición que toma cada secuencia.

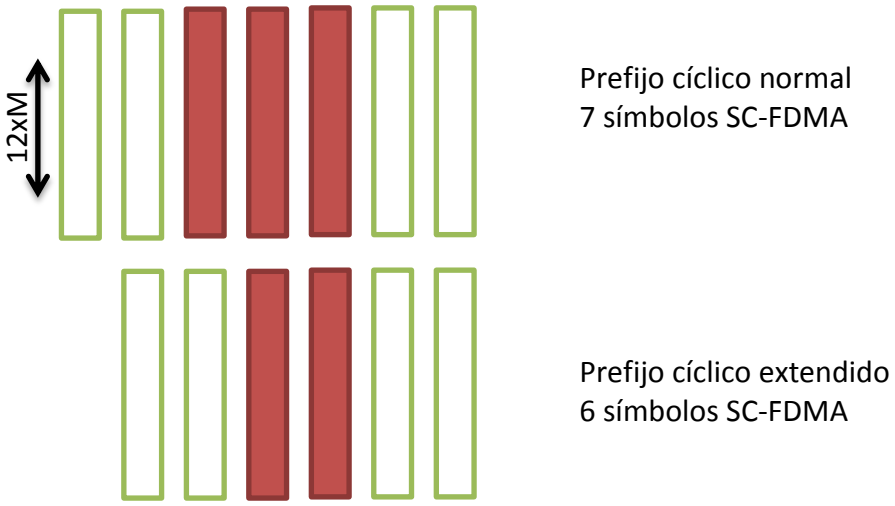


Ilustración 9: PUCCH1 y DRS

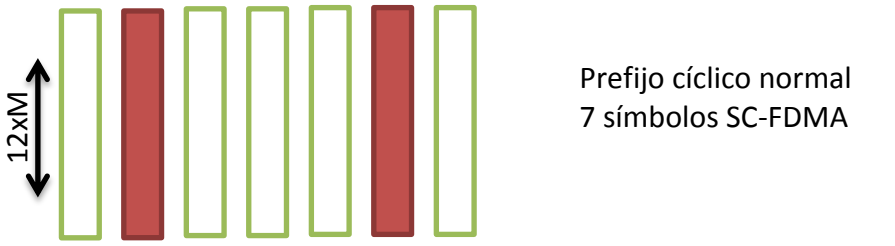


Ilustración 10: PUCCH2 y DRS

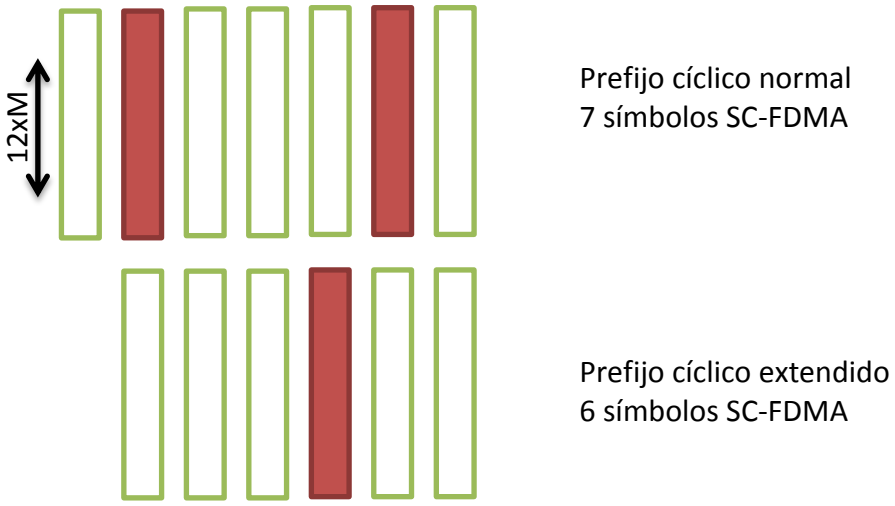


Ilustración 11: PUCCH3 y DRS

5. SRS.

Las señales de SRS (Sounding Reference Signal) son unas señales de piloto para realizar un reconocimiento del canal en frecuencia, de este modo la estación base puede conocer qué frecuencias poseen mayor rendimiento y cuáles no.

Estas señales cuando se quieren transmitir, entramos en la situación de "Shortened", en la cual, por cada 2 slots, el segundo se elimina un símbolo SC-FDMA de datos, para transmitir esta información.

Para generar dicha señal, debemos emplear otra vez la secuencia base, cuyo ángulo de rotación se genera de la siguiente manera:

$$\alpha = 2\pi \frac{n_{SRS}}{8}, \qquad n_{SRS} \epsilon [0, 8]$$

Para mapear esta secuencia en los 12xM portadoras disponibles, se realiza de manera alternada, una sub portadora con símbolos y otra sin símbolos (un cero). Es evidente que número de comienzo de la cuál queremos mapear, depende de una gran cantidad de información y parámetros. Nosotros lo hemos omitido por simplificación y directamente mapeados desde la primera posición.

6. Mapeado en el dominio de la frecuencia.

Partimos de la situación en que tenemos 2 slots, cada slot posee N símbolos SC-FDMA en el dominio del tiempo y 12xM portadoras en el dominio de la frecuencia. El mapeado de la información se va a hacer de 2 slots a la vez.

En primer lugar mapearemos el canal PUCCH, cada usuario dispone de un bloque de recurso para mapear su correspondiente PUCCH, y siempre se emplea los bloques de recursos que se disponen en los bordes, dejando el centro para los datos, y como existen dos bordes (el superior y el inferior) la mitad de los usuarios emplean un borde y la otra mitad emplean el otro borde, y alternándose para cada slot. Véase en la ilustración siguiente, tenemos 4 usuarios, para el primer slot, 2 transmiten abajo y dos transmiten arriba; en el siguiente slot se intercambian y así sucesivamente.

El espacio restante del medio, se emplea para transmitir los datos. A cada usuario se le asignan M bloques de recursos, si no está habilitado el salto de frecuencia, dichos M bloques de recursos se escogen equi espaciados para cada slot, de este modo estamos dando un redundancia en el dominio de la frecuencia. Sin embargo, si está habilitado el salto de frecuencia, se asignan los M bloques de recursos contiguos, y por cada slot realizan un salto, pero todos juntos.

Para realizar este salto de frecuencia, se emplean múltiples parámetros dados por capas superiores, en la simulación se va a emplear saltos aleatorios.

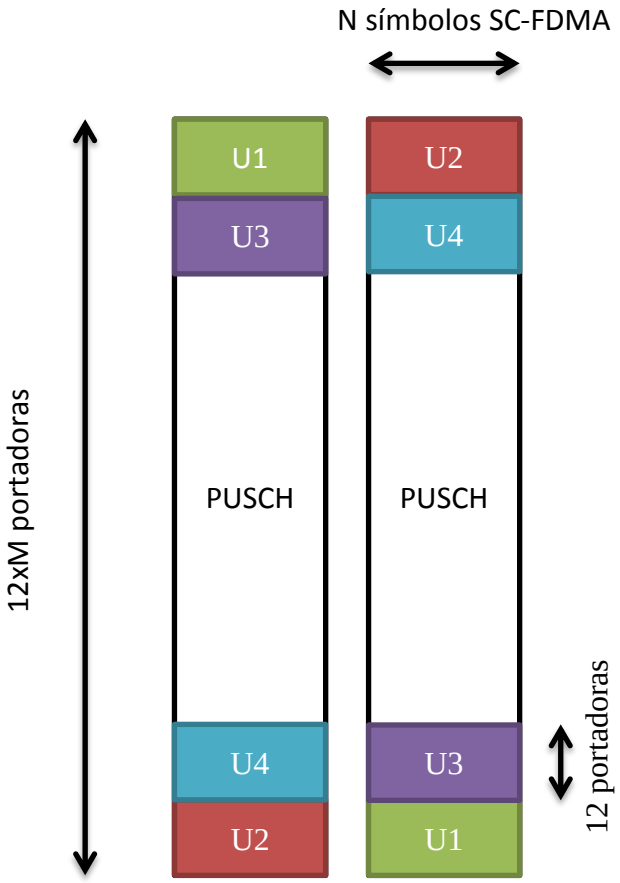


Ilustración 12: Mapeado de los datos en la frecuencia

Nótese, que el número de PUCCH no puede ser excesivamente elevado, ya que de este modo ocuparíamos muchas portadoras, y no habría espacio para transmitir los datos. Existen libros que determinan la cantidad máxima de bloques de recursos destinados para PUCCH, véase en la tabla inferior. Ahora bien el número de bloques es bastante subjetivo, ya que el estándar no comenta nada sobre el asunto, por tanto tomando como referencia de algunos libros, se van a escoger las cantidades mostradas a continuación.

BW_total	3MHz	5MHz	10MHz	15MHz	20MHz
$N_{UL\ RB}$	15 RBs	25 RBs	50 RBs	75 RBs	110 RBs
N_{PUCCH}	2 RBs	4 RBs	8 RBs	16 RBs	32 RBs

Tabla 6: Resumen de los RB

Además, el número de bloques no debe ser muy alto aunque se tengan un número elevado de usuarios, teniendo en cuenta que el formato 1 y 3 dispone de códigos de ortogonalización, por tanto se podría transmitir varios formatos en un mismo bloque de recursos, así reutilizando las cosas, se dispondrían aún más espacio para los datos.

Por último para obtener la trama final, se realiza una IDFT a cada símbolo SC-FDMA, dispondremos N tramas en el dominio del tiempo, a la cual se le debe concatenar el prefijo cíclico correspondiente a cada trama, y ordenarlo todo en un vector, así sucesivamente para cada slot.

7. Canal

Una vez que tenemos la trama en el transmisor, dicha trama debe atravesar un canal, y posteriormente tratado en el receptor para obtener los bits de información. Para realizar el canal, se ha seguido el anexo B de la referencia número 3 de este documento, que sugiere tres modelos de canal: EPA (Extended Pedestrian A model), EVA(Extended Vehicular A model) y ETU (Extended Typical Urban model); especificando para cada canal el patrón de retardo y su potencia.

Partiendo de estos tres modelos, se construye un vector que representa la respuesta del canal en tiempo discreto, formado por un conjunto de deltas. Posteriormente se interpola estas deltas para que cada posición del vector tome un valor no nulo, sino un valor promedio. La separación de cada muestra es de 1ns, esto es debido a que los patrones de retardo se especifican en nano segundos.

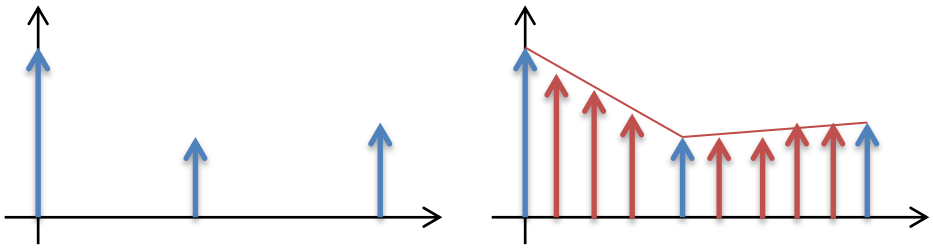


Ilustración 13: Interpolado h(t)

Una vez que tenemos la respuesta del canal interpolado para cada nano segundo, tendremos que diezmar la señal, muestreando a la tasa de ancho de banda del nuestro sistema. Esto es debido a que se trabaja en banda base complejo, por tanto la frecuencia de muestreo debe ser el doble que el ancho de banda en banda base, por tanto es justamente el ancho de banda que se emplea para el sistema de comunicaciones.

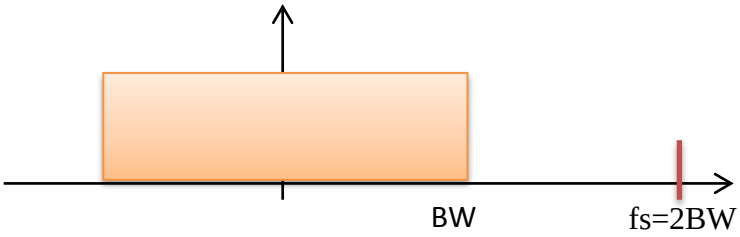


Ilustración 14: Frecuencia de muestreo

El factor de diezmado es la relación que existe entre la frecuencia de muestreo correspondiente a 1ns entre el ancho de banda del sistema.

$$N = \frac{1}{1ns * 2BW}$$

Por tanto si nuestro ancho de banda disminuye, el número de retardos es menor, esto tiene sentido ya que al tener menor ancho de banda, estamos siendo más selectivo en frecuencia, y por eso la respuesta del canal nos afecta menos, en el tiempo h(t) tendrá una menor longitud para anchos de banda bajos.

Sin embargo en un sistema real, existen un canal para cada usuario, ya que cada usuario está en un emplazamiento diferente, por tanto debemos construir un canal diferente para cada usuario. Para realizar este efecto, se va a multiplicar por una variable aleatoria a cada retardo de canal y para cada usuario, cuya media es nula, y desviación típica es la amplitud de cada retardo.

$$patrón:\ h = [h_1\ \ h_2\ \ \dots\ \ h_n]$$

$$a_{ij} \sim N\left(0, \frac{h_j}{2}\right) + jN\left(0, \frac{h_j}{2}\right), \qquad i = núm\ usuario, \qquad j = retardo\ del\ canal$$

$$H_i = [h_1 a_{i1}\ \ h_2 a_{i2}\ \ \dots\ \ h_n a_{in}], \qquad i = núm\ usuario$$

De este modo conseguimos que cada usuario tenga un canal cuya amplitud es totalmente aleatoria con respecto a otros, además la evolución de cada retraso también es aleatoria. Nótese que no se va a implementar la frecuencia doppler por simplificación.

Como el canal nos introduce una ganancia, se tendrá en cuenta en el cálculo de la varianza de ruido en función de la SNR especificada.

$$y = hx + n$$
$$n \sim N(0, \sigma^2), \qquad \sigma^2 = \frac{|h|^2}{SNR}$$

Una vez que cada usuario haya filtrado su señal por el canal y añadido su correspondiente ruido aleatorio, en el receptor hay que sumar todas las señales en el tiempo, ya que la señal cada usuario se superponen una con otras. Así en el receptor sólo se recibe únicamente una trama, pero esto no es un problema, ya que en el dominio de la frecuencia son totalmente separables, porque cada uno transmite en sus portadoras correspondientes.

Sin embargo, en la práctica, o desde el punto de vista de la simulación, no es necesario generar tantos usuarios ni tantos canales. Por un lado, si generamos n canales para n usuarios, cada canal es diferente uno de otros. Además de cada canal se genera para todo el ancho de banda del sistema, pero cada usuario sólo ocupa un trozo de la frecuencia, y por tanto sólo emplea un trozo del canal. Y por tanto cada trozo de canal empleado por cada usuario sigue siendo diferente uno de otros.

Como se aprecia en la ilustración 15, tenemos dos canales con dos usuarios, el usuario 1 usa frecuencias bajas, y el usuario 2 usa frecuencias altas, la respuesta de canal que usan son diferentes entre ellos.

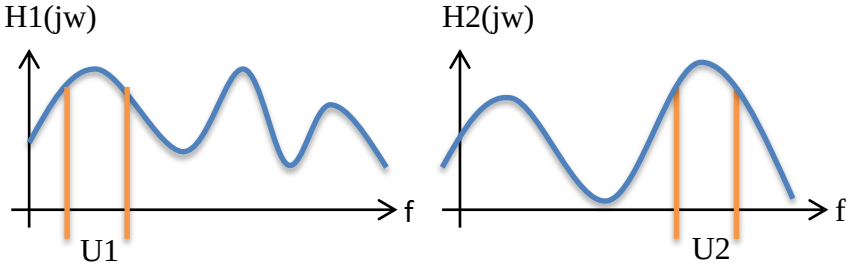


Ilustración 15: 2 canales 2 usuarios

Ahora bien, si sólo generamos un canal aleatoriamente, que no es constante en el dominio de la frecuencia, y como cada usuario sólo usa un trozo de ese canal, por tanto hemos conseguido las mismas condiciones que antes. Véase en la ilustración 16, tenemos un canal no constante en la frecuencia, y vemos que la respuesta equivalente de cada usuario es diferente uno de otros usando exclusivamente un único canal.

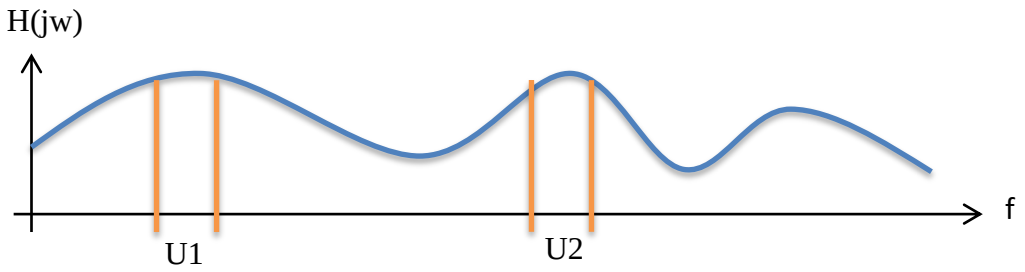


Ilustración 16: 1 canal N usuarios

La razón de usar esta simplificación, radica en la eficiencia de la simulación. Sabemos que Matlab es muy lento, por tanto todas las mejoras que hagamos aumentará en la eficiencia temporal de la simulación. Por tanto sólo generamos un canal, y sólo tenemos un usuario, o mejor dicho tenemos un súper usuario que mapea los datos, usurpando a los N usuarios. De este modo eliminamos bastantes bucles, operaciones matriciales, que agilizarán la simulación.

8. *Receptor y BER.*

Como ya se ha comentado anteriormente, en este proyecto vamos a simular en primer lugar la PAPR y posteriormente la BER (Bit Error Ratio) del sistema completo.

Persiguiendo los mismos objetivos que el punto anterior, debemos optimizar la simulación, reduciendo el tiempo empleado. Para ello se han realizado una serie de modificaciones al código original, siempre que no afecte a la BER.

- Se genera los símbolos QPSK / 16QAM / 64QAM aleatoriamente. Se va a eliminar el código de aleatorización "Gold" de longitud 31. Esto se emplea para aleatorizar los bits y evitar cadenas largas de todo unos o todo ceros, en nuestro caso se solventa con el método de aleatorización de Matlab que nos dan símbolos alternados.
- No se va a implementar el canal PUCCH, ya que poseen una gran protección a través de la redundancia en frecuencia y tiempo. Para obtener una probabilidad de error dado, necesitaríamos realizar tantas iteraciones que sería demasiado elevado.
- Sólo se va a implementar el canal PUSCH y se va a rellenar toda el slot entero de datos, así podemos estudiar un caso óptimo donde la DFT y la IDFT del transmisor o la IDFT y la DFT del receptor poseen la misma longitud. Además se puede obtener la respuesta del canal en toda la banda estimada, para compararlo con la real.
- Como ya se ha dicho anteriormente, sólo se va a simular un usuario con un canal, ya que este usuario se va a encargar de realizar el mapeado correspondiente.

Por tanto, en el receptor tendremos que realizar todas las operaciones inversas al transmisor, que se resumen a continuación:

- Se supone una sincronización perfecta, por tanto en la trama recibida debemos trocearla por slots.
- Una vez que tenemos la trama de cada slot, podemos eliminar el prefijo cíclico introducido.
- Si realizamos la DFT a la trama y ordenamos en un array de dos dimensiones, cuya estructura es justamente la representada en la ilustración 6.
- Una vez que tenemos todas las portadoras, y tomamos el conjunto de sub portadoras asignadas por las capas superiores, que deben ser las mismas en el transmisor y en el receptor. En esta simulación se va a emplear las mismas que en el momento de la generación, para ahorrar carga computacional.
- Una vez que tenemos el conjunto de portadoras, habrá que realizar una igualación de canal con la ayuda de los DRS, cuyo procedimiento ya se explicó anteriormente, mediante un igualador ZF.
- Antes de demodular, se realiza una IDFT para volver a pasar al dominio del tiempo.
- Y por último demodulamos para obtener los símbolos recibidos, los pasamos a bits y calcular la probabilidad de error.

3. Simulación en Matlab.

Para medir la PAPR, aplicamos la fórmula que hay a continuación:

$$PAPR = \frac{max(P_{trama})}{mean(P_{trama})} = \frac{max(|s|^2)}{E[|s|^2]}$$

Donde la trama "s" es el vector de símbolos tras realizarle la IDFT añadiéndole el prefijo cíclico, por tanto tenemos una PAPR para cada símbolo SC-FDMA en el tiempo.

1. Simulación de la PAPR.

- ***Simulación 1.***

La primera simulación, consta de analizar cómo es la PAPR para cada formato de PUCCH. Cómo se aprecia en la ilustración siguiente, el formato PUCCH 2 presenta una PAPR mucho más elevada que el resto. Y la raya sólida, como representa a todos los datos, es como una especie de media, por tanto se encuentra justamente en el medio entre PUCCH 2 y PUCCH 1 y 3.

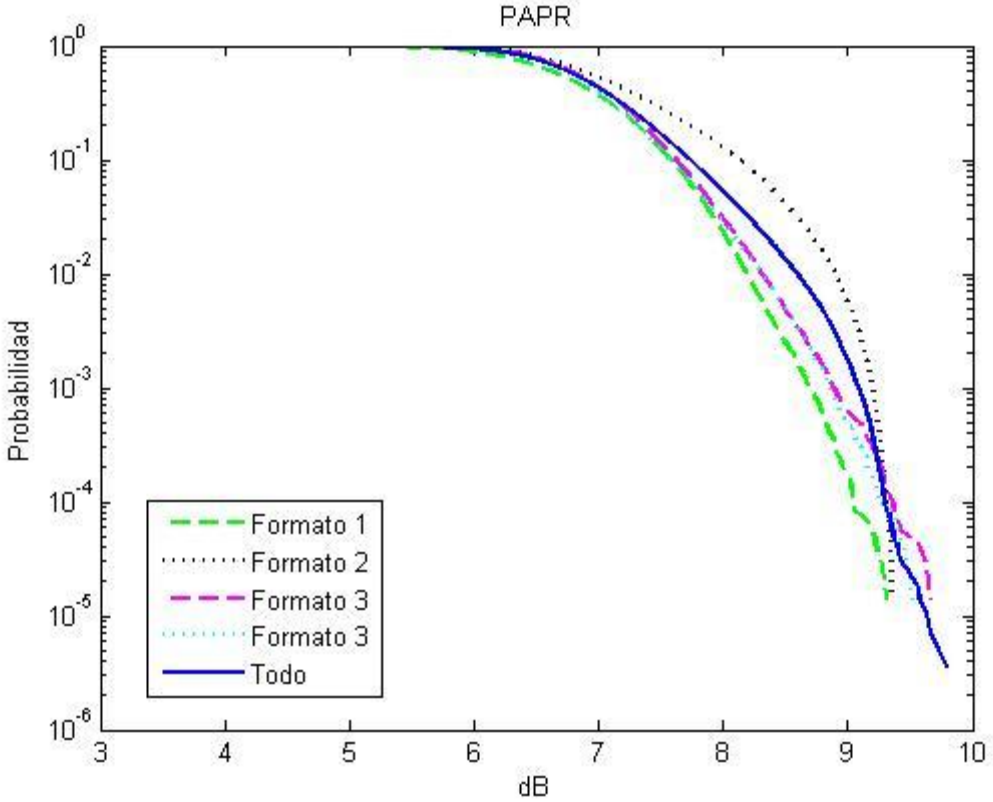


Ilustración 17: Simulación diferentes PUCCHs

Datos:
Iteraciones: 100
Ancho de banda: 10MHz
Prefijo cíclico: Normal
Modulación: QPSK
Shortened: NO
Número de usuarios: 4

La razón de ello, puede ser en que el formato 3 realiza una DFT de la información a igual que la información PUSCH, y el formato 1 sólo se transmite un único símbolo replicado en todo el recurso de bloque. Sin embargo en el formato 2, se está transmitiendo 11 símbolos diferentes y no se realiza ninguna DFT, por tanto es por este motivo que su PAPR es superior al resto.

- ***Simulación 2.***

Se va a simular un caso ideal, donde los datos ocupan toda la trama. Un único usuario envía su trama de PUCCH correspondiente, y el espacio restante lo ocupa de datos.

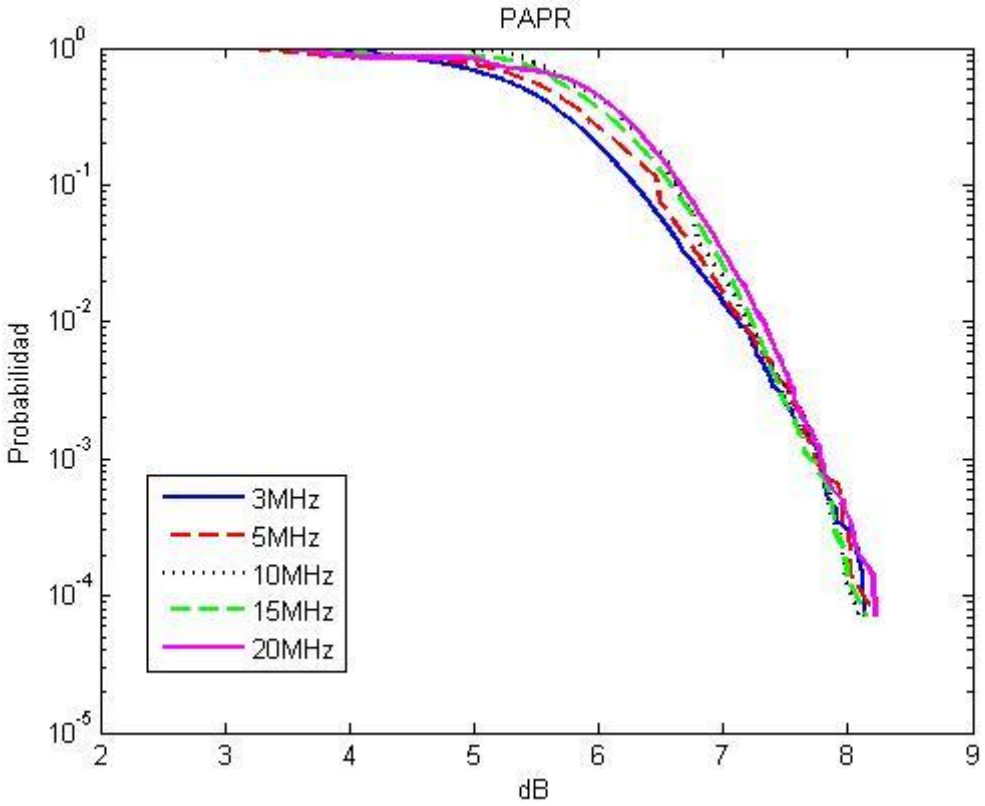


Ilustración 18: Simulación del caso ideal

Datos:
Iteraciones: 50000
Ancho de banda: Todos
Prefijo cíclico: Normal
Modulación: QPSK
Shortened: NO
Cada usuario tiene exclusivamente 4RBs

Como se aprecia en la ilustración 18, más o menos la PAPR converge desde el inicio hasta el final, el caso peor reside en unos 8.4dB, que es inferior a la modulación OFDM.

Y las variaciones o discrepancias entre una curva y otra, es debido a los formatos, como se eligen formatos aleatoriamente, puede que el formato 2 altere un poco las curvas, ya que estamos representando su media.

- ***Simulación 3.***

Tras ver el caso ideal, vamos hacia un caso real, se va a simular para diferentes anchos de banda disponibles, donde cada usuario transmite siempre 4 bloques de recursos.

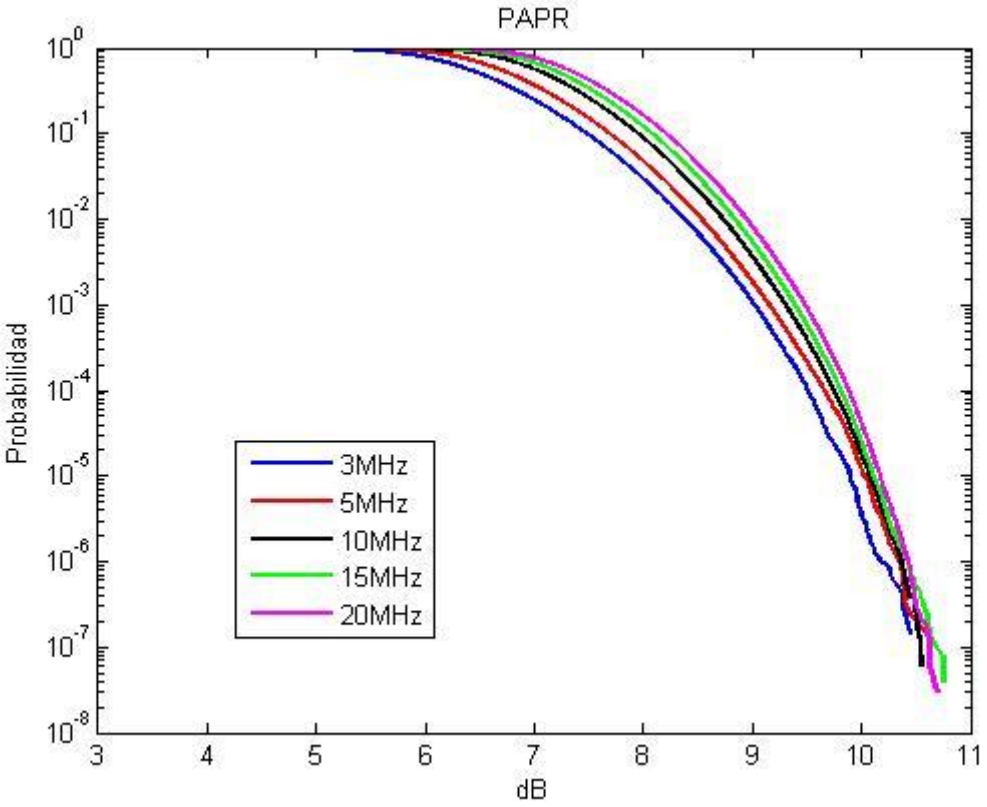


Ilustración 19: Simulación para diferentes BW

```
Datos:
Iteraciones: 50000
Ancho de banda: TODOS
Prefijo cíclico: Normal
Modulación: QPSK
Shortened: NO
Cada usuario tiene exclusivamente 4RBs
N_usuarios=[1, 2, 4, 6, 8];
```

Como se aprecia en la ilustración 19, al aumentar el ancho de banda, se presenta una peor PAPR, esto es debido a que se dispone de más espacio sin transmitir, y en esos lugares se inserta ceros. Dichos ceros hacen que la media descienda, y por tanto hacen que la PAPR aumente. Cuanto mayor ancho de banda se disponga, mayor es el número de ceros, y por tanto mayor PAPR.

Además se aprecia que existen picos que llegan casi a los 11dBs, unos 3dBs superior al caso ideal. También, que para una probabilidad de 10^{-4} se disponen de una magnitud de unos 10dBs y en el caso ideal sólo se dispone de 8.4dBs.

Estamos viendo claramente, que el caso real, la PAPR es ligeramente elevada, unos 3dB en el caso más pésimo, pero aun así sigue siendo ligeramente inferior a la modulación en OFDM.

- ***Simulación 4.***

En esta simulación, se va a estudiar los efectos de las señales de piloto SRS. Se va a realizar dos simulaciones para dos bandas diferentes, partiendo de los mismos datos que la simulación anterior, simplemente se le añade dos simulaciones con SRS.

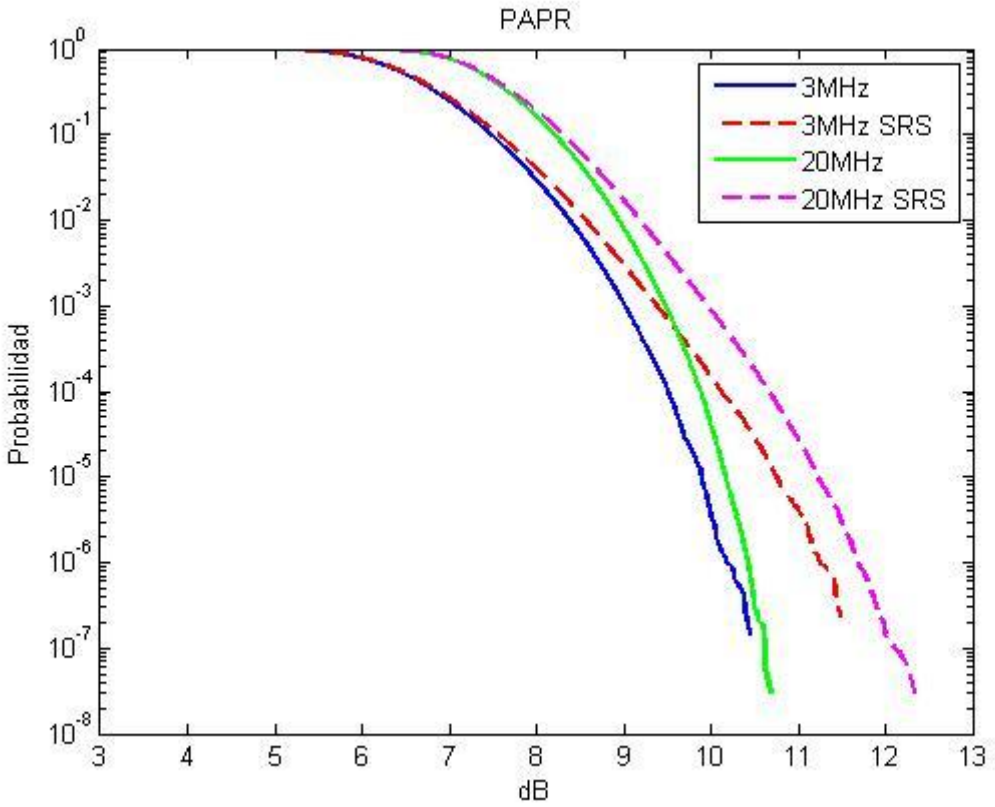


Ilustración 20: Simulación datos con SRS

Datos:
Iteraciones: 50000
Ancho de banda: 3 y 20MHz
Prefijo cíclico: Normal
Modulación: QPSK
Shortened: SI
Cada usuario tiene exclusivamente 4RB

Como se aprecia en la ilustración 20, podemos ver que si el usuario transmite los datos junto a los pilotos puede llegar hasta los 12.4dBs como máximo, sacando una diferencia de casi 2dBs con respecto al caso de no transmitir SRS.

Esto es debido a que las señales de piloto, sólo se rellenan la mitad de las portadoras, si se transmite información en las portadoras cuya posiciones pares, por tanto las portadoras de las posiciones impares son ceros. Al aumentar la cantidad de ceros, disminuye la media, y por tanto aumenta la PAPR.

- ***Simulación 5.***

En esta simulación, se va a estudiar la PAPR en función de las modulaciones existentes. Se va a realizar una simulación en una banda, para las modulaciones QPSK, 16QAM y 64QAM.

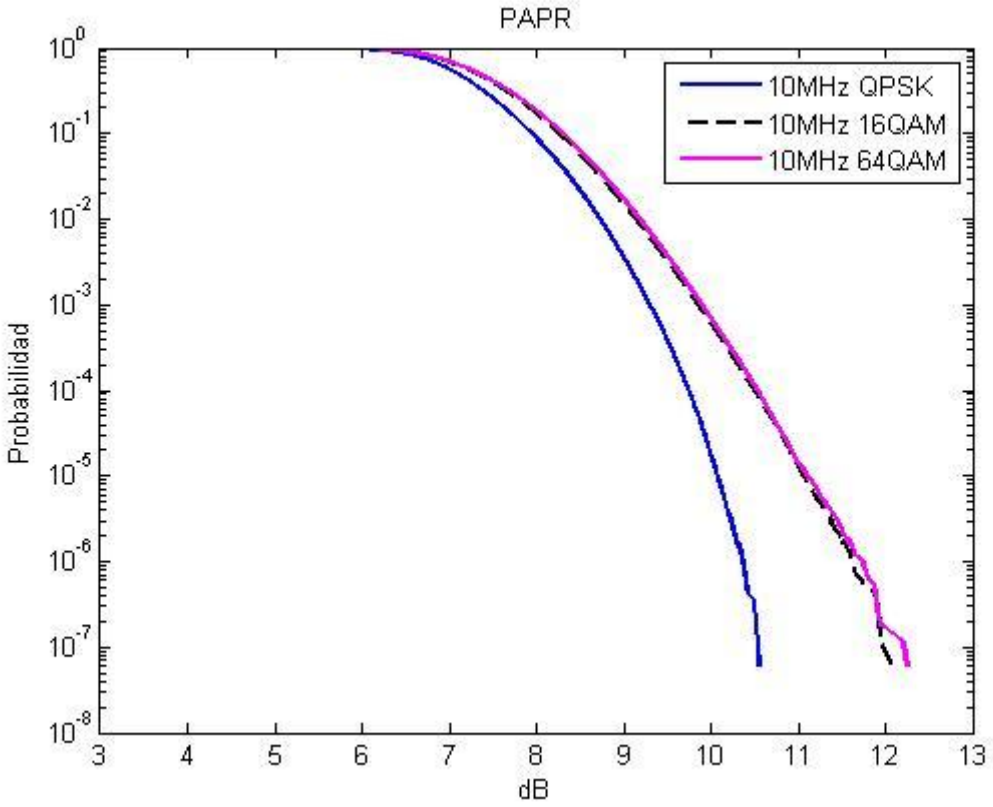


Ilustración 21: Simulación para diferentes modulaciones

Datos:
Iteraciones: 50000
Ancho de banda: 10MHz
Prefijo cíclico: Normal
Modulación: TODOS
Shortened: NO
Cada usuario tiene exclusivamente 4RB

Como se aprecia en la ilustración 21, la modulación QPSK es la que presenta un mejor comportamiento con respecto a la PAPR. Para las otras dos modulaciones, se comportan de la misma manera y superando a la QPSK hasta 1.4dB por encima.

Esto puede ser debido a que la modulación QPSK dispone de menos símbolos, menos niveles y una media mayor. Sin embargo para las otras dos modulaciones, se disponen de mayores niveles, y por tanto discrepan más sobre la media.

- ***Simulación 6.***

En esta simulación, se va a realizar el estudio de la PAPR en función de la cantidad del prefijo cíclico, la normal y la extendida.

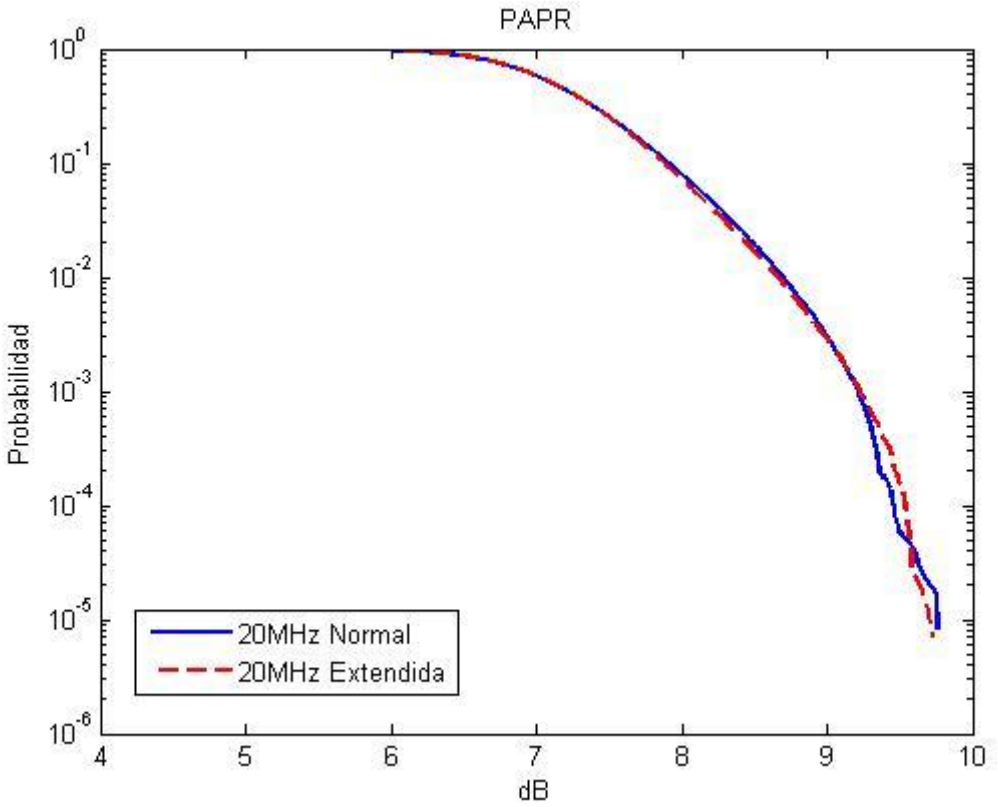


Ilustración 22: Simulación para diferentes longitudes de CP

Datos:
Iteraciones: 100
Ancho de banda: 20MHz
Prefijo cíclico: TODOS
Modulación: QPSK
Shortened: NO
Cada usuario tiene exclusivamente 4RB

Como se aprecia en la ilustración 22, y como uno ya se sabe antes de realizar la simulación, la PAPR no debe cambiar con respecto al prefijo cíclico, ya al replicar la información ni se altera la media ni el valor máximo de la misma.

2. *Simulación de la Probabilidad de error.*

- ***Simulación 1.***

En una primera aproximación, vamos a estudiar un canal sin ruido, se va a emplear por ejemplo un canal EPA. De este modo podremos estudiar cómo el igualador ZF estima el canal.

Como se aprecia en la ilustración 23, representamos el canal en el dominio de la frecuencia en todo el ancho de banda. Y viendo un poco lo resultados, vemos que el canal estimado posee una diferencia con respecto al canal real, podemos ver que el ZF no está dando los resultados esperados, como en OFDM.

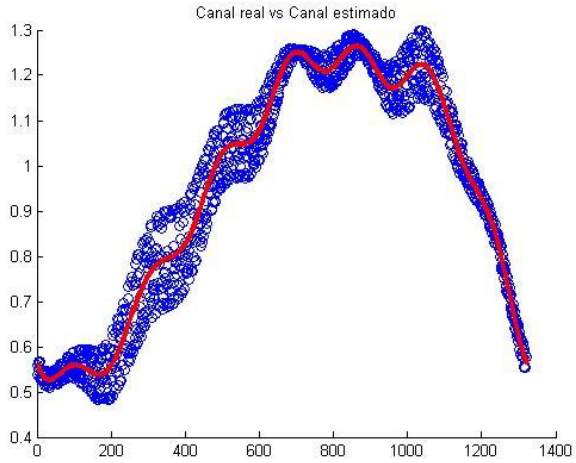


Ilustración 23: Estimación del canal en frecuencia

Por otro lado, representamos los símbolos QPSK transmitidos (‘x’) y los símbolos QPSK recibidos (‘o’), y como se aprecia en la ilustración 24, en ausencia de ruido, muchos de los símbolos están dispersos. Por tanto esto nos introduce una probabilidad de error inicial. Si posteriormente introducimos un ruido adicional, la probabilidad de error empeorará aún más.

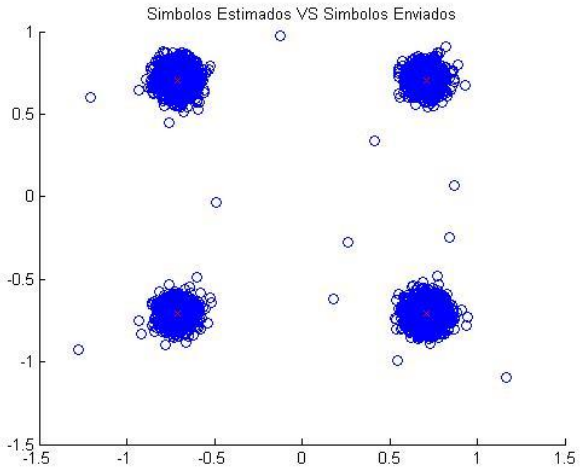


Ilustración 24: Constelación QPSK txón y rxón

- ***Simulación 2.***

En está simulación, se va a estudiar el ancho de banda del sistema. Debido en que LTE dispone de varios anchos de banda, cada uno ve un canal diferente, ya que poseen una velocidad de muestreo diferente.

Como se aprecia en la ilustración 25, se ha simulado los dos anchos de banda extremos. Como cabía de esperar, si empleamos un ancho de banda de 3MHz cae casi hasta 10^{-2} con una SNR de 10dB, sin embargo, si empleamos un ancho de banda de 20MHz el sistema tiene una probabilidad de error malísimo.

Es evidente que los resultados eran previsibles desde un punto de vista cualitativo, sin embargo desde un punto de vista cuantitativo, los resultados de la probabilidad de error obtenidos son demasiado altos, que nos muestran un sistema muy precario.

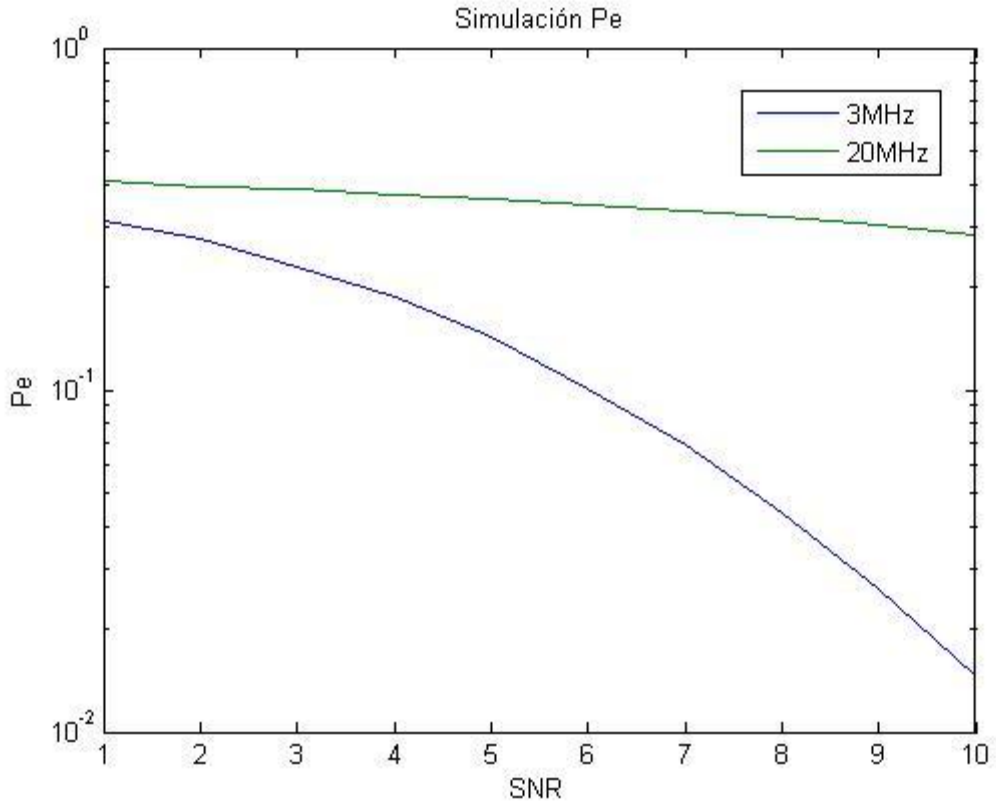


Ilustración 25: Simulación para dos BW extremos

Datos:
Iteraciones: 1000
Ancho de banda: 3MHz y 20MHz
Prefijo cíclico: NO
Modulación: QPSK

Para llevar a cabo esta simulación, se ha realizado 10^3 slots, rellenando todo el slot entero de símbolos QPSK, con un prefijo cíclico normal. La cantidad de slots simulados se mantiene a lo largo de todas las simulaciones posteriores.

- ***Simulación 3.***

En esta simulación se va a estudiar el comportamiento de los 3 canales propuestos por el 3GGP, EPA, EVA y ETU. Se va a simular para un ancho de banda de 3MHz y una modulación QPSK con un prefijo cíclico normal.

La principal razón por la que se elige un ancho de banda de 3MHz, es debido a que nos da unos mejores resultados que el de 20MHz, ya que si empleamos un ancho de banda de 20MHz no veríamos la diferencia entre los 3 canales, ya que todas nos darían una probabilidad de error alrededor del 50%.

Como se aprecia en la ilustración 26, los resultados son los esperados, el mejor canal sigue siendo el EPA y el peor es ETU, con una diferencia bastante significativa.

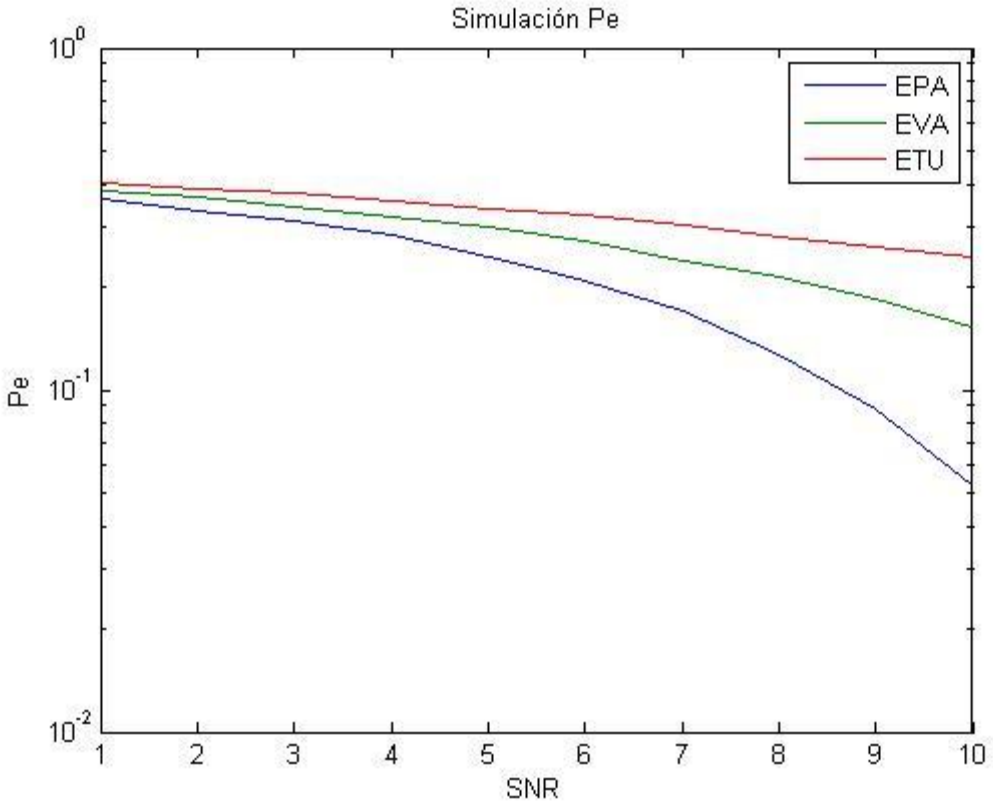


Ilustración 26: Simulación para los tres canales

Datos:
Iteraciones: 1000
Ancho de banda: 3MHz
Prefijo cíclico: NO
Modulación: QPSK

- ***Simulación 4.***

En esta simulación, se va a estudiar las diferentes modulaciones propuestas por el 3GPP, para un ancho de banda de 3MHz, dicha elección es debido por los mismos motivos que las simulaciones anteriores.

Los resultados, como se aprecia en la ilustración 27, los resultados son totalmente predecibles, ya que al emplear una modulación menos densa o con mayor distancia entre sus símbolos, reduce la probabilidad de error de manera significativa, hasta un orden de magnitud.

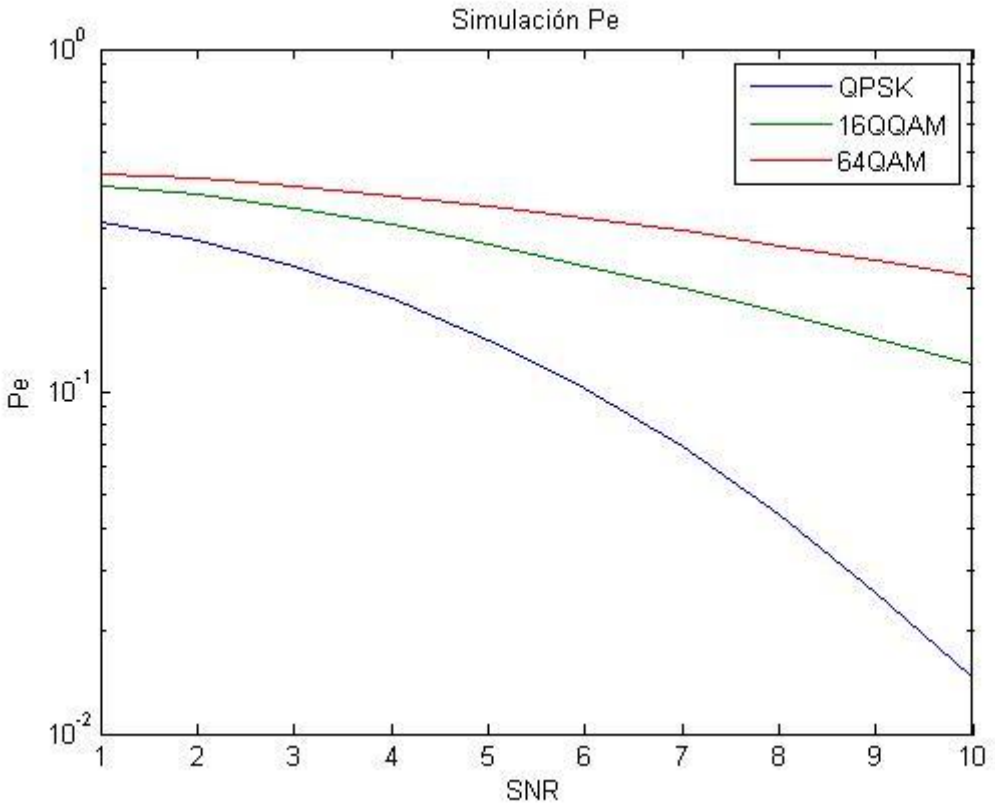


Ilustración 27: Simulación para diferentes modulaciones

Datos:
Iteraciones: 1000
Ancho de banda: 3MHz
Prefijo cíclico: NO
Modulación: TODAS

- ***Simulación 5.***

Se va a estudiar la probabilidad de error del sistema en función del prefijo cíclico empleado, ya que el 3GPP propone dos prefijos diferentes, la normal y la extendida.

Para realizar la simulación del prefijo extendido, nos obliga a emplear un ancho de banda mínimo de 5MHz, ya que la longitud del prefijo es de 512. Sin embargo, si empleamos un ancho de banda tan alto, no podremos apreciar los efectos de este prefijo, ya que la probabilidad de error ronda en torno al 50%.

Por tanto para realizar esta simulación, se va a emplear un ancho de banda de 10MHz, y se va a rebajar la longitud del prefijo cíclico a 300, de la misma longitud que la trama de datos. Y por otro lado generamos un canal aleatorio cuya longitud es de 200.

Como se aprecia en la ilustración 28, existe una gran diferencia y mejora entre dos tipos de prefijos cíclicos, como de cabía de esperar.

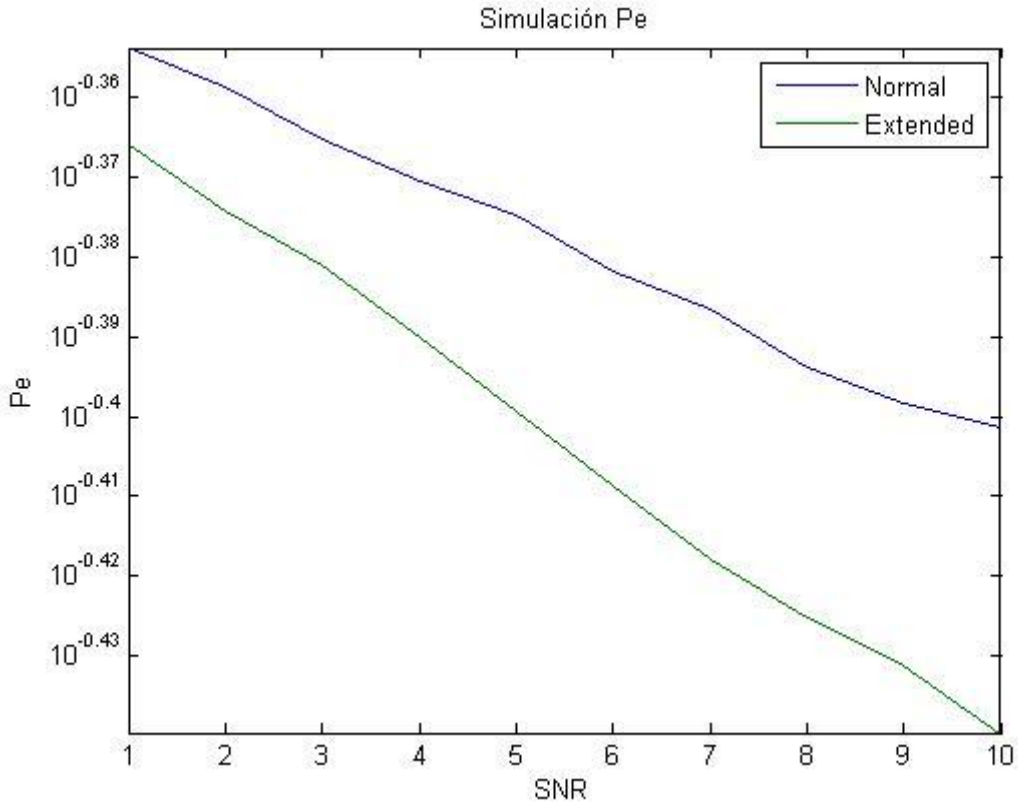


Ilustración 28: Simulación para diferentes prefijos cíclicos

Datos:
Iteraciones: 1000
Ancho de banda: 3MHz
Prefijo cíclico: TODAS
Modulación: QPSK

4. Descripción técnica de bloques VHDL.

1. Generador de bits aleatorios.

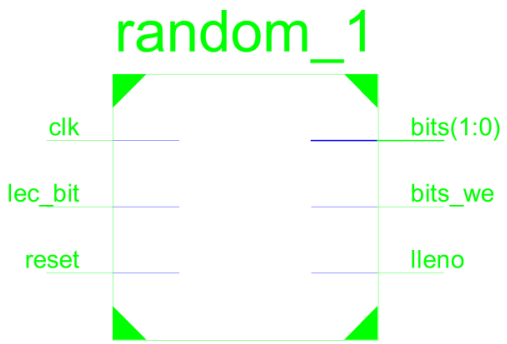


Ilustración 29: Generador de bits aleatorios

En primer lugar, vamos a necesitar una fuente que sea capaz de generar aleatoriamente bits para alimentar al transmisor. Para llevar a cabo este bloque, se va a tomar como referencia el generador de bits aleatorios del estándar [1], el aleatorizador Gold de longitud 31.

Nótese que el aleatorizador Gold está hecho para evitar secuencia largas de datos, aleatorizando estas secuencias proporcionadas por capas superiores. Nosotros vamos a emplear directamente esta secuencia aleatoria como datos, al fin y al cabo el resultado es el mismo.

Para generar este bloque se emplean los siguientes pasos:

- Se le proporciona al bloque un número entero inicial, que será nuestro c_init.
- Pasamos a binario este c_init, y lo almacenamos en nuestra x2.
- En cada golpe de reloj nos generamos dos bits en función de unas operaciones lógicas con el bit más significativo de x1 y de x2.
- Y por último eliminamos el bit más significativo de x1 y x2, e insertamos estos dos bits generados en el bit menos significativo de x1 y x2. De este modo estamos rotando los bits hacia la izquierda.
- La salida aleatoria será precisamente una operación lógica de los dos bits más significantes de x1 y x2.
- El circuito espera a que le indiquen que se desea leer la información. Cuando se lo indica, descarga la información y vuelve al tercer punto.

Las interfaces de entrada son las siguientes:

- Clk: el reloj del sistema.
- Reset: reset asíncrono del sistema.
- Lec_bit: una señal que nos indica que se desea leer los bits almacenados en el bloque, de este modo en el siguiente ciclo de reloj, dicho bloque descarga la información almacenada.

Las interfaces de salida son las siguientes:

- Bits: el bus de datos para la descarga, su ancho (2, 4, 6) en bits depende de la modulación, dicha información se pasa a través de la biblioteca.
- Bits_we: señal de habilitación de lectura.
- Lleno: señal que indica al bloque siguiente que ya hay bits preparados para leer.

Como podemos apreciar en la figura siguiente, que las señales bit, aux1, aux2, x1 y x2 cambian constantemente. El puntero cuenta desde 0 hasta 5, al tener una modulación 64QAM debemos dar a la salida 6 bits.

Y por último podemos apreciar que cuando se lanza el pulso de "Lleno", el bloque siguiente nos activa "Lec_bit" que inicia que quiere leer los 6 bits, y en el ciclo siguiente se le transmite por el bus de datos los 6 bits, y se activa "bits_we" que es la señal de habilitación de escritura.

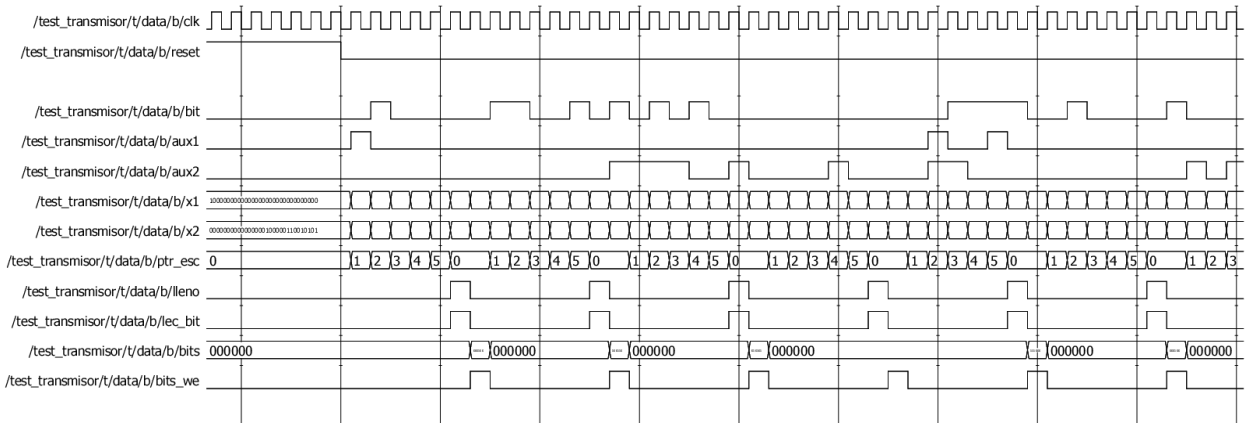


Ilustración 30: ModelSim Generador de bits aleatorios

2. *Modulador de bits.*

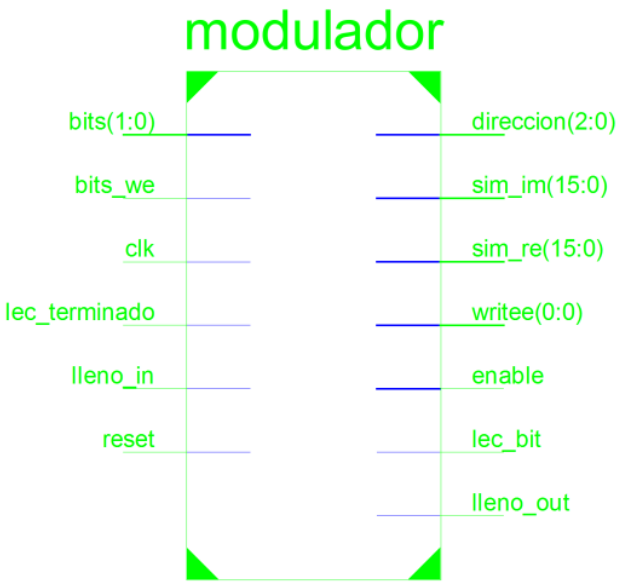


Ilustración 31: Modulador de símbolos complejos

Una vez que se ha generado los bits, el siguiente paso consiste en modularlo en símbolos complejos, como se indica en el estándar [1]. Para ello hay que tener en cuenta qué modulación hay que emplear, ya que los niveles son diferentes para cada modulación; y tener en cuenta cuántos símbolos se debe generar en cada fase, ya que la cantidad de estos símbolos deben ser igual a la longitud de la FFT.

La máquina de estados es la siguiente:

- Espera: el sistema debe esperar a que el bloque anterior le indique que los bits ya están generados y listos para modular.
- Modulado: se lee en el bus de datos los bits generados, y mediante unos switches se eligen el nivel o amplitud del símbolo, en función de la modulación seleccionada.
- Aumentado: tras modular, tendremos que aumentar un puntero o contador de símbolos modulados. Si no hemos alcanzado el tope del contador, tendremos que volver al estado de Espera, y volver a modular otro símbolo. Si ya hemos terminado de modular M símbolos, pasaremos al estado siguiente.
- Lectura: en este estado esperamos a que el bloque siguiente descargue los símbolos modulados. Mientras que el bloque siguiente no nos indique que ha concluido de leer todos los símbolos, se permanecerá en este estado indefinidamente. Cuando nos avisen que han terminado de leer todos estos símbolos, pasaremos al estado de Espera.

Para realizar este bloque, uno de los principales factores que hay que tener en cuenta consiste en elegir los niveles o amplitud de los símbolos. A través de pruebas empíricas, a partir de un número entero mayor que 720 aproximadamente, al introducirle en la FFT y posteriormente una IFFT, el resultado es totalmente diferente que el original. Este problema resulta del desbordamiento de la misma, el fabricante (Xilinx) nos sugiere que introduzcamos números pequeños y que posteriormente amplifiquemos.

Para evitar tener que hacer amplificaciones excesivamente grandes, se va a tomar como valor máximo el número 700. Por tanto el valor máximo se hace corresponder con el símbolo más alto de la constelación 64QAM, y a partir de ahí, se normaliza y se obtiene todos los valores intermedios:

- $QPSK$: ± 457
- $16QAM$: $\pm 205,\ \pm 615$
- $64QAM$: $\pm 100,\ \pm 300,\ \pm 500,\ \pm 700$

Las interfaces de entrada son las siguientes:

- Clk: el reloj del sistema.
- Reset: reset asíncrono del sistema.
- Bits: el bus de datos para la descarga, conectado al generador de bits aleatorios.
- Bits_we: señal de habilitación de lectura, conectado al generador de bits aleatorios.
- Lleno_in: señal que indica que los bits están listos, conectado al generador de bits aleatorios.
- Lec_terminado: señal que indica que ya se ha concluido con la lectura de todos los símbolos de la memoria RAM.

Las interfaces de salida son las siguientes:

- Direccion: el bus de direcciones que se conecta a dos memorias RAMs.
- Sim_re: señal de datos reales.
- Sim_im: señal de datos imaginarios.
- Writee: señal de habilitación de escritura.
- Enable: señal de habilitación de la memoria RAM.
- Lleno_out: señal que indica al bloque siguiente que ya están listos los M simbolos en la memoria RAM.
- Lec_bit: señal para avisar al generador de bits aleatorios que ya se ha leído los bits del bus.

Como se aprecia en la ilustración 32, podemos ver que tras obtener los 6 bits del generador de bits aleatorios, se procede a modular, por tanto cambian el número de los símbolos reales e imaginarios. Se aprovecha para escribir en memoria, se habilita las señales “writee” y “enable”, y por último se aumenta el puntero. Nótese que para realizar todas operaciones se dispone de una máquina de estados, véase en la señal de “estado_actual” como cambia de un estado a otro, en función de las operaciones realizadas.

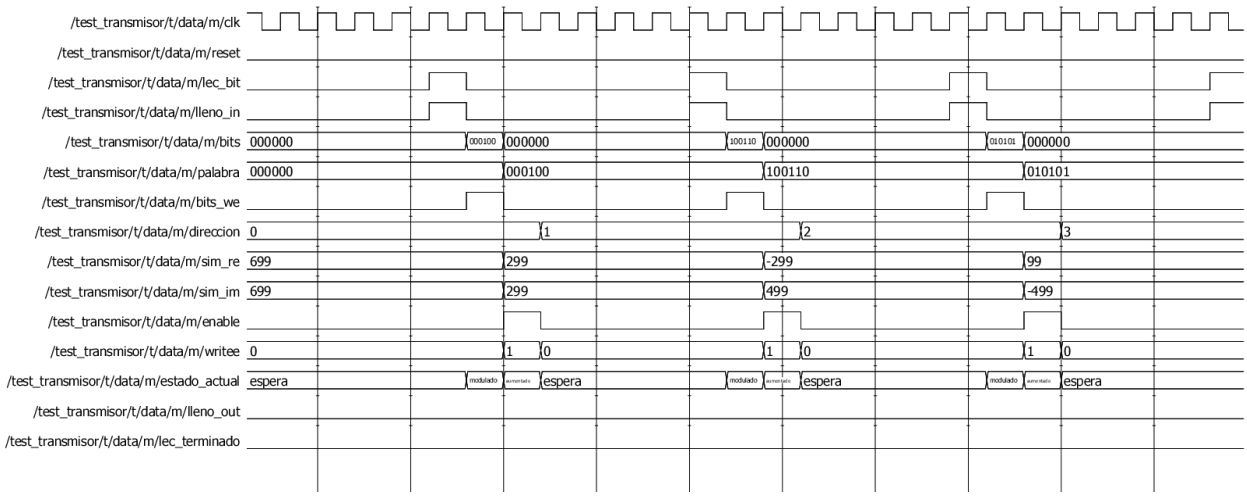


Ilustración 32: ModelSim Modulador de símbolos complejos

En la ilustración 33 se pretende dar una visión global del modulador de símbolos complejos, para ello nos tenemos que centrar en las dos últimas señales. Cuando hemos modulado M símbolos complejos, se activa la señal de “lleno_out” para avisar al bloque siguiente de que estamos esperando a que recoja estos símbolos. Y cuando el bloque siguiente nos active “lec_terminado”, en nuestro bloque nos volvemos a generar otra vez M símbolos y volvemos a esperar.

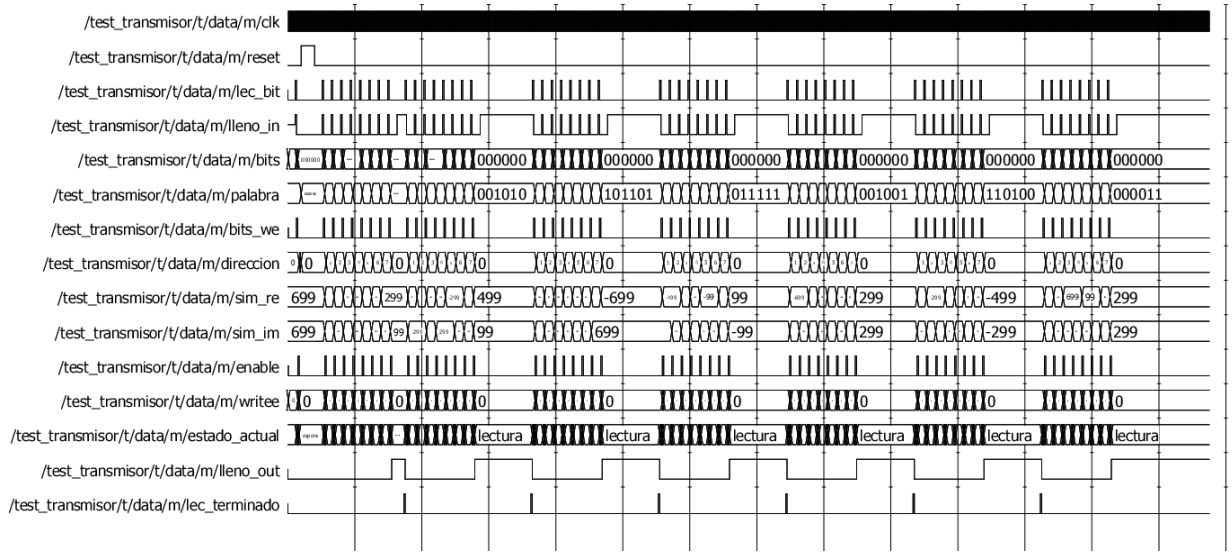


Ilustración 33: ModelSim Modulador visión global

3. *Pre codificador.*

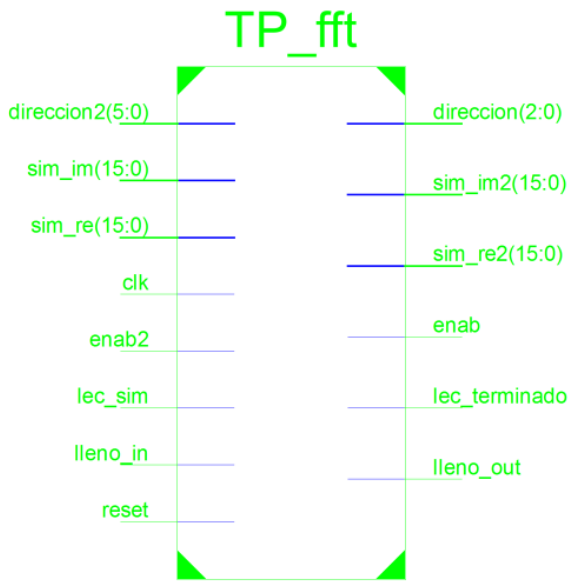


Ilustración 34: Pre codificador

En este bloque consiste principalmente en realizar una FFT a los símbolos almacenados en la memoria RAM por el modulador. Esta operación se hace tantas veces como el número de RBs asignados a un usuario móvil, tras esta operación se vuelve a almacenar toda la otra memoria RAM, que está integrado en este bloque.

La máquina de estados que gobierna no es más que una máquina de estados que gobierna la FFT, cuyos estados son las siguientes:

- Espera_lleno: el sistema debe esperar a que el bloque anterior le indique que los M símbolos complejos ya están almacenados en la memoria RAM.
- Espera_memo: el sistema debe realizar una espera de un ciclo de reloj, esto es debido a que cuando proporcionamos a la memoria RAM la dirección de la cual se desea leer, éste tarda hasta un ciclo de reloj en proporcionar el resultado.
- Inicio: estado en que se deben cargar la información previa a la FFT y para decirle que empiece a leer datos en el siguiente ciclo de reloj.
- Carga: la FFT está esperando en su entrada M símbolos, donde M es la longitud de la FFT establecida. Por tanto se necesita M ciclos de reloj para cargar los M símbolos.
- Cálculo: tras cargar los símbolos a la FFT, éste se toma un largo tiempo en calcular, por tanto debemos esperar pacientemente hasta que la FFT termine de realizar sus operaciones.
- Espera_des: tras terminar de operar, antes de descargar los datos de la FFT, se debe esperar 4 ciclos de reloj.
- Descarga: en el quinto ciclo de reloj es cuando se realiza la descarga efectiva de los datos, en el momento de la descarga, aprovechamos para empezar a almacenar estos símbolos en la memoria RAM interna del bloque. Si hemos aún no hemos llenado la memoria, volvemos al estado inicial, se debe realizar tantas veces hasta llenar la memoria. Cuando se llene pasaremos al estado siguiente.
- Lectura: cuando la memoria se haya llenado, debemos avisar al estado siguiente de que la memoria está llena y que los datos pueden ser leídos en cualquier momento. Y debemos

esperar hasta que haya concluido con su lectura. Para entonces, podremos volver al estado inicial y realizar todas las operaciones anteriores otra vez.

En este bloque, se ha tenido que realizar una gran modificación sobre el diseño de la modulación SC-FDMA, donde el estándar [1] se indica que cada RB posee 12 sub portadoras, y dicho número no es un número potencia de 2, por tanto no es posible generar el número 12 mediante 2^N. Tampoco podemos emplear una FFT de 16 puntos y eliminar 4 muestras tras la FFT, ya que en el receptor al faltar 4 muestras, la IFFT no nos daría el valor original del transmisor. Una posible solución a este problema, fue reducir las sub portadoras a 8, por tanto la longitud de la FFT será 8. Se ha elegido 8 y no 16, porque al ser un número más pequeño es mucho más fácil de configurar la cantidad total de las sub portadoras, nótese que en la Tabla 1 se especifica un número de RBs totales para cada ancho de banda, por tanto al cambiar el tamaño de una RB también cambia la cantidad total de sub portadoras. Se debe intentar mantener intacto la cantidad de sub portadoras asignadas a cada usuario y la cantidad de sub portadoras totales dado un ancho de banda. Otra forma de verlo consiste en mantener la misma proporcionalidad entre número de sub portadoras de usuario frente al número de sub portadoras totales, ya que si no mantenemos esta proporcionalidad, alteraríamos la PAPR, porque la cantidad de ceros introducidos en las sub portadoras libres influencia mucho en el resultado final.

En la simulación en Matlab, se está empleando 4RBs para cada usuario, por tanto tenemos 48 sub portadoras para la transmisión de la información. Como nuestro RB tiene un tamaño de 8 puntos, necesitaremos 6RBs para alcanzar las 48 sub portadoras. Por tanto la operación de la FFT se realizará 6 veces por cada slot con 8 símbolos en su entrada y 8 símbolos en su salida.

Las interfaces de entrada son las siguientes:

- Clk: el reloj del sistema.
- Reset: reset asíncrono del sistema.
- Lleno_in: señal que indica que los símbolos están listos, conectado a la RAM anterior.
- Sim_re: bus de datos reales.
- Sim_im: bus de datos imaginarios.
- Lec_sim: señal que indica que ya se ha concluido con la lectura de los símbolos de la RAM interna.
- Enab2: señal de habilitación de lectura de los datos de las dos memorias internas.
- Dirección: bus de dirección para leer los datos de las dos memorias internas.

Las interfaces de salida son las siguientes:

- Direccion: el bus de direcciones que se conecta a dos memorias RAMs anteriores para leer su contenido.
- Enab: señal de habilitación de lectura de las dos memoria RAMs anteriores para leer su contenido.
- Sim_re: bus de datos fft reales.
- Sim_im: bus de datos fft imaginarios.
- Lleno_out: señal que indica al bloque siguiente que ya están listos los 48 símbolos fft al bloque siguiente.
- Lec_terminado: señal para avisar al modulador que ya se ha concluido con la lectura de los símbolos.

Como se aprecia en la ilustración 35, en “Inicio” se activa todas las señales pertinentes para configurar la FFT, posteriormente en “carga” se introduce los datos por el bus de datos, y una buena forma para depurar que entra bien los 8 datos es ver en “xn_index” que no es más que un contador proporcionado por la FFT. Tras la carga, pasamos a un estado de “cálculo” que en la cual se activa una señal de “busy”. Nótese que no tenemos que olvidar de activar “lec_terminado” para avisar al bloque anterior que ya hemos usado los datos de la memoria RAM.

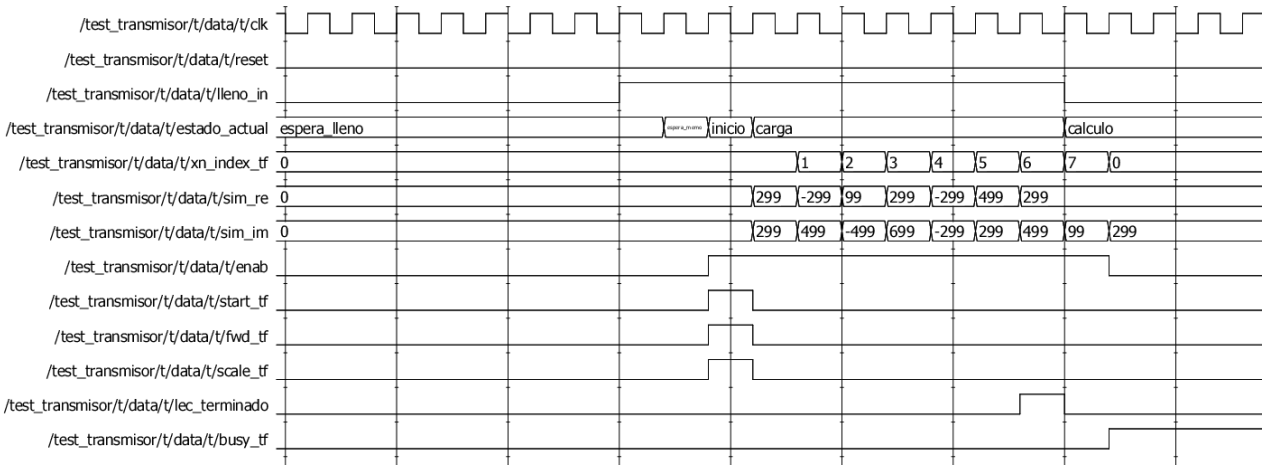


Ilustración 35: ModelSim FFT carga y cálculo

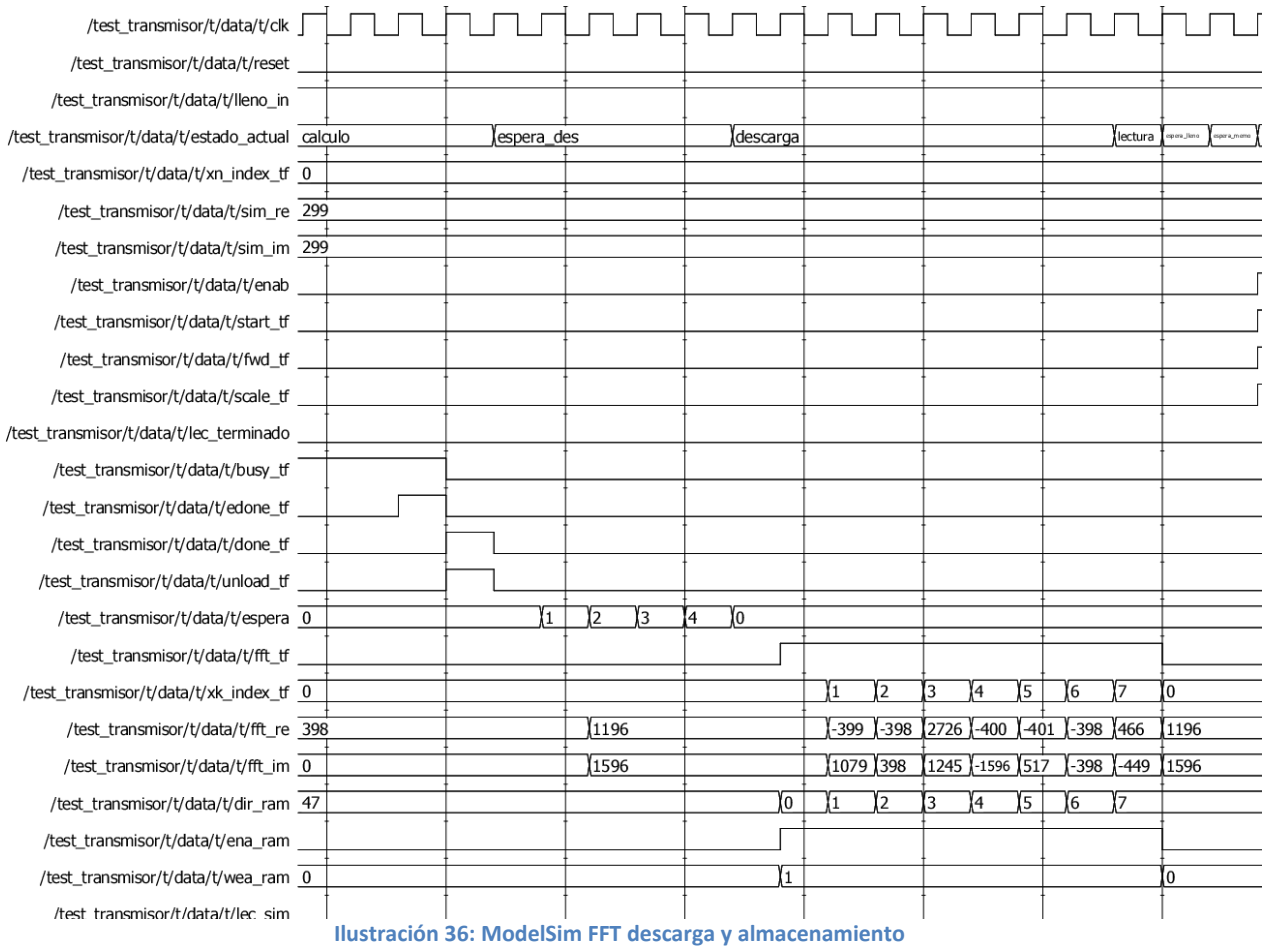


Ilustración 36: ModelSim FFT descarga y almacenamiento

Como se aprecia en la ilustración 36, tras terminar el cálculo de la FFT, se activa las señales “edone” y “done”, y es ahí donde debemos activar la señal de “unload” para poder recoger el resultado de la FFT. Debemos esperar 4 ciclos de reloj para que el resultado aparezca, y aprovechamos para escribir en la memoria RAM interna del componente.

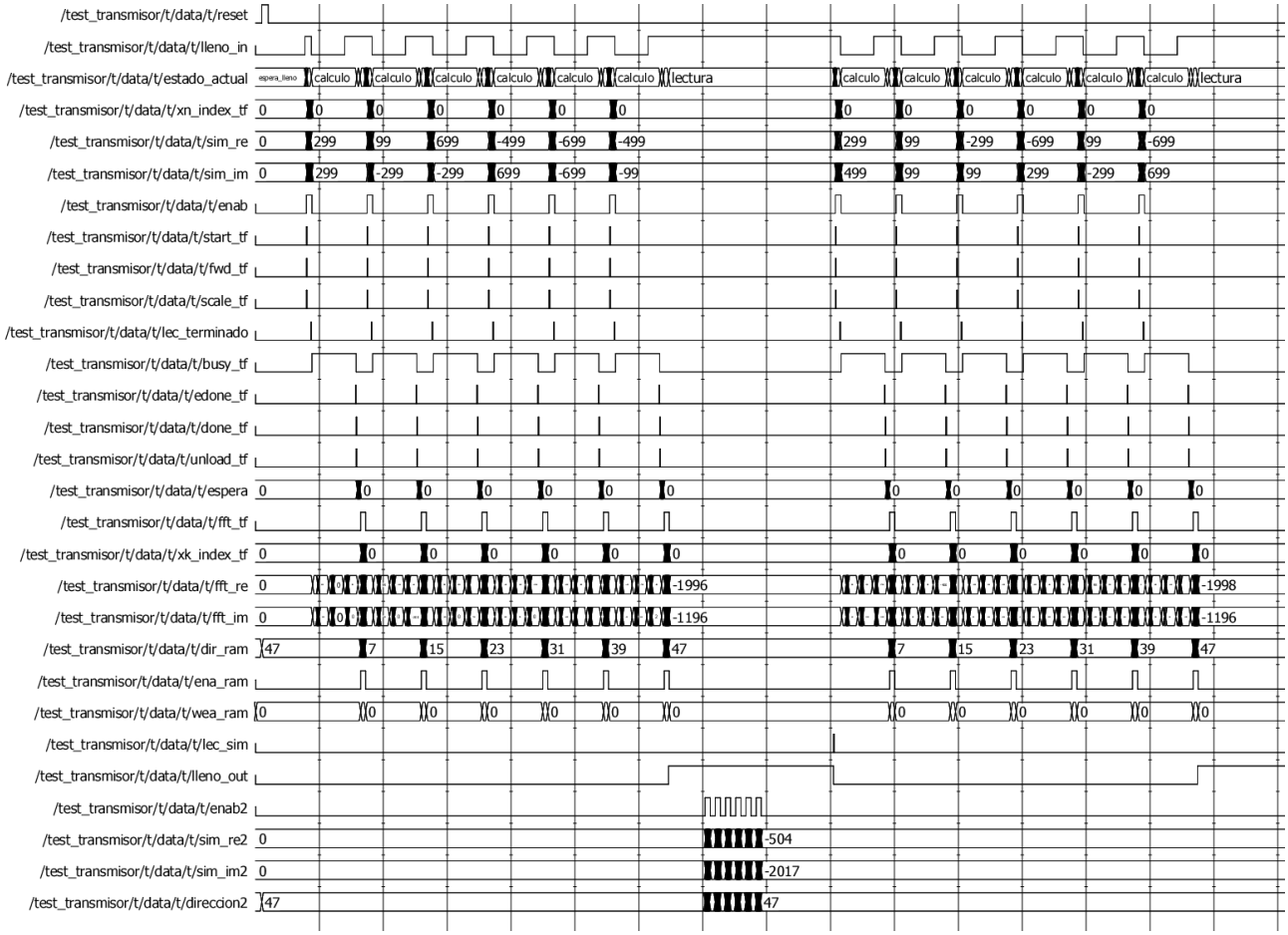


Ilustración 37: ModelSim Pre codificador visión global

Como se aprecia en la ilustración 37, tenemos una captura global de 2 slots. La FFT se realiza 6 veces y se acumula en una memoria RAM, podemos apreciar que el puntero va de 0 a 47. Cuando se ha almacenado, la información es descargada por el siguiente bloque. Y cuando nos avisen de que la información ya se ha descargado, el sistema vuelve a repetir las mismas operaciones otra vez.

4. Generador de datos.

Una vez que tenemos el generador de bits aleatorios, el modulador y el pre codificador, podemos conectar todos estos bloques y juntarlos en un bloque de jerarquía superior. De este modo, tenemos una entidad que se dedica a generar símbolos FFT y los almacena en una memoria RAM disponible para cualquier bloque posterior.

Además notar que las únicas entradas y salidas conectadas a este bloque son las entradas y salidas de la memoria RAM interna del Pre codificador, a parte del clk y reset.

Las interfaces de entrada son las siguientes:

- Clk: el reloj del sistema.
- Reset: reset asíncrono del sistema.
- Lec_sim: señal que indica que ya se ha concluido con la lectura de los símbolos de la RAM interna.
- Enab: señal de habilitación de lectura de los datos de las dos memorias internas.
- Dirección: bus de dirección para leer los datos de las dos memorias internas.

Las interfaces de salida son las siguientes:

- Sim_re: bus de datos fft reales.
- Sim_im: bus de datos fft imaginarios.
- Lleno_out: señal que indica al bloque siguiente que ya están listos los 48 símbolos fft al bloque siguiente.

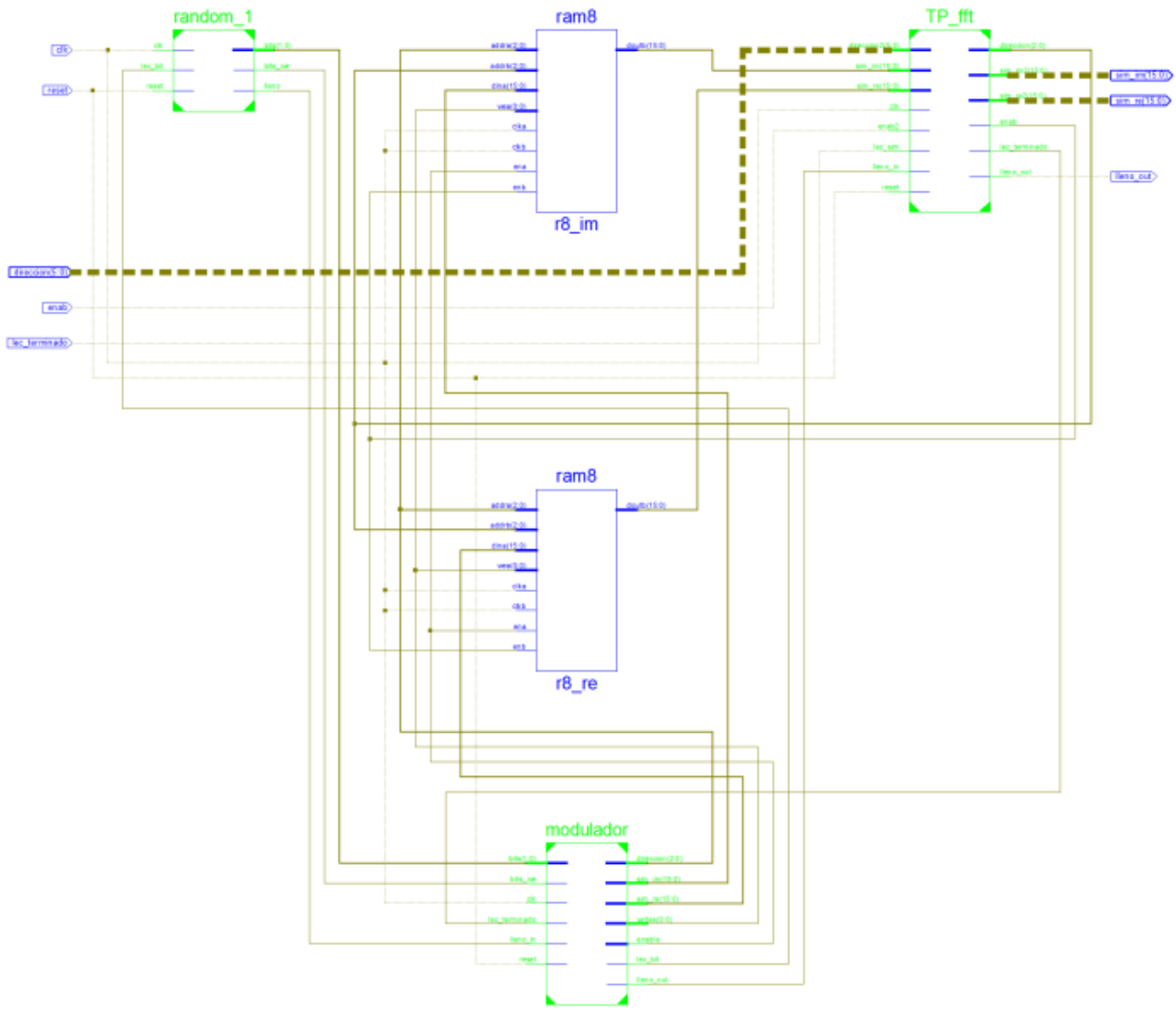


Ilustración 38: Generador de datos

5. *Pucch.*

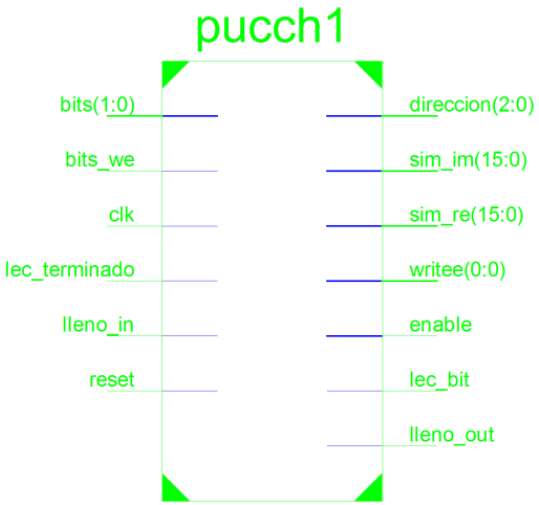


Ilustración 39: Pucch1, Pucch2 y Pucch3

Una vez que tenemos los datos, ahora necesitamos generarnos la señal de control. Las tres señales de control comparten un mismo bloque, poseen las mismas interfaces de entrada y salida, y lo único que cambia es su funcionalidad.

Las interfaces de entrada son las siguientes:

- Clk: el reloj del sistema.
- Reset: reset asíncrono del sistema.
- Bits: el bus de datos para la descarga, conectado al generador de bits aleatorios que siempre será de anchura 2.
- Bits_we: señal de habilitación de lectura, conectado al generador de bits aleatorios.
- Lleno_in: señal que indica que los bits están listos, conectado al generador de bits aleatorios.
- Lec_terminado: señal que indica que ya se ha concluido con la lectura de todos los símbolos de la memoria RAM.

Las interfaces de salida son las siguientes:

- Direccion: el bus de direcciones que se conecta a dos memorias RAMs.
- Sim_re: señal de datos reales.
- Sim_im: señal de datos imaginarios.
- Writee: señal de habilitación de escritura.
- Enable: señal de habilitación de la memoria RAM.
- Lleno_out: señal que indica al bloque siguiente que ya están listos los M simbolos en la memoria RAM.
- Lec_bit: señal para avisar al generador de bits aleatorios que ya se ha leído los bits del bus.

Los tres tipos de PUCCH poseen los mismos estados:

- Espera: estado en que se espera que los bits estén listos para ser leídos.
- Almacenado: estado en que se almacena el bit o los bits proporcionado por el bloque anterior.
- Calculado: estado en la que se obtiene el valor del símbolo modulado a partir del bit o bits almacenados previamente. Si no se ha alcanzado 8 símbolos (la señal de PUCCH ocupa 1RB)

se vuelve al estado inicial. En caso de haber alcanzado los 8 símbolos se va al estado siguiente.

- Lectura: estado en la que se avisa al bloque siguiente que ya está disponible los datos de control, y se espera hasta que sea leído. Cuando haya concluido la lectura, se vuelve al estado inicial.

Para intentar evitar emplear multiplicaciones con números complejos y para reducir su complejidad se han adoptado una serie de simplificaciones que NO afectan a la PAPR:

- Se elimina los códigos ortogonales, o se asume que van a ser siempre 1. De este modo no tendremos que realizar ninguna multiplicación.
- La generación de la secuencia base se sustituye por un vector de datos estáticos. En Matlab se genera 8 símbolos equi espaciados en la circunferencia unidad y se normaliza a 700, para mantener el mismo máximo que la modulación de los símbolos. Y se van a trabajar siempre con ese vector cuando se necesite utilizar la secuencia base.
- Se va a eliminar la rotación aleatoria que se introduce a la secuencia base.
- Se elimina todos los parámetros adicionales para simplificar el diseño: u, v, etc.

Para generar el bloque PUCCH1 se siguen los siguientes pasos:

- Sólo debemos leer un único bit, por tanto al estado de “espera” y “almacenamiento” sólo se pasa una vez por cada slot.
- Una vez que almacenamos el bit, tenemos que modularlo y multiplicarlo por la secuencia base. Para realizar esta multiplicación, no es necesario multiplicación en sí, sino mediante switches podemos realizar una serie de manipulaciones para obtener el resultado. Véase en la siguiente tabla.

Símbolo	Operación a realizar
1	Salida = entrada;
-1	Salida = 1-(not entrada)
J	Salida_imag=entrada_real; Salida_real=1-(not entrada_imag)
-J	Salida_imag=1-(not entrada_real); Salida_real=entrada_imag

Tabla 7: Operaciones sustitutivas de la multiplicación

- Según vamos generando los símbolos vamos almacenando en la memoria RAM.
- Para generar los DRS en los slots adecuados, se genera un contador de slots que gobierna un multiplexor, por tanto para ciertos números se da información de PUCCH, y para otros números se da información de DRS.
- También si nos activan SRS, el último slots se rellenará con ceros.
- Tiene en cuenta si el prefijo cíclico es normal (7 símbolos SC-FDMA) o extendido (6 símbolos SC-FDMA).

Para generar el PUCCH2, consiste en realizar una serie de modificaciones al PUCCH1 que consiste en las siguientes:

- Se debe leer dos bits por cada símbolo SC-FDMA, por tanto hay que leer dos bits 7 veces por cada slot, para un prefijo cíclico normal, y 6 para un prefijo extendido.

- La modulación cambia, ahora es QPSK normalizado a 1, por tanto para evitar tener que realizar una multiplicación se sustituye por sumas y restas elementales. Ya que la multiplicación de un número complejo cuya parte real e imaginaria posee el valor de la unidad por el segundo factor, el resultado de la misma no deja de ser una suma o resta de cada componente del segundo factor. Véase dos ejemplos:

$$(1+j)(a+jb) = a+jb+ja-b = (a-b)+j(a+b) \rightarrow \begin{Bmatrix} real = a-b \\ imag = a+b \end{Bmatrix}$$
$$(1-j)(a+jb) = a+jb-ja+b = (a+b)+j(b-a) \rightarrow \begin{Bmatrix} real = a+b \\ imag = a-b \end{Bmatrix}$$

- Las posiciones de la señal DRS son diferentes al PUCCH1, por tanto sólo tenemos que cambiar las condiciones del multiplexor, para que conmute en un slot diferente al anterior.

Para generar el PUCCH3, también consiste en realizar una serie de modificaciones al PUCCH1, que consiste en las siguientes:

- Se debe leer 16 bits por cada slot, estos 16 bits se convierten en 8 símbolos QPSK que serán replicados en frecuencia.
- Las operaciones sustitutivas a la multiplicación son las mismas que para el PUCCH2.
- Se deben realizar una rotación de los bits antes de ser mapeados.
- Las posiciones de la señal DRS son diferentes al PUCCH1, por tanto sólo tenemos que cambiar las condiciones del multiplexor, para que conmute en un slot diferente al anterior.
- Y se debe añadir un bloque de FFT a igual que el Pre codificador.

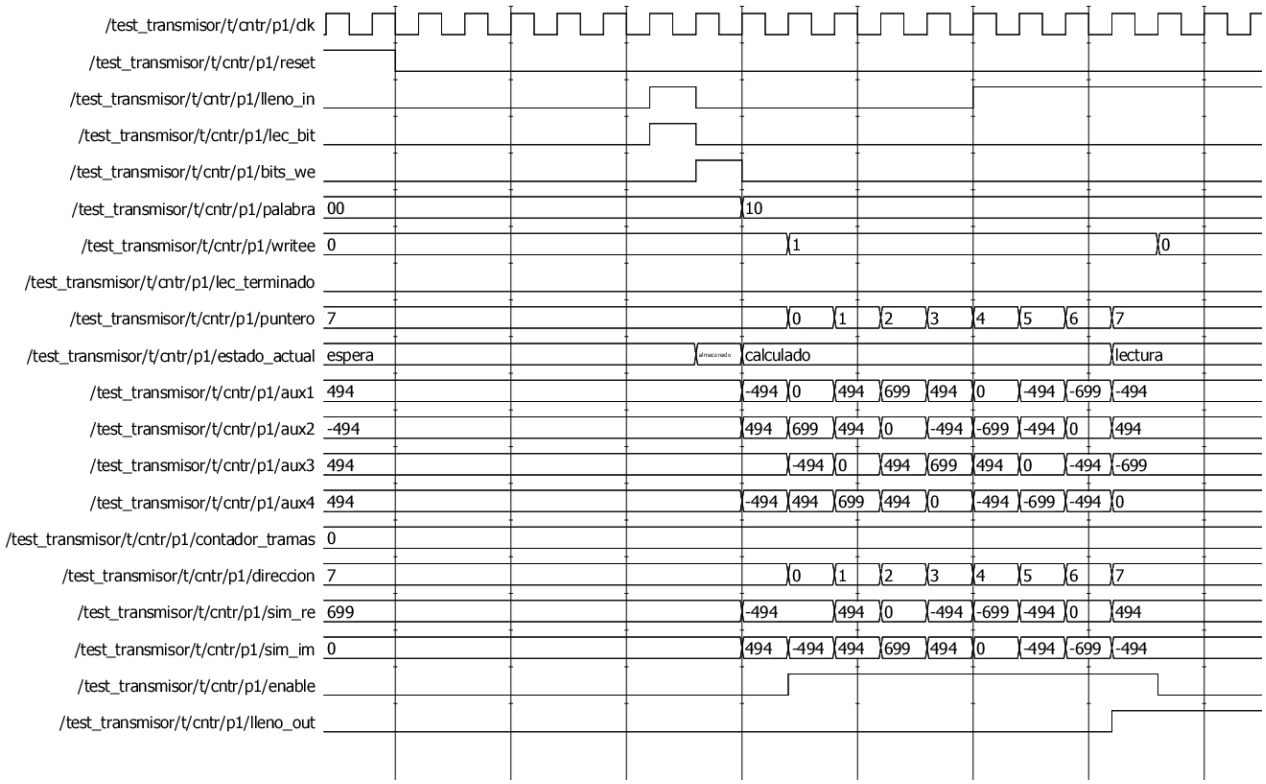


Ilustración 40: ModelSim Pucch1 generación de un simbolo SC-FDMA

Como se aprecia en la ilustración 40, leemos la palabra “10” que es diferente de cero, por tanto se va a realizar una serie de manipulaciones a la información antes de ser almacenada, las señales

"aux" nos muestra paso a paso cómo se altera la información, algunos cambian de signo, y otros se intercambian entre real e imaginario.

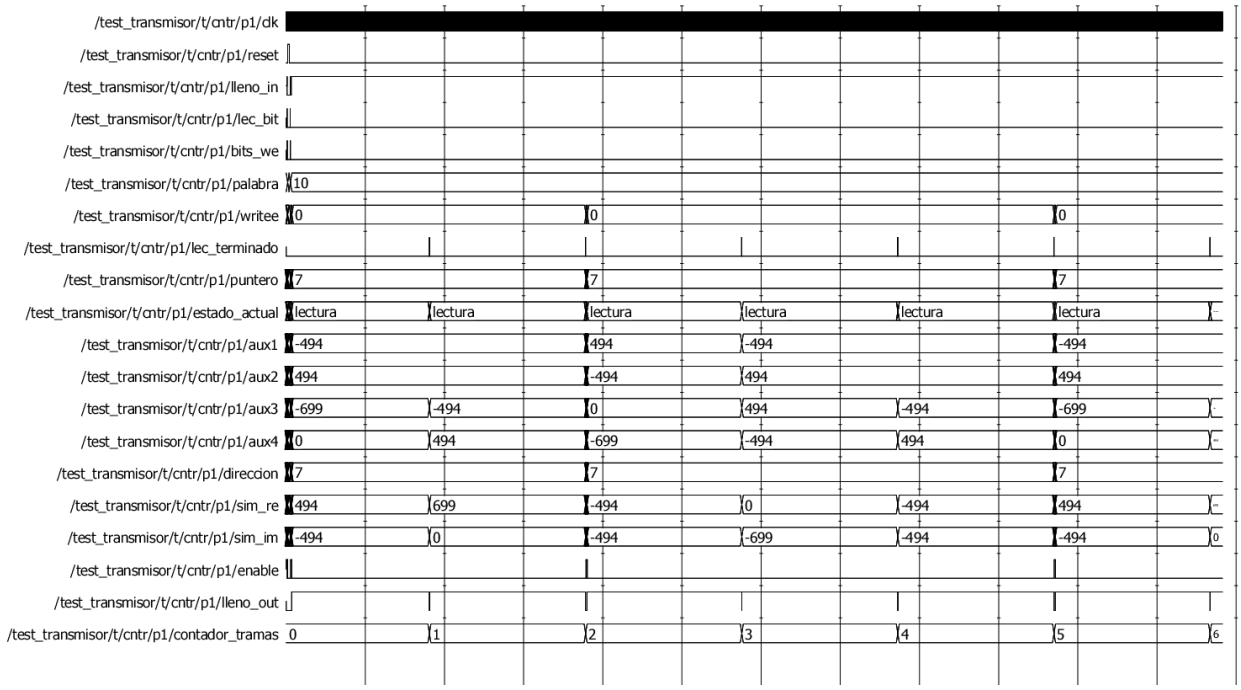


Ilustración 41: ModelSim Pucch1 visión global

En la ilustración 41, nos fijamos en primer lugar en el "contador_tramas", donde nos indica en qué trama estamos y podemos ver qué cada para cada caso:

- Cero: como estamos en el caso inicial, leemos del bit y calculamos la secuencia.
- Uno: como debe ser una réplica de la trama anterior, no se realiza ninguna operación, se mantiene los datos que hay en memoria.
- Dos: ahora tenemos que cargar en memoria la DRS.
- Tres: como sigue siendo DRS, no se hace nada.
- Cuatro: última trama de DRS, tampoco se hace nada.
- Quinta: se debe volver a cargar la secuencia calculada previamente, pero no se lee nada.
- Sexta: ídem que 2.
- Séptimo: no se ve en la simulación, pero debería ser ídem a 2 y 6.

6. *Generador de control.*

A igual que el caso de los datos PUSCH, también se va a generar un bloque con mayor nivel jerárquico que agrupe a todos los bloques que intervienen en la generación del PUCCH. Para ello se deben añadir dos bloques más que ya están definidos en el bloque de datos, de este modo reutilizamos el código y su funcionalidad, que son los siguientes:

- Generador de bits aleatorios: es el mismo bloque, pero siempre leemos los bits de 2 en 2. Además el valor inicial c_init es un parámetro genérico, y se le asigna un valor diferente para cada caso, así los bits generados son diferentes a los de los datos.
- Pre codificador: como en el PUCCH3 se necesita una FFT, podemos reutilizar el bloque del pre codificador sin realizar ningún cambio adicional.

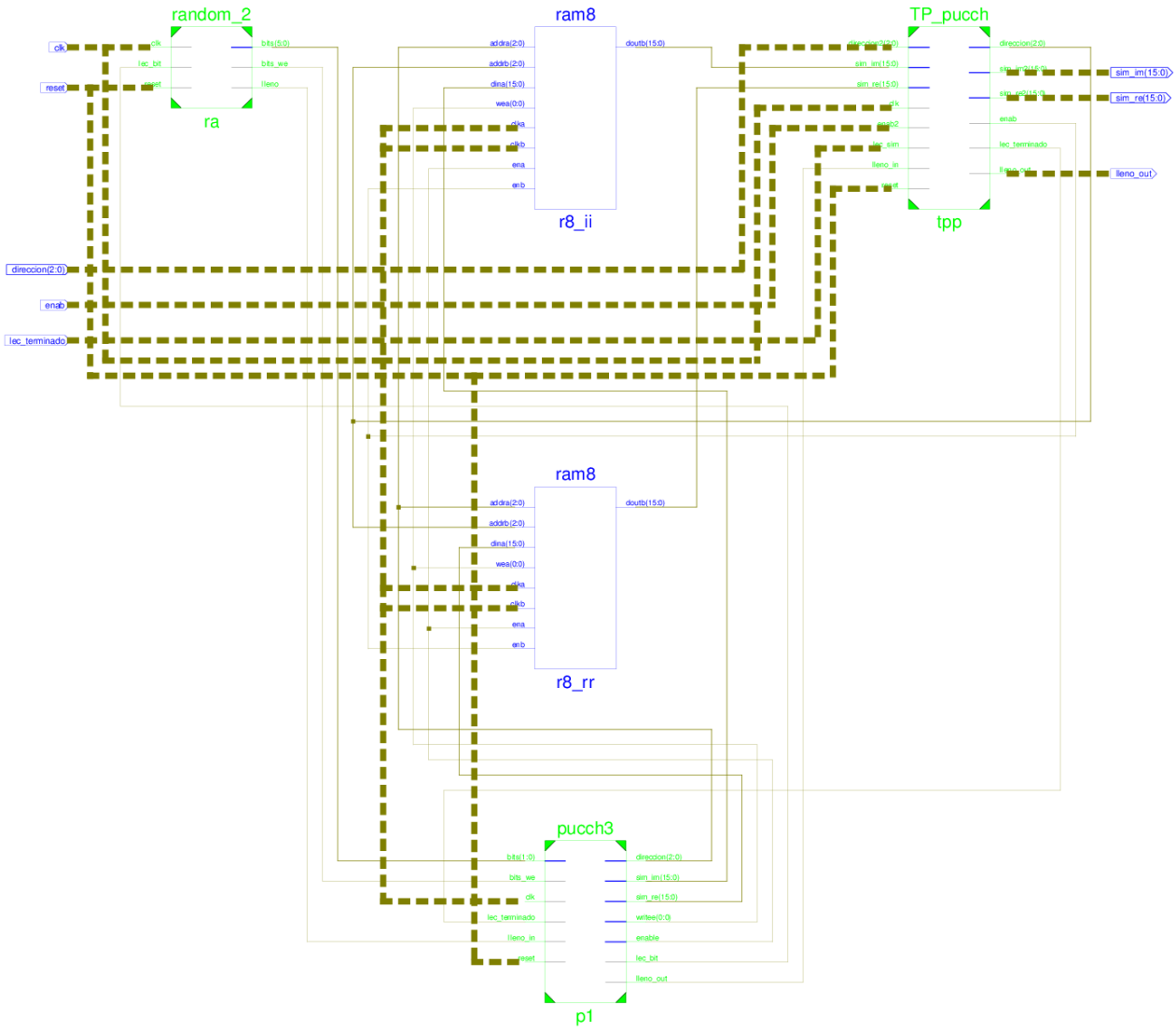


Ilustración 42: Generador de control

En la ilustración 42, se presenta la interconexión de los componentes para el PUCCH3. Y para el caso del PUCCH1 y 2 es simplemente eliminar la FFT adicional.

Además las interfaces de entrada y salida son las mismas que el generador de datos, por tanto no se van a describir otra vez las interfaces. Lo único que puede cambiar es el número de bits para la dirección de la memoria RAM, ya que en los datos estamos generando 48 datos PUSCH y aquí sólo generamos 8 datos PUCCH.

7. Mapeador.

El mapeador no deja de ser un super multiplexador de la información. Debe multiplexar diferentes fuentes de información que hay en su entrada y proporcionarlos en la salida de manera síncrona.

Como siguiente bloque del mapeador es una IFFT, y como ya sabemos que la FFT posee un estado de carga, debemos ajustar los tiempos para que la salida del mapeador se ajuste exactamente en

el tiempo de la carga de la IFFT. De este modo podremos ahorrarnos nos memorias RAM de alta capacidad.

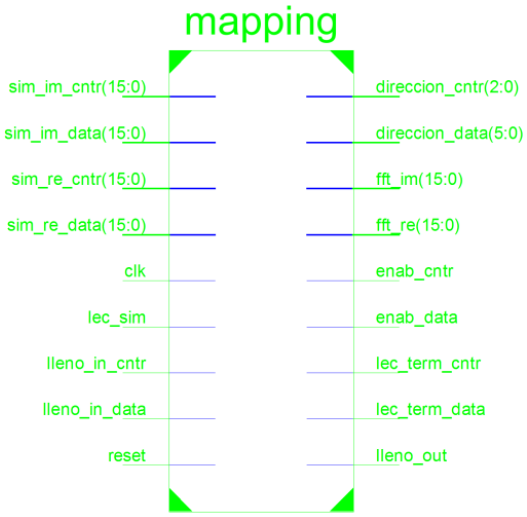


Ilustración 43: Mapeador en frecuencia de los datos

Uno de los factores que hay que tener en cuenta, es el tamaño de las sub portadoras totales no concuerda con el tamaño de la IFFT. Por ejemplo, para el caso de 3MHz se dispone de 176 sub portadoras, y se necesita una IFFT de 256 puntos. Como podemos ver, el tamaño de la IFFT posee 80 puntos más que nuestras sub portadoras, por tanto debemos introducir 40 ceros al inicio y 40 ceros al final.

La máquina de estados es la siguiente:

- Espera_lleno: en el estado inicial, el mapeador debe esperar a que tanto los datos PUSCH como el control PUCCH estén llenos.
- Lectura: estado de preparación, esperando que el bloque posterior indique al mapeador que empiece a multiplexar.
- Rellena: en este estado se empezará a multiplexar la información en función de unos punteros que nos guiarán en todo este proceso.

En el estado de “rellena” se deberá multiplexar las siguientes entradas de datos, guiándose de un contador que va de 0 a 255 para el caso de 3MHz (para otros anchos de banda, serán otros índices):

- Se empezará a rellenar los 40 ceros iniciales.
- Se rellenará PUCCH antes de los datos PUSCH si estamos en la trama par, en caso contrario se rellenará el PUCCH después. Para ello se tiene una variable binaria para recordar el estado.
- Se rellenara los datos PUSCH (6RB)/DRS/SRS, en función del contador de tramas.

No hay que olvidar que se debe rellenar la información DRS y SRS. Para ello se debe tener un contador de tramas, a igual en el caso de PUCCH, para saber cuándo se rellena PUSCH o DRS o SRS. Como en un slot tiene 7 símbolos SC-FDMA, los 3 símbolos del medio son DRS, y si estamos ante un caso de prefijo cíclico extendido de los 6 símbolos SC-FDMA, los 2 símbolos del medio son DRS. Y en caso que se active SRS, en el último símbolo SC-FDMA se debe leer el vector de la secuencia base intercalada por ceros.

La simulación en ModelSim se va a realizar junto al SC-FDMA, ya que éste alimenta a la IFFT y se puede ver conjuntamente.

Las interfaces de entrada son las siguientes:

- Clk: el reloj del sistema.
- Reset: reset asíncrono del sistema.
- Sim_re_data, Sim_im_data: bus de datos reales e imaginarios de los datos PUSCH.
- Sim_re_cntr, Sim_im_cntr: bus de datos reales e imaginarios de los datos PUCCH.
- Lleno_in_data: señal para avisar que los datos PUSCH ya están disponibles en la memoria.
- Lleno_in_cntr: señal para avisar que los datos PUCCH ya están disponibles en la memoria.
- Lec_sim: señal para indicar que se debe empezar la multiplexación.

Las interfaces de salida son las siguientes:

- Direccion_data: el bus de direcciones que se conecta a dos memorias RAMs de PUSCH.
- Direccion_cntr: el bus de direcciones que se conecta a dos memorias RAMs de PUCCH.
- FFT_re: señal de datos reales fft multiplexados.
- FFT_im: señal de datos imaginarios fft multiplexados.
- Enab_data: señal de habilitación de la memoria RAM de PUSCH.
- Enab_cntr: señal de habilitación de la memoria RAM de PUCCH.
- Lleno_out: señal que indica al bloque siguiente que ya está listo para multiplexar.
- Lec_term_data: señal para avisar al generador de datos que ya hemos terminado de leer la memoria de PUSCH.
- Lec_term_cntr: señal para avisar al generador de control que ya hemos terminado de leer la memoria de PUCCH.

8. SC-FDMA

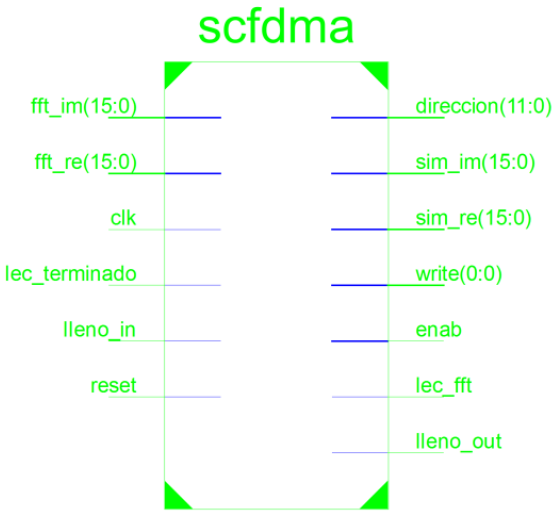


Ilustración 44: Modulación SC-FDMA

Este bloque es el que se encarga de realizar la IFFT, el bloque es el mismo que el de la FFT, sólo se debe indicar en su entrada qué operación se desea realizar. A parte de esta diferencia, es que en este módulo se le va a indicar que nos genere un prefijo cíclico de 140 puntos, de este modo no lo

tendremos que generar posteriormente. Y por tanto que la máquina de estados de este bloque es exactamente la misma que para el Pre codificador, pero se le añade un estado más: “descarga_cp”, para poder almacenar el prefijo cíclico antes de los datos.

Un parámetro de diseño fundamental de este bloque, es el reloj que se alimenta, ya que el ancho de banda de la señal en su salida corresponderá al valor del reloj clk que alimentamos a este circuito. Esto ocurre lo mismo que en OFDM, y por tanto la elección del reloj del sistema tendrá que adecuarse al ancho de banda especificado por el estándar, garantizando una separación frecuencia de 15KHz entre sub portadoras.

$$f_{clk} = 15KHz * Nfft \rightarrow \begin{Bmatrix} Nfft = 256 \rightarrow f_{clk} = 3.84MHz \\ Nfft = 2048 \rightarrow f_{clk} = 30.72MHz \end{Bmatrix}$$

Las interfaces de entrada son las siguientes:

- Clk: el reloj del sistema.
- Reset: reset asíncrono del sistema.
- Fft_re: el bus de datos reales, conectado al mapeador.
- Fft_im: el bus de datos imaginarios, conectado al mapeador.
- Lleno_in: señal que indica que el mapeador está listo para multiplexar.
- Lec_terminado: señal que indica que ya se ha concluido con la lectura de todos los símbolos de la memoria RAM.

Las interfaces de salida son las siguientes:

- Direccion: el bus de direcciones que se conecta a dos memorias RAMs.
- Sim_re: señal de datos reales.
- Sim_im: señal de datos imaginarios.
- Writee: señal de habilitación de escritura.
- Enable: señal de habilitación de la memoria RAM.
- Lleno_out: señal que indica al bloque siguiente que ya están listos los M simbolos en la memoria RAM.
- Lec_fft: señal para avisar al mapeador que empiece a multiplexar.

En la ilustración 45, podemos ver cómo se multiplexa la información y se carga en la FFT, podemos apreciar que primero se carga los datos de PUCCH y luego se carga los datos de PUSCH, dejando 1RB intercalado entre RB y RB.

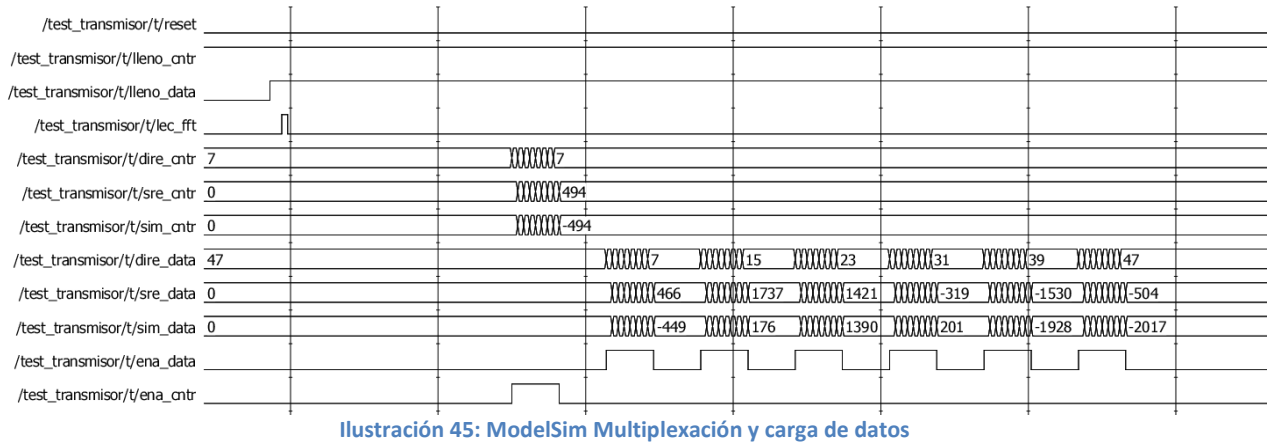


Ilustración 45: ModelSim Multiplexación y carga de datos

En la ilustración 46, podemos ver un ciclo completo de la IFFT, de aquí lo único importante que hay que destacar es el estado “descarga_cp”. Como la IFFT es de tamaño 256, antes de descargar estos 256 datos se debe descargar antes los 140 datos de prefijo cíclico, que corresponden con los últimos 140 símbolos de los 256. Por tanto, en nuestra salida, tendremos mayor número de información que recoger. El resto de la simulación es idéntico a Pre codificador.

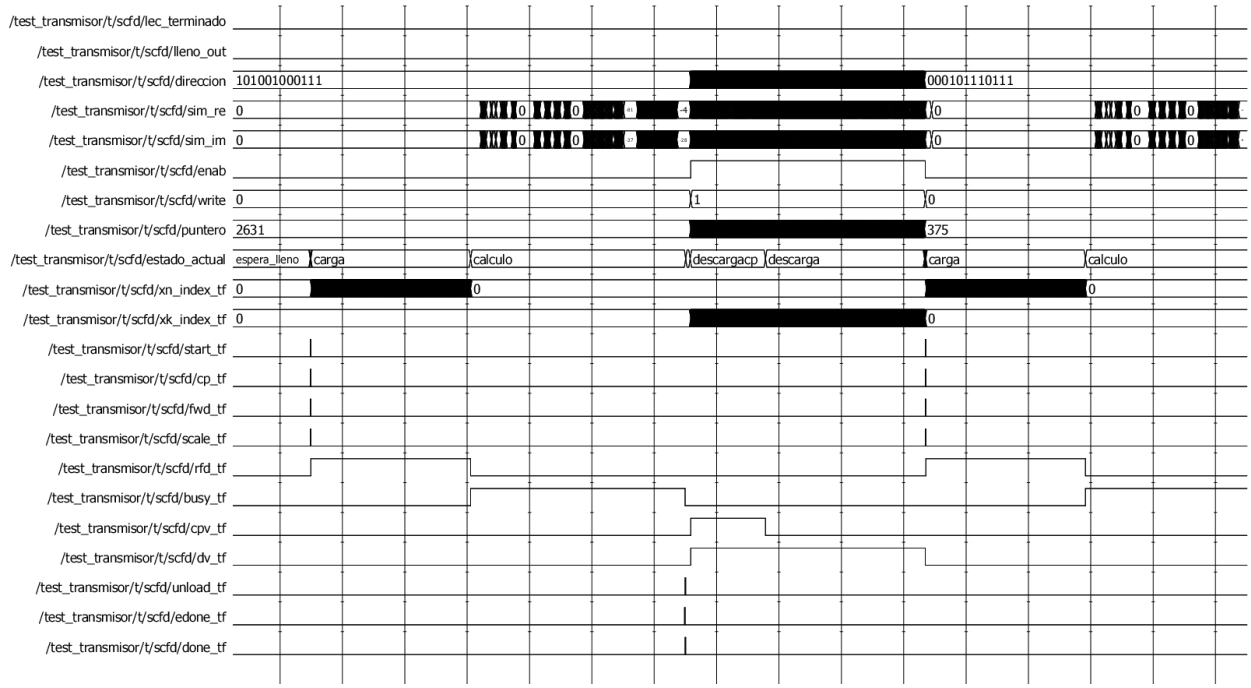


Ilustración 46: ModelSim SC-FDMA visión general

En la ilustración 47 se aprecia en primer lugar que los datos de PUCCH se cargan después de los datos PUSCH, al estar en otro slot, ha cambiado la paridad. Y en segundo lugar podemos ver cinco ciclos completos para la generación de un símbolo SC-FDMA.

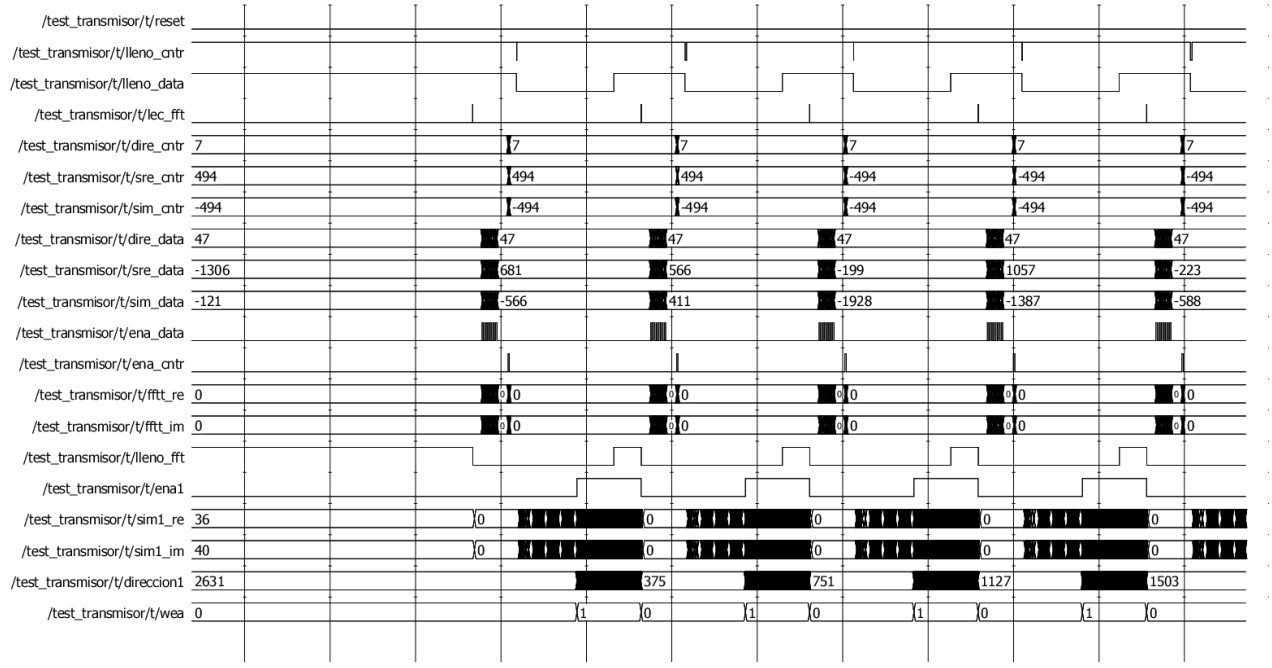


Ilustración 47: ModelSim simulación global

9. *Transmisor.*

El último bloque, consiste en juntar todo lo anterior en un único bloque, véase en la ilustración 48 y 50. Con esto conseguimos un único componente en la cual se inserta en los archivos de Lyrtech para poder compilarlo todo junto.

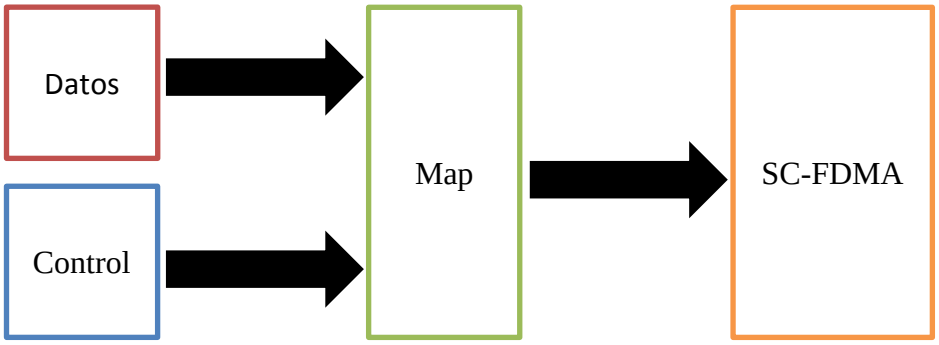


Ilustración 48: Diagrama de bloques

La única dificultad para generar este bloque consiste en generar dos relojes, uno para garantizar el ancho de banda del sistema, y otro para garantizar que el tiempo de slot se sea 0.5ms. Para generar estos relojes es simplemente realizando dos contadores, para poder ampliar el periodo del reloj original. Nótese que no es posible conseguir todas las frecuencias, ya que se dispone de un único reloj, y no siempre la frecuencia deseada es un sub múltiplo.

Por otro lado, tenemos una máquina de estados que tiene dos estados: parada y transmitir. Donde en la mitad del tiempo se transmite, y en la otra mitad del tiempo calcula todos los datos. Como se aprecia en la ilustración 48, en la mitad del tiempo transmitimos información, y en la otra mitad es cero.

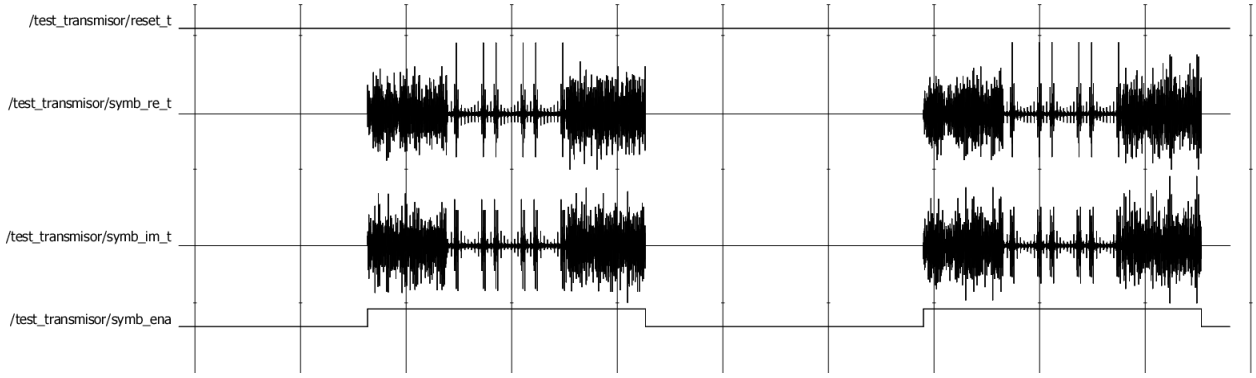


Ilustración 49: ModelSim Transmisor

Nótese que se debe introducir dos memorias RAM lo suficientemente grande como para almacenar hasta 1slot, que son 6 o 7 simbolos SC-FDMA con sus prefijos cíclicos correspondientes.

Las interfaces de entrada son las siguientes:

- Clk: el reloj del sistema.

Las interfaces de salida son las siguientes:

- Symb_re: señal real para transmitir.
- Symb_im: señal imaginaria para transmitir.
- Symb_ena: señal de habilitación para transmitir.

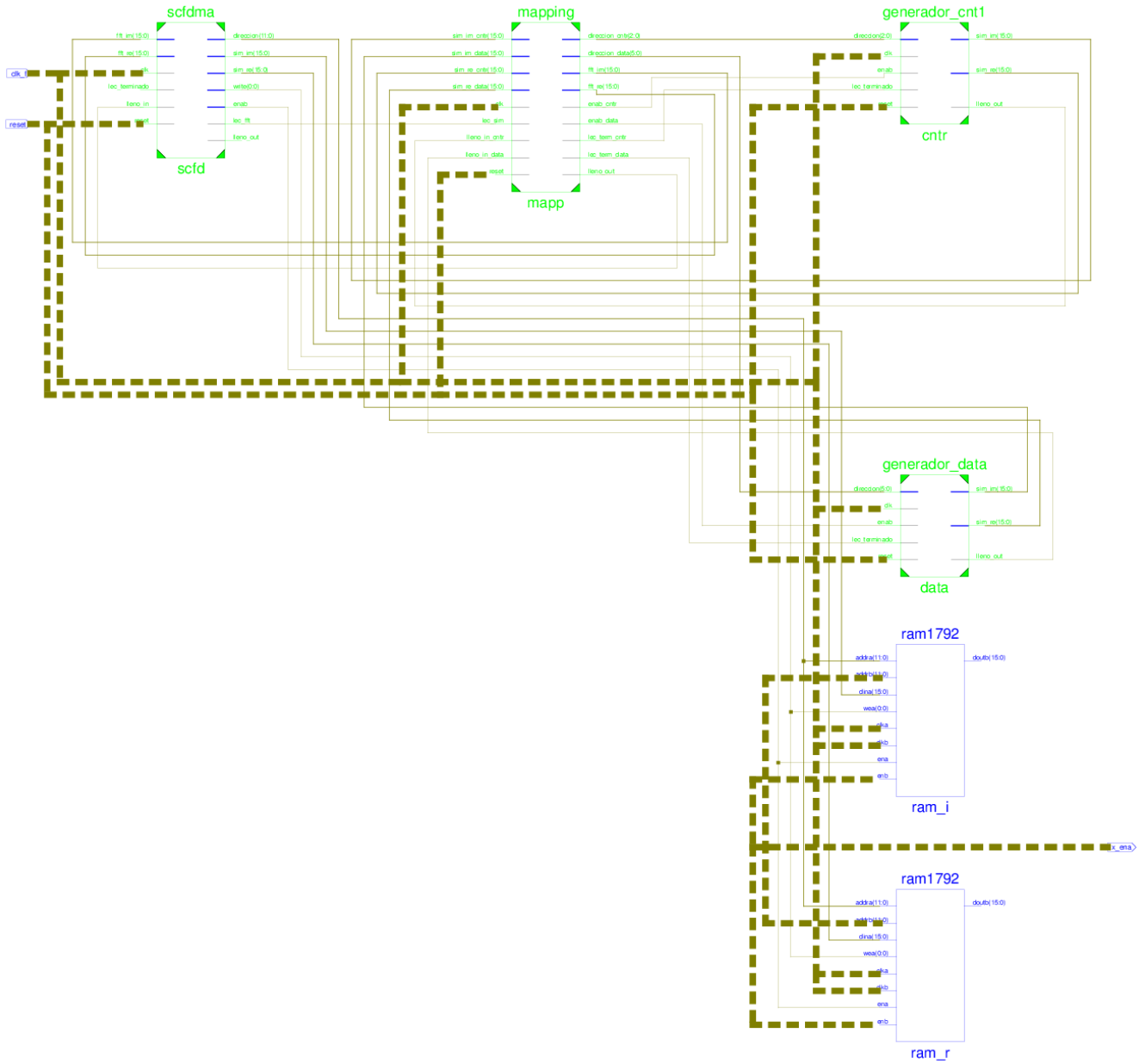


Ilustración 50: Transmisor

10. Ampliaciones.

En los apartados anteriores, se ha descrito cada bloque para el caso de 3MHz, que corresponde con 256 puntos para la FFT. Si se desea ampliar el ancho de banda, por ejemplo para 20MHz, todos los bloques funcionales se mantienen su estructura, y sólo hay que realizar una serie de modificaciones que se detallan a continuación:

- IP core FFT, cambiarlo por un bloque que soporte 2048 puntos.
- Ampliar el rango de los punteros que recorren desde 0 hasta 2047.
- Ampliar el rango del puntero para rellenar los ceros superiores e inferiores.
- Ampliación de las memorias RAM para el almacenamiento de los símbolos SC-FDMA.

Para contrastar los cambios sufridos entre el caso de 3MHz y 20MHz, simplemente hay que acudir al fichero de “biblioteca.vhd” de cada caso, para ver cuáles son los parámetros han cambiado y cuáles no.

5. Entorno de trabajo y resultados.

1. Entorno de trabajo.

Para llevar a cabo la implementación del sistema real, se realiza en un dispositivo Lyrtech SFF SDR, en la cual está formado por tres elementos principales: módulo de RF, módulo de conversión de datos (ADC y DAC) y un módulo de procesado digital. Véase en la ilustración 51 el resultado de su montaje y para más información acuda a las referencias [12], [13] y [14].

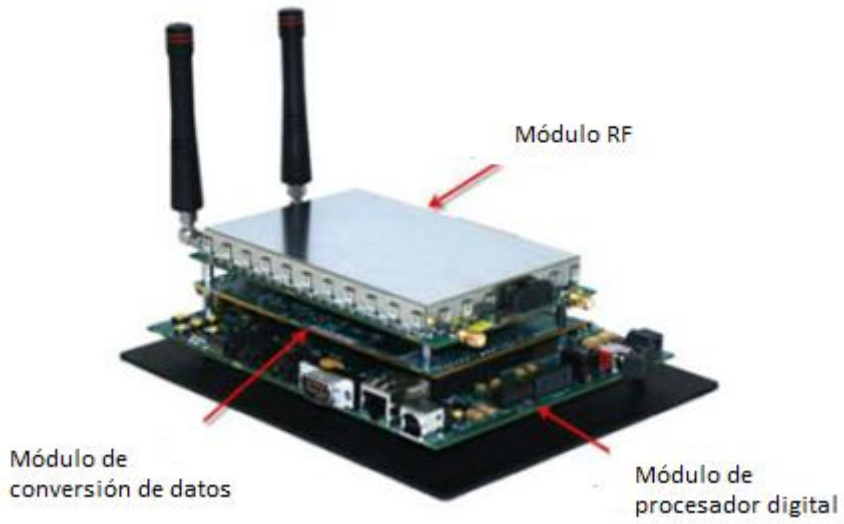


Ilustración 51: Lyrtech SFF SDR

Para configurar el dispositivo, se carga un archivo heredado de proyectos anteriores, dicho archivo inicializa cada uno de los componentes, además se elige que la frecuencia de transmisión del módulo de RF, siendo ésta de 370MHz.

En nuestro proyecto, nos vamos a centrar en la FPGA Virtex4 SX35, que está situado en el módulo de procesado digital. En ella vamos a cargar todos nuestros diseños en VHDL para poder realizar las simulaciones pertinentes.

Nótese que en el módulo de procesado digital está compuesto por una FPGA Virtex4 y un DSP5446. Se podría haber realizado la misma implementación en el DSP, sin embargo no se ha utilizado debido a dos grandes motivos que se detalla a continuación:

- El Code Composer Studio es mucho más difícil de manejar que el Xilinx ISE. Para poder utilizar el CCS con cierta soltura, se necesita mucha experiencia. A diferencia del Xilinx, es muy fácil de aprender y de manejar.
- Las FPGAs tiene una mayor velocidad de respuesta, se pueden ejecutar las tareas paralelamente, ya que cada bloque trabaja como una entidad autónoma. Sin embargo los DSP tiene como mucho dos CPUs, y para ejecutar todas tareas, tardarían mucho más que la FPGA, ya que los procesos se ejecutan en serie, y no en paralelo.

En comunicaciones, en la FPGA se suelen implementar códigos como: Red Solomon, Viterbi, etc. Que al fin y al cabo son algoritmos complejos que son críticos en el tiempo, y sólo la FPGA puede cumplir con dichos requisitos temporales.

2. *Caso ideal.*

A igual que hicimos en Matlab, en primer lugar vamos a diseñar un caso ideal, en la que sólo tenemos datos PUSCH y no tenemos datos de control, y se rellena toda la trama de información, sin dejar ninguna portadora sin rellenar. Por tanto una FFT y una IFFT del mismo tamaño, y no tenemos mapeadores de frecuencia, véase en la ilustración 52.

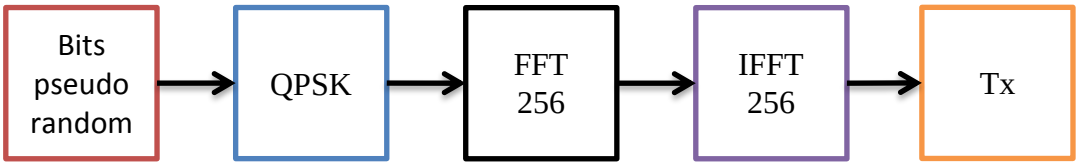
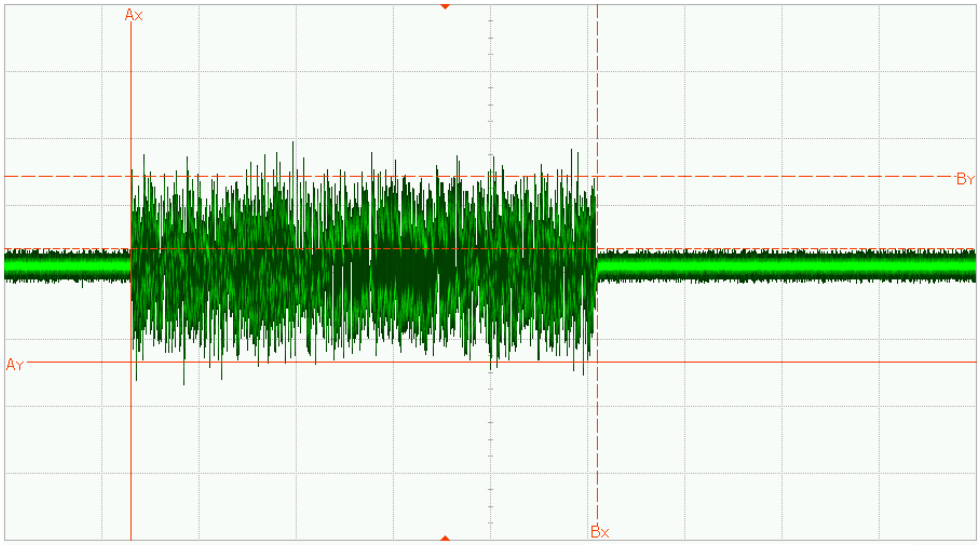


Ilustración 52: Diagrama de bloques del caso ideal

En la ilustración 53, podemos apreciar un único slot de 480us en el canal real, que se acerca mucho a los 0.5ms que indica el estándar [1]. La amplitud es de 138.71mV de pico a pico, no vamos a dar importancia la amplitud absoluta ya que dicho valor se puede modificar amplificando digitalmente en los registros en la FPGA o amplificando analógicamente en el DAC. Sin embargo, nos importa la amplitud desde un punto de vista relativo, los picos elevados e indeseados, como se aprecia en dicha ilustración, no vemos picos relativamente altos e indeseados en toda la trama. Esto nos indica a priori, que la modulación SC-FDMA está reduciendo los picos con respecto a OFDM. Pero no podemos olvidarnos que es un caso ideal que nunca se va a cumplir en la realidad ya que corresponde con un caso óptimo.



Acquisition: Sampling Mode real time, Normal
Memory Depth automatic 10.0000 Mpts
Sampling Rate automatic 10.0 GSa/s
Averaging off, Interpolation Auto

Channel 2: Scale 50.0 mV/, Offset 0.0 V, Skew 0.0 s

Horizontal: Scale 100 µs/, Position 46.6666666 µs, Reference center

Trigger: Mode edge, Source Channel 2, Level 18 mV, Slope rising
Sweep auto, Sensitivity normal, Holdoff Time 100 ns

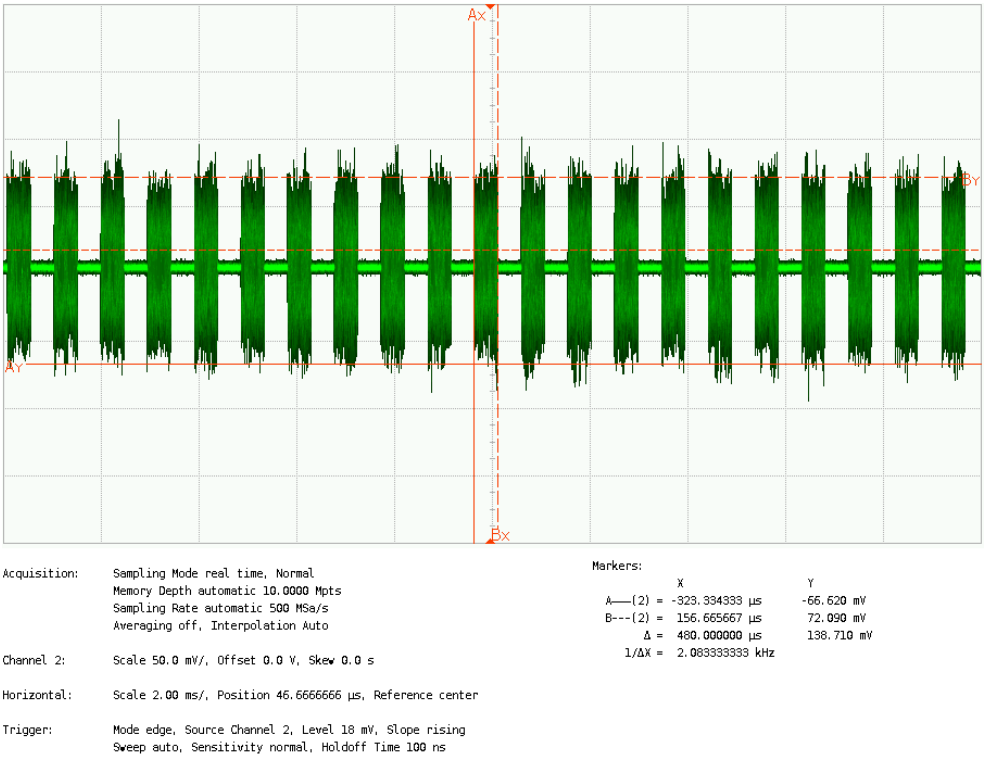
Markers:

	X	Y
A——(2) =	-323.3343334 µs	-66.62 mV
B---(2) =	156.6656666 µs	72.09 mV
Δ =	480.0000000 µs	138.71 mV
1/ΔX =	2.0833333333 kHz	

Ilustración 53: Un slot en banda base

Nótese que si hacemos más zoom en la señal, los pulsos generados por el dispositivo no son pulsos perfectos y planos, sino son pulsos con una pendiente diferente de cero. Para más información acuda al Apéndice A.

En la ilustración 54, no es más que alejar el zoom del ilustración 53, podemos apreciar que estamos transmitiendo de manera alternada, la FPGA aprovecha para calcular toda la información en los huecos. Aquí podemos ver que en toda la figura sólo hay un único pico en el tercer slot de los 21 en total.



Acquisition: Sampling Mode real time, Normal
Memory Depth automatic 10.0000 Mpts
Sampling Rate automatic 500 MSa/s
Averaging off, Interpolation Auto

Channel 2: Scale 50.0 mV/, Offset 0.0 V, Skew 0.0 s

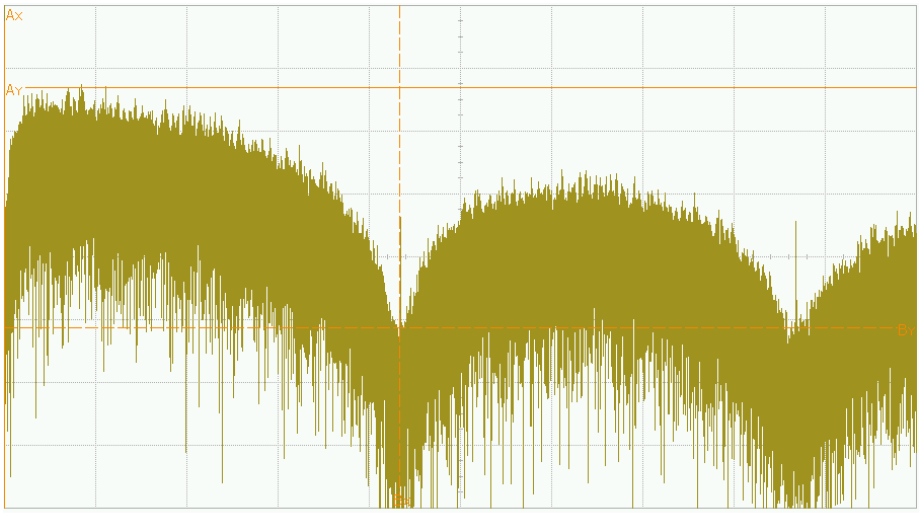
Horizontal: Scale 2.00 ms/, Position 46.6666666 µs, Reference center

Trigger: Mode edge, Source Channel 2, Level 18 mV, Slope rising
Sweep auto, Sensitivity normal, Holdoff Time 100 ns

Markers:

	X	Y
A——(2) =	-323.334333 µs	-66.620 mV
B---(2) =	156.665667 µs	72.090 mV
Δ =	480.000000 µs	138.710 mV
1/ΔX =	2.083333333 kHz	

Ilustración 54: Varios slots en banda base



Acquisition: Sampling Mode real time, Normal
Memory Depth automatic 863.753 kpts
Sampling Rate automatic 125 MSa/s
Averaging off, Interpolation Auto

Horizontal: Scale 691 µs/, Position -6.4977977778 ms, Reference center

Trigger: Mode edge, Source Channel 2, Level 18 mV, Slope rising
Sweep auto, Sensitivity normal, Holdoff Time 100 ns

Function 1: FFT magnitude Channel 2
Vertical Scale 9.17339 dB/, Offset -75.8225 dBm
Horizontal Scale 864 kHz/, Center Frequency 4.319419 MHz
Window Hamming, Resolution 217.1 Hz

Markers:

	X	Y
A——(f1) =	0 Hz	-51.114 dBm
B---(f1) =	3.74341 MHz	-86.180 dBm
Δ =	3.74341 MHz	-35.066 dB
1/ΔX =	267.1361 ns	

Ilustración 55: Ancho de banda en banda base

En la ilustración 55, se aprecia la FFT realizada por el osciloscopio, podemos apreciar que tenemos un ancho de banda entre nulos de 3.7434MHz, que coincide justamente con el reloj que se ha introducido en la IFFT.

$$f_{clk} = 37.5MHz \rightarrow f_{ifft} = \frac{f_{clk}}{10} = 3.75MHz \cong 3.7434MHz$$

En las ilustraciones anteriores, hemos representado sólo el canal real y no se ha representado el canal imaginario, ya que sería igual que el canal real en la forma de onda. Por tanto, todas las ilustraciones de este proyecto, sólo se representará la parte real siempre que estemos en banda base.

Una vez medido la señal en banda base, se va a caracterizar la señal en RF, ya que se dispone de un modulador IQ. Como se aprecia en la ilustración 56, se presenta un slot en radio frecuencia. Como se aprecia, disponemos de picos elevados y muy frecuentes en el slot, esto es debido a una distorsión desconocida del DAC, si hacemos zoom en el slot, alguno de esos picos son pulsos cuadrados y otros son deltas muy estrechas. Por tanto el DAC nos está estropeando la señal, introduciendo una distorsión en ella.

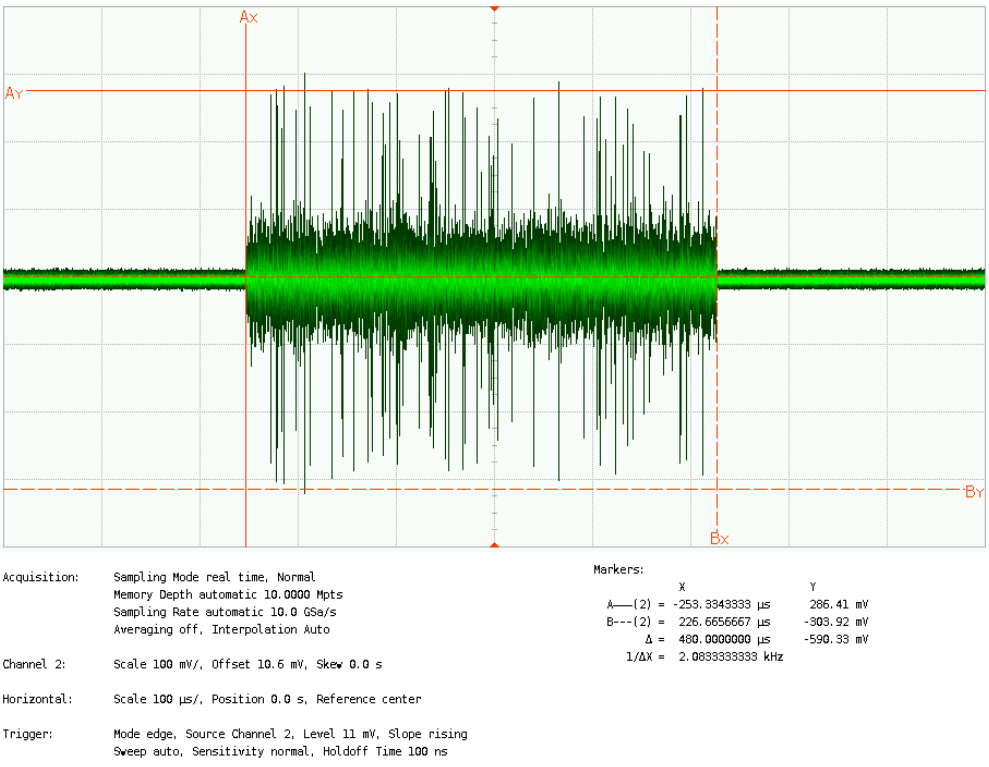
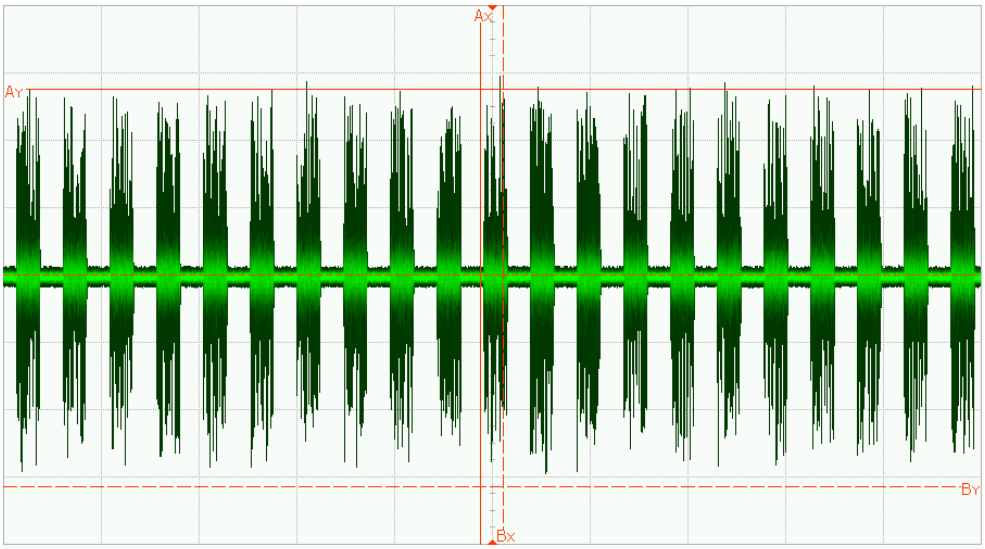


Ilustración 56: Un slot en radiofrecuencia

Como dicha señal se va a exportar en un archivo CSV, para tratarlo posteriormente con Matlab, estos picos o deltas nos van a estropear considerablemente la PAPR final. Por tanto para eliminar estas deltas se emplea una técnica muy común en el mundo del tratamiento digital de la señal para eliminar los "outliers" mediante la mediana. Se divide el slot en 7 trozos, y por cada trozo se ejecuta el algoritmo de la mediana que está detallado en el apéndice A.

En la ilustración 57 se aprecia varios slots seguidos de transmisión, y que a todas les contaminan picos indeseados introducidos por el DAC.



Acquisition: Sampling Mode real time, Normal
Memory Depth automatic 10.0000 Mpts
Sampling Rate automatic 500 MSa/s
Averaging off, Interpolation Auto

Channel 2: Scale 100 mV/, Offset 10.6 mV, Skew 0.0 s

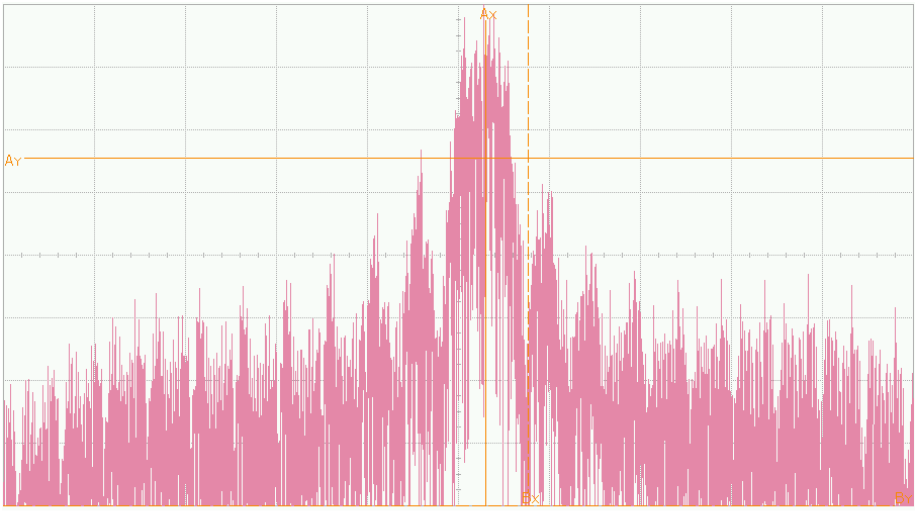
Horizontal: Scale 2.00 ms/, Position 0.0 s, Reference center

Trigger: Mode edge, Source Channel 2, Level 11 mV, Slope rising
Sweep auto, Sensitivity normal, Holdoff Time 100 ns

Markers:

	X	Y
A——(2) =	-253.334333 µs	286.41 mV
B---(2) =	226.665667 µs	-303.92 mV
Δ =	480.000000 µs	-590.33 mV
1/ΔX =	2.083333333 kHz	

Ilustración 57: Varios slots en radiofrecuencia



Acquisition: Sampling Mode real time, Normal
Memory Depth automatic 940.002 kpts
Sampling Rate automatic 10.0 GSa/s
Averaging off, Interpolation Auto

Horizontal: Scale 9.40 µs/, Position -6.7568698379 ms, Reference right

Trigger: Mode edge, Source Channel 2, Level 18 mV, Slope rising
Sweep auto, Sensitivity normal, Holdoff Time 100 ns

Function 4: FFT magnitude Channel 2
Vertical Scale 4.24883 dB/, Offset -57.6878 dBm
Horizontal Scale 7.84 MHz/, Center Frequency 409.053986 MHz
Window Hamming, Resolution 15.96 kHz

Markers:

	X	Y
A——(f4) =	372.1651 MHz	-51.114 dBm
B---(f4) =	375.8540 MHz	-86.180 dBm
Δ =	3.6889 MHz	-35.066 dB
1/ΔX =	271.084 ns	

Ilustración 58: Ancho de banda en radiofrecuencia

Una vez visto la señal en el tiempo, pasamos al dominio de la frecuencia mediante la FFT del osciloscopio, podemos apreciar en la ilustración 58 que nuestra portadora está en 372MHz, muy cercano de los 370MHz que le hemos configurado. En dicha ilustración se aprecia que la mitad del ancho de banda de nuestra señal en el primer cero es de 3.7MHz, el mismo ancho de banda entre

ceros que en el caso en banda base. Y por último indicar que existen armónicos, dicho armónicos puede ser introducido por diversos motivos, el más importante es el DAC. Según sus especificaciones, el DAC introduce armónicos cuando realiza una conversión de datos, y si a esto le añadimos que en el transmisor no se dispone de un filtro selectivo en frecuencia, estamos dejando pasar todos los armónicos producido por circuitos anteriores.

En la ilustración 59, podemos ver una ampliación de nuestra señal en el dominio de la frecuencia, podemos ver con mejor precisión, el ancho de banda entre nulos de nuestra señal.

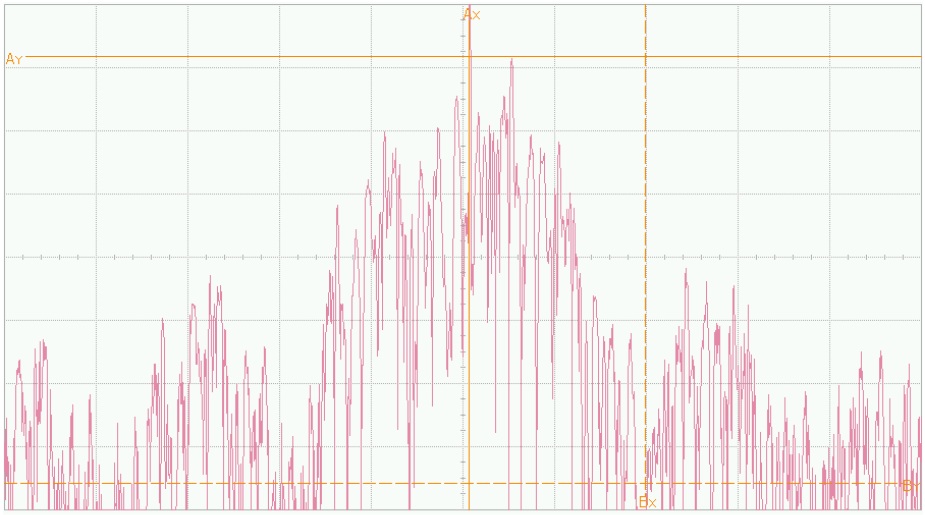


Ilustración 59: Ancho de banda en radiofrecuencia ampliado

Una vez que hemos visto cualitativamente nuestra señal generada por el dispositivo SFF SDR, recogemos la señal y se procede a calcular la PAPR, como ya se ha especificado anteriormente, a cada slot se le divide entre 7, ya que en un slot tenemos símbolos SC-FDMA. Y se aplica el algoritmo de la mediana para filtrar los picos indeseados. Como se aprecia en la ilustración 60, si comparamos el caso ideal generado por Matlab y el caso ideal generado por el dispositivo, la diferencia es de 1.5dB aproximadamente.

Es comprensible que el caso ideal generado por el dispositivo tenga una PAPR mayor, pero el principal culpable de este empeoramiento es el DAC. Sabemos que la señal de salida del canal real o del canal imaginario posee un valor ± 457 que corresponde con la amplitud de la modulación QPSK, ya que la FFT se cancela con la IFFT. En la simulación de ModelSim, se verifica que los datos poseen los niveles de la QPSK, y por tanto la señal que deberíamos de recibir debería tener solo y exclusivamente dos niveles de tensión, un nivel de tensión correspondiente para +457 y otro para -457. Como se ha apreciado en la ilustración 53 y 54, no se aprecia que haya dos niveles de tensión, sino que se aprecia diversos niveles de amplitud, incluso algún que otro pulso de elevada amplitud.

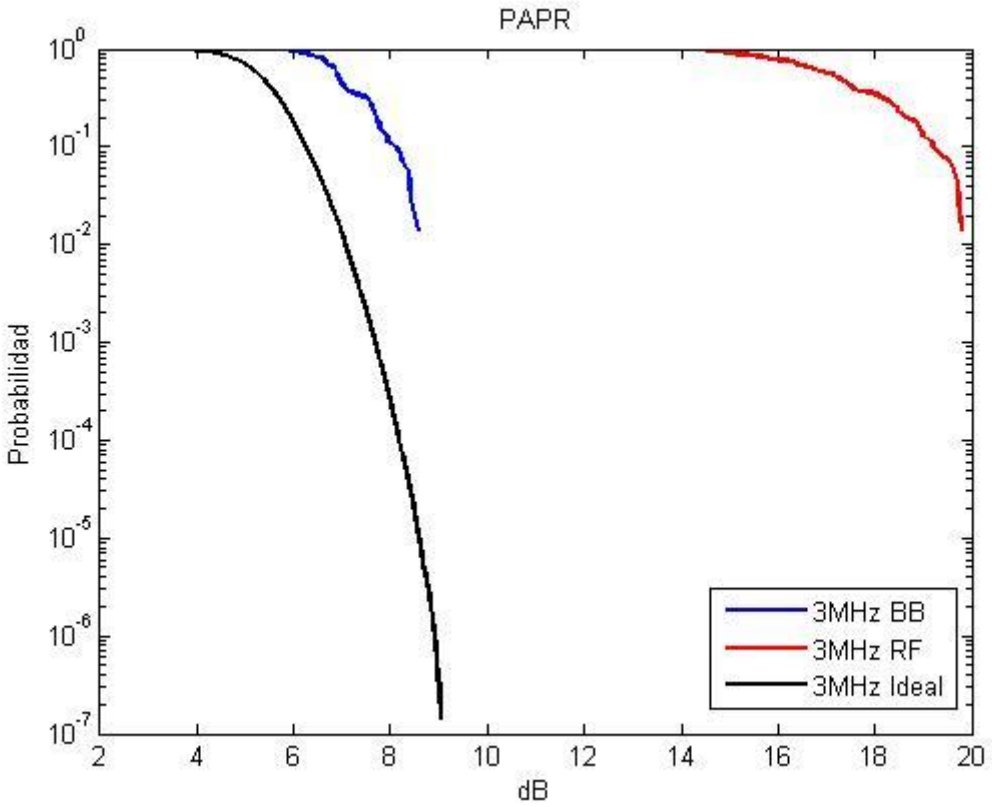


Ilustración 60: Comparativa caso ideal de Matlab vs caso ideal implementado

Por otro lado, podemos apreciar en la ilustración 60, la señal en radiofrecuencia posee una PAPR mucho más elevada que en banda base, donde la amplitud máxima es inferior a los 20dB. La señal en radiofrecuencia debe tener mayor PAPR que en banda base, ya que estamos multiplicando nuestra señal por una señal sinusoidal que es una señal monótono creciente, pasando por todos los niveles de amplitud intermedio. Por tanto estamos reduciendo la media y elevando la PAPR.

3. Caso real 256 puntos.

Tras estudiar el caso ideal, se va a adoptar un caso real introduciendo todos los bloques que se ha descrito en el capítulo anterior. Este caso se denominará como el caso estándar de 3MHz que tiene la siguiente configuración:

- La IFFT tiene 256 puntos
- Posee un canal de control PUCCH1.
- No tiene señales de piloto SRS.
- La modulación es QPSK.
- El prefijo cíclico es normal.
- DRS.

Para realizar simulaciones de otros casos, sólo tenemos que modificar uno de los parámetros y mantener constante el resto.

Como se aprecia en la ilustración 61, disponemos de un slot que dura 493us aproximadamente. Si nos fijamos en la forma de onda se parece mucho a la señal simulada por el ModelSim en la ilustración 49. Podemos ver que en el medio hay muchos ceros, ya que corresponde con los 3 símbolos SC-FDMA que son DRS, y en los extremos se dispone de una señal diferente de ceros, ya que son los 4 símbolos SC-FDMA que es PUSCH y PUCCH1.

Además podemos apreciar que ha crecido en número los pulsos de amplitud elevado con respecto al caso ideal. Esto es debido a que en el caso ideal transmitimos 256 símbolos QPSK, y aquí estamos transmitiendo sólo 48 símbolos, y el resto de sub portadoras están rellenas con ceros.

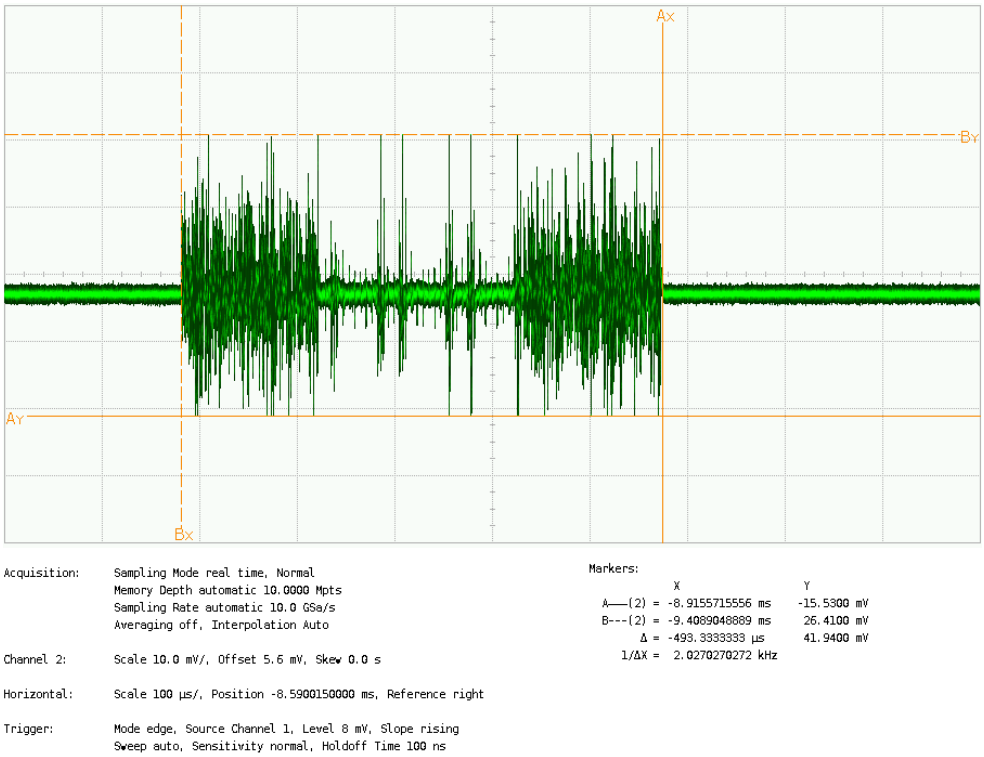
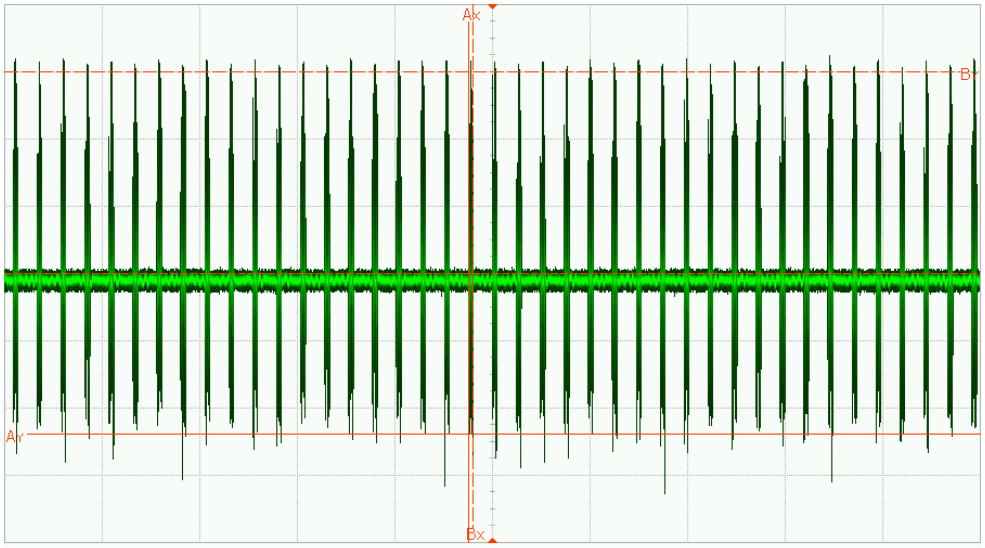


Ilustración 61: Un slot en banda base

En la ilustración 62 podemos ver cómo estamos transmitiendo varios slots, dejando un slot vacío entre slots de información. Podemos ver que esta vez, la variación de la señal es mayor que la variación de la señal del caso ideal, y por tanto esto empeorará la PAPR.

En la ilustración 63, se aprecia la señal en el dominio de la frecuencia, podemos apreciar que esta vez, el ancho de banda entre nulos es mayor que 5MHz. Para nuestra sorpresa el ancho de banda casi se duplica con respecto al caso ideal, esto es debido a la inserción del prefijo cíclico. Estamos comprimiendo en un slot de 0.5 ms la señal 7 símbolos SC-FDMA que posee una longitud de 256 puntos, a esto hay que añadirle la longitud del prefijo cíclico a cada símbolo SC-FDMA. Y es debido a esto a que el ancho de banda crece considerablemente.



Acquisition: Sampling Mode real time, Normal
Memory Depth automatic 10.0000 Mpts
Sampling Rate automatic 100 MSa/s
Averaging off, Interpolation Auto

Channel 2: Scale 10.0 mV/, Offset 5.4 mV, Skew 0.0 s

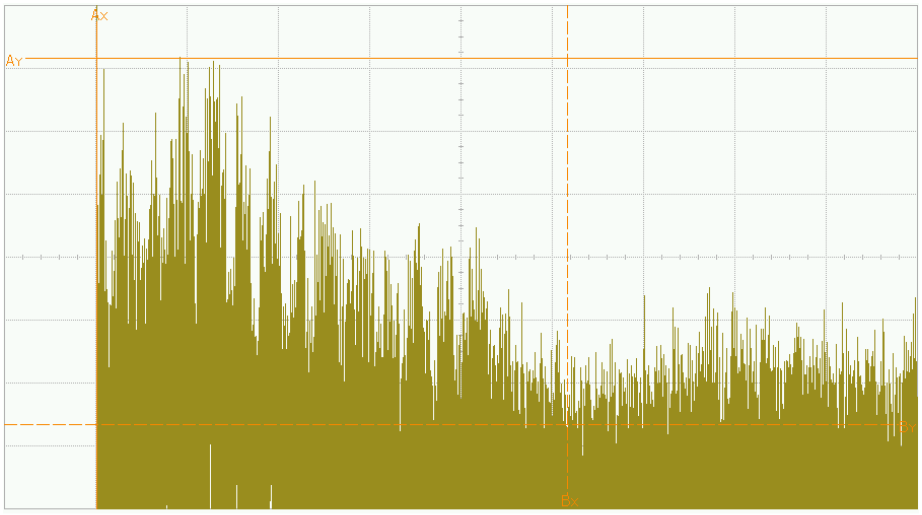
Horizontal: Scale 10.0 ms/, Position 0.0 s, Reference center

Trigger: Mode edge, Source Channel 2, Level 5.4 mV, Slope rising
Sweep auto, Sensitivity normal, Holdoff Time 100 ns

Markers:

	X	Y
A——(2) =	-2.45556556 ms	-18.470 mV
B---(2) =	-1.96667667 ms	35.400 mV
Δ =	488.88889 µs	53.870 mV
1/ΔX =	2.04545455 kHz	

Ilustración 62: Varios slots en banda base



Acquisition: Sampling Mode real time, Normal
Memory Depth automatic 1.00000 Mpts
Sampling Rate automatic 200 MSa/s
Averaging off, Interpolation Auto

Horizontal: Scale 500 µs/, Position -8.5900150000 ms, Reference right

Trigger: Mode edge, Source Channel 1, Level 8 mV, Slope rising
Sweep auto, Sensitivity normal, Holdoff Time 100 ns

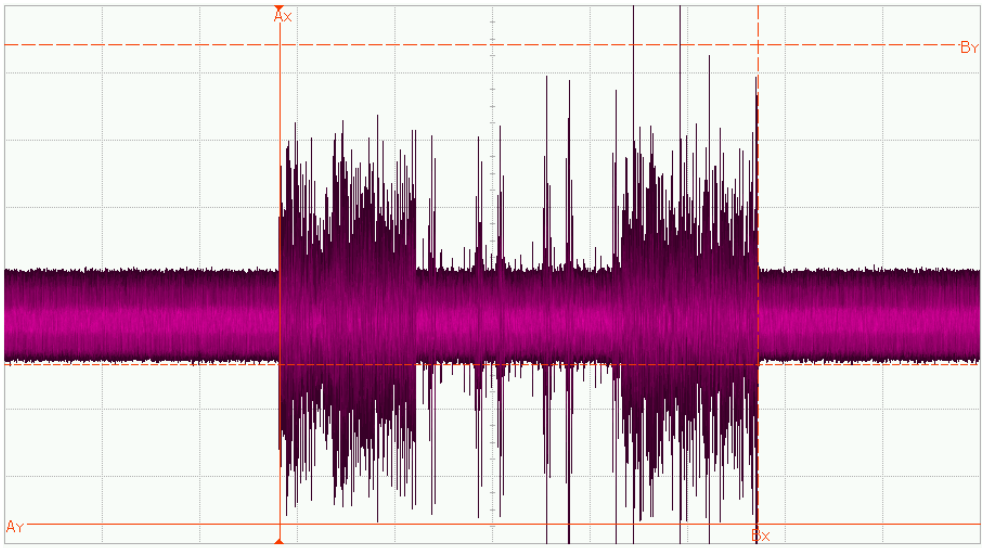
Function 1: FFT magnitude Channel 1
Vertical Scale 3.00000 dB/, Offset -75.9600 dBm
Horizontal Scale 989 kHz/, Center Frequency 8.886120 MHz
Window Hamming, Resolution 300.0 Hz

Markers:

	X	Y
A——(f1) =	0 Hz	-66.476 dBm
B---(f1) =	5.09494 MHz	-83.944 dBm
Δ =	5.09494 MHz	-17.468 dB
1/ΔX =	196.2730 ns	

Ilustración 63: Ancho de banda en banda base

Una vez que se ha visto la señal en la banda base, se va a mostrar las mismas señales en la radiofrecuencia. En la ilustración 64 se muestra un único slot y en la ilustración 65 se muestra varios slots. Lo que cabe de destacar es el aumento de picos o deltas con respecto a la banda base. Esto es debido a que el canal imaginario del DAC introduce más picos o deltas que el canal real, por tanto uno de los dos canales posee una mayor degradación.



Acquisition: Sampling Mode real time, Normal
Memory Depth automatic 10.0000 Mpts
Sampling Rate automatic 10.0 GSa/s
Averaging off, Interpolation Auto

Channel 4: Scale 10.0 mV/, Offset 13.4 mV, Skew 0.0 s

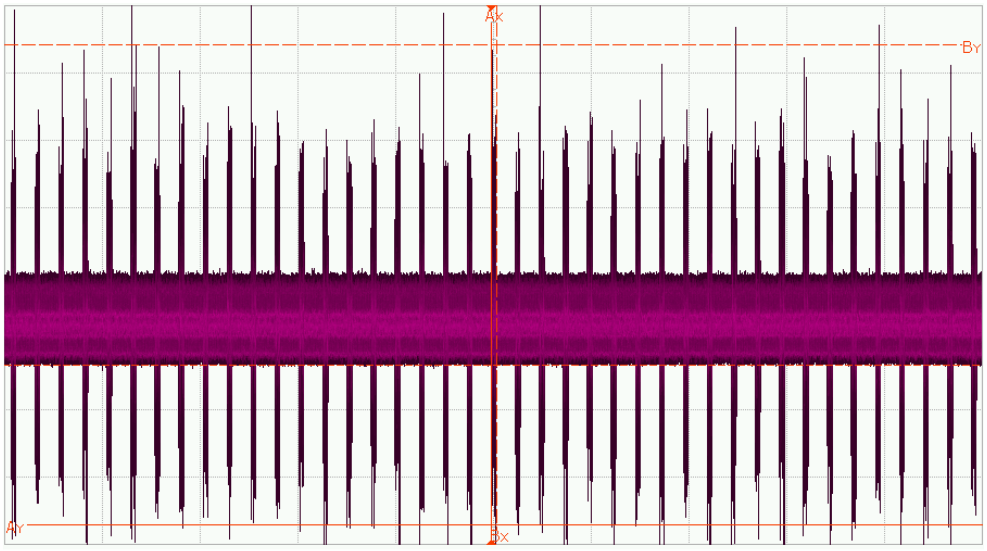
Horizontal: Scale 100 µs/, Position 218.8888889 µs, Reference center

Trigger: Mode edge, Source Channel 4, Level 0.0 V, Slope rising
Sweep auto, Sensitivity normal, Holdoff Time 100 ns

Markers:

	X	Y
A——(4) =	1.1101111 µs	-23.700 mV
B---(4) =	491.1101111 µs	47.590 mV
Δ =	490.0000000 µs	71.290 mV
1/ΔX =	2.0408163265 kHz	

Ilustración 64: Un slot en radiofrecuencia



Acquisition: Sampling Mode real time, Normal
Memory Depth automatic 10.0000 Mpts
Sampling Rate automatic 100 MSa/s
Averaging off, Interpolation Auto

Channel 4: Scale 10.0 mV/, Offset 13.4 mV, Skew 0.0 s

Horizontal: Scale 10.0 ms/, Position 218.8888889 µs, Reference center

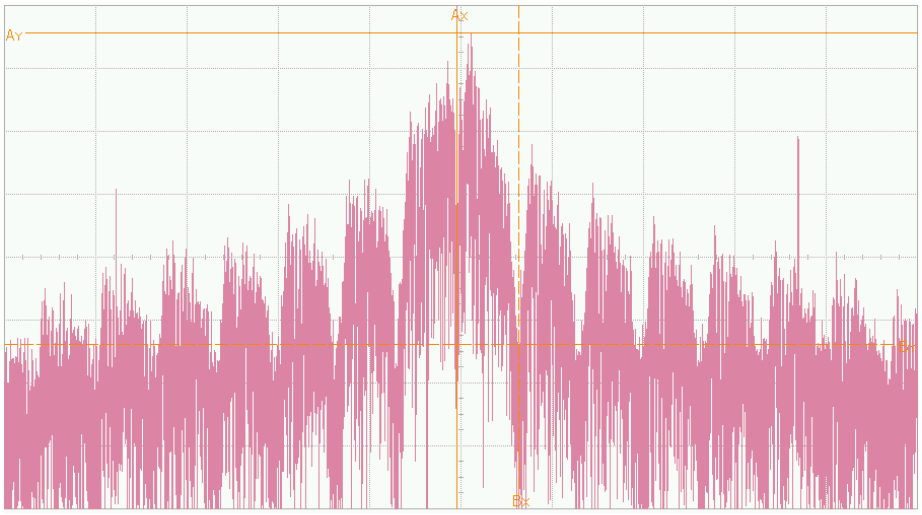
Trigger: Mode edge, Source Channel 4, Level 0.0 V, Slope rising
Sweep auto, Sensitivity normal, Holdoff Time 100 ns

Markers:

	X	Y
A——(4) =	1.11011 µs	-23.700 mV
B---(4) =	491.11011 µs	47.590 mV
Δ =	490.00000 µs	71.290 mV
1/ΔX =	2.04081633 kHz	

Ilustración 65: Varios slots en radiofrecuencia

Por último se muestra la señal en el dominio de la frecuencia montado en la portadora de 370MHz. En la ilustración 66 podemos apreciar nuestra señal y sus armónicos, y en la ilustración 67 podemos apreciar con mayor detalle nuestra señal en banda.



Acquisition: Sampling Mode real time, Normal
Memory Depth automatic 1.00500 Mpts
Sampling Rate automatic 5.00 GSa/s
Averaging off, Interpolation Auto

Horizontal: Scale 20.1 µs/, Position -733.0225681 µs, Reference right

Trigger: Mode edge, Source Channel 4, Level 7.3 mV, Slope rising
Sweep auto, Sensitivity normal, Holdoff Time 100 ns

Function 4: FFT magnitude Channel 4
Vertical Scale 7.46529 dB/, Offset -80.7698 dBm
Horizontal Scale 8.03 MHz/, Center Frequency 412.488204 MHz
Window Hamming, Resolution 7.463 kHz

Markers:

	X	Y
A——(f4) =	371.9814 MHz	-54.160 dBm
B---(f4) =	377.4240 MHz	-91.125 dBm
Δ =	5.4426 MHz	-36.965 dB
1/ΔX =	183.737 ns	

Ilustración 66: Ancho de banda en radiofrecuencia

Acquisition: Sampling Mode real time, Normal
Memory Depth automatic 833.003 kpts
Sampling Rate automatic 10.0 GSa/s
Averaging off, Interpolation Auto

Horizontal: Scale 8.33 µs/, Position -221.9535943 µs, Reference right

Trigger: Mode edge, Source Channel 2, Level 38.5 mV, Slope rising
Sweep auto, Sensitivity normal, Holdoff Time 100 ns

Function 4: FFT magnitude Channel 4
Vertical Scale 6.36164 dB/, Offset -73.0314 dBm
Horizontal Scale 2.34 MHz/, Center Frequency 383.769012 MHz
Window Hanning, Resolution 18.01 kHz

Markers:

	X	Y
A——(f4) =	371.9730 MHz	-92.317 dBm
B---(f4) =	377.3470 MHz	-55.507 dBm
Δ =	5.3741 MHz	36.811 dB
1/ΔX =	186.079 ns	

Ilustración 67: Ancho de banda en radiofrecuencia ampliado

La PAPR de este caso, se mostrará en los casos siguientes, ya que vamos a tomar este caso como el caso estándar y se realizarán las comparaciones con sus variantes.

4. *Caso real 256 puntos con SRS.*

Una vez que tenemos el caso estándar definido, se va a empezar a cambiar la configuración de la trama. En primer lugar se va a activar las señales de Piloto SRS. Como se aprecia en la ilustración 67, se dispone de un slot muy contaminado por picos o deltas indeseados. A simple vista parece que hay muchos picos, es sólo un efecto visual del osciloscopio, si aplicamos zoom en el slot, los picos o deltas son mucho más estrechos y son menos frecuentes de lo que parece.

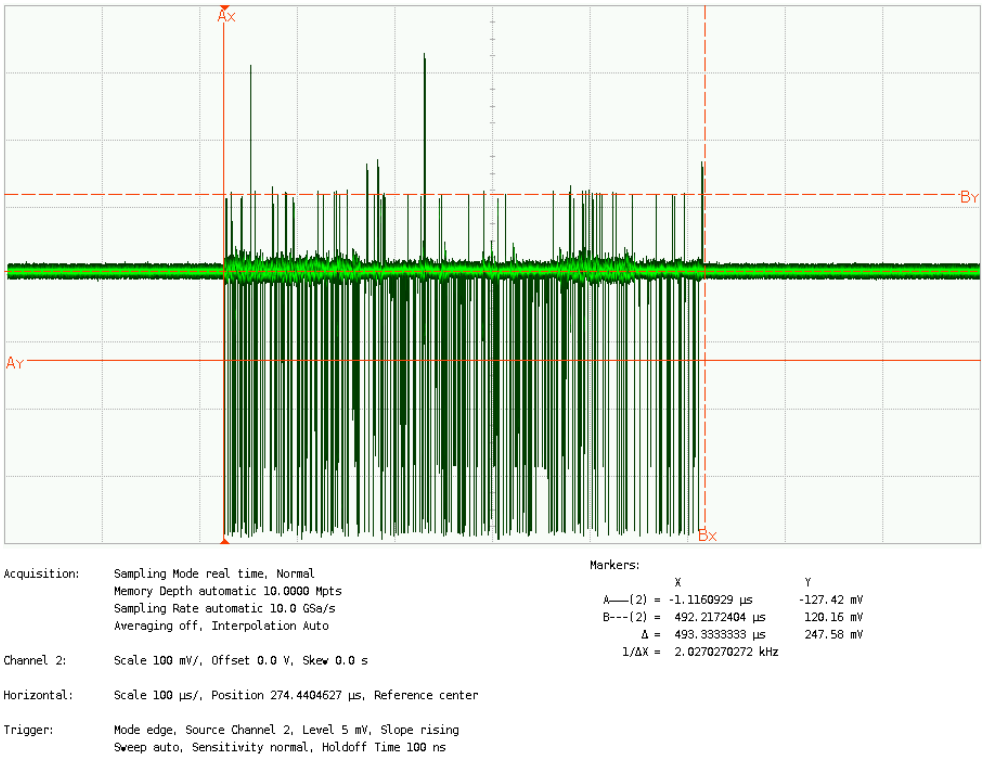


Ilustración 68: Un slot SRS

Aunque haya muchos picos en la zona inferior, podemos apreciar en la zona superior que hay mucho más ceros que el caso anterior, esto tiene sentido ya que símbolo SC-FDMA del piloto SRS contiene el 50% de ceros.

Como se aprecia en la ilustración 69, la señal con pilotos SRS posee mayor PAPR que la señal sin los pilotos SRS, para una probabilidad de 10^{-2} la diferencia entre una trama con SRS y otra sin SRS es de 1.2dB aproximadamente. En caso de la radiofrecuencia, la PAPR empeora 2dB con respecto a la banda base, y entre ellas mantiene las prestaciones en banda base.

Una de las cosas que nos llama la atención es la forma cualitativa de la PAPR de la señal en banda base que posee zonas casi planas, y en radio frecuencia está mucho más redondeado. Esto es debido a que al multiplicar con una señal sinusoidal, tenemos más valores intermedios, y por tanto se nos genera diferentes PAPR diferentes, generando así una curva más suave.

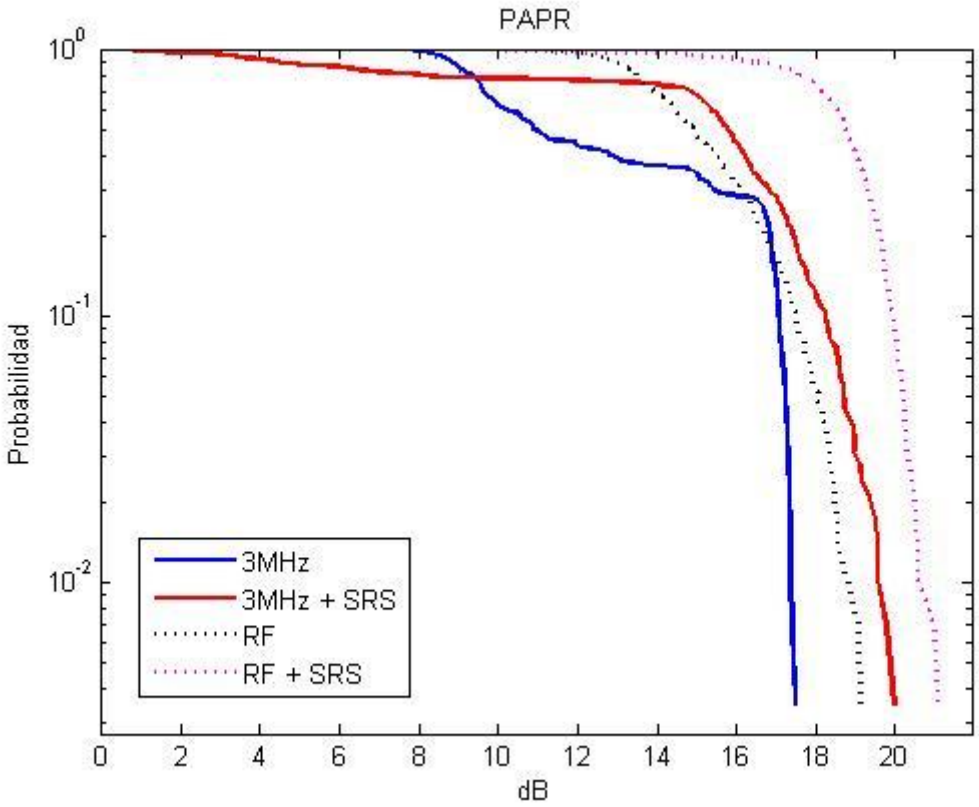


Ilustración 69: Comparativa SRS

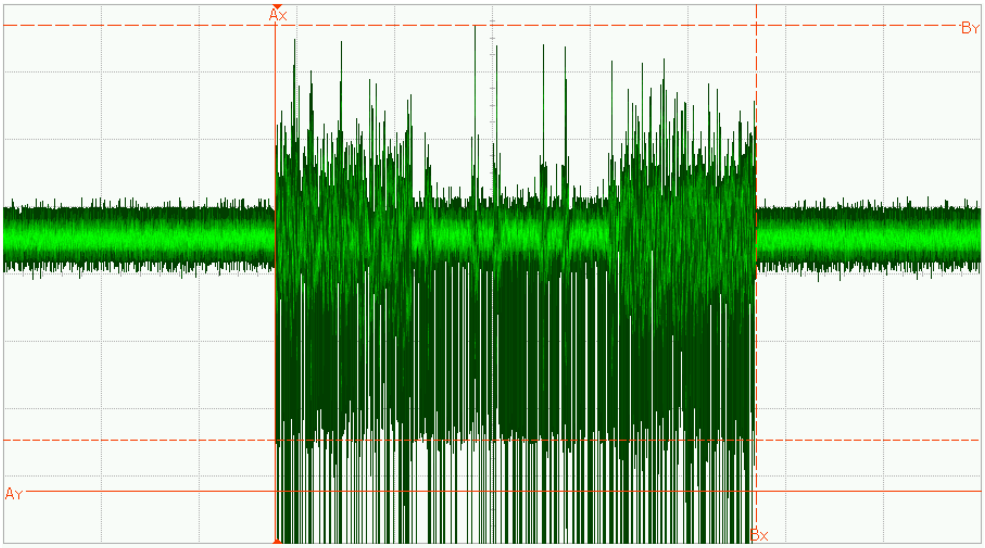
5. *Caso real 256 puntos con diferentes PUCCHs.*

En este caso se va a estudiar las diferentes PAPR en función de los diferentes esquemas de PUCCH que ha propuesto el estándar [1].

En la ilustración 70 se muestra un slot con un PUCCH del tipo 2, y en la ilustración 71 se muestra un slot con PUCCH del tipo 3. Como se aprecia a simple vista, en la ilustración 70 nos encontramos con el mismo caso de los picos o deltas que nos distorsiona la señal. Además en la ilustración 71, podemos apreciar que no hay ninguna distorsión, y además podemos notar la forma peculiar del slot. Dicha forma peculiar puede ser debido a la FFT extra que se introduce en el PUCCH del tipo 3.

Una vez que tenemos los tres tipos de formatos posibles del canal de control, se realiza una comparativa de los tres esquemas en banda base y en radio frecuencia. Podemos apreciar en la ilustración 72 que el formato 2 presenta la peor PAPR, y para nuestra sorpresa el formato 3 presenta la mejor PAPR a diferencia del caso simulado en Matlab, donde el formato 1 y 2 presentan casi la misma prestación.

Para una probabilidad de 10^{-2} en banda base el formato 3 mejora con respecto al formato 1 menos de 0.5dB, y el formato 2 empeora con respecto al formato 1 sobre 1dB. Para el caso de la radio frecuencia, se empeora unos 2dB con respecto a la banda base, y las relaciones entre los diferentes formatos se mantienen aproximadamente igual. Nótese que para el peor caso, la PAPR máxima es inferior a los 20dB, igual que en el caso ideal.



Acquisition: Sampling Mode real time, Normal
Memory Depth automatic 10.0000 Mpts
Sampling Rate automatic 10.0 GSa/s
Averaging off, Interpolation Auto

Channel 2: Scale 10.0 mV/, Offset -1.6 mV, Skew 0.0 s

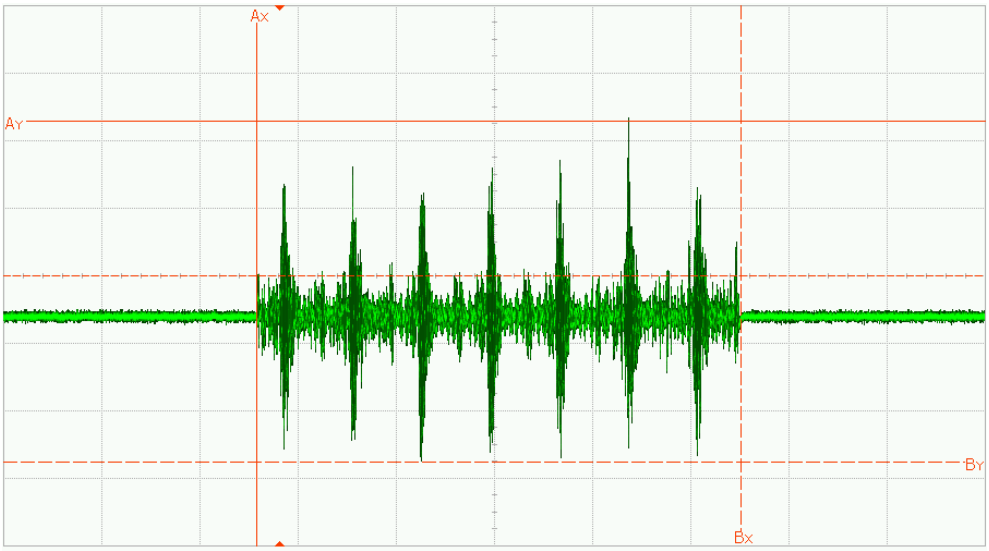
Horizontal: Scale 100 µs/, Position 219.9960179 µs, Reference center

Trigger: Mode edge, Source Channel 2, Level -26.2 mV, Slope rising
Sweep auto, Sensitivity normal, Holdoff Time 100 ns

Markers:

	X	Y
A——(2) =	-2.227204 µs	-33.860 mV
B---(2) =	489.995018 µs	35.340 mV
Δ =	492.222222 µs	69.200 mV
1/ΔX =	2.031602709 kHz	

Ilustración 70: Un slot PUCCH2 en banda base



Acquisition: Sampling Mode real time, Normal
Memory Depth automatic 10.0000 Mpts
Sampling Rate automatic 10.0 GSa/s
Averaging off, Interpolation Auto

Channel 2: Scale 50.0 mV/, Offset 35.0 mV, Skew 0.0 s

Horizontal: Scale 100 µs/, Position 218.2262294 µs, Reference center

Trigger: Mode edge, Source Channel 2, Level 35 mV, Slope rising
Sweep auto, Sensitivity normal, Holdoff Time 100 ns

Markers:

	X	Y
A——(2) =	-23.996993 µs	149.520 mV
B---(2) =	469.336341 µs	-102.900 mV
Δ =	493.333333 µs	-252.420 mV
1/ΔX =	2.027027027 kHz	

Ilustración 71: Un slot PUCCH3 en banda base

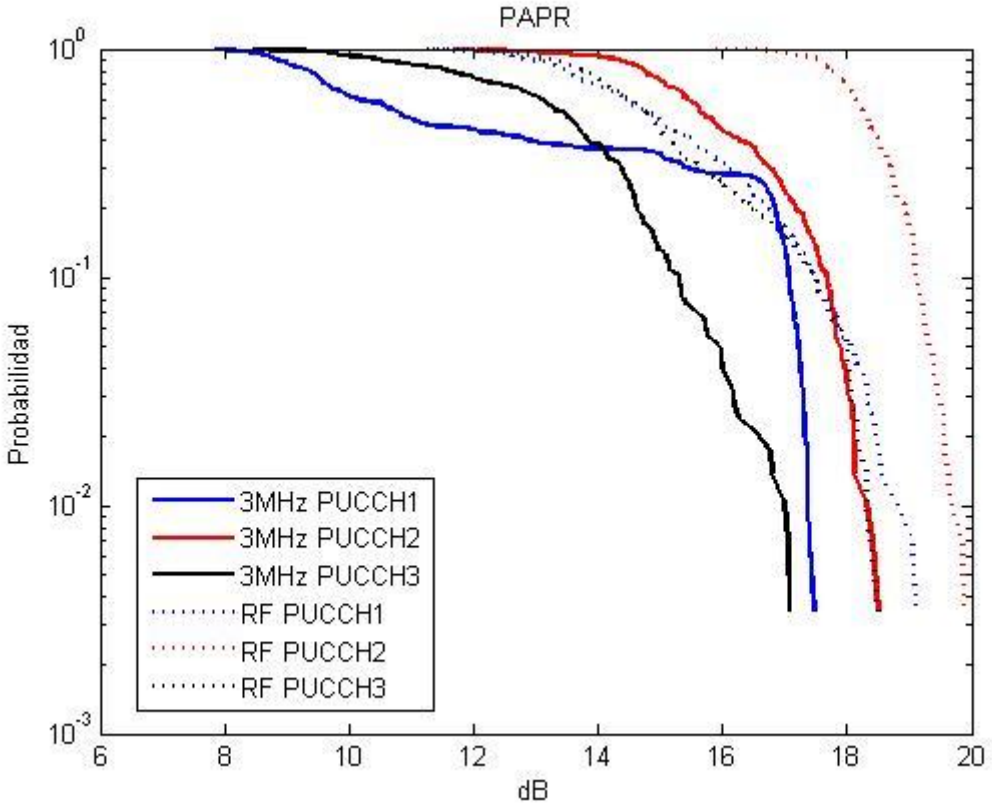


Ilustración 72: Comparativa diferentes PUCCH

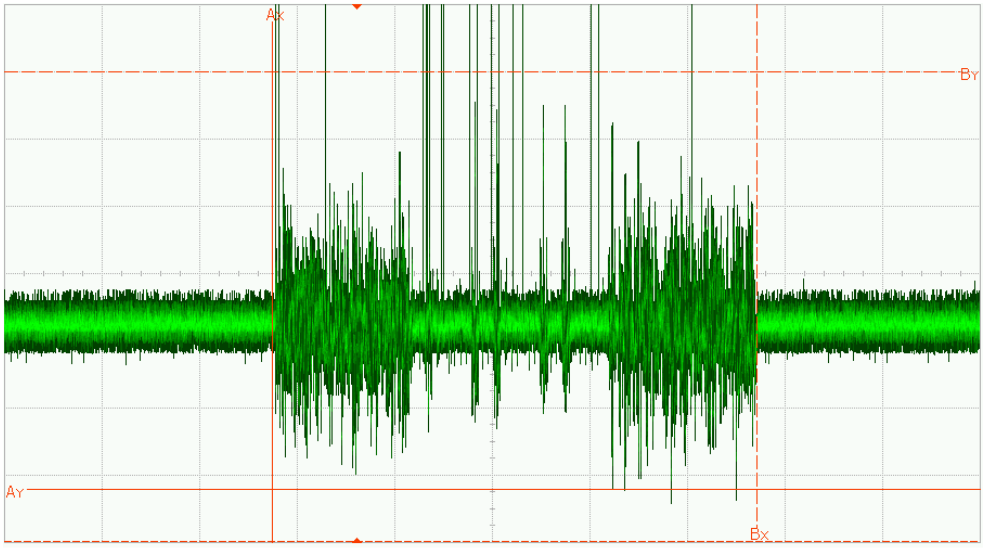
6. *Caso real 256 puntos con diferentes modulaciones.*

Por último, se va a implementar los 3 esquemas de modulación propuestos por el estándar [1]: QPSK, 16QAM y 64QAM.

En la ilustración 73 se muestra un slot para la modulación 16QAM y en la ilustración 74 se muestra una modulación 64QAM, podemos apreciar que en la ilustración 73 presenta menos picos o deltas que en la ilustración 74. Además en la ilustración 74 se presenta más niveles intermedios que la ilustración 73, muy acorde con los niveles intermedios de la modulación 64QAM.

Tras analizar los tres tipos de slots, correspondiente con cada formato de la señal de control PUCCH, podemos realizar una comparativa entre ellas. Como se aprecia en la ilustración 75, la PAPR de la modulación 16QAM y 64 QAM es casi la misma, sin embargo el problema reside en la modulación QPSK, ya que también posee los mismos niveles que las modulaciones de mayor complejidad, a diferencia de la simulación en Matlab, que posee menor PAPR. Esto es debido a la degradación del DAC, que ya se explicó anteriormente en el apartado 1 de éste capítulo, donde el DAC genera un número mayor de niveles de amplitud.

Y desde el punto de vista de la radio frecuencia, para una probabilidad de 10^{-2}, se eleva 2dB con respecto a la señal en banda base.



Acquisition: Sampling Mode real time, Normal
Memory Depth automatic 10.0000 Mpts
Sampling Rate automatic 10.0 GSa/s
Averaging off, Interpolation Auto

Channel 2: Scale 10.0 mV/, Offset 11.9 mV, Skew 0.0 s

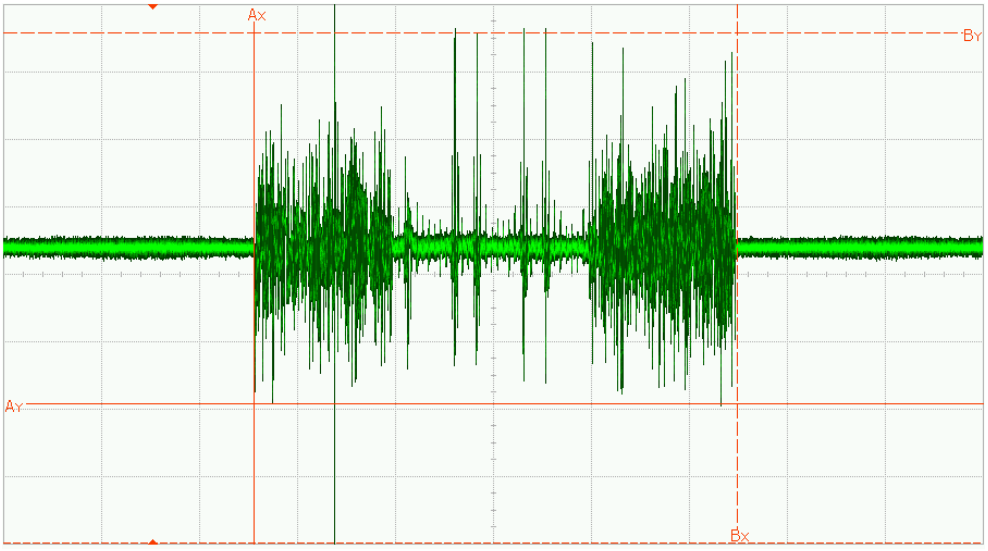
Horizontal: Scale 100 µs/, Position 138.3293333 µs, Reference center

Trigger: Mode edge, Source Channel 2, Level -28.0 mV, Slope rising
Sweep auto, Sensitivity normal, Holdoff Time 100 ns

Markers:

	X	Y
A——(2) =	-87.227222 µs	-20.160 mV
B---(2) =	409.439444 µs	41.940 mV
Δ =	496.666667 µs	62.100 mV
1/ΔX =	2.013422819 kHz	

Ilustración 73: Un slot 16QAM



Acquisition: Sampling Mode real time, Normal
Memory Depth automatic 10.0000 Mpts
Sampling Rate automatic 10.0 GSa/s
Averaging off, Interpolation Auto

Channel 2: Scale 10.0 mV/, Offset 0.0 V, Skew 0.0 s

Horizontal: Scale 100 µs/, Position 347.7801778 µs, Reference center

Trigger: Mode edge, Source Channel 2, Level -39.9 mV, Slope rising
Sweep auto, Sensitivity normal, Holdoff Time 100 ns

Markers:

	X	Y
A——(2) =	103.334733 µs	-19.190 mV
B---(2) =	596.668067 µs	35.810 mV
Δ =	493.333333 µs	55.000 mV
1/ΔX =	2.027027027 kHz	

Ilustración 74: Un slot 64QAM

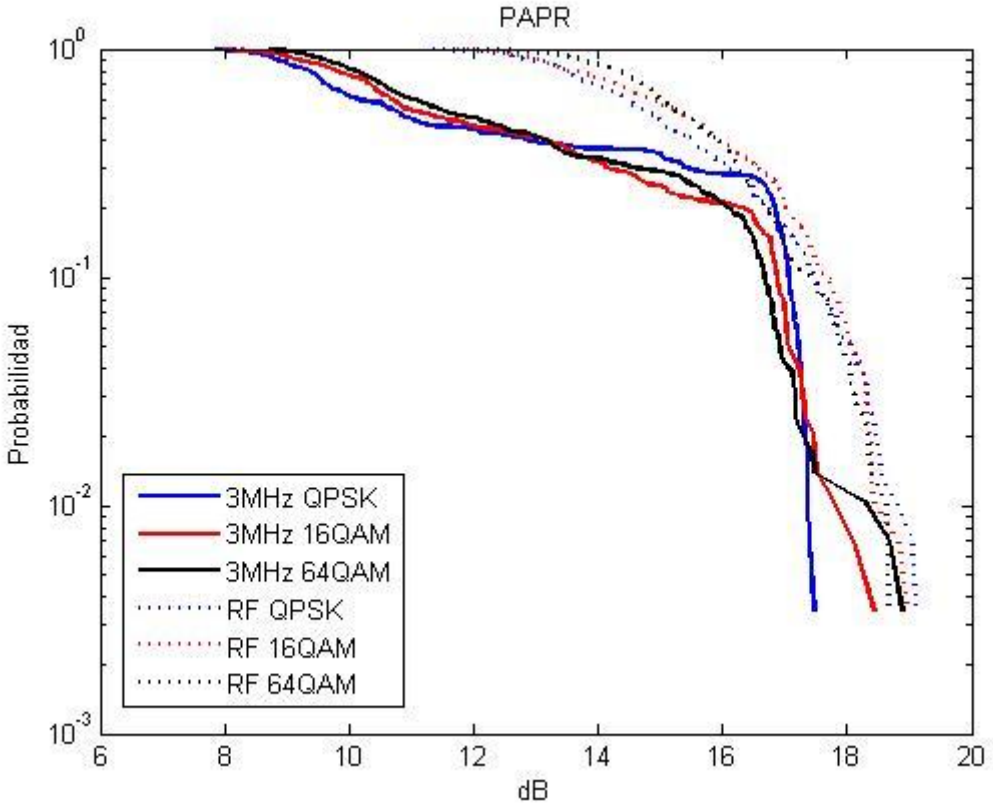


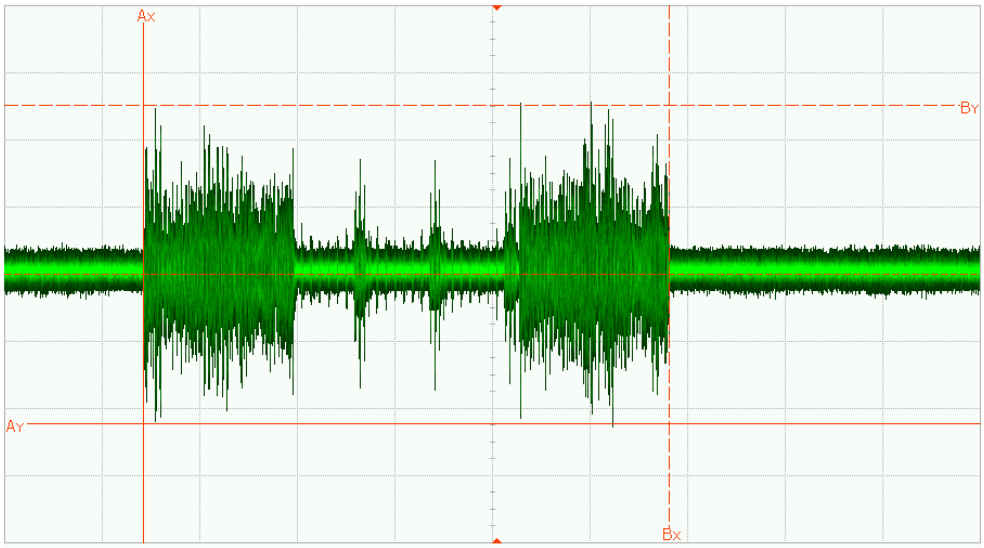
Ilustración 75: Comparativa modulaciones

7. *Caso real 2048 puntos.*

Una vez que se ha explotado todos los casos posibles, variando todos parámetros de configuración de la trama, se aumenta de ancho de banda. Se va a estudiar directamente el ancho de banda más elevado con un ancho de banda ideal de 20MHz, que corresponde con una IFFT de 2048 puntos.

En la ilustración 76, se genera una señal comprimido en los 0.5ms. Si comparamos este slot con el slot de 3MHz, dicho slot posee una mayor suavidad, se aprecia la imagen con mayor claridad, esto es debido al gran nivel de compresión de los símbolos, estamos introduciendo casi 1000 veces más símbolos que para el caso de los 3MHz.

En la ilustración 76, se aprecia varios slots. A diferencia de las ilustraciones cuyo ancho de banda de 3MHz, el espacio entre slots es mayor, ya que el tiempo necesario para generar toda esta información es mayor, y por tanto el tiempo de no transmisión es mucho mayor que un slot.



Acquisition: Sampling Mode real time, Normal
Memory Depth automatic 10.0000 Mpts
Sampling Rate automatic 10.0 GSa/s
Averaging off, Interpolation Auto

Channel 2: Scale 5.00 mV/, Offset 4.4 mV, Skew 0.0 s

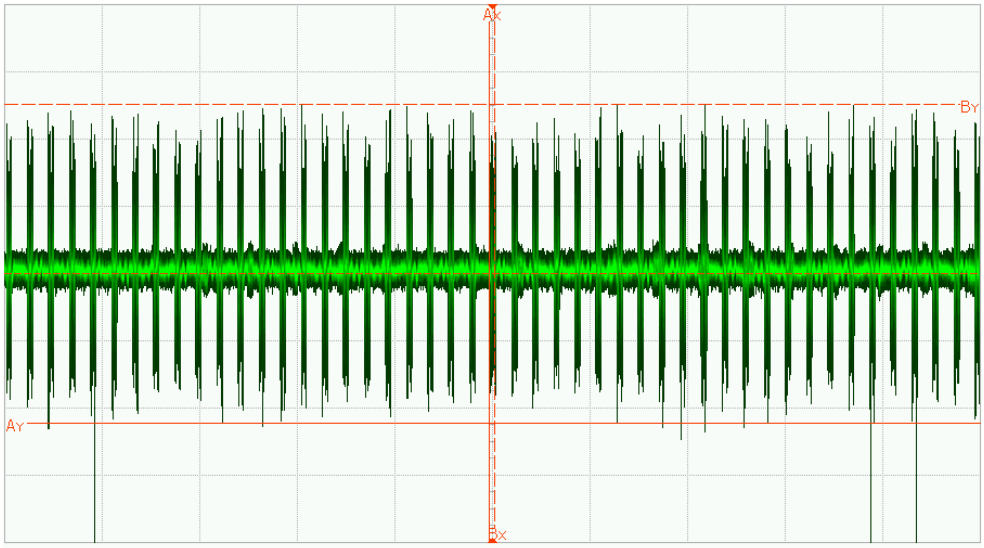
Horizontal: Scale 100 µs/, Position -4.4400000 µs, Reference center

Trigger: Mode edge, Source Channel 2, Level 4.4 mV, Slope rising
Sweep auto, Sensitivity normal, Holdoff Time 100 ns

Markers:

	X	Y
A——(2) =	-362.2187778 µs	-6.730 mV
B---(2) =	176.6701111 µs	16.980 mV
Δ =	538.8888889 µs	23.710 mV
1/ΔX =	1.8556701031 kHz	

Ilustración 76: Un slot en banda base



Acquisition: Sampling Mode real time, Normal
Memory Depth automatic 10.0000 Mpts
Sampling Rate automatic 100 MSa/s
Averaging off, Interpolation Auto

Channel 2: Scale 5.00 mV/, Offset 4.4 mV, Skew 0.0 s

Horizontal: Scale 10.0 ms/, Position -4.4400000 µs, Reference center

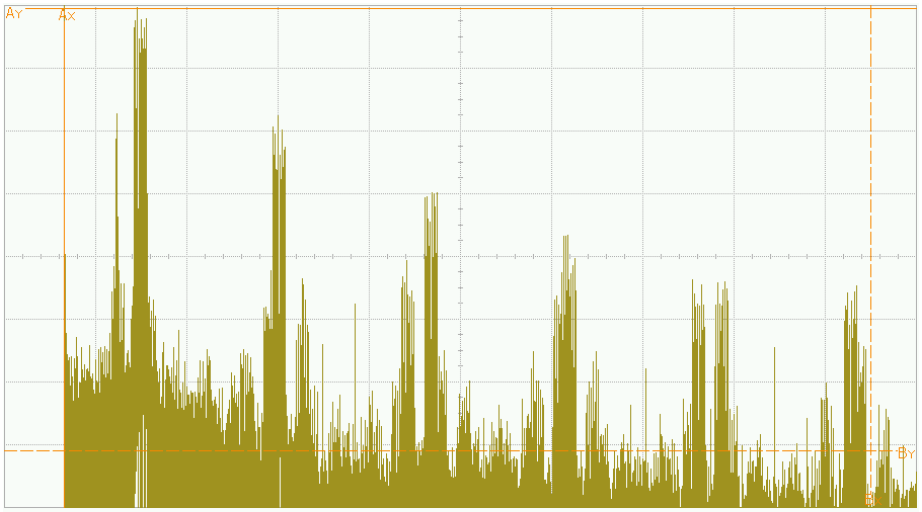
Trigger: Mode edge, Source Channel 2, Level 4.4 mV, Slope rising
Sweep auto, Sensitivity normal, Holdoff Time 100 ns

Markers:

	X	Y
A——(2) =	-362.21878 µs	-6.7300 mV
B---(2) =	176.67011 µs	16.9800 mV
Δ =	538.88889 µs	23.7100 mV
1/ΔX =	1.85567010 kHz	

Ilustración 77: Varios slots en banda base

Como se aprecia en la ilustración 78, el ancho de banda aumenta hasta los 78MHz, y en el diseño se ha introducido un reloj de 37.5MHz en el bloque de la IFFT. Este aumento del ancho de banda es debido a la inserción del prefijo cíclico y a la compresión del elevado número de símbolos en un tiempo de slot de 0.5ms.



Acquisition: Sampling Mode real time, Normal
Memory Depth automatic 883.752 kpts
Sampling Rate automatic 1.25 GSa/s
Averaging off, Interpolation Auto

Horizontal: Scale 70.7 µs/, Position -491.1131111 µs, Reference center

Trigger: Mode edge, Source Channel 2, Level 4.9 mV, Slope rising
Sweep auto, Sensitivity normal, Holdoff Time 100 ns

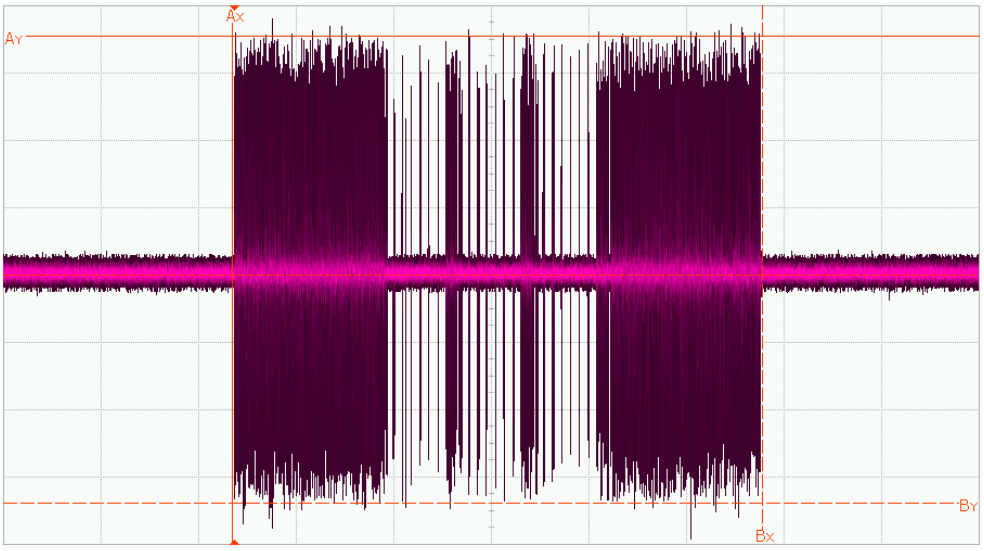
Function 1: FFT magnitude Channel 2
Vertical Scale 5.38306 dB/, Offset -82.4508 dBm
Horizontal Scale 8.83 MHz/, Center Frequency 38.333083 MHz
Window Hamming, Resolution 2.122 kHz

Markers:

	X	Y
A——(f1) =	0 Hz	-61.179 dBm
B---(f1) =	78.0680 MHz	-99.121 dBm
Δ =	78.0680 MHz	-37.942 dB
1/ΔX =	12.80935 ns	

Ilustración 78: Ancho de banda en banda base

Una vez analizado la señal en banda base, se va a presentar la señal en radio frecuencia, como se aprecia en la ilustración 79, se dispone de un slot en radio frecuencia. Podemos apreciar que se ha aumentado la amplitud de la señal con respecto a la banda base.



Acquisition: Sampling Mode real time, Normal
Memory Depth automatic 10.0000 Mpts
Sampling Rate automatic 10.0 GSa/s
Averaging off, Interpolation Auto

Channel 4: Scale 50.0 mV/, Offset 3.7 mV, Skew 0.0 s

Horizontal: Scale 100 µs/, Position 263.3330432 µs, Reference center

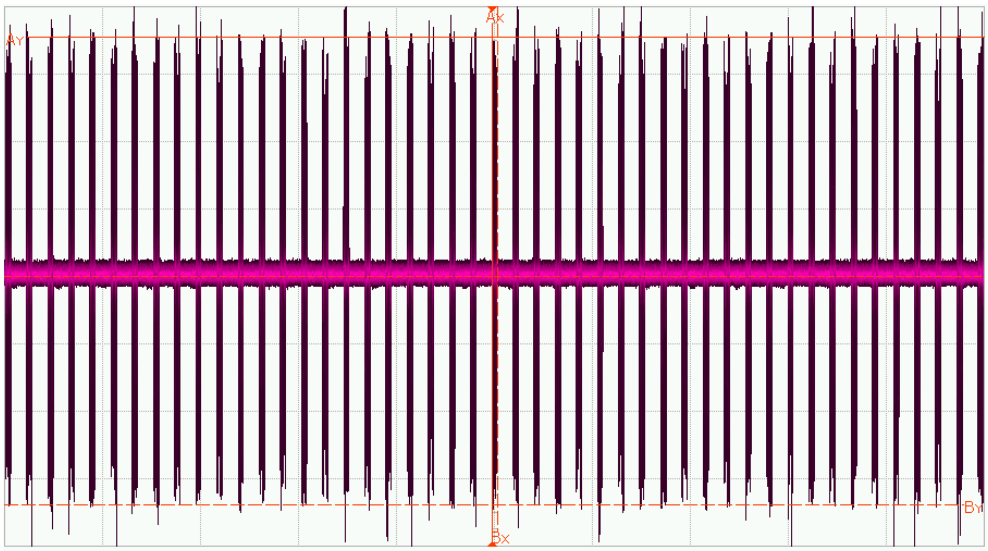
Trigger: Mode edge, Source Channel 4, Level 4 mV, Slope rising
Sweep auto, Sensitivity normal, Holdoff Time 100 ns

Markers:

	X	Y
A——(4) =	-2.224512 µs	181.12 mV
B---(4) =	541.109821 µs	-165.65 mV
Δ =	543.334333 µs	-346.77 mV
1/ΔX =	1.840487410 kHz	

Ilustración 79: Un slot en radiofrecuencia

En la ilustración 80 se aprecia varios slots, podemos apreciar que la amplitud de los slots ha aumentado, y que su variabilidad también ha aumentado con respecto al caso de banda base.



Acquisition: Sampling Mode real time, Normal
Memory Depth automatic 10.0000 Mpts
Sampling Rate automatic 100 MSa/s
Averaging off, Interpolation Auto

Channel 4: Scale 50.0 mV/, Offset 3.7 mV, Skew 0.0 s

Horizontal: Scale 10.0 ms/, Position 263.3330432 µs, Reference center

Trigger: Mode edge, Source Channel 4, Level 4 mV, Slope rising
Sweep auto, Sensitivity normal, Holdoff Time 100 ns

Markers:

	X	Y
A——(4) =	-2.22451 µs	181.120 mV
B---(4) =	541.10982 µs	-165.650 mV
Δ =	543.33433 µs	-346.770 mV
1/ΔX =	1.84048741 kHz	

Ilustración 80: Varios slots en radiofrecuencia

Una vez visto en el tiempo, se va a representar la señal en el dominio frecuencia, podemos apreciar que la señal está montada en la portadora 370MHz, y el ancho de banda es la misma que para el caso de banda base.

Acquisition: Sampling Mode real time, Normal
Memory Depth automatic 30.003 kpts
Sampling Rate automatic 20.0 GSa/s
Averaging off, Interpolation Auto

Horizontal: Scale 150 ns/, Position 240.2019861 µs, Reference center

Trigger: Mode edge, Source Channel 4, Level 4 mV, Slope rising
Sweep auto, Sensitivity normal, Holdoff Time 100 ns

Function 4: FFT magnitude Channel 4
Vertical Scale 7.31636 dB/, Offset -63.1790 dBm
Horizontal Scale 29.0 MHz/, Center Frequency 372.157467 MHz
Window Hamming, Resolution 1.000 MHz

Markers:

	X	Y
A——(f4) =	371.84 MHz	-73.446 dBm
B---(f4) =	450.78 MHz	-35.566 dBm
Δ =	78.94 MHz	37.880 dB
1/ΔX =	12.667 ns	

Ilustración 81: Ancho de banda en radiofrecuencia

Ahora podemos comparar la señal en función de los dos anchos de banda de la señal. Podemos apreciar en la ilustración 82 que la PAPR en banda base es casi la misma, esto es debido a que ambas señales se comprimen en un mismo tiempo de slot de 0.5ms, a diferencia de Matlab. Para el caso de 20MHz en Matlab, se tiene una frecuencia de muestreo mayor que para 3MHz, por tanto se dispone de más ceros en cada símbolo SC-FDMA que reducen la media y a su vez aumentan la PAPR. Lo único que se puede destacar de la PAPR en banda base, es que la señal de 20MHz dispone de picos mayores, donde la diferencia máxima entre los dos tipos de ancho de banda es de 1dB aproximadamente.

Y en el dominio de la radio frecuencia, la PAPR intermedia de los 3MHz es inferior a los 20MHz, para una probabilidad de 10^{-1} la diferencia es de casi 1dB. Sin embargo ambas curvas convergen para una PAPR máxima de 19dB aproximadamente.

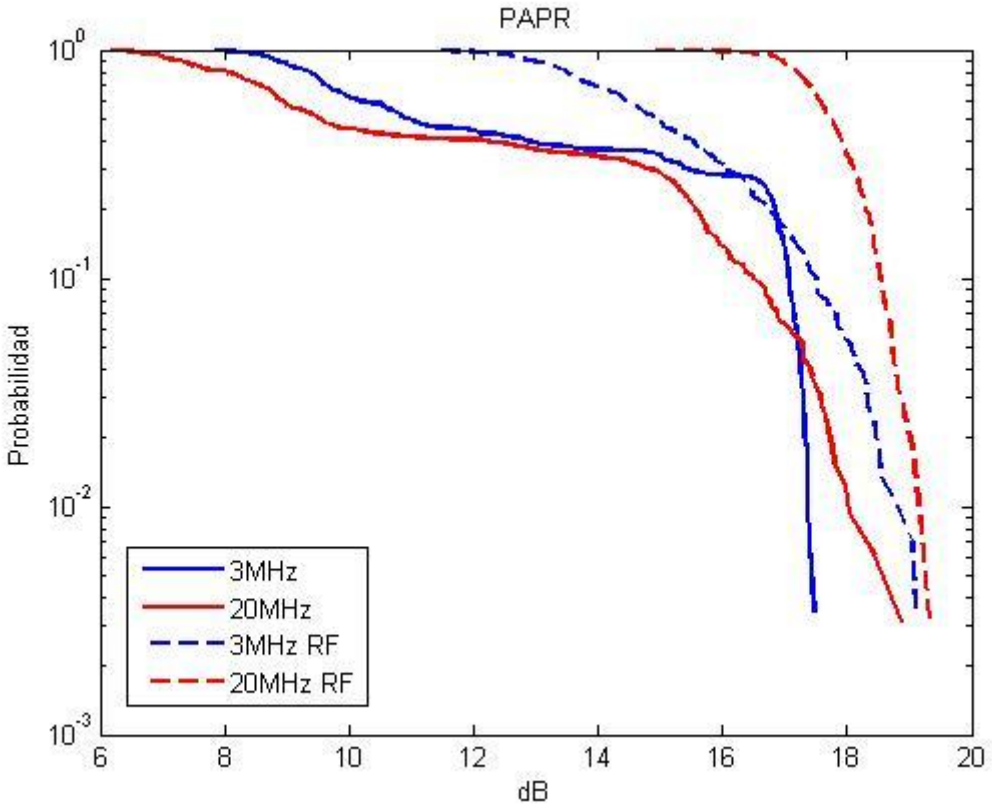


Ilustración 82: Comparativa anchos de banda

6. Principales dificultades.

La primera dificultad ha sido entender y comprender el estándar de 3GGP [1], ya que emplean un lenguaje muy ambiguo y muy confuso. La notación matemática de la misma es muy compleja, y para entender una simple multiplicación de matrices requiere una concentración mayor de lo necesario. Además de la notación, existe otra dificultad debida a los numerosos parámetros sin definir por el estándar, ya que LTE no es un estándar tan sólido como UMTS, y quizás dicha versión no sea la definitiva, ya que se deja muchos cabos sin atar. Por tanto existe una gran cantidad de parámetros del sistema que uno debe de inventar con cierto criterio.

Al realizar una implementación en hardware, sin duda la gran complicación es la configuración y manipulación del dispositivo Lyrtech SFF SDR. Esto es debido a que la placa es muy compleja, disponiéndose de muchos bloques funcionales que han de ser configurados correctamente. Para atajar en parte esta dificultad se ha empleado archivos heredados de experiencias previas. Cabe destacar que hay que tener mucho cuidado con cada línea de código dedicado a la configuración del dispositivo, ya que si se omite sin querer cualquier línea, todo el diseño dejará de funcionar correctamente.

Y la última gran dificultad fue comprender las IP Cores de Xilinx, que en nuestro caso particular fue complicado trabajar con la FFT de Xilinx [15]. En primer lugar, el manual de usuario de Xilinx para cada componente es poco claro, y no especifica bien cómo se debe manipular el componente. Por tanto la mejor manera es crearse un código auxiliar que itere con el bloque y simularlo con ModelSim. En segundo lugar, cada componente de Xilinx sufre grandes cambios con respecto la versión de la misma. Al principio se empleó la FFT del laboratorio cuya versión es 4.3 y el problema de usar esta versión en el caso ideal de 255 puntos, es que al poner en cascada una FFT y una IFFT, el resultado a la salida del IFFT no son los símbolos QPSK transmitidos. Posteriormente se empleó la versión 7.1 y al ponerlos en cascada, el resultado si era correcto. Esto es debido a que el resultado de la FFT de la versión 7.1 te lo proporciona en el orden inverso, primero te saca el coeficiente 255 y por último te saca el coeficiente 0; a diferencia de la versión 4.3 que te lo proporcional en orden natural, que habría que invertirlo manualmente, para que nos garantice un resultado correcto.

Para el lector, quizás parezca pequeños e insignificantes detalles que hay que solventar. Sin embargo, desde el punto de vista del desarrollo, todos estos detalles son muy difíciles de descubrir, ya que hay tantos parámetros que hay que tener en cuenta, que uno no puede abarcar a todo.

7. Conclusión y líneas de trabajo futuras.

En el presente proyecto fin de carrera, se han realizado dos tipos de simulaciones y la implementación del transmisor en un dispositivo SFF SDR. En la primera hemos simulado la PAPR producida en el transmisor, estudiando un poco cómo se comporta esta nueva modulación definida por el 3GPP. En una segunda simulación, se ha estudiado la probabilidad de error del sistema en función de varios factores o parámetros del sistema. Y por último se ha implementado en un dispositivo para estudiar su PAPR en un caso real.

En la primera simulación, para el caso ideal (ilustración 18) con una probabilidad de 10^{-4} se consigue una amplitud cuyo PAPR superan los 8dB. Por otro lado, como se aprecia en la ilustración 83 (referencia [8]), para una modulación OFDM y empleando en cada subportadora una modulación QPSK, para la misma probabilidad se consigue una PAPR de 11.5dB aproximadamente.

Ahora bien, si comparamos con un caso real en la simulación, como se aprecia en la ilustración 21, para una probabilidad de 10^{-1}, en la modulación QPSK con 10MHz se consigue una PAPR de 7.8dB aproximadamente. Si vamos a la ilustración 83 con la misma probabilidad, se consigue una PAPR de 8,7 dB aproximadamente.

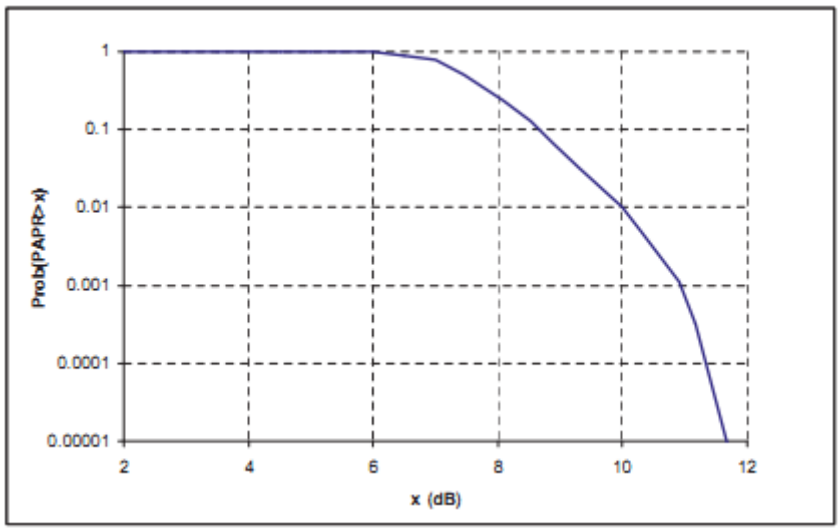


Ilustración 83: PAPR de OFDM [8], N=256 K=64portadoras

Por tanto, desde un punto de vista ideal de la simulación de Matlab, empleando una DFT con mayor longitud y empleando un mayor número de sub portadoras, somos capaces de reducir la PAPR hasta 4dB aproximadamente. Si observamos la ilustración 84 que pertenece a la referencia [8], conseguimos los mismos resultados.

Ahora bien, si comparamos un caso real de la simulación de Matlab, en la que sólo empleamos parte del ancho de banda, dejando el resto a ceros, aun así seguimos obteniendo una mejora de casi 1dB comparado con la referencia [8].

Tras estas dos comparaciones, podemos concluir que la modulación SC-FDMA mejora significativamente, a pesar de estar en peores condiciones se consigue un valor de PAPR inferior comparado con la modulación OFDM.

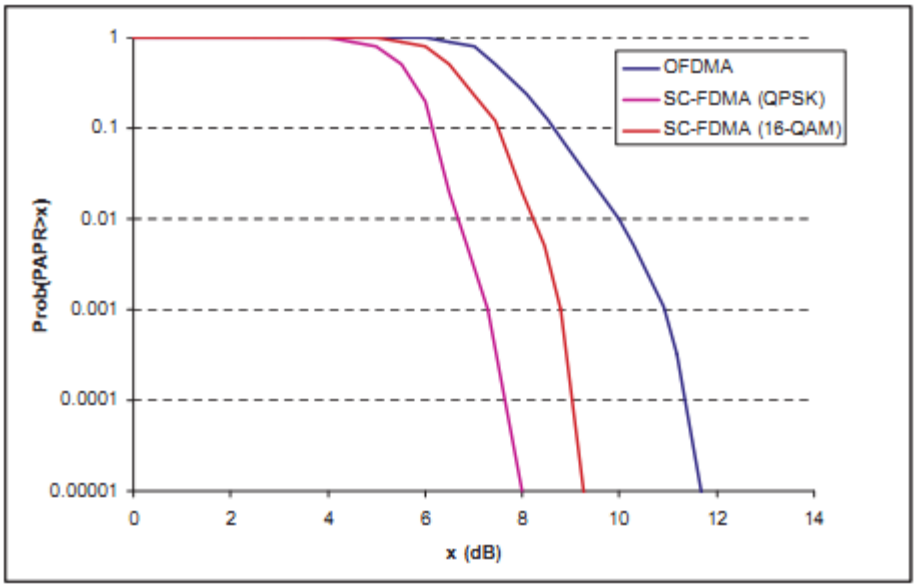


Ilustración 84: Comparativa OFDM vs SC-FDMA [8]

En una segunda simulación, hemos estudiado la probabilidad de error del sistema. Desde un punto cualitativo, los resultados de todas las simulaciones eran predecibles antes de la simulación, validando así todos nuestros conocimientos asociados a esta materia.

Sin embargo nos hemos llevado una gran sorpresa, debido a los malos resultados numéricos obtenido tras las simulaciones. Como ya se ha dicho, desde un punto de vista cualitativo es correcto, pero no desde un punto de vista cuantitativo. El causante de este fenómeno, es por culpa de los bloques FFT/IFFT que se realiza en el transmisor y receptor, ya que de alguna manera desconocida estropea el resultado, haciendo así que el igualador ZF no funcione con las prestaciones que se desea, como en OFDM.

Es comprensible que no todo salga tan bien como uno se espera, cuando se ha diseñado el sistema SC-FDMA se ha realizado partiendo de los conocimientos de OFDM, y claro, uno piensa que debería funcionar como OFDM, y eso no tiene porqué. Por tanto, para mejorar el rendimiento o la probabilidad de error del sistema, se debe hacer algo más sofisticado que ZF, indagando un poco en este mundillo, parece ser que ya se está investigando sobre igualadores basados en MMSE-DFE, turbo igualadores, o variantes de igualadores clásicos, etc. Incluso hay investigaciones para reducir este error desde otro punto de vista, olvidándonos de todos los conocimientos de OFDM, dando un giro más radical.

Como el diseño de los igualadores no es cosa fácil, y encima está muy poco maduro, en el presente proyecto no se ha realizado otros igualadores, y se deja esta parte como una mejora que habrá que realizar en el futuro.

Y por último para acercarnos mejor a la realidad, se realiza la implementación del transmisor de SC-FDMA en un dispositivo SFF SDR. Podemos apreciar en las ilustraciones que en el caso ideal, en banda base la PAPR empeora unos 1.5dB; y en radiofrecuencia la PAPR está acotada a 20dB. Estos resultados son bastantes comprensibles, ya que como se ha dicho, en la realidad siempre los resultados deben ser peores que en la simulación.

Probabilidad	Ideal Matlab	Ideal FPGA	Diferencia
10^{-2}	7dB	8.5dB	1.5dB

Tabla 8: Comparativa del caso ideal

Desde el punto de vista del canal de control PUCCH, se mantiene que PUCCH2 es el que posee mayor PAPR, sin embargo PUCCH3 posee mucho mejor PAPR en la implementación que en la simulación, siendo ésta inferior al PUCCH1.

Ahora si comparamos el caso correspondiente para un ancho de banda de 3MHz para una probabilidad de 10^{-2}, para un caso real:

- Existe una diferencia máxima de 10dB entre la señal generada con Matlab y la señal generada con la FPGA. Podemos concluir, que existe mucho más picos en el caso práctico que en el caso simulado, como cabría de esperar.
- Entre la señal con SRS y sin SRS, existe una diferencia de 0.2dB en el caso de Matlab, y 1.5dB en el caso de la FPGA. Podemos apreciar en el caso práctico, la diferencia es mucho mayor que en la simulación, activando la SRS nos aumenta mucho más la PAPR.

SRS	Matlab	FPGA	Diferencia
NO	8.8dB	17.5dB	8.7dB
SI	9dB	19dB	10dB
Diferencia	0.2dB	1.5dB	

Tabla 9: Comparativa del caso real

Para el resto de parámetros de configuración del sistema, los resultados de Matlab concuerdan con los de la implementación en la FPGA, sólo desde un punto relativo. Por ejemplo, al cambiar de modulación o ancho de banda, la PAPR varía muy poco. Por tanto, de forma relativa, los resultados concuerdan exceptuando los 10dB de diferencia entre el caso simulado y el caso de la implementación. Que en parte puede ser debido a las imperfecciones del hardware, destacando el DAC que nos introduce picos, valores intermedios indeseados y que los pulsos no son planos.

Adicionalmente se puede concluir que la PAPR también depende fuertemente de los módulos de conversión de datos y de radiofrecuencia. Para poder mejorar la PAPR, en un futuro se podrían insertar otros módulos para mejorar la señal producida. Se podría programar la misma señal en la misma Virtex4, pero se debería probar con otros modelos de hardware:

- Emplear un módulo de conversión de datos que posea mejores prestaciones, cuya respuesta en la frecuencia disponga de un mejor comportamiento.
- Emplear un módulo de radiofrecuencia que disponga de mayor número de filtros, y de un ancho de banda más reducido que el actual, de este modo podremos filtrar todo el ruido de las bandas ajenas a la nuestra.

A simple vista, estas mejoras pueden causar en un aumento del coste del dispositivo hardware en comparación con lo que se tiene actualmente. Sin embargo esta apreciación es totalmente errónea, ya que hoy en día, existen módulos de radiofrecuencia y conversión de datos que poseen mayores prestaciones y siendo más económicas que los actuales.

Como sólo se ha realizado el transmisor SC-FDMA, por tanto en un futuro no próximo se debería implementar también el receptor SC-FDMA, teniendo en cuenta la dificultad extra del igualador de canal.

Para acabar, podemos concluir definitivamente que la nueva modulación SC-FDMA reduce la PAPR con respecto a la OFDM, pero hay que tener en cuenta que estamos pagando un precio, la complejidad de los equipos. Como ya se ha visto, estamos introduciendo muchas más operaciones, como la FFT/IFFT en nuestro terminal. Hoy en día ya existen procesadores de cuatro núcleos, por tanto desde un punto de vista de cálculo no sería un problema, sin embargo esto tendría un efecto muy negativo en la autonomía. Las baterías son de las pocas cosas del mundo que no cumplen con la ley de Moore, por tanto su capacidad no aumentan como el resto de componentes electrónicos. Por tanto si aumentamos el consumo con operaciones más complejas, estaríamos consumiendo más energía que antes, y por tanto la batería de nuestro móvil se agotaría inmediatamente. En la actualidad los móviles que se conectan a 3G o UMTS poseen un consumo excesivo de batería, haciendo necesario una carga por día, y eso que no realizan ninguna FFT/IFFT. Por tanto para poder llevar a cabo la implementación de esta nueva modulación, habría que duplicar al menos la capacidad de las baterías.

Apéndice A: Algoritmo de la mediana.

En comunicaciones digitales, para luchar contra el ruido y los efectos del canal, se implementan diferentes mecanismos para construir estimadores y decisores de gran prestación para realizar la demodulación de los símbolos en el receptor.

Un caso muy común en comunicaciones, véase en la ilustración 84, tenemos una modulación BPSK y por culpa del ruido impulsivo tenemos alguna muestra que está muy lejos de lo normal, dichas muestras se denominan "outliers" que son muestras fuera de rango y son atípicas. Ahora bien, en nuestro decisor debemos en primer lugar determinar cuál va a ser ese umbral de decisión para luego tomar la decisión en función de dicho umbral. Como sabemos que nuestro espacio muestral está contaminado con "outliers", no podemos calcular la media de todos los datos, ya que la media no se situaría en el centro de -1 y +1, la posición ideal de nuestro umbral de decisión, causando así una probabilidad de error cerca al 0.5. Por tanto debemos usar la mediana, que no tiene en cuenta la amplitud de los datos, sino la cantidad de datos, dejando a una mitad el 50% de los datos a un lado, y a otro lado los otros 50%. Aunque el umbral de decisión construido por la mediana tampoco se sitúa en el lugar ideal, pero sí está más cerca de dicho umbral, causando una probabilidad de error cercana a 0. Por tanto podemos concluir que la mediana es una operación matemática muy apta para eliminar los efectos de los "outliers".

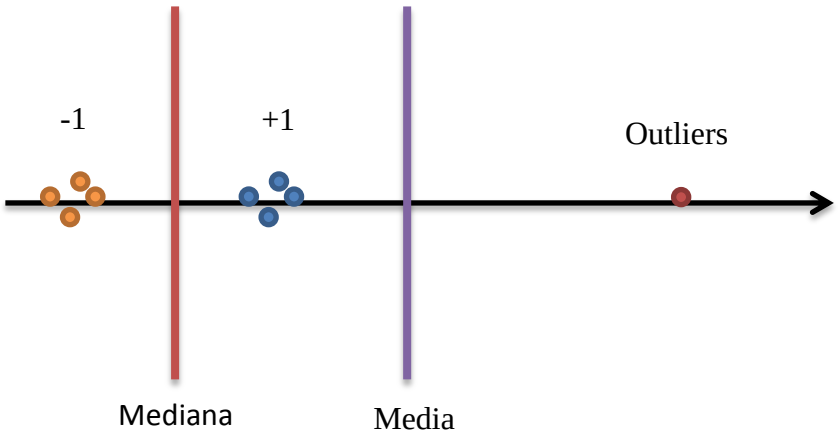


Ilustración 85: BPSK: comparación media y mediana

En nuestro proyecto, el DAC en vez de generarnos pulsos cuadrados, nos genera picos de tensión muy elevados, véase en la ilustración 85. Además de dichos picos, el pulso no es cuadrado, la parte superior no es plana, sino se nos genera una curva, cuya pendiente no es cero. Es evidente que estos dos efectos van a generar un error extra en el cálculo de la PAPR. Para eliminar el pico elevado podemos aplicar el algoritmo de la media, que se detalla a continuación. Sin embargo, no podemos hacer nada con la zona curva del pulso, ya que es algo intrínseco al convertidor.

Desde un punto vista práctico, de diseño global, estos se podría filtrar mediante un dispositivo RF, ya que dicha capa presenta filtros paso banda, cuyo el ancho de banda debería ser lo más estrecho posible. Como en nuestro dispositivo no disponemos de un filtro tan selectivo en frecuencia, no podemos filtrar nada.

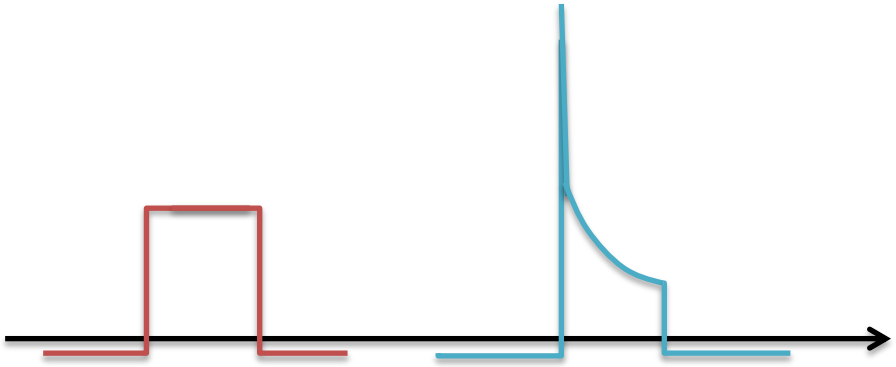


Ilustración 86: Pulso normal y pulso DAC

Los pasos del algoritmo de la mediana son los siguientes:

- Dado un vector en Matlab, ordenamos de menor a mayor el vector. De este modo, todos los máximos se acumulan al final del vector.
- De dicho vector, extraemos N máximos para calcular la mediana de todas ellas.
- Cogemos el máximo global, la última muestra del vector, y preguntamos si dicho máximo esta lo bastante cerca de la mediana, o si está muy lejos.
- Si la muestra no está cerca de la mediana, descartamos el máximo y cogemos el siguiente máximo absoluto, la penúltima muestra del vector.
- Se realiza tantas iteraciones necesarias, hasta encontrar un máximo que esté lo bastante cerca de la mediana, y se procede a calcular la PAPR.

Los umbrales se seleccionan en función de cómo es la trama, como se tiene la trama ordenada, es posible ver a simple vista, cuantas muestras son picos elevados. Y en función de dicha cantidad se elige unos valores determinados.

Apéndice B: Resultados de la síntesis.

Se presentan los recursos utilizados y el tiempo máximo de procesamiento de la FPGA para el caso real de 256 puntos con un PUCCH3, ya que es el que presenta mayor número de bloques empleados.

Logic Utilization	Used	Available	Utilization
Number of Slice Flip Flops	6,896	30,720	22%
Number of 4 input LUTs	8,056	30,720	26%
Number of occupied Slices	5,840	15,360	38%
Number of Slices containing only related logic	5,840	5,840	100%
Number of Slices containing unrelated logic	0	5,840	0%
Total Number of 4 input LUTs	8,630	30,720	28%
Number used as logic	7,187		
Number used as a route-thru	574		
Number used as 16x1 RAMs	16		
Number used as Shift registers	853		
Number of bonded IOBs	211	448	47%
IOB Dual-Data Rate Flops	4		
IOB Master Pads	12		
IOB Slave Pads	12		
Number of BUFG/BUFGCTRLs	8	32	25%
Number used as BUFGs	7		
Number used as BUFGCTRLs	1		
Number of FIFO16/RAMB16s	29	192	15%
Number used as RAMB16s	29		
Number of DSP48s	15	192	7%
Number of DCM_ADVs	2	8	25%
Number of RPM macros	3		
Average Fanout of Non-Clock Nets	2.92		

Tabla 10: Resultado síntesis 256 puntos + PUCCH3

$$T_{min} = 1.972ns \rightarrow f_{max} = 507.1MHz$$

Como se aprecia en la tabla 8, el uso de los recursos de la FPGA es relativamente bajo, el consumo medio es del 25% aproximadamente. Por tanto la FPGA no está sobrecargada, y se podría implementar más bloques en su interior.

Por otro lado, apreciamos que la frecuencia máxima hasta unos 507MHz, y actualmente estamos usando 3.7MHz. Vemos que trabajamos dos órdenes de magnitud por debajo del máximo, se podría aumentar la velocidad en el caso de que tengamos mayor cantidad de datos que procesar.

Y por último se muestra el resultado para el caso de 2048 puntos, ya que corresponde con el ancho de banda más alto propuesto por el estándar.

Logic Utilization	Used	Available	Utilization
Number of Slice Flip Flops	6,191	30,720	20%
Number of 4 input LUTs	7,451	30,720	24%
Number of occupied Slices	5,337	15,360	34%
Number of Slices containing only related logic	5,337	5,337	100%
Number of Slices containing unrelated logic	0	5,337	0%
Total Number of 4 input LUTs	7,910	30,720	25%
Number used as logic	6,776		
Number used as a route-thru	459		
Number used as 16x1 RAMs	16		
Number used as Shift registers	659		
Number of bonded IOBs	211	448	47%
IOB Dual-Data Rate Flops	4		
IOB Master Pads	12		
IOB Slave Pads	12		
Number of BUFG/BUFGCTRLs	9	32	28%
Number used as BUFGs	8		
Number used as BUFGCTRLs	1		
Number of FIFO16/RAMB16s	40	192	20%
Number used as RAMB16s	40		
Number of DSP48s	12	192	6%
Number of DCM_ADVs	3	8	37%
Number of RPM macros	3		
Average Fanout of Non-Clock Nets	3.02		

Tabla 11: Resultado síntesis 2048 puntos

$$T_{min} = 7.426ns \rightarrow f_{min} = 134.66MHz$$

Como podemos apreciar, a groso modo los recursos consumidos por la FPGA son casi los mismos que el caso anterior. Esto puede ser debido a que la gran diferencia entre el caso de los 256 puntos con 2048 puntos es el aumento de la memoria RAM y el aumento de los puntos de la FFT. Si nos damos cuenta, ambos bloques son de la IP Core de Xilinx, no está implementado por nosotros. Podemos concluir que es altamente recomendable utilizar las IP Cores de Xilinx para operaciones muy complejas.

Sin embargo, podemos notar que la frecuencia máxima ha caído 4 veces con respecto al caso anterior, esto es debido a la gran cantidad de datos que hay que procesar.

Apéndice C: Presupuesto

A continuación se va a detallar el presupuesto del proyecto. En la tabla 12 se especifica los costes de personal, asumiendo que el coste por hora es de 15€. En la tabla 13 se detalla el precio de todo los materiales y equipos empleados en el presente proyecto.

Concepto	Horas	Precio
Estudio y documentación	240	3.600€
Implementación en Matlab	120	1.800€
Implementación en VHDL	480	7.200€
Documentación	50	750€
	Subtotal:	13.350€

Tabla 12: Costes de personal

Concepto	Precio
Ordenador Core i3 RAM:4GB	600€
Dispositivo SFF SDR y licencias	8.400€
Osciloscopio Infinium DSO90604A	42.000€
Local 5 meses	600€
Otros gastos	1.000€
Subtotal:	52.600 €

Tabla 13: Costes de material

En la última tabla se obtiene el coste total del proyecto, añadiéndole el 20% de costes indirectos para poder mantener un margen de operación.

Concepto	Precio
Gastos de personal	13.350€
Gastos de material	52.600 €
Costes indirectos 20%	13.190 €
Base Imponible:	79.140 €
IVA 18%:	14.245 €
TOTAL:	93.385 €

Tabla 14: Costes totales

El coste total del presente proyecto asciende a la cantidad de **NOVENTA Y TRES MIL TRES CIENTOS OCHENTA Y CINCO EUROS (93.385 €).**

Acrónimos.

3GGP:	3rd Generation Partnership Project
ADC:	Analog to Digital Converter
BPSK:	Binary Phase Shift Keying
BW:	Band Width
CCS:	Code Composer Studio
CP:	Cyclic Prefix
DAC:	Digital to Analog Converter
DFE:	Decision Feedback Equalizer
DSP:	Digital signal processing
DFT:	Discrete Fourier Transform
DRS:	Demodulation Reference Signal
EPA:	Extended Pedestrian A model
ETU:	Extended Typical Urban model
EVA:	Extended Vehicular A model
FDD:	Frequency Division Duplexing
FFT:	Fast Fourier Transform
FPGA:	Field Programmable Gate Array
LTE:	Long Term Evolution
MIMO:	Multiple Input Multiple Output
MMSE:	Minimum Mean Square Error
OFMA:	Orthogonal Frequency Division Multiple Access
PAPR:	Peak to Average Power Ratio
PUCCH:	Physical uplink Control channel
PUSCH:	Physical Uplink Shared Channe
QAM:	Quadrature Amplitude Modulation
QPSK:	Quadrature Phase Shift Keying
RB:	Resource Block
RF:	Radio Frecuency
SFF SDR:	Small Form Factor Software Defined Radio
SC-FDMA:	Single Carrier Frequency Division Multiple Access
SRS:	Sounding Reference Signals
TDD:	Time Division Duplexing
UMTS:	Universal Mobile Telecommunications System
VHDL:	VHSIC Hardware Description Language
VHSIC:	Very High Speed Integrated Circuit
ZF:	Zero Forcing

Referencias.

1. 3GPP TS 36.211 V10.3.0 (2011-09): Physical Channels and Modulation (Release 10)

2. 3GPP TS 36.212 V10.3.0 (2011-09): Multiplexing and channel coding (Release 10)

3. 3GPP TS 36.101 V10.4.0 (2011-09): User Equipment (UE) radio transmission and reception (Release 10)

4. http://www.steepestascent.com/content/default.asp?page=s2_10_1

5. 3GPP LTE: Introducing Single-Carrier FDMA.
   Moray Rumney BSc, C. Eng, MIET. Lead Technologist, Agilent Technologies. 2008

6. LTE for UMTS – OFDMA and SC-FDMA Based Radio Access.
   Harri Holma and Antti Toskala both of Nokia Siemens Networks, Finland.
   2009 John Wiley & Sons Ltd.

7. LTE – The UMTS Long Term Evolution.
   Stefania Sesia. ST-NXP Wireless /ETSI, France.
   2009 John Wiley & Sons Ltd.

8. LTE: NUEVAS TENDENCIAS EN COMUNICACIONES MÓVILES.
   Ramón Agusti.
   2010 Fundación Vodafone España.

9. SINGLE CARRIER FDMA A NEW AIR INTERFACE FOR LONG TERM EVOLUTION.
   Hyung G. Myung. Qualcomm /Flarion Technologies, USA.
   2008. John Wiley & Sons, Ltd

10. PFC: IMPLEMENTACIÓN DE UN SISTEMA DE COMUNICACIÓN EN UN DISPOSTIVO RADIO
    Autor: Rony Fonseca Arboleda, Tutora: Ana García Armada.
    Universidad Carlos III Madrid, 2012.

11. Using Xilinx IP Cores. Amontec, 2001.

12. Lyrtech – ADACMaster III Users Guide, 2008.

13. Lyrtech – SFF SDR User's Guide, 2008.

14. Lyrtech – SFF SDR API Guide, 2007.

15. Xilinx – FFT core 7.1, 2012.

16. Xilinx – RAM core 4.3, 2010.

17. Xilinx – Virtex4 Configuration User's Guide, 2009.