



ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL
Facultad de Ingeniería en Electricidad y Computación

**“Diseño e implementación en FPGA’s de un sistema
escalable Modulador/Demodulador OFDM, como herramienta
académica e investigativa”**

TESIS DE GRADO

Previa la obtención del Título de:

INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

Presentado por:

Fauzi Antonio Nemer Villarroel

Director de Tesis:

Ing. Boris Ramos

Guayaquil – Ecuador

2007

AGRADECIMIENTO

A mis compañeros del subcomponente Telecomunicaciones del Proyecto VLIR ESPOL por la ayuda logística y la disponibilidad de los equipos necesarios para la implementación de este Proyecto de Tesis.

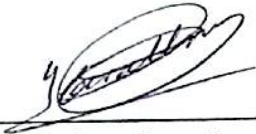
DEDICATORIA

A Dios, porque de el
proceden todas las cosas.

A mi esposa y amiga,
Maritza, por su constante
apoyo y confianza.

A mis padres, porque
siempre han querido lo
mejor para mí.

TRIBUNAL DE GRADUACION



Ing. Holger Cevallos
SU-DECANO DE LA FIEC



Ing. Boris Ramos
DIRECTOR DE TESIS



Ing. Rebeca Estrada
MIEMBRO PRINCIPAL

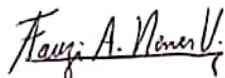


Ing. Cesar Martín
MIEMBRO PRINCIPAL

DECLARACION EXPRESA

“La responsabilidad del contenido de esta Tesis de Grado me corresponde exclusivamente; y el patrimonio intelectual de la misma a la Escuela Superior Politécnica del Litoral”

Art. 12 del Reglamento de Graduación de la ESPOL



Fauzi A. Nemer Villarroel

RESUMEN

Debido a la diversidad de frecuencia y a los efectos del prefijo cíclico sobre el ISI, OFDM (Orthogonal Frequency Division Multiplexing) es ampliamente estudiado como base de soluciones banda ancha en ambientes selectivos. Por esta razón, este proyecto pretende iniciar un estudio profundo en esta área, gracias a la implementación de un sistema escalable modulador/demodulador OFDM mediante el uso de FPGA's (Field Programmable Gate Array). Para el diseño de dicho sistema se ha tomado como referencia las especificaciones WirelessMan OFDM del estándar IEEE 802.16, ya que estas se apegan a los objetivos planteados en el proyecto.

El sistema implementado está constituido por 4 bloques principales: Transmisor (TX), Receptor (RX), Clocking y Settings. El bloque de settings se encarga de definir y proporcionar a los demás bloques las especificaciones básicas. El bloque de clocking por su parte se encarga de la sincronía del sistema mediante la distribución de los diferentes relojes al TX y al RX. El TX se encarga de formar y emitir los bloques OFDM en el dominio del tiempo. Finalmente el RX se encarga de la recuperación de los datos transmitidos en los símbolos de datos del transmisor.

Los denominados bloques OFDM están constituidos por dos tipos de símbolos OFDM: símbolos de Training y símbolos de Datos. Los símbolos de Training se constituyen en la cabecera del bloque y son formados por una cadena conocida de NFFT subportadoras C_k . Los símbolos de datos contienen la información de los datos codificados y modulados en OFDM. Estos símbolos están conformados por subportadoras C_k de Guarda, Pilotos y datos, formando en total NFFT componentes.

Para la evaluación del sistema se hizo uso de las herramientas del CSP. Es posible dividir las pruebas del sistema en tres grupos: Frecuencia de Operación del Sistema implementado, generación de los símbolos y bloques OFDM y evaluación de los datos recuperados. La primera parte se encarga de evaluar el rango de la frecuencia de operación función de los diferentes parámetros del sistema. La segunda parte, explora el proceso de generación de los símbolos y el bloque OFDM. Para esto se utilizan pruebas modulares que evalúan el proceso en cada etapa. Finalmente, en la tercera parte, se revisarán los datos recuperados en el receptor. Es decir, se evaluará la respuesta del sistema en perfecta sincronía y ante la presencia de un desplazamiento en el tiempo (TOE) provocado. Para esto, se manipulara una señal de control entre el TX y el RX que simula al sincronizador ya que este no es incluido en el actual esquema.

ÍNDICE GENERAL

RESUMEN	VI
ÍNDICE GENERAL	VI
ÍNDICE DE TABLAS	X
ÍNDICE DE FIGURAS.....	XII
ABREVIATURAS	XVI
INTRODUCCION	19
1. OFDM Y EL ESTÁNDAR IEEE 802.16.....	21
1.1. EL ESQUEMA DE MODULACIÓN POR DIVISIÓN ORTOGONAL DE FRECUENCIAS.....	21
1.1.1. Transformada rápida de Fourier FFT/IFFT	24
1.2. EFECTOS DE OFDM EN CANALES DE DESVANECIMIENTO MULTITRAYECTORIA.....	34
1.3. EL ESTÁNDAR IEEE 802.16 [9].....	36
1.3.1. Símbolo OFDM en el dominio del Tiempo.....	37
1.3.2. Símbolo OFDM en el dominio de la Frecuencia.....	38
1.3.3. Parámetros del Símbolo OFDM.....	39
1.3.4. Codificación de Canal.....	39
2. DESCRIPCIÓN DEL SISTEMA	44
2.1. MODELO PROPUESTO	44
2.2. HARDWARE Y SOFTWARE UTILIZADO	46
2.3. DIAGRAMA DE BLOQUES DEL MODELO IMPLEMENTADO.....	49
2.4. PARÁMETROS DEL SISTEMA: BLOQUE “SETTINGS”	50
2.5. SINCRONIZACIÓN DEL SISTEMA: BLOQUE “CLOCKING”	51

3.	MODULADOR OFDM: BLOQUE “TRANSMISOR”	59
3.1.	FUENTE DE DATOS	63
3.4.	PILOTOS Y GUARDA	70
3.5.	ESCALAMIENTO	73
3.6.	BLOQUE IFFT	80
3.8.	INSERCIÓN DE GUARDA	88
4.	DEMODULADOR OFDM: BLOQUE “RECEPTOR”	95
4.1.	BLOQUE FFT	99
4.2.	CONTROLADOR FFT	99
4.3.	MAPEADOR DE CONSTELACIÓN	102
4.4.	DECODIFICADOR.....	106
4.5.	CONTROLADOR DE DATOS.....	112
4.6.	MEDICIÓN DE ERRORES	114
5.	FUNCIONAMIENTO DEL SISTEMA IMPLEMENTADO	118
5.1.	FRECUENCIA DE OPERACIÓN DEL SISTEMA IMPLEMENTADO	127
5.2.	GENERACIÓN DE LOS SÍMBOLOS Y BLOQUES OFDM	136
5.2.1.	Codificación de Datos.....	141
5.2.2.	Generación de las Portadoras de Piloto y Guarda.....	152
5.2.3.	Generación del Símbolo de Training.....	161
5.2.4.	Generación del Bloque OFDM en el dominio de la Frecuencia.....	166
5.2.5.	Generación del Bloque OFDM en el dominio del Tiempo.....	178
5.3.	EVALUACIÓN DE LOS DATOS RECUPERADOS.....	187
	CONCLUSIONES Y RECOMENDACIONES	202
	BIBLIOGRAFÍA.....	208

ÍNDICE DE TABLAS

Tabla I	Parámetros del Símbolo OFDM en el estandar IEEE 802.16	39
Tabla II	Puertos del bloque Settings.....	51
Tabla III	Valores máximos en las constelaciones gray.	65
Tabla IV	Bloque de Escalamiento	79
Tabla V	Puertos Principales Del Core FFT utilizado	81
Tabla VI	Puertos de Salida del Controlador Contador Tx.....	82
Tabla VII	Datos a cargarse en la IFFT según el valor de mux	87
Tabla VIII	Puertos del Codificador	107
Tabla IX	Pruebas de Rango de Frecuencias ante la variación de la Guarda y La Codificación	130
Tabla X	Pruebas de Rango de Frecuencias ante la variación de la Estructura del Bloque	132
Tabla XI	Pruebas de Rango de Frecuencias ante la variación de los Parámetros de Sincronización	134
Tabla XII	Puertos del bloque “Simulador Control Portadoras”	138
Tabla XIII	Valor de Variable Limite en funcion del Modo del bloque “Simulador_Control_Portadoras”	139
Tabla XIV	Codificador de Salida Codec_bit (“Simulador_Control_Portadoras”).....	140
Tabla XV	Parámetros del bloque Settings definidos para los experimentos de generacion de los bloques OFDM en el tiempo y la frecuencia	167

Tabla XVI	Codificador de la señal Inicio_simbolo en función de la señal TOE en simbolos OFDM con Guarda diferente de cero	189
Tabla XVII	Valores Promedio Resultantes de las pruebas de BER.....	199

ÍNDICE DE FIGURAS

Fig. 1. 1	Esquema tradicional de implementación de OFDM.....	22
Fig. 1. 2	Comparación entre la DFFT y el algoritmo de la FFT [3].....	25
Fig. 1. 3	Esquema de decimación en frecuencia para N=8 usando filtros butterfly [3].....	27
Fig. 1. 4	Diagrama de bloques de un modulador I/Q	32
Fig. 1. 5	Símbolo OFDM en el dominio del tiempo	37
Fig. 1. 6	Símbolo OFDM en el dominio de la frecuencia	39
Fig. 1. 7	Constelaciones Gray para la implementación de BPSK, QPSK, 16-QAM y 64QAM	40
Fig. 1. 8	Secuenciador Pseudoaleatorio de la secuencia de pilotos	41
Fig. 1. 9	Preámbulo Completo de Estándar IEEE 802.16 para aplicaciones WirelessMan.....	42
Fig. 1. 10	Estructura de los preámbulos en el tiempo para el DL en el IEEE 802.16 WirelessMan OFDM.....	43
Fig. 2. 1	Constelación Gray para 256-QAM	45
Fig. 2. 2	Kit Xtreme DSP Virtex Pro	47
Fig. 2. 3	Vista del Software ISE (Project Navigator) de Xilinx durante el desarrollo del proyecto	48
Fig. 2. 4	Diagrama de Bloques del Sistema.....	50
Fig. 2. 5	Esquemático de la etapa de Sincronización: Bloque de Clocking.....	55
Fig. 2. 6	Diagrama de Estados de Enables_Clocking.....	57
Fig. 2. 7	Simulación en ModelSim del bloque Enables_clocking.....	58
Fig. 3. 1	Estructura en tiempo y frecuencia del bloque OFDM transmitido.....	60
Fig. 3. 2	Diagrama de bloques del Transmisor	61
Fig. 3. 3	Diagrama de Bloques del Codificador.....	67

Fig. 3. 4	Diagrama ASM del Controlador del Codificador	68
Fig. 3. 5	Maquina de estados del Controlador Contador TX.....	84
Fig. 3. 6	Diagramas de bloques del modulo Inserción de guarda	89
Fig. 3. 7	Diagramas de estados del controlador del modulo de Inserción de guarda	91
Fig. 4. 1	Diagrama de bloques del Receptor.....	97
Fig. 4. 2	Diagrama ASM del controlador FFT	100
Fig. 4. 3	Ejemplo de división de áreas para el mapeo de Ck en 16-QAM.	104
Fig. 4. 4	Esquemático del bloque mapeador de constelación	104
Fig. 4. 5	Diagrama de Bloques del Decodificador.....	108
Fig. 4. 6	Diagrama ASM del Controlador del Decodificador	109
Fig. 4. 7	Diagrama ASM del Controlador de Datos.....	113
Fig. 4. 8	Esquemático del bloque Medición de errores.....	115
Fig. 5. 1	Esquema de las pruebas del sistema mediante el CSP.....	119
Fig. 5. 2	Cable Paralel IV de Xilinx.....	119
Fig. 5. 3	Diagrama de procesos para las pruebas mediante el CSP.....	121
Fig. 5. 4	Mapeo de señales mediante el CSP Inserter	122
Fig. 5. 5	Estructura de Reseteo sugerida por el XTREME DSP	123
Fig. 5. 6	Clocking propuesto por el XTREME DSP	124
Fig. 5. 7	Ventanas del Fuse para el control de los Resets y el reloj de entrada.	125
Fig. 5. 8	Datos recuperados: (a) Sistema en perfecta sincronía, (b) Sistema desincronizado.	128
Fig. 5. 9	Pruebas de Rango de Frecuencias ante la variación de la Guarda y la Codificación.	131
Fig. 5. 10	Pruebas de Rango de Frecuencias ante la variación de la Estructura del Bloque	133
Fig. 5. 11	Pruebas de Rango de Frecuencias ante la variación de los parámetros Div y Codec_bit.....	135

Fig. 5. 12 Diagrama ASM de la máquina secuencial del bloque “Simulador Control Portadoras”	139
Fig. 5. 13 Esquemático de las pruebas del bloque de Codificación de Datos.	142
Fig. 5. 14 Diagrama de Tiempo de las Pruebas de Codificación de Datos .	144
Fig. 5. 15 Diagramas de tiempo de Codificación de Datos para diferentes esquemas de codificación.	145
Fig. 5. 16 Tabla del CSP Analyzer para las pruebas de Codificación.	146
Fig. 5. 17 Constelaciones obtenidas a partir de las pruebas de Codificación de datos.	149
Fig. 5. 18 Análisis en EXCEL de los datos reexportados en la prueba de Codificación de datos.	151
Fig. 5. 19 Esquemático de las pruebas de Pilotos y Guarda.	153
Fig. 5. 20 Pruebas de Pilotos y Guarda mediante el CSP Analyzer.	156
Fig. 5. 21 Constelaciones resultantes del bloque Pilotos y Guarda.	157
Fig. 5. 22 Salida del bloque Pilotos y Guarda en función de ω_k	158
Fig. 5. 23 Análisis en EXCEL en las pruebas de Pilotos y Guarda.	160
Fig. 5. 24 Esquemático de las pruebas del Training.	161
Fig. 5. 25 Pruebas del Training mediante el CSP Analyzer.	163
Fig. 5. 26 Constelaciones resultantes del bloque Training.	164
Fig. 5. 27 Análisis en EXCEL de los datos exportados en las pruebas de Training.	165
Fig. 5. 28 Gráfica de buses Dir_Ck y Mux en función del tiempo.	169
Fig. 5. 29 Componentes de la señal C_k procesada en el tiempo.	170
Fig. 5. 30 Diagrama de tiempo de las pruebas en frecuencia.	171
Fig. 5. 31 Captura de la ventana Listing de las señales Dir_Ck, Mux y Ck.	173
Fig. 5. 32 Evaluación del selector de MUX durante un símbolo de datos, mediante filtros en Excel.	175

Fig. 5. 33 Evaluación de los habilitadores En_Datos y En_pilotos mediante filtros en Excel.	176
Fig. 5. 34 Constelaciones transmitidas en función del selector MUX.	178
Fig. 5. 35 Diagrama de buses de Dir_Ck y Dir_Xn	181
Fig. 5. 36 Diagrama de tiempo de las pruebas de Bloque y Símbolo en el tiempo.	182
Fig. 5. 37 Tablas del CSP Analyzer para las pruebas de generación de símbolo y bloque en el tiempo.	183
Fig. 5. 38 Condiciones para X_{RE} , XN_{IM} , $I[N]$ y $Q[N]$ en función del periodo de muestreo N	185
Fig. 5. 39 Análisis del proceso de inserción de Guarda mediante Excel.	186
Fig. 5. 40 Constelaciones recuperadas en codificación BPSK.	190
Fig. 5. 41 Constelaciones recuperadas en codificación QPSK.	191
Fig. 5. 42 Constelaciones recuperadas en codificación 16QAM.	192
Fig. 5. 43 Constelaciones recuperadas en codificación 64QAM.	193
Fig. 5. 44 Constelaciones recuperadas en codificación 256QAM.	194
Fig. 5. 45 Efectos del TOE en el procesamiento de la FFT en el RX.	196
Fig. 5. 46 Resultado de las pruebas de BER en función del TOE.	200

ABREVIATURAS

ADC	Analog to Digital Converters
ASCII	American Standart Code for Information Interchange
BER	Bit Error Rate
BPSK	Phase Shift Keying
CSP	Chip Scope Pro
DAB	Digital Audio Broadcasting
DAC	Digital to Analog Converters
DC	Direct Current
DCM	Digital Clock Manager
DFT	Discrete Fourier Transform
DL	DownLink
DSP	Digital Signal Processing
DVB-T	Digital Video Broadcasting - Terrestrial
ETSI	European Telecommunications Standards Institute
FEC	Forward Error Correction
FFT	Fast Fourier Transform
FPGA	Field Programmable Gate Array
I	Inphase

IBM	International Business Machines
ICI	InterCarrier Interference
ICON	Integrated Controller Pro
IEEE	Institute of Electrical and Electronics Engineers
IFFT	Inverse Fast Fourier Transform
ILA	Integrated Logic Analyzer
ISE	Integrated Software Environment
ISI	InterSimbol Interference
LAN	Local Area Network
LFSR	Linear Feedback Shift Register
LUT	LookUp Table
NLOS	Non Line Of Sight
OFDM	Orthogonal Frequency Division Multiplexing
Q	Quadrature
QAM	Quadrature Amplitude Modulation
QPSK	Quadrature Phase Shift Keying
RAM	Random Access Memory
RF	Radiofrecuencia
RX	Receptor
SNR	Signal to Noise Ratio
TOE	Timing Offset Error
TX	Transmisor

UL	UpLink
USB	Universal Serial Bus
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuit
VIO	Virtual Input Output
VLSI	Very Large Scale Integration
WLAN	Wireless Local Area Network
XST	Xilinx Synthesis Tool

INTRODUCCION

Las exigencias actuales del mercado de las Telecomunicaciones provocan una visible tendencia a la banda ancha. No obstante, en algunos casos su implementación puede resultar técnicamente complicada, lo cual se ve reflejado en altos costos, provocando una demanda y servicio insatisfecho. En los países en vías de desarrollo como el nuestro esta situación es más notoria, dejando consecuencias nefastas en el campo de la educación, pilar fundamental para el desarrollo de los pueblos.

Las comunicaciones inalámbricas han brindado una solución parcial, dado que su alcance es más versátil por no necesitar infraestructura física entre los puntos. Sin embargo, este tipo de implementación es accesible para apenas un 30% de los posibles suscriptores debido a la exigencia de línea de vista. Orientados a este problema, las soluciones basadas en el esquema OFDM son objeto de continuo estudio y desarrollo alrededor del mundo.

Por esta razón, el presente proyecto de tesis tiene como objetivo iniciar una investigación seria en esta área, tomando a OFDM como posible solución de banda ancha en ambientes selectivos. Para lograr este objetivo, nuestro

estudio se basa en el uso de FPGAs (Field Programmable Gate Array), dada la versatilidad de este integrado programable como elemento de diseño y desarrollo. De esta forma, nuestra meta puntual será el diseño e implementación de un sistema Modulador/Demodulador OFDM mediante el uso de FPGAs. Por otra parte, el diseño del mismo debe de garantizar la escalabilidad a mediano y largo plazo ya que nuestro objetivo final es dejar fundada las bases para futuros análisis. Debido a esto, el esquema resultante debe tener como objetivo constituirse en una herramienta académica e investigativa que fomente los futuros estudios en el área.

Sin duda, el punto de partida para cualquier diseño o implementación es el estudio profundo del sistema a investigar. Por esta razón, en el capítulo I, se cubrirá de forma teórica el concepto de OFDM y sus propiedades en ambientes selectivos. Además, también se introducirán las especificaciones de la capa física del estándar IEEE 802.16 para aplicaciones WirelessMan OFDM, ya que estas han sido tomadas como referencia para nuestro estudio. En base de este estudio teórico, en el capítulo II, se definirán las características generales del diseño y se introducirán las herramientas de hardware y software utilizadas. Posteriormente, en los capítulos III y IV se describirá profundamente el bloque Modulador y Demodulador diseñado. Finalmente, en el capítulo V, se documentaran las pruebas del sistema implementado en hardware.

1. OFDM y el Estándar IEEE 802.16

1.1. El Esquema de Modulación por División Ortogonal de Frecuencias

La modulación por división de frecuencias ortogonales OFDM, es un concepto antiguo el cual fue introducido a principio de la década de los 60's. OFDM se basa en la técnica de modulación multiportadora utilizada en enlaces militares de alta frecuencias [1]. En esta técnica el ancho de banda disponible es dividido en segmentos en los cuales se transmiten subportadoras ortogonales entre sí. Al decir que las subportadoras son ortogonales, nos referimos al hecho de que teóricamente el ancho de banda de cada una de las portadoras es lo suficientemente pequeño para que estas no se traslapen.

A pesar de esto, OFDM no sería explotado sino hasta el advenimiento de técnicas DSP tales como la transformada rápida de Fourier FFT y el desarrollo de circuitos integrados de muy larga escala (VLSI). La

explicación de esto se debe a que el esquema tradicional (fig. 1.1) de OFDM es implementado de forma analógica mediante el uso de un arreglo de generadores sinusoidales y demoduladores coherentes de gran tamaño (uno por cada portadora). En definitiva, dado este esquema, la implementación de OFDM es muy cara y compleja en términos prácticos.

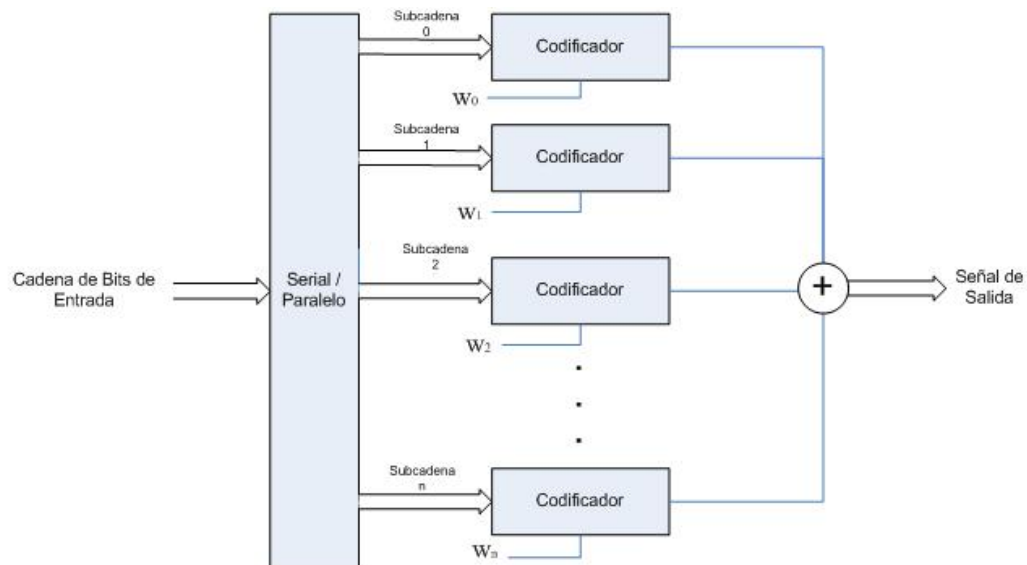


Fig. 1. 1 Esquema tradicional de implementación de OFDM

Este panorama cambio drásticamente mediante uso de la FFT/IFFT como base para la implementación de OFDM. Diferentes aplicaciones tales como DAB (digital audio broadcasting), DVB-T (digital video

terrestrial broadcasting), wireless LAN (WLAN) IEEE 802.11 y WiMax IEEE 802.16 han adoptado OFDM satisfactoriamente.

La ventaja de aplicar la FFT/IFFT en la realización de OFDM radica en el hecho de que su implementación es mucho mejor en términos económicos. Por otra parte, la implementación se reduce a analizar los bits a codificar como NFFT componentes complejos C_k en el dominio de la frecuencia, los cuales son mucho más sencillos de examinar que la señal a transmitirse en el dominio del tiempo. Esto se debe a que la señal en el dominio del tiempo corresponde a una suma de señales sinusoidales donde la información de la portadora k correspondiente a la frecuencia ω_k viajaría a través de la magnitud A_k y la fase ϕ_k dependiendo del tipo de codificación a utilizarse:

$$x(t) = \sum_k A_k \cos(\omega_k t + \phi_k) \quad (1.1)$$

Debido a la importancia de la FFT a continuación se expondrá una descripción de la misma.

1.1.1. Transformada rápida de Fourier FFT/IFFT

La transformada rápida de Fourier es simplemente un algoritmo para la rápida evaluación numérica de integrales de Fourier desarrollado en los laboratorios de IBM. La ventaja de la FFT con respecto a las series y transformadas discretas de Fourier, está en la implementación de interesantes ideas que habían sido dejadas de lado por aspectos prácticos, principalmente por el tiempo que se toma para calcular dichas transformadas. La FFT, por otra parte, ha demostrado adaptarse perfectamente a la ejecución digital eficiente lo cual ha provocado el desarrollo de técnicas de análisis de señales y sistemas discretos a un ritmo acelerado [2]. Un ejemplo tangible de esto es la implementación de OFDM, el cual es posible mediante el uso de un bloque IFFT/FFT en el modulador y el demodulador. De esta manera, se obtiene una implementación lo suficientemente rápida y con las notorias ventajas que acarrea el desarrollo a nivel DSP. Como se puede observar en la fig. 1.3 la diferencia de velocidad de cálculo entre la tradicional transformada discreta y la FFT aumenta según el número de muestras a analizar. Esto se debe a que mientras en una crece el número de operaciones necesarias para la resolución de forma exponencial, en la otra lo hace de forma prácticamente lineal.



Fig. 1.2 Comparación entre la DFFT y el algoritmo de la FFT [3]

Si tomamos como referencia la definición de la transformada discreta, tendremos que la DFT de una secuencia de N puntos $\{x[n]\}$, $0 < n < N - 1$ esta definida por:

$$X[u] = \sum_{n=0}^{N-1} x[n] e^{j2\pi nu/N} \quad (1.2)$$

Donde $X[u]$ denota el u th valor de la secuencia de N valores resultantes de la DFT. Asimismo, los N puntos de la secuencia

$\{x[n]\}$ en nuestro caso representan a N muestras tomadas a partir de una onda $x(t)$ con una frecuencia de muestreo f_s .

De manera similar, la inversa de la transformada discreta de Fourier, a partir de una secuencia de N puntos $\{X[u]\}$, $0 < u < N - 1$, esta dada por:

$$x[n] = \frac{1}{N} \sum_{u=0}^{N-1} X[u] e^{-\frac{j2\pi nk}{N}} \quad (1.3)$$

La idea básica de la implementación de la DFT mediante el uso del algoritmo de la FFT, es la de separar la suma ((1.2) y (1.3)) en dos partes. Dependiendo del tipo de división, al algoritmo de la FFT se lo define como decimación en el tiempo o en la frecuencia. Si este proceso de división se repite recursivamente para cada una de las sumatorias tendremos que eventualmente; si N es una potencia de 2, al final se obtendría un producto de términos simples. De esta manera el resultado final consistiría simplemente en la suma de dichos productos. La fig. 1.3 nos muestra un esquema de decimación en frecuencia para la implementación de una DFT con $N_{FFT}=8$ y mediante el uso de

los denominados filtros butterfly. Mayor información sobre este tipo de aplicaciones es posible encontrarla en [1], [3] y [4], así como en la literatura actual.

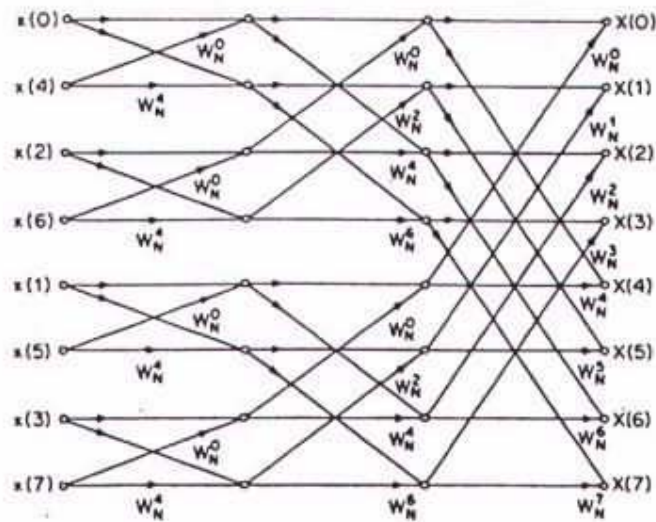


Fig. 1.3 Esquema de decimación en frecuencia para $N=8$ usando filtros butterfly [3]

1.1.1.1. Interpretación de la FFT y la IFFT

La clave básica para entender el funcionamiento de la IDFT, es interpretando secuencia $\{X[u]\}$ como un estimador de la transformada de Fourier de la señal $x(t)$ (versión continua de la secuencia discreta $\{x[n]\}$) expresado a través de N subportadoras escaladas. Para ilustrar esto, definamos la variable k en función de

la variable u (dominio de la frecuencia en (1.3)), como se indica a continuación:

$$k = \begin{cases} u & ; 0 \leq u \leq \frac{NFFT}{2} \\ u - NFFT & ; \frac{NFFT}{2} < u < NFFT \end{cases} \quad (1.4)$$

De esta manera, el dominio de la variable k estaría determinado por todos los enteros comprendidos en $(-NFFT/2, NFFT/2]$. Por otra parte, si dividimos la sumatoria de (1.3) en dos partes y aplicamos el cambio de variable especificado en (1.4), tendremos que:

$$\begin{aligned} x[n] &= \frac{1}{NFFT} \sum_{u=0}^{\frac{NFFT}{2}} X[u] e^{-\frac{j2\pi n u}{NFFT}} + \\ &\quad \frac{1}{NFFT} \sum_{u=\frac{NFFT}{2}+1}^{N-1} X[u] e^{-\frac{j2\pi n u}{NFFT}} \\ &= \frac{1}{NFFT} \sum_{k=0}^{\frac{NFFT}{2}} X[k] e^{-\frac{j2\pi n k}{NFFT}} + \\ &\quad \frac{1}{NFFT} \sum_{k=-\frac{NFFT}{2}+1}^{-1} X[k + NFFT] e^{-\frac{j2\pi n (k+NFFT)}{NFFT}} \end{aligned} \quad (1.5)$$

Si observamos el segundo factor y tomamos en cuenta que por definición $X[u]$ es periódica cada N puntos [3], tendremos que:

$$X[k + N] = X[k]$$

$$e^{-\frac{j2\pi n(k+NFFT)}{NFFT}} = e^{-\frac{j2\pi nk}{NFFT}} \cdot e^{-j2\pi n} = e^{-\frac{j2\pi nk}{NFFT}} \quad (1.6)$$

De todo esto podríamos expresar que:

$$x[n] = \frac{1}{NFFT} \sum_{k=1-\frac{NFFT}{2}}^{\frac{NFFT}{2}} X[k] e^{-\frac{j2\pi nk}{NFFT}} \quad (1.7)$$

Es decir, si definimos al componente complejo C_k con el valor $X[k]$ correspondiente a la variable k , tendríamos que:

$$x[n] = \frac{1}{NFFT} \sum_{k=1-\frac{NFFT}{2}}^{\frac{NFFT}{2}} C_k e^{-\frac{j2\pi nk}{NFFT}} \quad (1.8)$$

De aquí se deduce que la señal $x(n)$ correspondería a la sumatoria de N fasores C_k , con frecuencia f_k , en el

instante $t=n$. Es decir, podríamos definir a la secuencia compleja $\{C_k\}$ como la representación fasorial de N subportadoras equidistantemente espaciadas en el ancho de banda abarcado por $-\frac{f_s}{2} < f < \frac{f_s}{2}$, donde f_s representa la frecuencia de muestreo. Es decir:

$$f_k = k \frac{f_s}{N} \quad (1.9)$$

De esta forma, la secuencia digital de fasores $\{C_k\}$ transportaría la información de la transformada $X(f)$, mientras que la secuencia digital $\{x[n]\}$ acarrea la información de la señal continua $x(t)$.

Si tomamos en cuenta la interpretación fasorial de (1.8), el valor máximo de la secuencia $x[n]$ se daría en el instante $t=n$ donde los NFFT fasores tengan la misma fase. De aquí se tiene que, un método para determinar la máxima magnitud posible de un bloque IFFT es asignando las entradas C_k con el máximo valor posible y con la misma fase entre sí, de esta

forma el primer valor $x[0]$ de la secuencia $\{x[n]\}$ corresponderá al valor máximo:

$$x(0) = \frac{1}{NFFT} \sum_{k=1-\frac{NFFT}{2}}^{\frac{NFFT}{2}} C_k e^{-\frac{j2\pi(0)k}{NFFT}} = \frac{1}{NFFT} \sum_{k=1-\frac{NFFT}{2}}^{\frac{NFFT}{2}} C_k \quad (1.10)$$

Este método será de gran utilidad en el capítulo III, a la hora de analizar aspectos de escalamiento en los bloques FFT/IFFT, es decir la máxima ganancia del bloque. Como se explicará posteriormente esto es de gran importancia porque determina cuan eficiente es el sistema.

1.1.1.2. Implementación de OFDM mediante la FFT

Se puede observar que la implementación de OFDM se podría realizar mediante un bloque IFFT, lo cual mostraría una secuencia en el tiempo correspondientes a la suma de N fasores C_k donde la información estaría almacenada en la fase y en la amplitud de los mismos. Sin embargo, si tomamos como referencia el concepto básico de OFDM introducido al principio de este capítulo, se desea tener en la etapa RF una suma

de señales reales sinusoidales y no a través de una secuencia compleja $\{x[n]\}$. Una manera de conseguir esto último, es mediante la implementación de un bloque modulador I/Q en la frecuencia intermedia, para posteriormente con el uso de un oscilador subir la señal en frecuencia a la etapa RF.

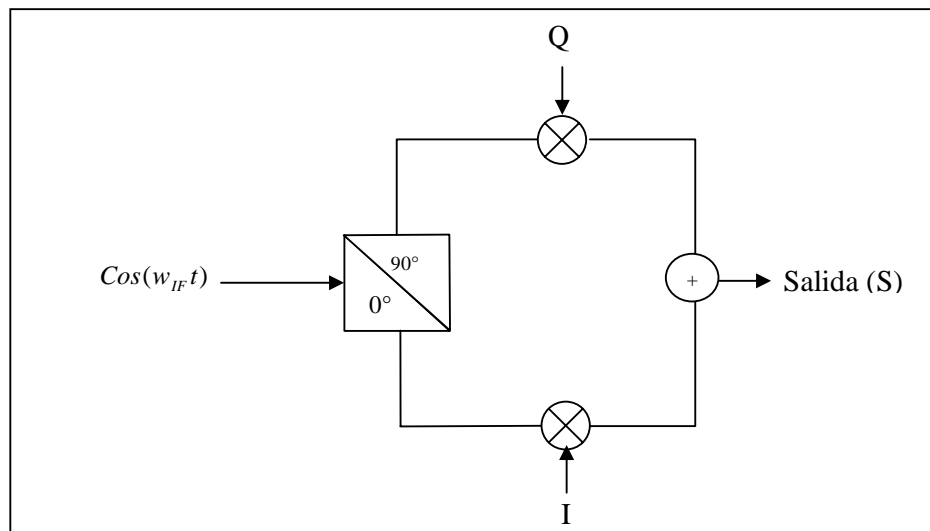


Fig. 1. 4 Diagrama de bloques de un modulador I/Q

Como se puede observar en la fig. 1.4 la salida S de un modulo I/Q en función de las entradas I y Q esta dada por:

$$\begin{aligned}
 S &= I.Cos(w_{IF} t) + Q.Cos(w_{IF} t + \pi / 2) \\
 S &= I.Cos(w_{IF} t) + Q.Sen(w_{IF} t)
 \end{aligned}
 \tag{1.11}$$

Si definimos a la señal compleja $x(t)$ como la versión continua de la secuencia $\{x[n]\}$ obtenida a partir de la IFFT, tendríamos que:

$$x(t) = \sum_{k=0}^{N-1} A_k e^{(w_k t + \phi_k)} \quad (1.12)$$

Además, si definimos a las señales $I(t)$ y $Q(t)$ como la parte real e imaginaria de $x(t)$ respectivamente tendríamos que:

$$I(t) = \sum_{k=0}^{N-1} A_k \cdot \text{Cos}(w_k t + \phi_k) \quad (1.13)$$

$$Q(t) = \sum_{k=0}^{N-1} A_k \cdot \text{Sen}(w_k t + \phi_k) \quad (1.14)$$

Se puede demostrar que si reemplazamos estos resultados en (1.11), se tiene que las señales I y Q resultantes de la parte real e imaginaria de la IFFT alimentadas al bloque I/Q producen una señal OFDM de la siguiente forma:

$$S(t) = \sum_{k=0}^{N-1} A_k \cdot \text{Cos}(\{w_{IF} - w_k\}t + \phi_k) \quad (1.15)$$

Finalmente en el modulador, se necesitaría un convertidor de subida para dejar la señal en la frecuencia RF deseada. El esquema del demodulador OFDM puede ser aplicado de manera similar mediante el uso de un bloque demodulador I/Q. De esta manera, la implementación de OFDM se dividiría en dos partes: la parte DSP y la parte analógica.

1.2. Efectos de OFDM en Canales de Desvanecimiento Multitrayectoria

Como hemos visto, OFDM es una técnica de modulación que consiste en dividir una señal de banda ancha en N subportadoras ortogonales. En sistemas inalámbricos multitrayectoria de banda ancha, esta técnica es usualmente preferida sobre las soluciones de portadora simple. Este hecho se debe a que la combinación de la diversidad en frecuencia que provee OFDM con la técnica del prefijo cíclico permite una mayor robustez en este tipo de ambientes selectivos [5].

El hecho de dividir el ancho de banda del símbolo OFDM en N subportadoras implica que cada subportadora tendrá la N -ésima parte del ancho de banda del símbolo completo. Si analizamos que el desvanecimiento de frecuencia selectiva en corta escala, se da en una señal cuando el ancho de banda de la misma es mayor al ancho de

banda de coherencia, se puede concluir que dependiendo del N cualquier canal que experimente un desvanecimiento de frecuencia selectiva, puede ser dividido en múltiples canales con desvanecimiento plano. El desvanecimiento de frecuencia selectiva debe evitarse a toda costa ya que produce interferencia intersímbolo (ISI) y una irreducible tasa de error (BER) [6] [7] [8].

Sin embargo, el hecho de aumentar deliberadamente el N tiene serias desventajas ya que mientras mayor sea N , mas difícil será mantener la ortogonalidad entre las subportadoras. Este es un asunto crítico ya que la ortogonalidad permite al sistema OFDM estar libre de ICI. Por esta razón, se utiliza un prefijo cíclico junto a OFDM.

Esta técnica es muy simple, ya que consiste en concatenar al inicio del símbolo OFDM un prefijo cíclico que esta formado por los últimos puntos de dicho símbolo. El objetivo de este método es el de disminuir el ISI mediante la espera de un tiempo, para que los componentes multitrayectoria correspondientes al símbolo anterior sean eliminados. Es decir, el receptor debe de esperar un tiempo de guarda T_g para tomar en cuenta los datos a demodular. Para que esta técnica tenga completo éxito, el tiempo de guarda debe ser mayor al esparcimiento de retardo del canal.

El intervalo de guarda puede realizarse mediante la inserción de ceros. Sin embargo, el uso del prefijo cíclico tiene la propiedad de transformar la convolución lineal con el canal en una convolución circular [1]. Esta propiedad permite la implementación de ecualizadores en el dominio del tiempo orientados a aplicaciones OFDM. Este tipo de ecualizador tiene la función de acortar la respuesta efectiva al impulso del canal, de esta forma reduce el valor requerido para T_g , ganando así eficiencia en el sistema.

1.3. El Estándar IEEE 802.16 [9]

Una de las razones por la cual el estándar IEEE 802.16 es objeto de continuo estudio, es su capacidad de implementar banda ancha en ambientes inalámbricos sin línea de vista (NLOS). El éxito de Wimax (que recoge las especificaciones del IEEE 802.16 y su equivalente de la ETSI [10] para soluciones fijas inalámbricas) ha provocado una inmensa escalada de estudios investigativos alrededor de OFDM como base de soluciones banda ancha, en ambientes donde la implementación inalámbrica era impensable.

Debido a esto, en nuestro estudio se ha tomado como referencia principal las especificaciones de la capa física del estándar para aplicaciones WirelessMan OFDM [9]. De aquí se infiere la importancia

de desarrollar alguno de estos criterios especialmente en lo referente a la descripción y estructura del símbolo OFDM y la codificación de canal utilizada.

1.3.1. Símbolo OFDM en el dominio del Tiempo

Para crear la forma de onda OFDM el estándar especifica el uso de la IFFT. El tiempo de duración de la señal resultante es conocido como tiempo de símbolo útil T_b . Con el objetivo de recolectar las componentes Multitrayectoria de la señal recibida, el método de inserción de guarda es implementado mediante la inserción de un prefijo cíclico. Este prefijo es formado por la fracción del símbolo útil en los últimos instantes T_g (fig. 1.5).

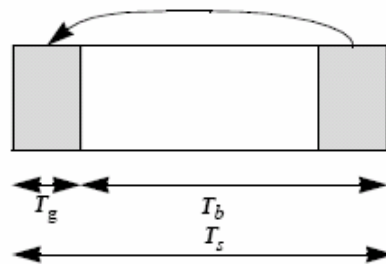


Fig. 1. 5 Símbolo OFDM en el dominio del tiempo

Otra ventaja del prefijo cíclico es que si el receptor comete un error en la sincronización de datos e interpreta el inicio de señal durante la guarda, el efecto en frecuencia sería un desfase determinado por la frecuencia de la subportadora y el tiempo del desfase en el dominio del tiempo. Si en vez del prefijo cíclico se

utilizaría un relleno con cero, el efecto en la salida de la FFT fuera indeterminado.

A pesar de esto, es importante tener en cuenta que la implementación de la guarda conlleva una pérdida de E_b / N_0 y SNR ya que en el tiempo T_g se transmite energía que no será utilizada en el receptor. Debido a esto, tampoco es deseable que T_g sea muy elevado.

1.3.2. Símbolo OFDM en el dominio de la Frecuencia

El símbolo OFDM está conformado por un determinado número de subportadoras. Este número define el tamaño de la FFT por lo cual corresponde a NFFT. El estándar define tres tipos de subportadoras:

- **Subportadoras de Datos:** Utilizadas para transmisión de datos.
- **Subportadoras de Piloto:** Utilizadas para propósito de estimación.
- **Subportadoras de Nulas:** Utilizadas para bandas de guarda, subportadoras inactivas y portadora DC(C_0)

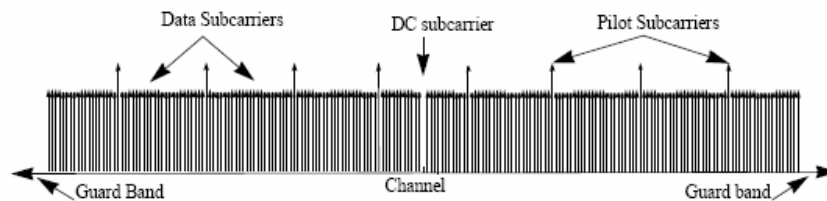


Fig. 1. 6 Símbolo OFDM en el dominio de la frecuencia

1.3.3. Parámetros del Símbolo OFDM

La Tabla I muestra los parámetros utilizados en el estándar para la implementación del símbolo OFDM.

Tabla I
Parámetros del Símbolo OFDM en el estándar IEEE 802.16

Parámetro	Valor
NFFT	256
N _{Used}	200
G	1/4, 1/8, 1/16, 1/32
Número de subportadoras de guarda en frecuencias inferiores	28
Número de subportadoras de guarda en frecuencias Superiores	27
Indices K de subportadoras de guarda	-128,-127,...,-101 +101,+102,...,127
Indices k de subportadoras de piloto	-88,-63,-38,- 13,13,38,63,88

1.3.4. Codificación de Canal

En la etapa de codificación de canal, el estándar especifica las técnicas utilizadas para la aleatorización, FEC, intercalado, modulación y estructura del preámbulo. Debido a que el actual

proyecto no tiene como objetivo abarcar las propiedades básicas del proceso de aleatorización, FEC e intercalado; nuestra descripción se centrará en la modulación de datos y la estructura del preámbulo.

1.3.4.1. Modulación de Datos

Para la modulación de los datos de entrada, el estándar especifica la implementación de BPSK, QPSK, 16-QAM y 64-QAM en código gray. La implementación de 64-QAM es opcional para el caso de las bandas no licenciadas.

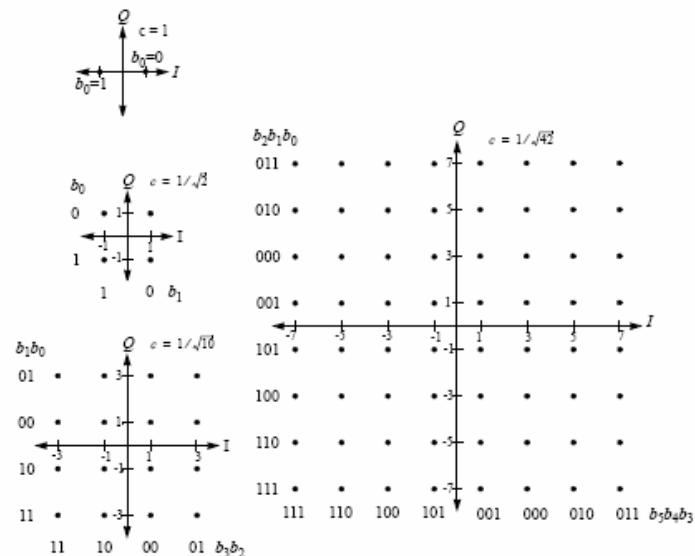


Fig. 1.7 Constelaciones Gray para la implementación de BPSK, QPSK, 16-QAM y 64QAM

1.3.4.2. Modulación de Pilotos

Las subportadoras de pilotos deben de ser insertadas en la constitución de los símbolos de acuerdo a las posiciones especificadas en la tabla I. Por otra parte, los valores de los pilotos en las tramas de bajada y subida son especificados por (1.16) y están en función de la secuencia binaria pseudoaleatoria $\{\omega_k\}$.

$$\text{DL: } C_{-88} = C_{-38} = C_{63} = C_{88} = 1 - 2\omega_k \text{ y}$$

$$C_{-63} = C_{-13} = C_{13} = C_{38} = 1 - 2\bar{\omega}_k$$

$$\text{UL: } C_{-88} = C_{-38} = C_{13} = C_{38} = C_{63} = C_{88} = 1 - 2\bar{\omega}_k \text{ y}$$

$$C_{-63} = C_{-13} = 1 - 2\bar{\omega}_k \quad (1.16)$$

Por otra parte, El valor ω_k es determinado por el secuenciador pseudoaleatorio de la fig. 1.8 el cual esta definido por el polinomio $X^{11} + X^9 + 1$. La diferencia entre la trama de bajada (DL) y de subida (UL) radica en el valor de inicialización del secuenciador.

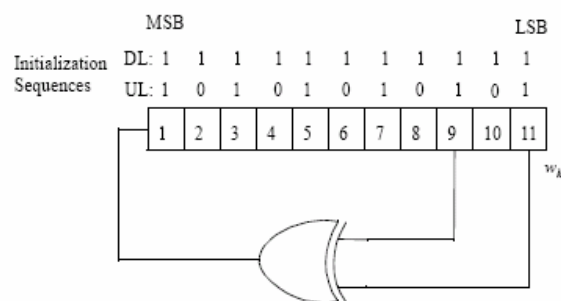


Fig. 1. 8 Secuenciador Pseudoaleatorio de la secuencia de pilotos

parte el segundo símbolo es producto de tomar solamente las portadoras pares. El resultado en el tiempo es que el primer símbolo está conformado por cuatro repeticiones de 64 puntos, mientras que el segundo por dos repeticiones de 128 puntos (fig. 1.10).

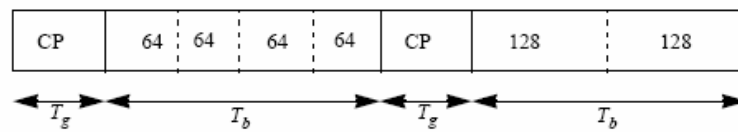


Fig. 1.10 Estructura de los preámbulos en el tiempo para el DL en el IEEE 802.16 WirelessMan OFDM.

En definitiva, la utilización del preámbulo completo se aplica siempre y cuando no se traten de tramas de subcanalización. En este caso, los preámbulos no son tomados a partir del preámbulo completo sino de una secuencia especialmente definida para la subcanalización. No obstante, como en este proyecto la subcanalización no es tomada en cuenta, no se abarcará la estructura de sus preámbulos. Si se desea profundizar sobre la estructura de los símbolos de los diferentes preámbulos se recomienda revisar en el estándar IEEE 802.16, en la sección 8.3.3.6.

2. Descripción del Sistema

2.1. Modelo Propuesto

El modelo propuesto se basa en la implementación de un modulador y demodulador OFDM. El sistema OFDM solamente abarca la parte DSP y para la implementación hace uso del algoritmo de la IFFT/FFT. El número de puntos de la FFT corresponde al valor NFFT definido en la tabla I. Por otra parte, las salidas del transmisor y las entradas del receptor son expresadas a través de las señales I y Q.

El Transmisor se encarga de producir los símbolos, los cuales a su vez forman los denominados bloques OFDM. Los bloques están formados por dos tipos de símbolos OFDM: los símbolos de Training y los Símbolos de Datos. Los Símbolos de Training conforman la cabecera del bloque y corresponden al preámbulo completo mostrado en la

sección 1.3.4.3. Los símbolos de Datos llevan la información de los datos codificados y tienen la estructura definida en la Tabla I.

La codificación de canal utilizada se limita a modular los bits de entrada en modulación BPSK, QPSK, 16QAM, 64QAM y 256QAM. Las constelaciones utilizadas en el modelo para los casos BPSK, QPSK, 16QAM y 64QAM son especificadas en la fig. 1.7 y corresponden a constelaciones gray. Para el caso de 256QAM también se ha escogido la modulación Gray, los valores de esta constelación se pueden observar en el estándar IEEE 802.16 2004 para aplicaciones WirelessMan-SCa:

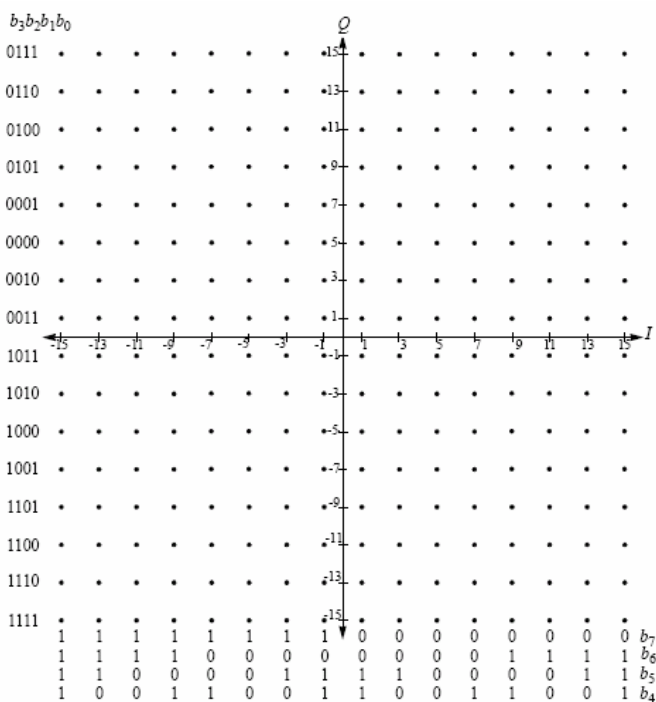


Fig. 2. 1 Constelación Gray para 256-QAM

Una vez que los símbolos son conformados en el dominio de la frecuencia y transformados al dominio del tiempo, el transmisor realiza la inserción del prefijo cíclico de guarda.

El sistema propuesto no abarca el proceso de Sincronización entre el transmisor y el Receptor. A pesar de esto, para base de futuros proyectos, el TX tiene la funcionalidad de crear los símbolos de Training necesarios para la sincronización. La detección del inicio del símbolo útil es simulada mediante una señal de control conectada directamente entre el TX y el RX. Por otra parte, el controlador de los datos en el receptor no toma en cuenta los símbolos de Training. En definitiva, el Receptor, descarta la guarda de todos los símbolos e ignora los símbolos de Training, para una correcta recuperación de los datos.

Una vez que los datos son recuperados de los símbolos OFDM, el receptor compara la señal recibida con la secuencia enviada. De esta forma se tiene la capacidad de calcular el BER que será el parámetro a medir en la etapa de evaluación.

2.2. Hardware y Software utilizado

En lo referente al hardware, el FPGA utilizado fue el Virtex II Pro [11] de Xilinx, de manera más específica se hizo uso del Kit Xtreme DSP Virtex

II Pro [12]. Por esta razón, muchas de las especificaciones del proyecto han sido orientadas al uso de dicho hardware. No obstante, debido a lo modular del sistema, sus bloques o incluso todo el esquema puede ser implementado en otro FPGA o Kit con muy poca o ninguna adaptación. En este aspecto es importante tener en cuenta que los resultados obtenidos en la etapa de síntesis e implementación corresponden al uso del FPGA y del Kit especificado. El sistema presentado es completamente digital y no hace uso de los convertidores DAC y ADC proporcionados por el Kit. A pesar de esto, como se dijo anteriormente, el diseño ha sido orientado al uso de los mismos.



Fig. 2. 2 Kit Xtreme DSP Virtex Pro

Para la etapa de diseño se hizo uso de herramientas de simulación tales como Matlab, Simulink y Modelsim (Simulaciones de modelos VHDL). De igual manera, todo el proyecto fue diseñado mediante el uso del Project Navigator de Xilinx y la técnica de diseño fue Top-Down, es decir fue un diseño jerárquico de arriba hacia abajo. La mayoría de los módulos y submódulos han sido diseñados mediante esquemáticos o archivos de código VHDL. En el caso de los esquemáticos se utilizó la herramienta de Xilinx XST (Xilinx Squematic Tool). Además de esto, existen algunos módulos implementados mediante el uso de “Cores”, que son modelos parametrizables provistos por el Core Generator de Xilinx.

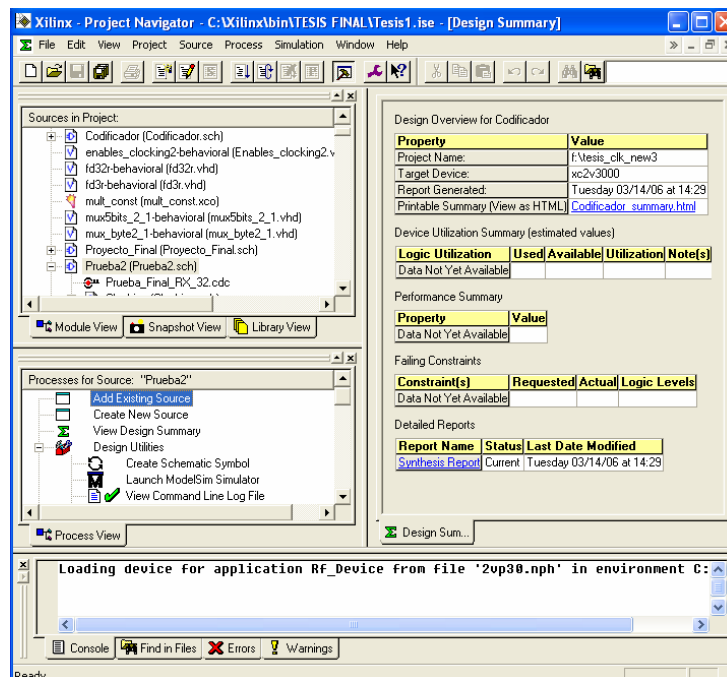


Fig. 2. 3 Vista del Software ISE (Project Navigator) de Xilinx durante el desarrollo del proyecto

Para la etapa de evaluación del Sistema se utilizó el Chip Scope Pro Analyzer de Xilinx. Esta herramienta permite cargar al computador algunas señales desde el interior del chip cuando se cumpla cierta condición. En este proyecto, las especificaciones de las señales a ser captadas como datos o como disparadoras (señales utilizadas en la evaluación de la condición) son mapeadas mediante un core insertado luego del proceso de síntesis por el ChipScope Pro Inserter de Xilinx.

2.3. Diagrama de Bloques del modelo Implementado

Como se puede observar en la fig. 2.4 el sistema implementado está constituido por 4 bloques principales: Transmisor, Receptor, Clocking y Settings. El bloque de settings se encarga de definir y proporcionar a los demás bloques las especificaciones básicas del sistema. El bloque de clocking por su parte se encarga de la sincronía del sistema mediante la distribución de los diferentes relojes al Transmisor y al Receptor. El Transmisor se encarga de formar y emitir los bloques conformados por los símbolos OFDM en el tiempo, a partir de una fuente de datos binaria de entrada. Finalmente el receptor se encarga de la recuperación de los datos transmitidos en los símbolos de datos del transmisor.

Los bloques de Settings y Clocking serán desarrollados a continuación porque ellos definen las características y parámetros del sistema. Así mismo, los bloques del Transmisor y del receptor serán ampliamente desarrollados en los capítulos III y IV respectivamente.

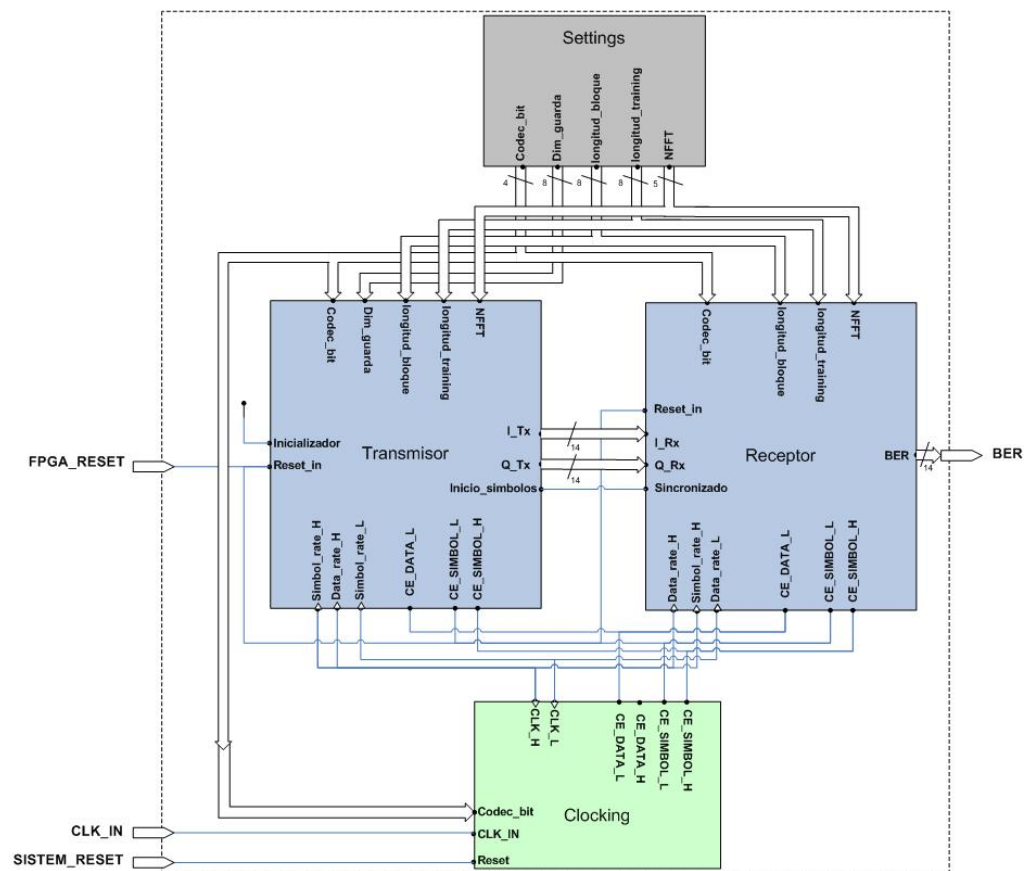


Fig. 2. 4 Diagrama de Bloques del Sistema

2.4. Parámetros del Sistema: Bloque “Settings”

El bloque “settings” se encarga de definir y proporcionar a los demás bloques las especificaciones básicas del sistema. La tabla II especifica

los puertos de salida del bloque e indica la definición de cada uno de sus parámetros. Los parámetros son especificados en VHDL, por esta razón son estáticos y para hacer algún tipo de cambio debe realizarse nuevamente el proceso de Síntesis e Implementación. Tener en cuenta que este bloque es completamente independiente del TX y el RX, por lo cual se facilitan futuras adecuaciones, estudios e implementaciones de dichos bloques.

Tabla II
Puertos del bloque Settings

Nombre	Longitud	Descripción
Dim_guarda	8	Número de puntos de la guarda a insertarse. Valores válidos entre 0 y NFFT.
NFFT	5	Señal que representa al número de puntos de la FFT. El valor de NFFT es igual $\log_2(\text{número de puntos})$. Valor valido: 8 Puerto para usos futuros.
Longitud_training	8	Numero de símbolos OFDM de training. Valores validos: entre 0 y Longitud_bloque.
Logintud_bloque	8	Numero de símbolos OFDM del bloque. Valores validos: entre 0 y 256.
Codec_bit	4	Número de bits codificados. Valores validos: 1, 2, 4, 6 y 8. 1 para BPSK, 2 para QPSK, 4 para 16-QAM, 6 para 64-QAM, 8 para 256-QAM.

2.5. Sincronización del Sistema: Bloque “Clocking”

El bloque de clocking se encarga de la sincronía del sistema mediante la distribución de los diferentes relojes al Transmisor y al Receptor. Debido a que el sistema implementado tiene el objetivo de realizar los esquemas de codificación BPSK, QPSK, 16-QAM, 64-QAM y 256-QAM,

es necesario desarrollar 5 diferentes dominios de frecuencias. Esto se debe a que la tasa de bits debe ser “Codec_bit” veces más rápida que la tasa de Símbolo.

Para obtener esto, la primera idea que se tiene en mente, es generar los 5 tipos de frecuencias y usar un sistema de multiplexación en función de “Codec_bit”. De esta manera, al transmisor y al receptor se enrutarían solo dos señales de reloj: la correspondiente a la tasa de Símbolo y la correspondiente a la tasa de bits.

No obstante, si bien es cierto que esta solución es simple, también acarrea serios problemas. El utilizar múltiples señales de reloj implica la necesidad de mantener una perfecta sincronía entre ellas, lo cual resulta complejo. Otra desventaja es que sería necesario tener diferentes entradas de reloj con las frecuencias deseadas o en su defecto se debería implementar diferentes DCM (Digital Clock Manager) en el FPGA utilizado. Sin embargo, la mayor desventaja quizás sea el hecho de que el rendimiento del sistema decaería notablemente. Esto se debe a que en un sistema multiplexado, el sintetizador del código VHDL interpretaría a las señales de reloj como señales combinatoriales. En este aspecto, es necesario tener en cuenta que los FPGAs en su interior tienen buses diferenciados para las señales de reloj y para las

señales combinatoriales. De esta forma, las señales de reloj son tratadas de manera especial para que puedan trabajar a las mayores velocidades posibles. Si el sintetizador interpreta la señal de reloj como combinatorial, esta propiedad se pierde y el rendimiento del sistema se ve afectado notablemente.

Por estas razones, la opción de utilizar diferentes señales de reloj ha sido descartada. En su defecto, el bloque de Clocking trabaja con una sola frecuencia de reloj, mientras que las diferentes frecuencias necesarias para la codificación y el entramado de los símbolos, son simuladas a través de la generación de señales habilitadoras.

No obstante, este método también tiene una desventaja. La única forma de implementar el método es estableciendo al reloj principal con una frecuencia mayor o igual a la frecuencia máxima a simular. De esta forma, el rendimiento del sistema estará limitado por todos los bloques del sistema a pesar de que estos no necesariamente necesiten trabajar a una frecuencia tan alta. Por ejemplo, si tomamos como referencia nuestro caso, solo la etapa de codificación requiere una frecuencia elevada mientras que el resto de bloques les basta operar con una fracción de dicha frecuencia. Es decir, en teoría no habría problema con tener una frecuencia limitante de 10MHz en cualquiera de los bloques si

se tuviese una frecuencia limitante de 80MHz en el codificador, ya que sería posible la implementación de todos los esquemas de codificación alcanzando una tasa de bits de hasta 80Mbps. A pesar de esto, al aplicar un solo reloj al sistema la frecuencia máxima estaría limitada a 10MHz por lo cual no se podría procesar una tasa de datos mayor a 10Mbps.

En la actual implementación, el bloque de Clocking produce 4 señales habilitadoras: 2 para la tasa de símbolo (CE_Symbol_H y CE_Symbol_L) y 2 para la tasa de datos (CE_Data_H, CE_Data_L). Teniendo en cuenta esto, de ahora en adelante cuando se indique que un bloque es síncrono con cierta señal de reloj especificada, se debe interpretar que este bloque es alimentado con el reloj del sistema (correspondiente a la lógica utilizada) y con una señal habilitadora correspondiente al reloj especificado. Por ejemplo, cuando nos refiramos a Symbol_Rate.H en realidad se esta haciendo referencia a la acción de la pareja CLK.H y CE_Symbol_H.

La fig. 2.5 muestra los subcomponentes del bloque de Clocking. La señal de reloj de entrada es conectada a un DCM con el objetivo de ser regenerada. El DCM es un procesador digital que analiza la información de la fase de cada ciclo del reloj de entrada con resultados

completamente predecibles. Una de las ventajas del DCM, es que cuando trabaja dentro de sus parámetros de operación no se ve afectado por los cambio de voltajes o temperatura [13]. En nuestra implementación, el objetivo de regenerar la señal de reloj es el de eliminar el “sesgo” (Conocido ampliamente como “Deskew” en ingles) en el reloj de entrada.

La salida del DCM produce dos señales de reloj que serán utilizada en el sistema: CLK0 y CLK180. La señal CLK0 corresponderá a la señal de reloj del sistema en lógica positiva (CLK.H), mientras que CLK180 sería la señal de reloj en lógica negativa (CLK.L).

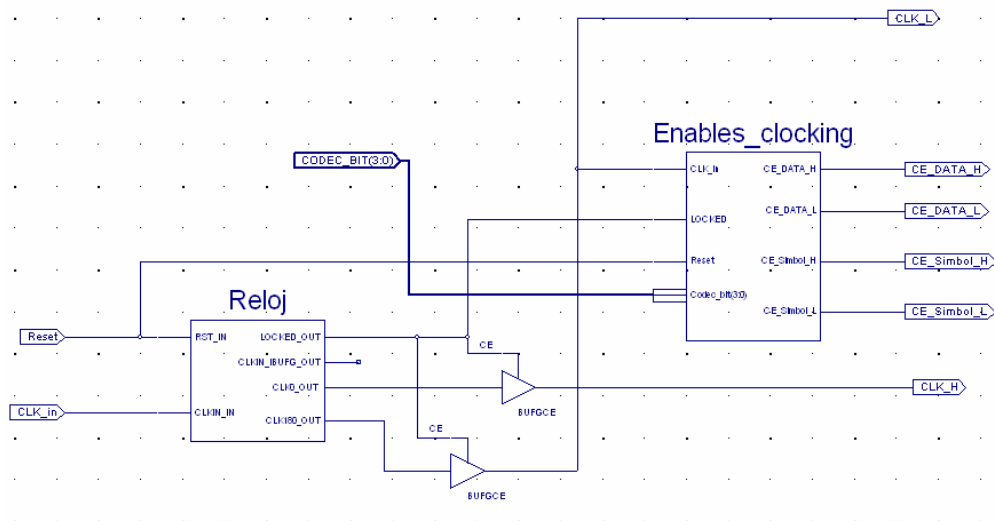


Fig. 2. 5 Esquemático de la etapa de Sincronización: Bloque de Clocking

La generación de las 4 señales habilitadoras esta a cargo del bloque “Enables_Clocking”, el cual es básicamente la máquina de estado mostrada en la fig. 2.6.

Esta máquina secuencial es síncrona con el CLK.L y consta de tres estados: El estado inicial E0, el Estado de la habilitación del símbolo E1 y el estado de conteo y habilitación de datos E2.

La máquina de estado permanecerá en el estado E0 hasta que la entrada locked sea habilitada. Esta señal indica cuando el DCM genera datos válidos en la salida de reloj. De igual manera, en cualquier momento que la señal de reset sea habilitada la maquina se trasladará al estado inicial E0. Si el reset no esta habilitado y locked es ‘1’, entonces la generación de los habilitadores debe de empezar, es decir debe de haber una transición al estado E1.

Luego de esto, el funcionamiento de la máquina oscilará entre los estados E1 y E2. En el estado E1 se mantendrá un flanco de reloj mientras que en el estado E2 se mantendrá $div - 1$ flancos de reloj. Es decir la secuencia de oscilación se repetirá cada div veces. Como la señal CE_Symbol_H solamente es habilitada en E1, la salida de la misma corresponderá a una onda cuadrada con un periodo de div y un

tiempo de encendido de un flanco de reloj. Por otra parte, la señal CE_Data_H también es habilitada en E1, mientras que en E2 se mantendrá habilitada por $\text{Codec_bit} - 1$ flancos más para luego tener un valor de '0'. De esta forma, la señal CE_Data_H corresponderá a una onda cuadrada con periodo div y tiempo de encendido de Codec_bit flancos de reloj.

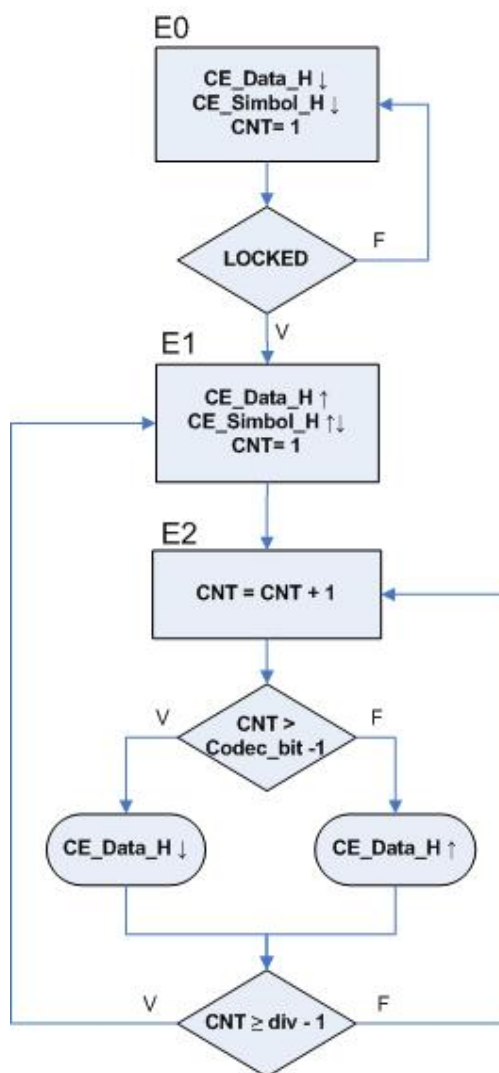


Fig. 2. 6 Diagrama de Estados de Enables_Clocking

Finalmente, los otros dos habilitadores (CE_Simbol_L y CE_Data_L) corresponden a las mismas señales (CE_Simbol_H, CE_Data_H) pero sincronizadas con CLK.H mediante el uso de dos registros. De esta manera, existe un retardo de medio flanco de reloj entre las dos parejas y las transiciones de sus relojes ocurrirán cuando la señal este completamente estable.

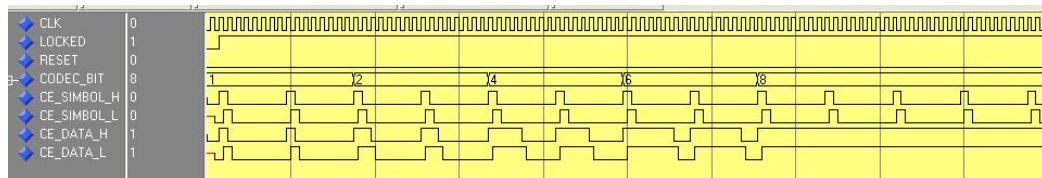


Fig. 2.7 Simulación en ModelSim del bloque Enables_clocking

Analizando los valores de Div y Codec_bit, podemos deducir que para que el sistema funcione correctamente debe darse la siguiente condición:

$$Div \geq \max(\text{Codec_bit}) \quad (2.1)$$

Por otra parte, las tasas de transferencias de datos y de símbolo estarían dadas por:

$$\text{Simbol_rate} = \frac{CLK}{Div} \quad (2.2)$$

$$\text{Data_rate} = \text{Codec_bit} \cdot \frac{CLK}{Div} \quad (2.3)$$

3. Modulador OFDM: Bloque “Transmisor”

El transmisor se encarga de formar y mostrar los bloques en el tiempo correspondientes a los símbolos OFDM resultantes. Como se indicó anteriormente, los bloques producidos por el transmisor son conformados por dos tipos de símbolos OFDM: símbolos de Training y símbolos de Datos. Los símbolos de Training se constituyen en la cabecera del bloque y son formadas por una cadena conocida de NFFT subportadoras C_k . El objetivo del training es el de proporcionar un medio para que la sincronización entre el Transmisor y el Receptor sea posible. Los símbolos de datos contienen la información de los datos codificados y modulados en OFDM. Estos símbolos están conformados por subportadoras C_k de Guarda, Pilotos y datos, formando en total NFFT componentes C_k tal como lo indica el estándar IEEE 802.16.

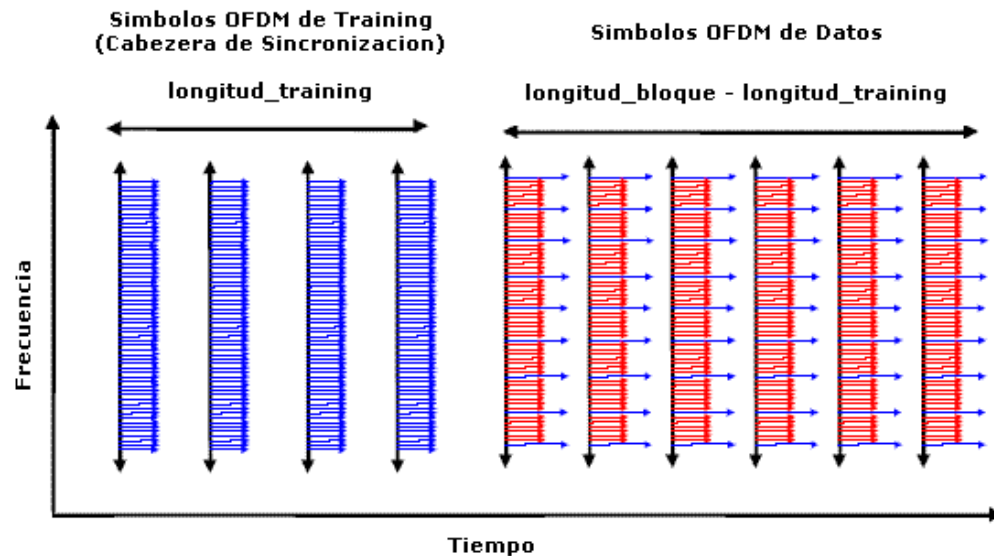


Fig. 3. 1 Estructura en tiempo y frecuencia del bloque OFDM transmitido.

La fig. 3.2 muestra el diagrama de bloques del transmisor. Los bits de entrada son producidos de forma serial por la fuente de datos. Estos bits son alimentados al bloque del codificador, el cual se encarga de producir las constelaciones nativas c_k de datos. En el mismo nivel podemos observar los bloques de training y “pilotos y guarda”, que como sus nombres lo indican están encargados de producir las constelaciones nativas c_k de training, pilotos y guarda respectivamente.

Para el entramado de los datos en el dominio de la frecuencia, el controlador del transmisor (Controlador Contador TX) hace uso del banco de selectores y de los tres bloques expuestos anteriormente. El control de dichos bloques lo realiza a través de las señales mux, dir_Ck y los habilitadores (En_pilotos,

En_datos). Antes de iniciar con el entramado, el controlador debe de encargarse de inicializar el bloque IFFT. Para esta finalidad hace uso de las señales esquema_FFT, Inicializador_FFT e Inicio_FFT.

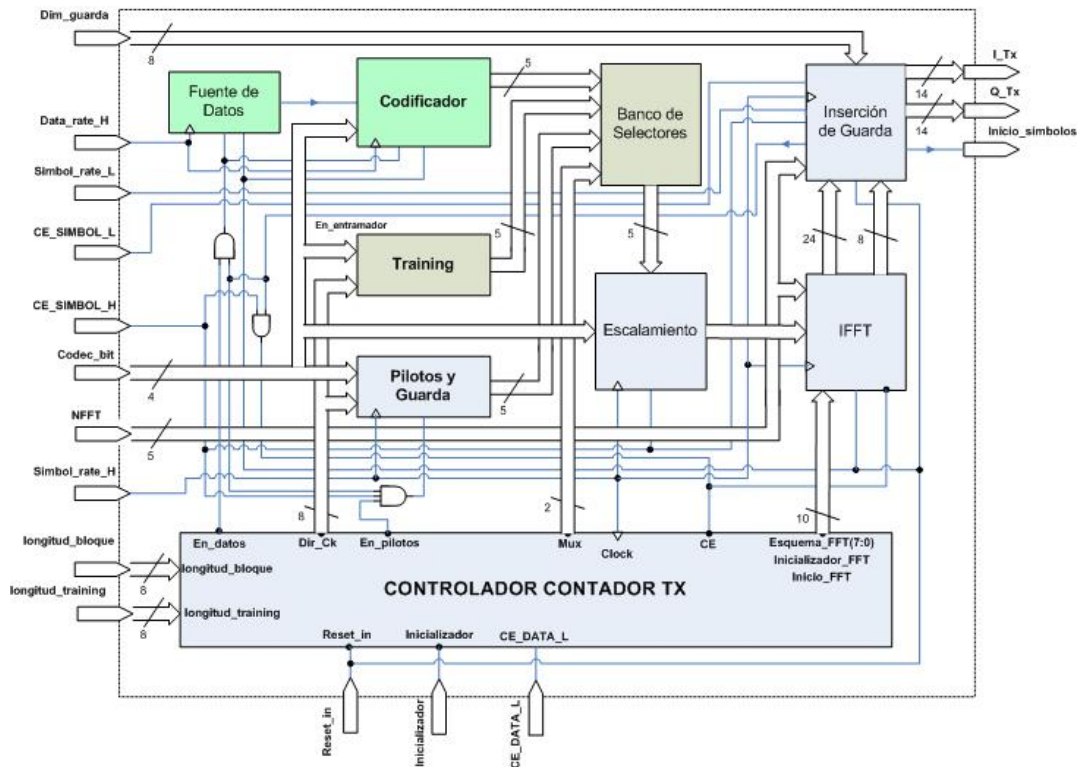


Fig. 3.2 Diagrama de bloques del Transmisor

Una vez sincronizado el bloque IFFT, comienza el entramado teniendo en cuenta la longitud del bloque y del training. El controlador tiene dos contadores internos encargados de contabilizar el índice k de la subportadora C_k (dir_Ck) y el número de símbolo producido en el bloque. Si el símbolo corresponde al training, el banco selector muestra en su salida las

componentes c_k del bloque de training. Por otra parte, si el controlador evalúa que el símbolo es de datos, este habilita la señal en_pilotos (solamente un flanco de reloj por símbolo de datos) y muestra a la salida del banco de selectores las señales correspondientes al bloque codificador o “Pilotos y Guarda” en función del contador interno del índice k . De esta forma en la entrada del bloque de escalamiento se tiene el símbolo OFDM en el dominio de la frecuencia, representado en forma serial por los diferentes componentes nativos c_k .

El bloque de escalamiento se encarga de multiplicar los valores en la entrada por un factor F en función del esquema de codificación, convirtiendo los componentes fundamentales c_k en los fasores C_k a transmitirse. De esta forma, la implementación del bloque multiplicador permite el trato independiente de los bloques Codificador, Training y “Pilotos y Guarda” con los bloques IFFT e Inserción de guarda. Gracias al escalamiento dichos bloques han podido ser diseñados de manera independiente y apegada al estándar, facilitando así su posible reutilización.

El bloque IFFT se encarga de transformar el símbolo OFDM del dominio de la frecuencia al dominio del tiempo. Es decir, transforma los NFFT fasores C_k en NFFT complejos $x[n]$. Finalmente se tiene el bloque de guarda que se encarga de la inserción del prefijo cíclico a partir de la misma secuencia

$\{x[n]\}$. De esta manera, los símbolos extendidos transmitidos serialmente en el tiempo son constituidos por $NFFT + dim_guarda$ valores $x[n]$. Debido a que el entramador (el resto de los bloques) y el bloque de inserción de guarda tienen la misma frecuencia de reloj, es necesario que el bloque de guarda detenga al entramador mientras la guarda es insertada. Esto es realizado mediante la señal `en_entramador` que se deshabilita durante la inserción del prefijo. De igual manera, es necesario indicarle al RX el inicio de cada símbolo útil para que este pueda suplir la ausencia del sincronizador. Esta función es realizada por el bloque de inserción de guarda mediante la señal `inicio_simbolo`. A continuación, se mostrará más detalladamente las especificaciones del funcionamiento de cada uno de los bloques del TX.

3.1. Fuente de Datos

El bloque de Fuente de datos es implementado mediante el uso del core LFSR en configuración Fibonacci. Mediante la utilización de este bloque es posible generar una secuencia binaria pseudoaleatoria. Los bits son generados de forma serial y son sincrónicos con el flanco positivo en la entrada del reloj [14]. La ventaja de realizar una secuencia pseudoaleatoria radica en el hecho de que se puede evaluar el sistema en condiciones prácticamente aleatorias (como sería en el caso real) y a la vez poder predecir lo que se debería obtener a la salida del receptor en condiciones de prueba. Si para efectos de prueba

se desea cambiar el polinomio característico, dicho cambio debe ser realizado antes del proceso de síntesis.

3.2. Training

Este bloque tiene la función de generar el símbolo de training en el dominio de la frecuencia. Los símbolos de Training se utilizan como cabecera de los bloques en el tiempo y tienen el objetivo de proporcionar un medio para que la sincronización entre el Transmisor y el Receptor sea posible. Como se indico anteriormente los valores generados por este bloque tienen correspondencia con el estándar IEEE 802.16. De forma más específica, el símbolo producido por este bloque es directamente proporcional al preámbulo completo mostrado en la fig. 1.9.

El bloque recibe como entrada la señal dir_Ck , y de forma combinatorial produce la salida del Training concerniente a dicha dirección. La señal dir_Ck corresponde al índice de la transformada discreta de fourier y puede ser definida en función del índice k como se indica a continuación:

$$dir_ck(k) \begin{cases} 256 + k; & k < 0 \\ k & ; k \geq 0 \end{cases} \quad (3.1)$$

El resultado del bloque esta expresado en función de la codificación utilizada y corresponde al preámbulo de la fig. 1.9 multiplicado por el valor expresado por la Tabla III. El valor max es definido en función del tipo de codificación y corresponde al máximo valor de las constelaciones definidas para datos en el modelo propuesto.

Tabla III
Valores máximos en las constelaciones gray.

Codificación	Max	Max_negado
BPSK	1	-1
QPSK	1	-1
16-QAM	3	-3
64-QAM	7	-7
256-QAM	15	-15

Esta multiplicación es necesaria ya que se desea que a la entrada de la IFFT siempre se tenga un mismo valor máximo para las constelaciones a modular. Como se verá más adelante, el bloque multiplicador se encarga de esto mediante la transformación de las constelaciones c_k en los favores C_k a través de la multiplicación de un factor en función del tipo de codificación. Como el bloque multiplicador es implementado luego del multiplexor su efecto se reflejará también en las constelaciones primitivas del Training y de los pilotos. Es decir, tanto en el caso del training como en los pilotos es necesario tener en cuenta el tipo de codificación para que el efecto de multiplicación

selectiva (en función de la codificación) sea cancelado y a la salida del multiplicador se tenga un único valor definido.

El bloque de Training, al igual que todos los multiplicadores utilizados en este proyecto, es básicamente una LUT implementada en VHDL. La utilización de este esquema combinatorial a diferencia de un sistema multiplicador secuencial se debe a su bajo retardo y mayor eficiencia en la utilización de recursos. Además, como se tiene un conjunto limitado de entradas y se conoce el factor de la multiplicación, su implementación es relativamente fácil.

3.3. Codificador

Este módulo tiene la función de codificar una cadena serial de bits, en un conjunto de valores complejos (Componentes c_k) correspondientes al tipo de codificación. El bloque tiene la capacidad de realizar codificación BPSK, QPSK, 16-QAM, 64-QAM y 256-QAM. La salida producirá una constelación c_k válida luego de que sean ingresados "Codec_bit" bits en la entrada.

La fig. 3.3 muestra el diagrama de bloques del Codificador. Los bloques verdes corresponden a los componentes sincrónicos con el reloj negado, los celestes a los sincrónicos con el reloj de lógica

positiva, mientras que los bloques grises corresponden a los componentes asincrónicos

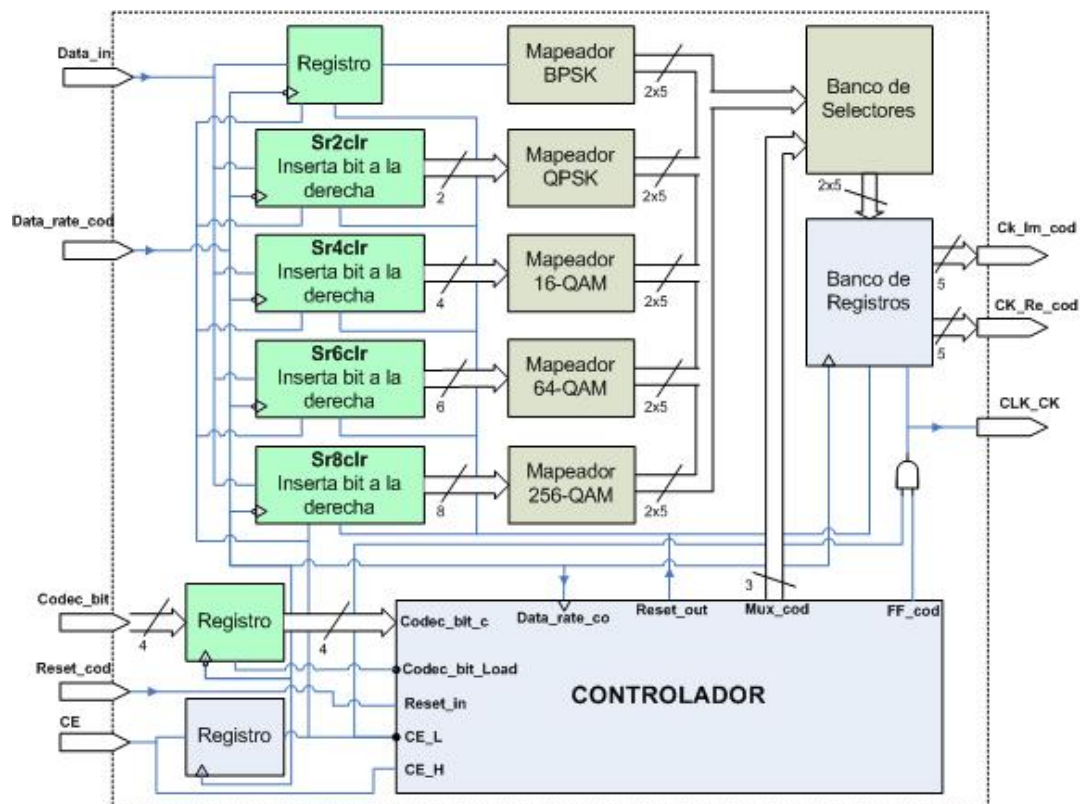


Fig. 3.3 Diagrama de Bloques del Codificador

La máquina secuencial del sistema consta básicamente de tres estados: el estado de reseteo E0, el estado de procesamiento E1 y el estado de muestreo E2. Los estados E0 y E2 son similares ya que en ellos la máquina se prepara para el procesamiento del siguiente símbolo. Por esta razón en ambos estados se muestrea el tipo de codificación (*Codec_bit*) mediante el uso de la señal habilitadora

Codec_bit_Load. De esta forma el bloque tiene la posibilidad de cambiar dinámicamente la codificación entre el procesamiento de un símbolo y otro sin que esto afecte al sistema. En este proyecto no se realiza un cambio dinámico de la codificación, sin embargo esta propiedad permitiría que el bloque sea reutilizado en futuros proyectos.

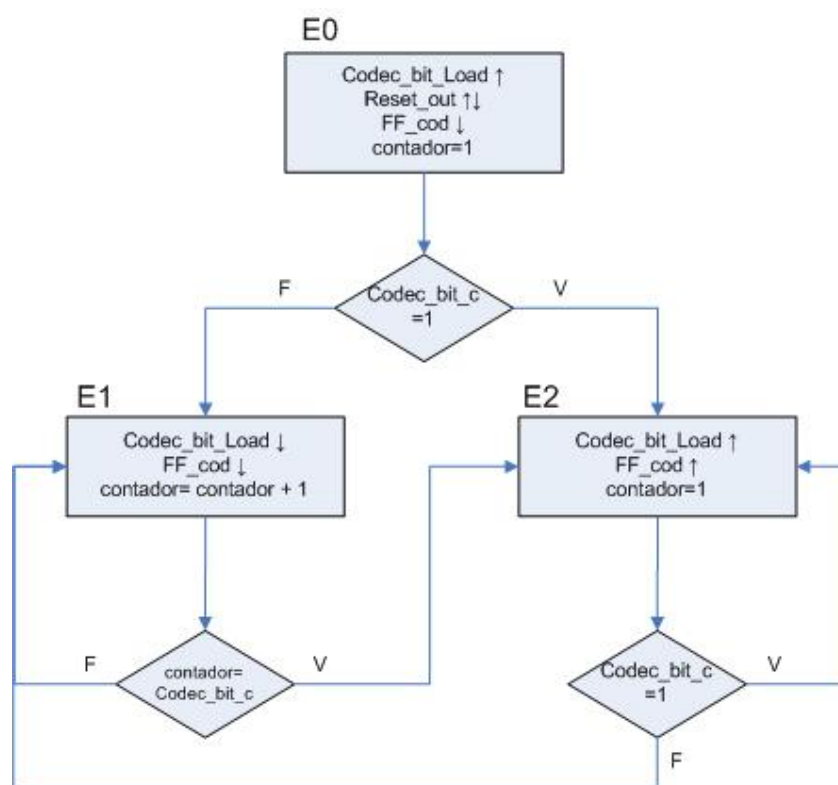


Fig. 3. 4 Diagrama ASM del Controlador del Codificador

La diferencia básica entre el estado E1 y E0 radica en que en E0 se resetea el Registro de salida, mientras que en E1 este es habilitado mediante el uso de la señal FF_cod. De esta forma la salida del

bloque en E0 es cero, mientras que en E1 corresponde al complejo c_k codificado.

Por otra parte, el estado E1 corresponde a la fase de procesamiento del sistema y funciona mediante la activación de un contador interno. En este estado el registro de salida no es habilitado y mantiene el anterior componente c_k muestreado. Una vez terminado el procesamiento el nuevo componente c_k debe ser muestreado, es decir se debe de pasar al estado E2. Para que esto suceda el contador debe tener un valor igual a `Codec_bit`. De esta forma, al realizarse la transición a E2 los registros de desplazamiento habrán realizado justamente `Codec_bit` inserciones a la derecha. Debido a esto, a la entrada del bloque mapeador correspondiente a la codificación utilizada, se encontrará la versión paralela de los “`Codec_bit`” bits ingresados seriamente. Como los bloques mapeadores construyen de forma combinatorial las constelaciones, una de las entradas del banco de selectores corresponde al nuevo componente c_k que debe ser muestreado. Finalmente, el símbolo codificado debe ser enrutado a la entrada del registro de salida para su posterior muestreo. Esto se consigue mediante el uso de la señal `Mux_cod` que es función del tipo de codificación utilizado. En este aspecto es importante anotar que debido a que el tipo de codificación

para el componente c_{k+1} es muestreado justamente cuando el valor de c_k debe ser mostrado a la salida, es necesario que la señal Mux_cod no varíe al mismo tiempo que Codec_bit_c sino a partir del siguiente flanco positivo.

De esta forma, gracias a los registros de desplazamientos y la topología mostrada en la Fig. 3.3 es posible muestrear el símbolo c_k a la salida del registro. Es importante tener en cuenta que en el caso de BPSK no es necesario crear una versión paralela de la entrada ya que esta es de un bit. En este caso el estado de procesamiento tiene que ser obviado y se debe mantener el estado E1. Por esta razón tanto en E1 como en E0 debe de preguntarse primeramente si Codec_bit corresponde a BPSK para decidir el estado siguiente.

3.4. Pilotos y Guarda

Este bloque se encarga de la inserción de los componentes c_k de pilotos y guarda (subportadoras nulas) en el dominio de la frecuencia de los símbolos OFDM. El comportamiento del mismo se basa en la estructura de los símbolos OFDM expuesta en la sección 1.3. En el caso de los pilotos, las especificaciones implementadas en este bloque han sido tomadas a partir de lo indicado en la sección 1.3.4.2 para la trama de DL.

El polinomio de la secuencia ω_k es implementado mediante el mismo core de Xilinx utilizado para generar la secuencia pseudoaleatoria de los datos de entrada. La secuencia ω_k es disparada gracias a la señal en_pilotos , la cual debe ser habilitada solo una vez al principio de cada símbolo de datos. Esto se justifica debido al hecho de que ω_k debe cambiar en cada símbolo OFDM de datos pero debe mantenerse constante en el transcurso del mismo.

Teniendo en cuenta lo expuesto, de (1.16) se concluye que:

$$\begin{aligned}
 \text{Si } \omega_k = 0 & & C_{-88} = C_{-38} = C_{63} = C_{88} = 1 \\
 & & C_{-63} = C_{-13} = C_{13} = C_{-38} = -1 \\
 \text{Si } \omega_k = 1 & & C_{-88} = C_{-38} = C_{63} = C_{88} = -1 \\
 & & C_{-63} = C_{-13} = C_{13} = C_{-38} = 1 \quad (3.2)
 \end{aligned}$$

De aquí se infiere que la componente imaginaria es siempre cero. Por otro lado la componente real será función del índice k, es decir de la señal dir_ck como se demuestra a continuación:

$$\text{Si } \omega_k = 0 \Rightarrow \text{Si } \text{dir_ck} = 13 \text{ ó } 63 \text{ ó } 218 \text{ ó } 243 \Rightarrow \text{Out_Re} = 1$$

$$\text{Si } \text{dir_ck} = 38 \text{ ó } 88 \text{ ó } 168 \text{ ó } 199 \Rightarrow \text{Out_Re} = -1$$

$$\text{Si } \omega_k = 1 \Rightarrow \text{Si } \text{dir_ck} = 13 \text{ ó } 63 \text{ ó } 218 \text{ ó } 243 \Rightarrow \text{Out_Re} = -1$$

$$\text{Si } \text{dir_ck} = 38 \text{ ó } 88 \text{ ó } 168 \text{ ó } 199 \Rightarrow \text{Out_Re} = 1 \quad (3.3)$$

Por otro lado si consideramos los símbolos en el estado de guarda tenemos que:

$$\begin{aligned} C_{-127} &= \dots = C_{-101} = 0 \\ C_0 &= 0 \\ C_{101} &= \dots = C_{127} = 0 \end{aligned} \quad (3.4)$$

Es decir, traducido en términos de la señal dir_ck tendríamos lo siguiente:

$$\text{Si } \text{dir_ck} = 0 \vee 129 \leq \text{dir_ck} \leq 155 \Rightarrow \text{Out_Im} = \text{Out_Re} = 0 \quad (3.5)$$

En definitiva, la salida del bloque debe generar los valores especificados en (3.3) y (3.5) multiplicados por el valor máximo de cada constelación (tabla III).

En realidad el bloque por definición no tiene especificación alguna para el resto de valores dir_ck ; sin embargo, para que la transición de

este bloque es el de maximizar la eficiencia en el uso de los 14 bits de resolución a la salida del TX. El uso de este bloque permite mantener la modularidad e independencia de la IFFT e inserción de guarda con el resto de los bloques. Gracias a esto, los bloques del codificador, training y “Pilotos y Guarda” han podido ser diseñados de manera independiente y apegada al estándar, facilitando así su posible reutilización.

La dimensión de 14 bits, ha sido escogida debido a que corresponde a la resolución de los DAC's y ADC's proveídos en el kit Xtreme DSP Virtex II Pro. Es importante anotar que a pesar de que el bloque se ajusta a 14 bits, la salida del mismo es de 16 bits, es decir tiene dos bits de relleno. Esto se debe a que las dimensiones más cercanas a 14 bits en las entradas del bloque IFFT, son de 16 y 12 bits [15]. La dimensión de 12 bits queda descartada ya que implicaría una pérdida de dos bits de resolución en el procesamiento de los datos o en la excursión de la salida.

Debido a que el factor multiplicativo determina el porcentaje de excursión de la señal de salida, la correcta elección de dicho factor es de vital importancia para el desempeño del sistema. Si tomamos en cuenta que lo que se desea es maximizar la magnitud de las

componentes $x[n]$ real e imaginaria (para maximizar la excursión de los 14 bits de salida), el camino a seguir para nuestro análisis sería el determinar la máxima respuesta de la IFFT ante la presencia de la secuencia primitiva c_k . Así, gracias a la linealidad de la transformada de Fourier y tomando en cuenta la limitante de 14 bits, es posible definir el valor idóneo para la variable multiplicativa F .

Como analizamos en el capítulo I, una manera de determinar la magnitud máxima de la IFFT es ingresando NFFT componentes c_k de la misma fase y de esta forma observar la salida $x[0]$. Sin embargo, esto no es posible ya que el valor de los pilotos dependen de una secuencia polinomial y no pueden tener la misma fase; además existen algunos componentes nulos.

A pesar de esto, se puede obtener el máximo valor de la IFFT, si asumimos que los N_{used} componentes c_k coinciden en fase entre sí

justamente en el instante $t = n \frac{1}{f_s}$, donde la respuesta de los pilotos en

el tiempo tengan el valor máximo. Otra condición para que esto se de, sería que el valor máximo de los pilotos este en fase con el valor

$x(t = n \frac{1}{f_s})$ resultante en la IFFT de los N_{used} fasores

c_k correspondiente a datos. Si tomamos en cuenta además que las componentes c_k de la guarda tienen valor cero, por linealidad podríamos expresar lo siguiente:

$$\max(x_n) = \underbrace{\max(x_{n1})}_{\substack{\text{Re spuesta máxima} \\ \text{en el tiempo de los datos}}} + \underbrace{\max(x_{n2})}_{\substack{\text{Re spuesta máxima} \\ \text{en el tiempo de los pilotos}}} \quad (3.7)$$

De esta forma el máximo valor de la secuencia x_{n1} se puede determinar con la metodología explicada en el capítulo I, mientras que el $\max(x_{n2})$ es fácilmente extraíble a través de las simulaciones en Matlab ya que existen solo dos casos dependiendo del valor de ω_k .

De aquí tenemos que para el caso de BPSK el máximo valor de la secuencia x_{n2} es 0.015625. Por otra parte, el máximo valor de x_{n1} estará determinado por el valor X_0 de la secuencia c_k cargada con "1" en todas las Nused posiciones k. De aquí se puede obtener que el valor $x[0]$ corresponde a:

$$\begin{aligned}
 \max(x_{n1}) &= x[0] \\
 &= \frac{1}{N_{FFT}} \sum_{k=0}^{N_{used}} c_k \\
 &= \frac{N_{used}}{N_{FFT}} \\
 &= 0.75
 \end{aligned}$$

Reemplazando estos resultados en 3.7, tenemos que:

$$\max(x_n) = 0.7 + 0.015625 = 0.765625$$

Razón por la cual, si tomamos en cuenta el efecto del factor F, el máximo valor X_n a la salida del bloque para el caso de BPSK estaría dado por:

$$\max(X_n) = 0.765625 F \quad (3.8)$$

Si tomamos en cuenta la premisa que la magnitud de la salida $X[n]$ debe estar limitada a 13 bits, finalmente tendremos la siguiente condición para F:

$$0.765625 * F < 2^{13} - 1 = 8191 \quad (3.9)$$

Otro aspecto que debe ser tomado en cuenta, es el hecho de que se desea que la salida en el dominio del tiempo tenga un mismo valor máximo para todos los casos de codificación. Esto, con el objetivo de facilitar la implementación en la etapa RF. Teniendo en cuenta esto, si observamos el hecho que la máxima entrada c_k para el caso de

QPSK, 16-QAM, 64-QAM y 256-QAM es 1, 3, 5, 7 y 15 respectivamente, finalmente tendremos:

$$\begin{aligned}\max[x[n]_{BPSK}] &= \max(x[n]_{QPSK}) \\ &= \frac{1}{3} \max(x[n]_{16QAM}) \\ &= \frac{1}{7} \max(x[n]_{64QAM}) \\ &= \frac{1}{15} \max(x[n]_{256QAM})\end{aligned}$$

Por otra parte, si expresamos la respuesta máxima del sistema para cada caso de codificación y tomamos en cuenta que esta debe ser siempre la misma, podríamos escribir las siguientes ecuaciones:

$$\begin{aligned}\max(X[n]_{BPSK}) &= f_{BPSK} * \max(x[n]_{BPSK}) = \\ \max(X[n]_{QPSK}) &= f_{QPSK} * \max(x[n]_{QPSK}) = \\ \max(X[n]_{16QAM}) &= f_{16QAM} * \max(x[n]_{16QAM}) = \\ \max(X[n]_{64QAM}) &= f_{64QAM} * \max(x[n]_{64QAM}) = \\ \max(X[n]_{256QAM}) &= f_{256QAM} * \max(x[n]_{256QAM}) =\end{aligned}$$

De todo esto finalmente se concluye lo siguiente:

$$f_{BPSK} = f_{QPSK} = 3f_{16QAM} = 7 * f_{64QAM} = 15f_{256QAM} \quad (3.10)$$

Por otra parte se desea que el resultado de la multiplicación que se carga a la entrada de la IFFT sea un entero, ya que de esta manera no se pierde información en el proceso de multiplicación. Así, si definimos a F como el factor fundamental y equivalente a f_{BPSK} se concluye que dicho valor ser múltiplo de 3, 7 y 15. Entonces, la tarea se reduce a escoger un factor F que cumpla esta condición y la expuesta en (3.9).

Finalmente, en la tabla IV podremos observar los valores de salida y los respectivos factores para todas las entradas y los tipos de codificación posibles. Estos valores corresponden a un factor F de 10500, para el cual se tiene una excursión máxima del 98.14% para todos los casos.

Tabla IV
Bloque de Escalamiento

Codificación	Ck entrada	factor	Ck Salida
PSK / QPSK	1	10500	10500
	-1	10500	-10500
16-QAM	3	3500	10500
	1	3500	3500
	-1	3500	-3500
	-3	3500	-10500
64-QAM	7	1500	10500
	5	1500	7500
	3	1500	4500
	1	1500	1500
	-1	1500	-1500
	-3	1500	-4500
	-5	1500	-7500
	-7	1500	-10500
256-QAM	15	700	10500

	13	700	9100
	11	700	7700
	9	700	6300
	7	700	4900
	5	700	3500
	3	700	2100
	1	700	700
	-1	700	-700
	-3	700	-2100
	-5	700	-3500
	-7	700	-4900
	-9	700	-6300
	-11	700	-7700
	-13	700	-9100
	-15	700	-10500

3.6. Bloque IFFT

El bloque IFFT tiene la función de transformar los símbolos OFDM del dominio de la frecuencia al dominio del tiempo. El bloque recibe de manera serial y ascendente los NFFT fasores C_k correspondientes al símbolo OFDM en frecuencia y luego del tiempo de procesamiento, muestra también de forma serial y ascendente la función inversa de la Transformada rápida de Fourier.

El bloque fue implementado mediante el uso del core “Logic Core Fast Fourier Transform v2.1” de Xilinx. En [15] se muestra la documentación detallada de este core. La tabla V muestra el detalle de los puertos principales del core utilizado. Para el caso del bloque

IFFT el valor de FWD_INV es '1'. Además los puertos NFFT_WE, FWD_INV_WE y SCALE_SCH_WE están cortocircuitados entre sí y corresponden a la señal Inicializador_FFT del bloque “Controlador Contador TX”

Tabla V
Puertos Principales Del Core FFT utilizado

Nombre	Longitud	Tipo	Descripcion
XN_RE[15:0]	16	Entrada	Bus de Entrada – componente real
XN_IM[15:0]	16	Entrada	Bus de Entrada – componente imaginario
START	1	Entrada	Señal Inicio FFT (Activo en alto) - Para flujo de datos continuo, START indicara el inicio de la carga de datos, luego de la misma, el procesamiento y la descarga serán automáticas
NFFT	4	Entrada	El número de puntos de la transformada- El valor de NFFT es igual \log_2 (número de puntos). El bloque tiene la capacidad de computar valores de 256,128 y 64 puntos.
NFFT_WE	1	Entrada	Enable de escritura para NFFT (Activo en alto)
FWD_INV	1	Entrada	Señal de control que indica si FFT o IFFT es ejecutado- Cuando FWD_INV=1, la transformada directa es realizada. Si FWD_INV=0, se realiza la IFFT
FWD_INV_WE	1	Entrada	Enable de escritura para FWD_INV (Activo en alto)
SCALE_SCH	8	Entrada	Esquema de escala: “00000000” Para realizar la FFT “10101010” Para realizar la IFFT
SCALE_SCH_WE	1	Entrada	Enable de escritura para SCALE_SCH (Activo en alto)
SCLR	1	Entrada	Reset Maestro (Activo en alto)
CE	1	Entrada	Enable del Reloj (Activo en alto)

CLK	1	Entrada	Reloj
XK_RE[15:0]	16	Salida	Bus de Salida – componente real.
XK_IM[15:0]	16	Salida	Bus de Salida – componente imaginario.
XN_INDEX	8	Salida	Indice de los datos de entrada
XK_INDEX	8	Salida	Indice de los datos de salida

3.7. Controlador Contador Tx

Este bloque constituye el cerebro de todo el sistema y tiene la función de controlar la formación del símbolo útil OFDM. Esta labor básicamente se divide en dos: inicializar la IFFT y controlar la formación del símbolo útil en el dominio de la frecuencia. Como se muestra en la tabla VI, para el control de la IFFT el controlador utiliza las señales esquema_FFT, Inicializador_FFT e Inicio_FFT. Por otra parte, la formación del símbolo OFDM en el dominio de la frecuencia es controlada a través de las señales mux, dir_Ck y de los habilitadores En_pilotos y En_datos.

Tabla VI
Puertos de Salida del Controlador Contador Tx

Nombre	Longitud	Descripción
Mux	2	Señal selectora del banco de selectores.
Dir_Ck	8	Muestra la dirección del valor C_k que se está entramando. Valores válidos entre 0 y 255.
Inicializador_FFT	8	Señal inicializadora del bloque FFT. Le indica al core que grave los valores del NFFT, la escala y el tipo de transformada (directa o inversa).

Esquema_FFT	8	Le indica al core FFT cual debe de ser el esquema de escalamiento que debe utilizar.
Inicio_FFT	1	Le indica al core cuando debe de iniciar su funcionamiento. El core debe de ser previamente inicializado.
En_datos	1	Señal habilitadora de la fuente de datos.
En_pilotos	1	Habilitador de las secuencia ω_k para la generación de los pilotos.

Para la generación de estas señales controladoras, el bloque contiene una máquina secuencial (fig. 3.5) y un decodificador interno. La máquina secuencial consta de 5 estados y de dos contadores internos encargados de contabilizar el índice k de la subportadora C_k (dir_Ck) y el número de símbolo producido en el bloque. El estado E0 corresponde al estado de reseteo del transmisor, mientras que los estados E1 y E2 son utilizados para la inicialización de la IFFT y los estados E3 y E4 para la formación del símbolo útil OFDM en el dominio de la frecuencia. El controlador del sistema permanecerá en el estado E0 hasta que observe un '1' en la señal de entrada Inicializador. En nuestra implementación esta señal es cargada con +VCC, por esta razón su existencia no refleja inherencia alguna sobre nuestro sistema y fue implementada para aplicaciones futuras. De igual manera, como en todos los controladores implementados, si la

señal de reset es seleccionada la máquina secuencial iría automáticamente al estado inicial, sin importar el estado actual.

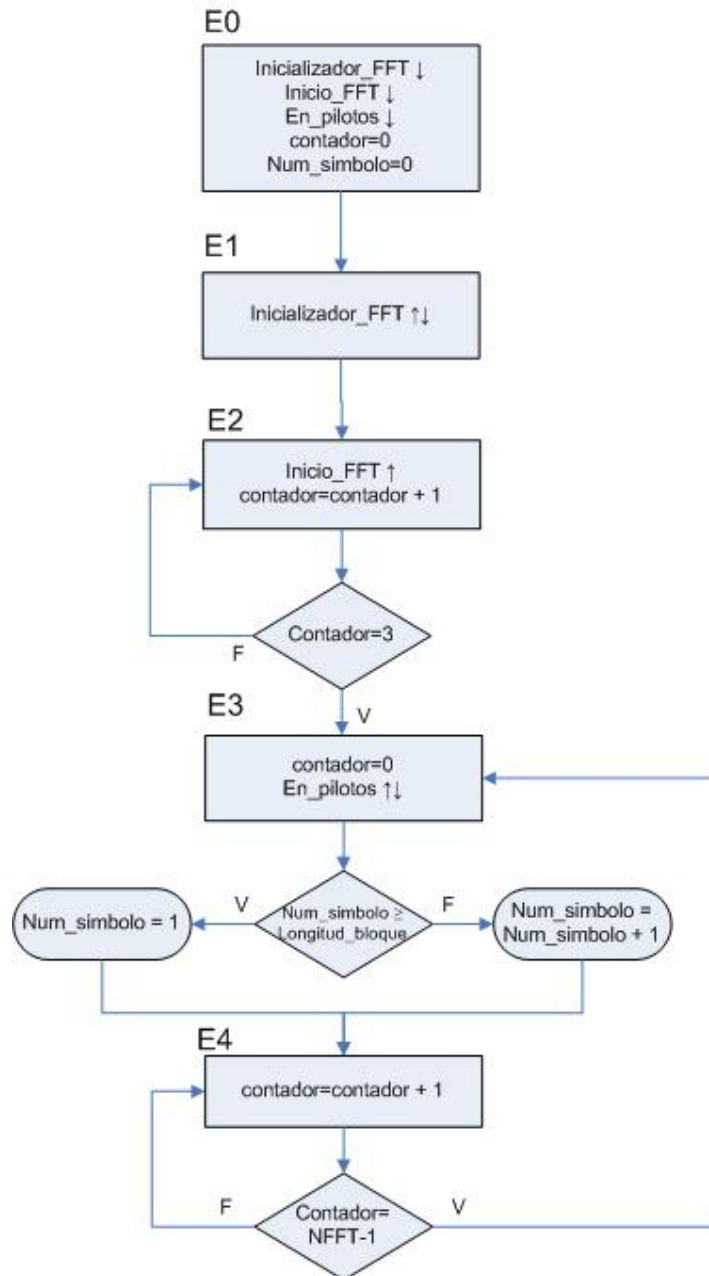


Fig. 3.5 Máquina de estados del Controlador Contador TX

Para el control del bloque IFFT el decodificador interno especifica los valores de las señales esquema_FFT, Inicializador_FFT e Inicio_FFT. Como se indicó en la tabla V, para la implementación de la IFFT el esquema de escala (puerto SCALE_SCH) debe de ser cargado con la cadena "10101010". Por esta razón, el controlador carga esta cadena de forma continua a través de la salida esquema_FFT. Es decir, esta señal es un parámetro definido en código VHDL y no es función de variable alguna.

Por otra parte, como se indica en la sección 3.6 y en la tabla VI, la señal Inicializador_FFT se encarga de indicar al core FFT que cargue los valores de NFFT, escala y tipo de transformada. El proceso inicialización del core es realizado en el estado E1, por esta razón la salida Inicializador_FFT es cargada con '1' solamente este estado.

Una vez que el core FFT es inicializado, este queda a la espera de que el puerto START sea cargado con '1' para continuar con el procesamiento de los datos. Debido a esto, la señal Inicio_FFT es cargada en el puerto START del core y tendrá un valor '1' en todos los estados excepto en E0 y E1 para un procesamiento continuo de datos. Por otra parte, debido a que el bloque necesita una transición de 4 flancos de reloj para procesar los datos de entrada [15], la máquina

secuencial debe de esperar en el estado E2 hasta que la señal contador tenga el valor de 3. Este valor es definido porque existe un flanco de reloj de retraso en la entrada de la IFFT por el registro del bloque de escalamiento, de esta manera se tienen en total los 4 flancos de retraso exigidos por el core.

Una vez que el modulo IFFT este inicializado y listo para procesar datos, el controlador procederá a crear indefinidamente los bloques OFDM en los estados E3 y E4. Esto lo logra gracias al uso de las señales contador y Num_simbolo que son contadores internos de la máquina de estado (fig. 3.5). En estos estados de procesamiento, el sistema formará la secuencia $\{0, 1 \dots N_{FFT} - 1\}$ para la señal contador y $\{1, 2 \dots longitud_bloque\}$ para la señal Num_simbolo. Además, la secuencia Num_simbolo variará cada vez que culmine un periodo de la secuencia de contador. De esta forma, el contador representa a la dirección k de las subportadoras y Num_simbolo a la posición del símbolo OFDM formado en el bloque.

En base al comportamiento de la máquina de estado el decodificador interno debe de mostrar las salidas Mux, dir_ck, en_datos y en_pilotos para la formación de los símbolos en el dominio de la frecuencia. Ante la explicación anterior, queda claro el hecho de que la salida dir_ck

corresponderá al valor de la señal contador. Además, debido a que la secuencia de pilotos depende de un valor pseudoaleatorio en cada símbolo de datos, la señal en_pilotos corresponde a un pulso en el estado E3, es decir al iniciar cada símbolo.

Por otra parte, las señales Mux y en_datos dependerán de los valores de las señales contador (dir_Ck) y Num_simbolo, tomando en cuenta las especificaciones de la tabla I y la tabla VII. Es importante anotar que la señal en_datos debe durar Codec_bit flancos de reloj para que se procesen justamente Codec_bit bits en cada símbolo. Este comportamiento se logra mediante una operación AND entre la señal interna en_datos_int y una versión de CE_Data_L retrazada medio flanco de reloj. Este retraso es necesario ya que en_datos_int es síncrono con el flanco negativo de reloj, es decir un flanco después de CE_Data_H y medio flanco después de CE_Data_L.

Tabla VII
Datos a cargarse en la IFFT según el valor de mux

Mux	Datos a cargarse
"00"	Pilotos o guarda provenientes del bloque "Pilotos y Guarda"
"01"	Símbolo de training proveniente del bloque "Training"
"10"	Datos codificados provenientes del bloque "Codificador"

De esta manera, si $Num_simbolo < longitud_training$, Mux debe tener un valor de "01" y en_datos_int debe ser '0' para todo caso excepto en el estado E1. Esto se debe a que E1 solamente dura un flanco de reloj y es necesario cargar al codificador los primeros Codec_bit bits, para que en E3 la salida del codificador no muestre basura sino el primer valor complejo c_k válido. Por otra parte, si $Num_simbolo \geq longitud_training$ el valor mux y en_datos_int dependerá del índice k de las portadoras C_k , es decir de la señal contador. Cuando contador corresponda a un valor k definido para datos, en_FFT debe ser '1' y mux debe ser "11"; mientras que si contador corresponde a valores k de pilotos o guardas mux debe ser "00" y en_datos_int='0'.

3.8. Inserción de Guarda

El bloque "Inserción de guarda" tiene la función de crear el símbolo OFDM extendido mediante la inserción del prefijo cíclico. Como se indicó en el capítulo I, el objetivo de este método es el de esperar un tiempo, para que los componentes Multitrayectoria de la señal deseada sean eliminados. El bloque toma la secuencia de números complejos $\{X[n]\}$ de cada símbolo OFDM en el tiempo e inserta "dim_guarda" componentes al principio. Estos componentes

corresponden a una copia de los últimos “dim_guarda” valores de dicha secuencia (fig.1.5).

Para lograr este funcionamiento, el bloque “Inserción de guarda” hace uso de la estructura mostrada en la fig. 3.6. Los bloques celestes corresponden a los sincrónicos con el flanco positivo, mientras que el verde es síncrono con el reloj negado y el café es combinatorial.

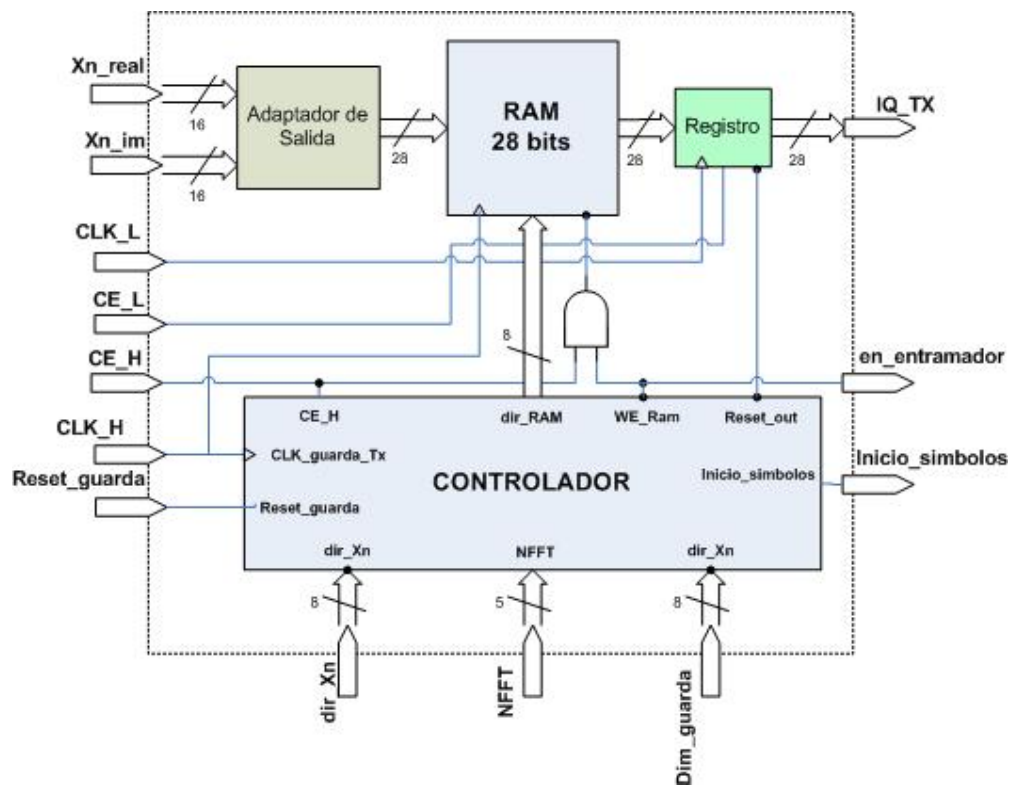


Fig. 3. 6 Diagramas de bloques del modulo Inserción de guarda

Los datos de entrada pasan primeramente al bloque “Adaptador de Salida” en donde los bits de relleno insertados en el escalamiento son retirados. Como su nombre lo indica, el objetivo de este bloque es adaptar la salida al uso del hardware escogido para las pruebas. Una vez adaptados, los complejos correspondientes a la secuencia $\{X[n]\}$ son capturados en una memoria RAM de 28bits y 256 líneas. Esta RAM es implementada mediante el core de Xilinx Distributed Memory v7.1 [16], y permite que el símbolo útil OFDM sea capturado a partir de la IFFT. Para evitar espurias en las entradas de la RAM, en dicho core se ha configurado la utilización de registros internos que capturan la entrada, la dirección y la señal WE cada flanco positivo de reloj. Una vez almacenados, los datos de salida son muestreados desde la RAM gracias el uso del Registro de Salida y mediante la manipulación de la señal `dir_ram`. En el proceso de muestreo, primeramente son mostrados los últimos “`dim_guarda`” datos y luego de forma ascendente el símbolo original.

Para obtener este funcionamiento, el Controlador consta de 3 Estados: el estado de solo captura (E0), el estado de solo muestreo (E1) y el estado de captura y muestreo (E2). En el estado inicial de solo captura, el bloque comienza a procesar por primera vez, por lo cual tiene como objetivo el capturar el primer símbolo en la RAM. Para

esto, la dirección de la RAM (dir_Ram) es cargada con el índice n de la secuencia $\{X[n]\}$ mientras que el habilitador WE_RAM esta siempre esta en '1' porque la RAM esta en continua captura. En todo este proceso, la salida del bloque es cero ya que el registro de salida es reseteado. Una vez que el último valor de la secuencia del primer símbolo sea capturado ($dir_Xn=255$) el controlador debe de cambiar de estado.

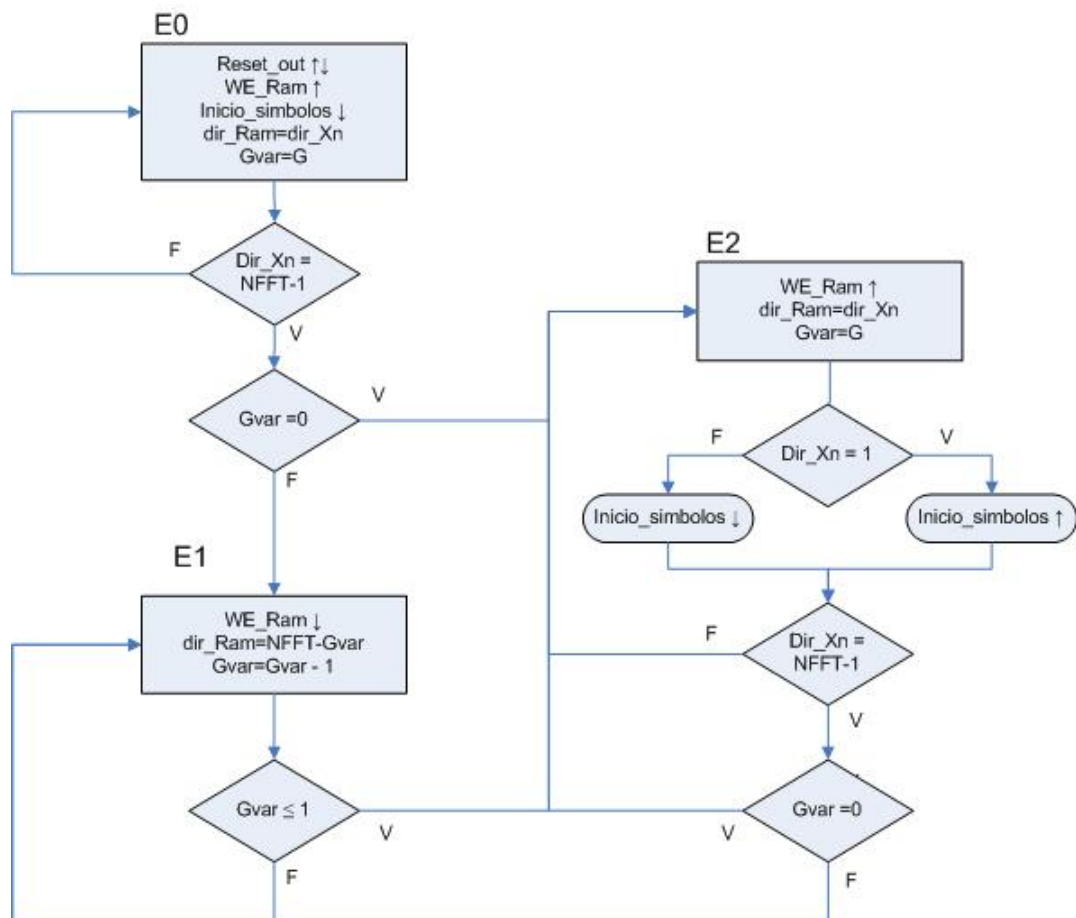


Fig. 3.7 Diagramas de estados del controlador del módulo de Inserción de guarda

Si el parámetro de entrada dim_guarda es mayor que cero entonces la guarda debe ser insertada. Debido a que el entramador e “Inserción de Guarda” trabajan con la misma frecuencia, es necesario que la generación de los valores $\{X[n]\}$ sea deshabilitada, por ende la captura también debe de ser detenida. En otras palabras, la máquina de estados debe de cambiar al estado E1. En este estado, la señal WE_RAM es siempre ‘0’ ya que los datos de entrada no deben ser capturados. Por otra parte, para que los valores muestreados por el registro correspondan a los últimos “ dim_guarda ” componentes, se cumple que:

$$Dir_Ram = NFFT - Gvar \quad (3.11)$$

Donde $Gvar$ varía cada flanco positivo de reloj y es función del estado actual. Si se trata de un estado diferente a E1 entonces $Gvar$ corresponde a dim_guarda , mientras que en el caso de E1 esta variable se decrementa. Una vez que el último valor de la guarda sea insertado ($Gvar=1$), la secuencia original debe ser muestreada. Sin embargo, debido a que se debe de tener un flujo continuo de datos en la salida, es necesario también comenzar a capturar la nueva secuencia $\{X[n]\}$ correspondiente al siguiente símbolo. En definitiva, el controlador debe de pasar al estado E2.

Una vez en el estado E2, la señal WE_Ram debe habilitarse nuevamente ya que los datos deben ser capturados. El arreglo de relojes entre la RAM y el registro de salida, permite que los valores $X[n]$ del símbolo j sean primeramente muestreados en la salida para luego ser remplazado por el valor $X[n]$ del símbolo $j + 1$. De esta forma los datos son muestreados y grabados paralelamente y de forma ascendente. En este estado, al igual que en E1, dir_Ram es igual a dir_Xn. De igual manera cuando todo el símbolo j sea muestreado y el símbolo $j + 1$ sea capturado ($dir_Xn=255$) el controlador debe de cambiar de estado.

Tanto en E0 como en E2, si el parámetro dim_guarda es cero, la máquina debe de pasar a E2 ya que no es necesario insertar la guarda. Por otra parte, como la captura en la RAM debe de ser de datos válidos de la secuencia $\{X[n]\}$, es necesario que exista una perfecta sincronía entre la producción de la secuencia y la captura de la misma. Esta sincronía se logra cargando la salida en_entramador con la señal WE_Ram. De esta manera, la señal en_entramador detendrá la producción de los símbolos solamente cuando se este mostrando la guarda en la salida del TX. La señal en_entramador es

habilitadora de todos los bloques secuenciales del transmisor a excepción de “Inserción de Guarda” que es quien la produce.

Por otra parte, el controlador del bloque se encarga también de generar la señal inicio_simbolo, la cual indica cuando empieza a transmitirse el primer complejo $x[0]$ del símbolo útil. En este proyecto, esta señal es conectada directamente al receptor y tiene la finalidad de indicarle al mismo el inicio del símbolo útil para que se pueda descartar la guarda sin la necesidad de utilizar un sincronizador. La ecuación lógica de esta señal evalúa cuando la máquina secuencial se encuentra en el estado de muestreo E2 y la dirección de la secuencia $X[n]$ (dir_Xn) sea “00000001”.

El valor de 1 para la variable dir_Xn se debe al hecho de que todas las entradas en la RAM están desfasadas un flanco de reloj por los registro de entrada, por ende la salida también experimenta este retraso. Esto implica que cuando dir_Xn sea 1 en E2 en realidad la RAM esta mostrando el valor $X[0]$, que corresponde al inicio del símbolo útil.

4. Demodulador OFDM: Bloque “Receptor”

El Receptor se encarga de recuperar los datos de entrada a partir del bloque transmitido serialmente en el tiempo. Como se ha indicado anteriormente, el receptor no incluye el proceso de sincronización y por ende no utiliza ni los símbolos de training ni el prefijo cíclico. En su defecto, presume la existencia de un sincronizador externo que se encarga de determinar el inicio de un símbolo útil de datos. Dicho sincronizador se encargaría de generar una señal disparadora luego de detectar los símbolos de training y desechar la guarda de cada símbolo de datos. En definitiva, este disparador corresponde a la entrada sincronizada. En nuestro caso, el disparador sincronizado es simulado a través de la señal inicio_simbolo producida por el bloque Inserción de Guarda del Transmisor, ya que esta señal se habilita al inicio de un símbolo útil.

Sin embargo, la señal inicio_simbolo se encarga solamente de desechar la guarda y no discrimina si el símbolo es de training o de datos. Esto es solucionado gracias al controlador de datos del receptor que asume la existencia de una secuencia de cabecera. Este controlador no habilita la decodificación de datos cuando se esta procesando un símbolo de cabecera, es decir cuando el índice del símbolo en un bloque es menor o igual a la dimensión de la cabecera. De aquí tenemos que los símbolos de esta cabecera si son transformados al dominio de la frecuencia pero no son decodificados. Claramente se infiere que en nuestro caso se define a esta dimensión de cabecera con el mismo valor de la dimensión del training. De esta manera, los símbolos del training no afectan en la recuperación de los datos e inclusive es posible observar al training recuperado en el dominio de la frecuencia. Para el caso de que se implemente un sincronizador como el especificado en la parte superior, bastaría simplemente con redefinir la dimensión del preámbulo con un valor de cero en el controlador de datos.

Una vez que el disparador de entrada le indica al receptor el inicio del símbolo útil, este debe ser transformado al dominio de la frecuencia por el bloque FFT. De esta forma, a la salida de la transformada se tiene la secuencia fasorial recuperada $\{\tilde{C}_k\}$. Cada fasor \tilde{C}_k es enrutado al bloque combinatorial mapeador de constelación que se encarga de transformar el

fasor recuperado al componente \tilde{c}_k correspondiente, esto acorde a las especificaciones de las constelaciones utilizadas y al factor F del bloque multiplicador de escalamiento en el transmisor.

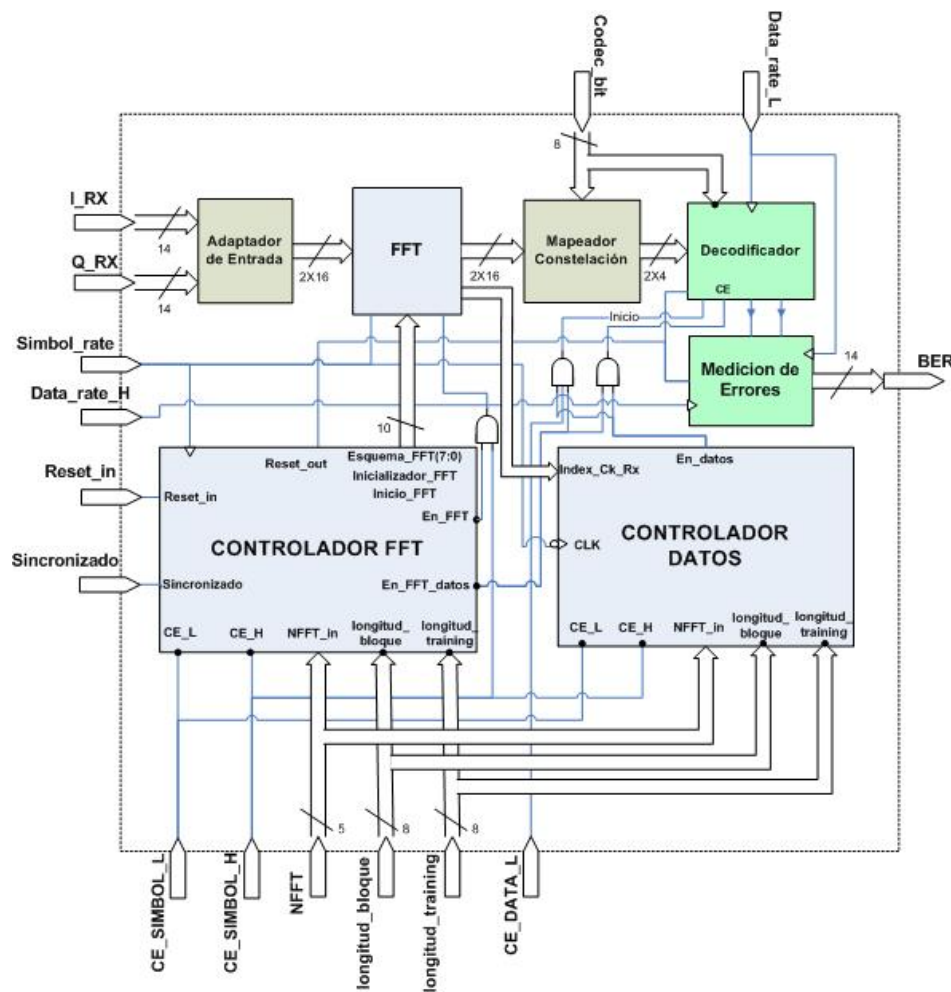


Fig. 4. 1 Diagrama de bloques del Receptor

Gracias al mapeador de constelación el decodificador ha sido diseñado de una manera independiente, apegada al estándar y en perfecta correspondencia con el codificador del transmisor. Como resultado del

proceso de decodificación, a la salida de este bloque se tienen los bits recuperados y una señal disparadora que indica cuando un nuevo bit es decodificado. Finalmente, estas señales son direccionadas al bloque de detección de errores que se encarga de regenerar la secuencia binaria transmitida y compararla con la secuencia recuperada para medir el BER.

Para la realización del proceso indicado en la parte superior es fundamental el uso de los dos controladores del receptor: controlador FFT y controlador de datos. Como su nombre lo indica, el controlador FFT se encarga de inicializar el core FFT, así como indicarle cuando debe procesar en función de la entrada disparadora “sincronizado”. Por su parte, el controlador de datos se encarga de generar el habilitador `en_datos` para el decodificador tomando en cuenta el índice del símbolo y el índice de la subportadora procesada.

Debido a las mismas razones expuestas en el caso del transmisor para el uso de la IFFT, es necesario utilizar un adaptador de entrada antes de que los datos sean direccionados al FFT. Este bloque se encarga insertar dos bits de relleno, debido a las especificaciones en la dimensión de la entrada del core. Por la complejidad e importancia de los demás bloques, estos serán expuestos a continuación.

4.1. Bloque FFT

El bloque FFT tiene la función de transformar los símbolos OFDM recuperados del dominio del tiempo al dominio de la frecuencia. El bloque recibe de manera serial y ascendente los NFFT componentes $x[n]$ correspondientes al símbolo útil OFDM en el tiempo y luego del tiempo de procesamiento, muestra también de forma serial y ascendente la Transformada rápida de Fourier de dicha secuencia.

Al igual que el bloque IFFT, para su implementación se utilizó el core “Logic Core Fast Fourier Transform v2.1” de Xilinx. Para el caso del bloque IFFT el valor de FWD_INV es ‘0’. Además los puertos NFFT_WE, FWD_INV_WE y SCALE_SCH_WE están cortocircuitados entre sí y corresponden a la señal Inicializador_FFT del Controlador FFT.

4.2. Controlador FFT

El controlador FFT se encarga de inicializar y controlar el core FFT para el procesamiento de la transformada de Fourier del símbolo útil. Podemos dividir el funcionamiento de la máquina secuencial del controlador en tres etapas: la etapa de inicialización (Estados E0, E1 y E2), la etapa de espera mientras se descarta el prefijo

(Estado E3) y la etapa de procesamiento de la transformada durante el símbolo útil (Estados E4 y E5).

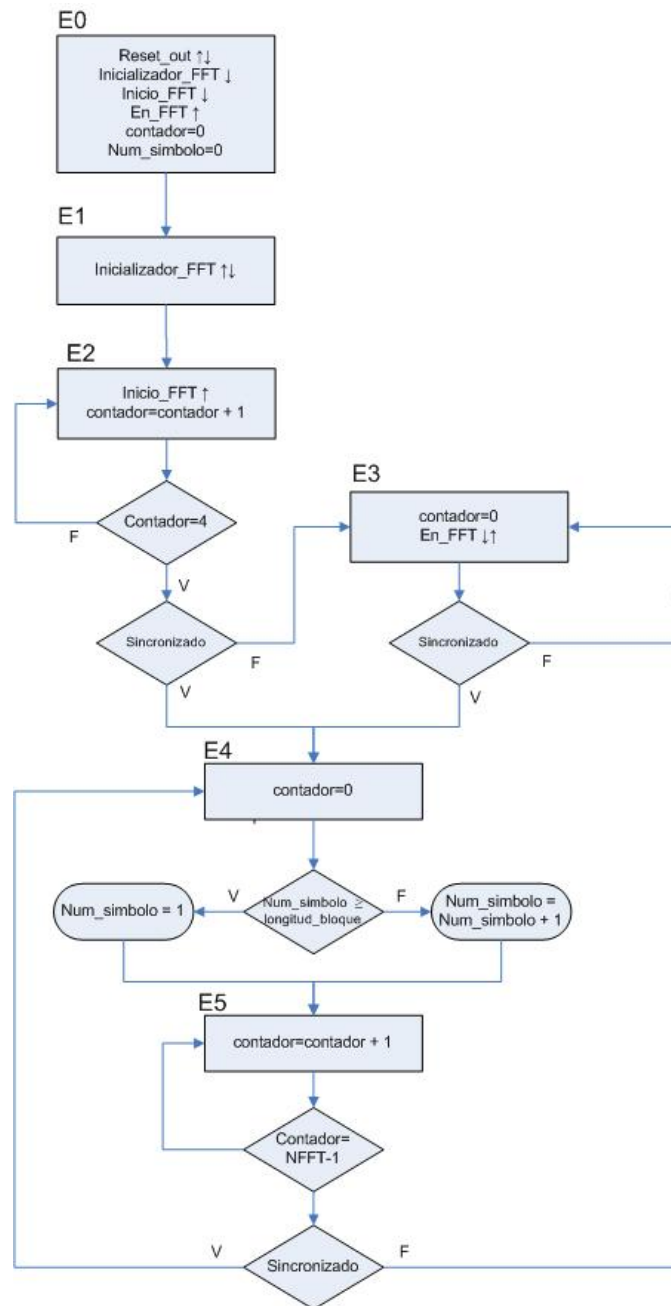


Fig. 4. 2 Diagrama ASM del controlador FFT

Como su nombre lo indica, la etapa de inicialización se encarga de inicializar el core FFT para su posterior funcionamiento. El comportamiento de los estados de inicialización (E0, E1 y E2) es análogo al de sus estados homólogos en el Controlador Contador del Transmisor.

De igual manera, las salidas generadas en dicho proceso (Inicializador_FFT, Inicio FFT, Esquema_FFT) tienen exactamente el mismo comportamiento. Básicamente la diferencia radica en que en E2 se espera que el contador sea 4 y no 3 ya que en el RX no hay un flanco extra en la entrada del core, como en el caso del TX.

Una vez que el core está inicializado el controlador se debe de encargar de indicarle cuando debe de procesar la transformada de Fourier del símbolo útil. Para realizar esto utiliza las etapas de espera y de procesamiento del símbolo mediante el control de la entrada sincronizado. Esta señal le indica al controlador el inicio del símbolo útil y se utiliza para simular un sincronizador externo que la generaría luego de analizar el preámbulo de sincronización y descartar el prefijo cíclico de cada símbolo a procesar. Por esta razón, al estado E3 se denomina la etapa de espera o descarte de prefijo ya que la máquina secuencial esperara en dicho estado hasta observar que la entrada sincronizado sea habilitada. Cuando

esto suceda, la etapa de procesamiento debe de inicializarse. En esta etapa, el habilitador de la FFT (en_FFT) es habilitado y gracias a la configuración de los estados E4 y E5 durara justamente el tiempo del símbolo útil, es decir $NFFT$ flancos de reloj. Por otra parte, La señal en_FFT_datos también se habilita en E3 y tiene el mismo comportamiento que en_FFT , la diferencia radica es que esta varía en el siguiente flanco positivo del sistema luego de la variación de en_FFT (En_FFT esta sincronizado con la máquina secuencial del controlador, es decir $Simbol_L$).

Una vez que culmine la etapa de procesamiento se interrogará a la señal “sincronizado” para saber si se debe proceder a la carga continua del siguiente símbolo útil (para el caso que dim_guarda sea 0) o debe de ir al estado de espera E3. De esta manera, la máquina oscilara entre la etapa de espera y la de procesamiento en función de sincronizado a excepción del caso en que la señal de reset sea habilitada. En este caso, la máquina secuencial se dirigirá al estado de reseteo E0 y comenzará nuevamente la etapa de inicialización.

4.3. Mapeador de Constelación

Este bloque se encarga de transformar el fador \tilde{C}_k al correspondiente componente \tilde{c}_k recuperado en función de las

constelaciones transmitidas. El efecto del ruido provoca que el fasor \tilde{C}_k recibido desde el bloque FFT sufra variaciones a partir del valor C_k . Estas variaciones son de naturaleza aleatoria y pueden ser selectivas en frecuencia, es decir:

$$\tilde{C}_k = C_k + ruido(k) \quad (4.1)$$

El sistema de recepción digital debe estar diseñado para asumir la presencia de dicho ruido y tratar de recuperar el valor C_k a partir de la señal \tilde{C}_k recibida. En este aspecto es necesario tener en cuenta que la asunción general es que los valores C_k son transmitidos de forma equiprobable. En el caso de las codificaciones utilizadas esto se logra mediante la división del plano I-Q en 2^{Codec_bit} zonas geométricas idénticas y centradas en cada uno de los posibles valores C_k . De esta manera si \tilde{C}_k pertenece al área correspondiente a C_k se debe interpretar que el valor transmitido correspondió a C_k . Esto último es básicamente lo que realiza el mapeador de constelación, la diferencia radica en que también toma en cuenta el factor multiplicativo utilizado en el transmisor y en la misma función de la LUT realiza la división de forma implícita.

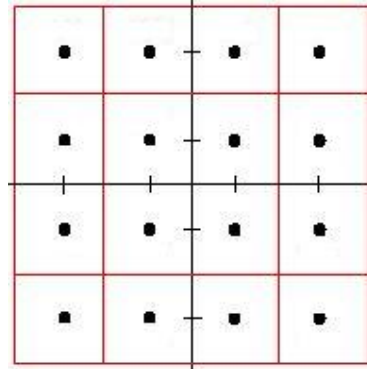


Fig. 4.3 Ejemplo de división de áreas para el mapeo de C_k en 16-QAM.

Debido a que las relaciones de orden necesarias para evaluar en que zona se encuentra el fasor \tilde{C}_k son idénticas para el caso de I y Q, es posible definir un solo mapeador unidimensional para cada eje (fig. 4.4).

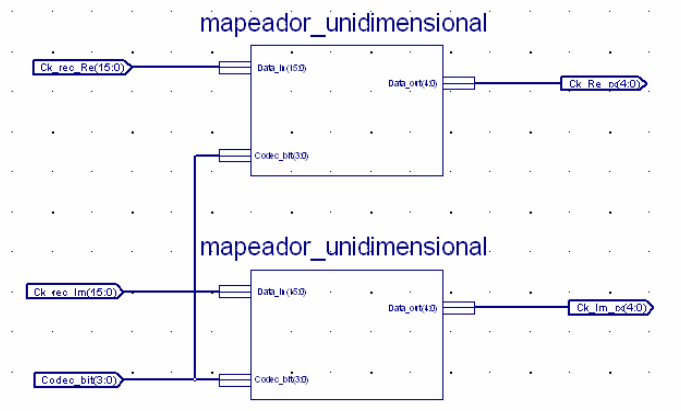


Fig. 4.4 Esquemático del bloque mapeador de constelación

El único aparente inconveniente que acarearía esta implementación sería para el caso de B-PSK ya que en el eje Q no se debe evaluar nada (es siempre cero) y en el eje I si. Sin embargo, por esta misma razón el valor imaginario resultante es descartado en el decodificador, por lo cual es indiferente el valor de Q en el caso de B-PSK.

Para la definición de los valores imaginarios y reales del componente C_k resultante, el mapeador unidimensional debe evaluar el tipo de codificación y los valores de I y Q dependiendo el caso. En definitiva, la función de salida $\text{Out}(\text{In}, \text{Codec_bit})$ de este mapeador esta definida por:

$$\text{Out}(\text{In}, 1) = \text{Out}(\text{In}, 2): \begin{cases} 1 & ; \text{In} > 0 \\ 0 & ; \text{resto de valores de In} \end{cases}$$

$$\text{Out}(\text{In}, 4): \begin{cases} 3 & ; \text{In} > 7000 \\ 1 & ; 7000 \geq \text{In} > 0 \\ -1 & ; 0 \geq \text{In} > -7000 \\ -3 & ; \text{resto de valores de In} \end{cases}$$

$$\begin{aligned}
 \text{Out}(In, 6): & \left\{ \begin{array}{l} 7 \quad ; \quad In > 9000 \\ 5 \quad ; \quad 9000 \geq In > 6000 \\ 3 \quad ; \quad 6000 \geq In > 3000 \\ 1 \quad ; \quad 3000 \geq In > 0 \\ -1 \quad ; \quad 0 \geq In > -3000 \\ -3 \quad ; \quad -3000 \geq In > -6000 \\ -5 \quad ; \quad -6000 \geq In > -9000 \\ -7 \quad ; \quad \text{resto de valores de In} \end{array} \right. \\
 \\
 \text{Out}(In, 8): & \left\{ \begin{array}{l} 15 \quad ; \quad In > 9800 \\ 13 \quad ; \quad 9800 \geq In > 8400 \\ 11 \quad ; \quad 8400 \geq In > 7000 \\ 9 \quad ; \quad 7000 \geq In > 5600 \\ 7 \quad ; \quad 5600 \geq In > 4200 \\ 5 \quad ; \quad 4200 \geq In > 2800 \\ 3 \quad ; \quad 2800 \geq In > 1400 \\ 1 \quad ; \quad 1400 \geq In > 0 \\ -1 \quad ; \quad 0 \geq In > -1400 \\ -3 \quad ; \quad -1400 \geq In > -2800 \\ -5 \quad ; \quad -2800 \geq In > -4200 \\ -7 \quad ; \quad -4200 \geq In > -5600 \\ -9 \quad ; \quad -5600 \geq In > -7000 \\ -11 \quad ; \quad -7000 \geq In > -8400 \\ -13 \quad ; \quad -8400 \geq In > -9800 \\ -15 \quad ; \quad \text{resto de valores de In} \end{array} \right. \quad (4.2)
 \end{aligned}$$

4.4. Decodificador

Este módulo tiene la función de decodificar una cadena de bits a partir del componente complejo \tilde{c}_k . Como se puede observar en la tabla VIII, el modulo consta de seis entradas y dos salidas. El

bloque tiene la capacidad de realizar decodificación BPSK, QPSK, 16-QAM, 64-QAM y 256-QAM. El proceso de decodificación comenzará en el flanco de reloj (data_rate_dec) que se perciba un '1' en la entrada inicio. Luego de esto se mostrará el primer bit codificado b_0 en el siguiente flanco negativo de reloj, de esta manera el proceso se repite hasta que se hayan mostrado los Codec_bit bits que se desean decodificar.

Tabla VIII
Puertos del Codificador

Nombre	Longitud	Tipo	Descripción
Ck_Re_dec	5	entrada	Componente real del valor de entrada. Valor I de las constelaciones especificadas en la sección 2.1.
Ck_Im_cod	5	entrada	Componente Imaginario del valor de entrada. Valor Q de las constelaciones especificadas en la sección 2.1.
inicio	1	entrada	Disparador de inicio. Indica al decodificador la presencia de un símbolo válido a la entrada y el inicio del procesamiento.
Codec_bit_d	4	entrada	Número de bits a decodificar. Señal correspondiente al parámetro Codec_bit.
data_rate_dec	1	entrada	Reloj del Decodificador
CE	1	entrada	Habilitador del reloj del decodificador.
reset_in_dec	1	entrada	Señal de reset.
bits_decod	1	Salida	Bits decodificados resultantes. La cadena resultantes es mostrada de forman ascendente, es decir: $b_0, b_1, \dots, b_{Codec_bit}$
CLK_Dec	8	salida	Presenta un pulso cada vez que se muestra un bit válido en la salida decodificada.

Durante el procesamiento y muestra de datos la señal inicio será ignorada y los componentes \tilde{c}_k de entrada deberán permanecer invariantes para un correcto funcionamiento del sistema. Dada las características seriales de la salida la velocidad de entrada se limita a un máximo dado por la tasa de símbolo del sistema.

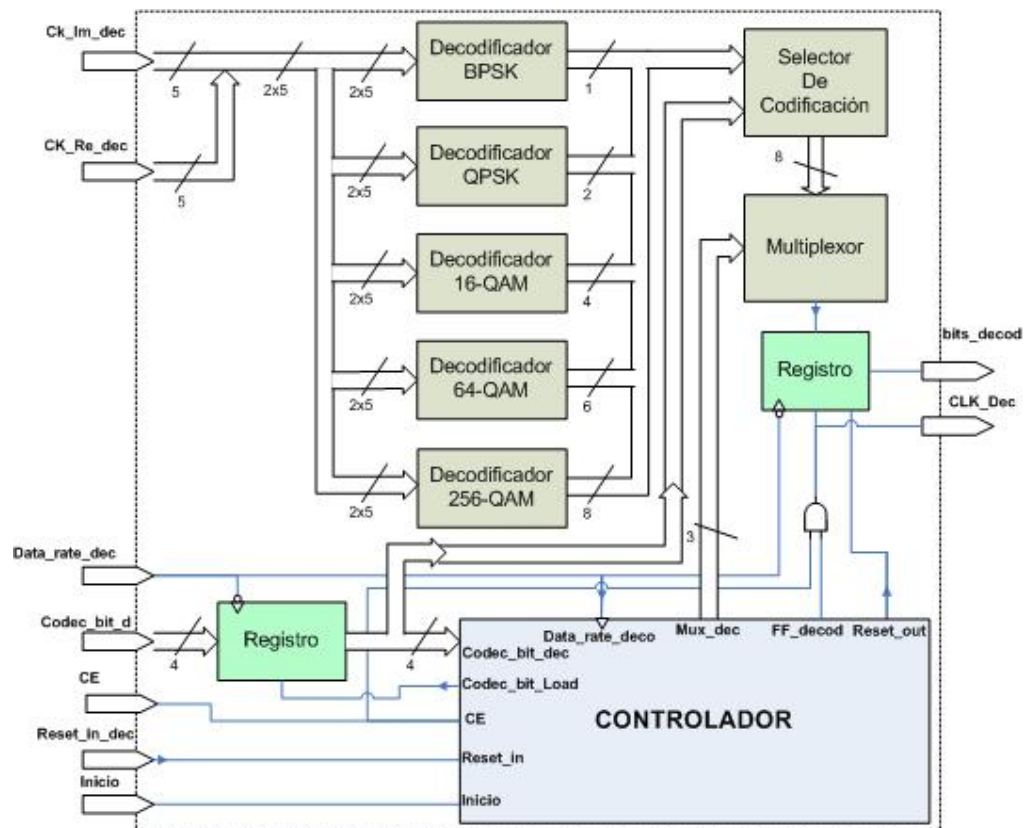


Fig. 4.5 Diagrama de Bloques del Decodificador

La máquina secuencial del sistema consta de tres estados (fig. 4.6): El estado de reseteo y espera E0, y los estados de

procesamiento E1 y E2 (E1 para el procesamiento de los Codec_bits - 1 bits menos significativos y E2 para el procesamiento del bit más significativo).

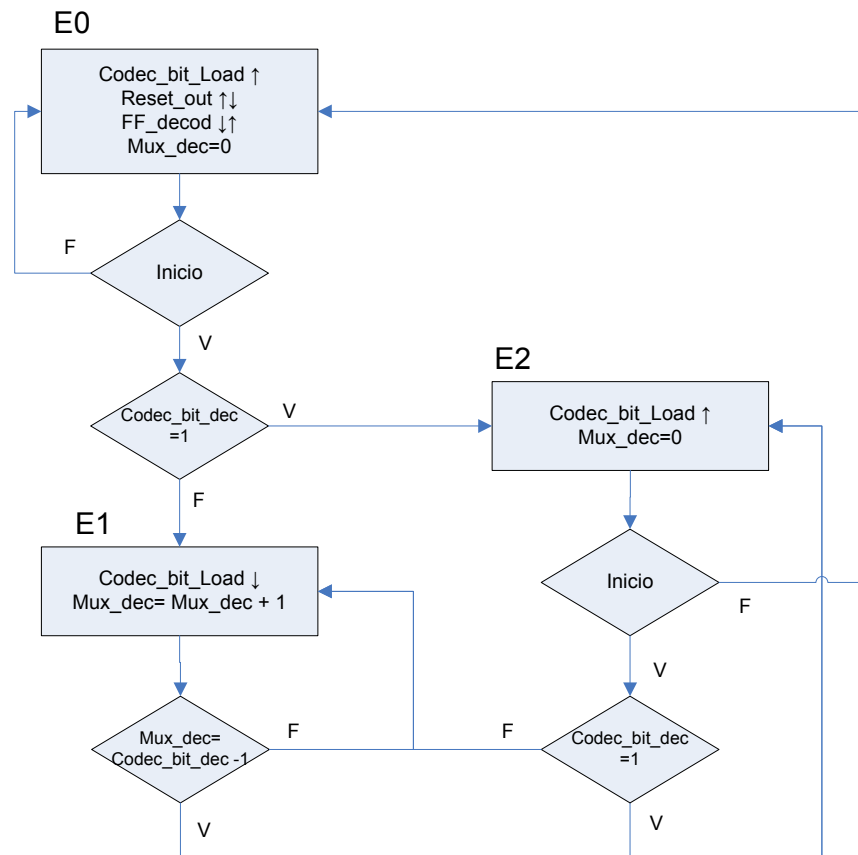


Fig. 4. 6 Diagrama ASM del Controlador del Decodificador

En el estado inicial E0 todo el sistema es reseteado. El sistema estará en este estado indefinidamente mientras la entrada Reset_in_dec tenga un valor '1' o la señal de inicio tenga un valor '0'. De igual manera, la señal de reset mandará a la máquina

secuencial al estado inicial sin importar que valor tengan las otras entradas o el estado actual del sistema.

En el caso que no se cumpla ninguna de las condiciones expuestas anteriormente, el sistema pasará al estado E1 o E2 en el siguiente flanco dependiendo del esquema de codificación utilizado, es decir el parámetro `Codec_bit`. Este parámetro es cargado al decodificador a través de la entrada `Codec_bit_d` cuando la salida del controlador `Codec_bit_load` tenga un valor de '1'. El muestreador `Codec_bit_load` es habilitado por el controlador un flanco de reloj antes del inicio del procesamiento de un componente de entrada. Es decir, cuando no se ha procesado ningún componente c_k (Estado de reseteo E0) o cuando el último bit del componente anterior es procesado (Estado de procesamiento E2).

Una vez que el nuevo valor del esquema de codificación es cargado es necesario evaluar si se trata del caso de B-PSK, para definir si el procesamiento debe pasar por el estado E1 o ir directamente a E2. Esto se debe a que para el caso de B-PSK solo se decodifica un bit y por ende no es aplicable el estado E1 ya que $\text{Codec_bit} - 1 = 0$.

Como se puede observar en la fig. 4.5, los componentes complejos \tilde{c}_k son cargados simultáneamente a los decodificadores paralelos (BPSK, QPSK, 16-QAM, 64-QAM y 256-QAM), que corresponden a tablas LUT definidas a partir de las constelaciones implementadas en este proyecto. Estos 5 bloques combinatoriales alimentan al Selector de Codificación, el mismo que proyectara una señal de 8 bits en función del valor `Codec_bit` cargado previamente. Los “`Codec_bit`” bits menos significativos de la señal de salida del Selector de Codificación corresponderán a la salida del bloque propio a la modulación usada, mientras que los restantes $8 - \text{“Codec_bit_dec”}$ bits tendrán un valor de ‘0’.

De esta manera, la versión paralela de la modulación implementada es alimentada al bloque multiplexor como una subcadena formada por los “`Codec_bit`” bits menos significativos. Finalmente, con la ayuda de este bloque, la versión paralela es multiplexada en el tiempo mediante la señal `Mux_dec` que corresponde a un contador tal como se observa en la máquina de estados del controlador. En los estados de procesamiento E1 y E2 este contador `Mux_dec` realiza la secuencia $0, 1, \dots, \text{Codec_bit}$ con lo cual a la salida del multiplexor se tiene la versión serial

decodificada $b_0, b_1, \dots, b_{Codec_bit}$. Finalmente, esta señal es muestreada en el registro de Salida mediante el uso de la señal FF_decod que es habilitada en los estados de procesamiento E1 y E2.

4.5. Controlador de Datos

El controlador de datos se encarga del control del decodificador mediante la señal en_datos. Para la generación de esta señal, el controlador toma en cuenta el índice del símbolo y el índice del componente \tilde{c}_k procesado. Podemos dividir el funcionamiento del controlador en dos partes: la máquina secuencial y el codificador de en_datos.

La máquina de estados del controlador consta de 3 estados: El estado inicial E0 y los estados de procesamiento E1 y E2. La función de la máquina de estados es mantener el control del contador interno Num_simbolo el cual le indica al decodificador del controlador que símbolo se esta procesando en ese momento. Para esto, el controlador contabiliza el número del símbolo (Si el símbolo procesado no es el último se incrementa el contador Num_simbolo) en E1 y luego espera hasta que el símbolo haya culminado, es decir, cuando la entrada indice_Ck sea $NFFT - 1$.

Indice_Ck corresponde al índice de la secuencia \tilde{c}_k recuperada por el core FFT, es decir el puerto XK_INDEX del bloque FFT.

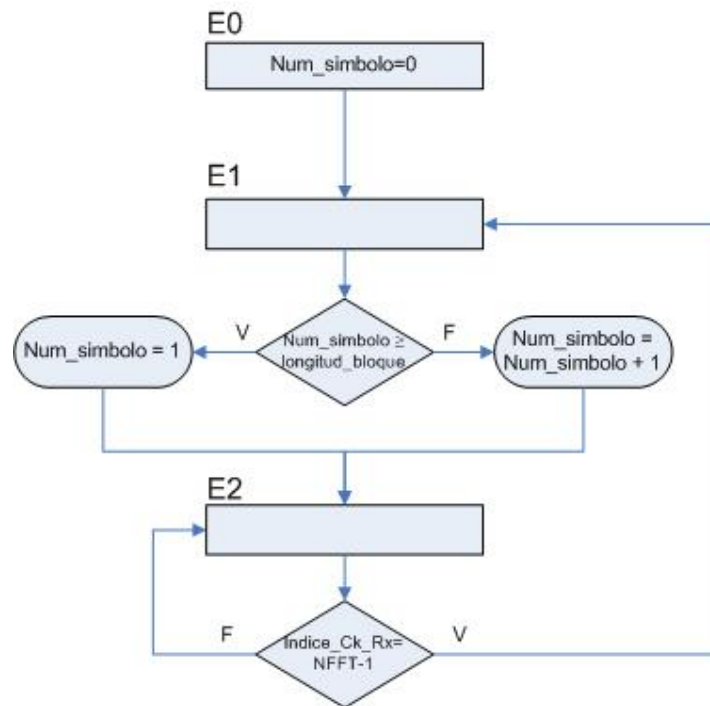


Fig. 4.7 Diagrama ASM del Controlador de Datos

Una vez culminada la secuencia del símbolo, la máquina secuencial regresa al estado E1 para que el nuevo símbolo sea contabilizado y el proceso se repita indefinidamente. Si el anterior símbolo procesado fue el último, es decir si Num_simbolo es igual a la dimensión del bloque, entonces el contador Num_simbolo no debe ser incrementado sino cargado con 1 ya que se trata del primer símbolo del siguiente bloque. De esta manera, la máquina

secuencial permanecerá en los estados de procesamiento hasta que la señal de reset sea habilitada, en este caso el próximo estado será E0 sin importar las demás condiciones. Por tratarse del estado inicial, en E0 el valor de Num_simbolo corresponde a 0 ya que ningún símbolo es procesado.

Por su parte, el decodificador del controlador se encarga de producir la salida en_datos a partir de la entrada indice_Ck y el contador Num_simbolo generado por la máquina de estado. El comportamiento del decodificador para la señal en_datos es similar al del caso de su señal homologa en el Controlador contador del transmisor. Si $Num_simbolo < longitud_training$, en_datos debe ser '0' ya que el core FFT esta procesando los símbolo de training. Por otra parte, si $Num_simbolo \geq longitud_training$ en_datos dependerá del índice k de las portadoras C_k , es decir de indice_Ck. En otra palabras, cuando indice_Ck corresponda a un valor k definido para datos en_Datos debe ser '1', sino debe ser '0'.

4.6. Medición de Errores

El bloque de Medición de Errores se encarga de regenerar la secuencia binaria transmitida y compararla con la secuencia recuperada para medir el BER. Para esto hace uso de las señales

bits_rec y CLK_dec provenientes del Decodificador que se constituyen en la entrada del bloque Medición de Errores.

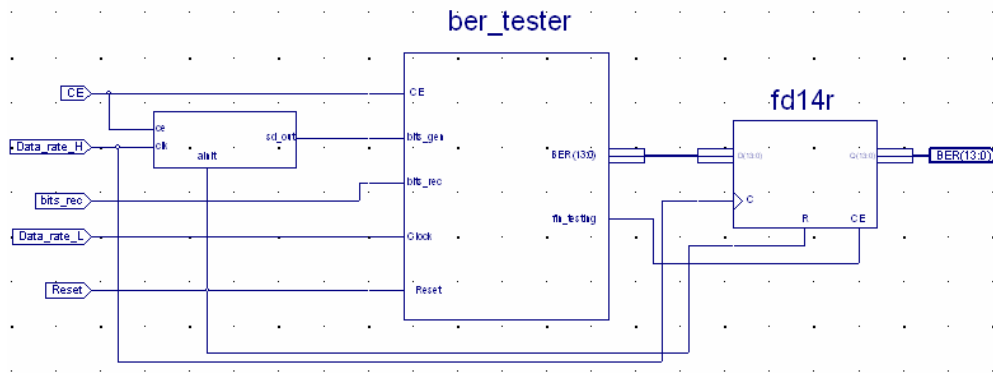


Fig. 4. 8 Esquemático del bloque Medición de errores.

Para la regeneración el bloque hace uso de una instancia de la fuente de datos implementada en el transmisor (Sección 3.1) mediante el uso del core LFSR. Debido a que el registro de salida del decodificador utiliza lógica negada, es necesario que el core LFSR use la misma señal de reloj. Esto debido a que la secuencia regenerada debe estar en sincronía con la recuperada. De igual manera, es necesario que la señal habilitadora del core sea la misma que la utilizada en el registro de salida, es decir la entrada CLK_dec. Teniendo en cuenta estas dos premisas, a la salida del core LFSR se obtiene la secuencia regenerada en sincronía con la recuperada para el posterior análisis del BER.

Como su nombre lo indica, el bloque Ber_tester se encarga de calcular el BER mediante la comparación de los bits recuperados y los regenerados. El bloque Ber_tester esta constituido por una máquina sincrónica que contabiliza cuantas inconcordancias existen entre los bits recuperados (bits_rec) y los regenerados (bits_gen) en una muestra de cierta dimensión determinada. El tamaño de la muestra esta definido por el parámetro interno max_bit y debe ser un múltiplo de 10 para que el número de concordancias contabilizado pueda ser interpretado como la tasa de error. De esta manera el parámetro max_bit define el grado de resolución teniendo un máximo dado por los 14 bits de salida, es decir un valor max_bit de 10000 con una resolución de 10^{-4} .

Para la ejecución interna la máquina secuencial utiliza dos contadores internos: bits y tasa_errores. El contador bits contabiliza el índice del valor muestreado cada flanco positivo de reloj. De esta manera, el contador es incrementado hasta que llega al valor definido por max_bit; es decir si bits es igual a max_bit en un flanco positivo de reloj, este debe ser cargado con 1 ya que se trata del primer valor de la siguiente muestra. Por otra parte el contador tasa_errores contabiliza el número de concordancias acumuladas en la muestra. Es decir, en cada flanco de reloj evalúa si existe una

inconcordanca e incrementa el contador, de no ser así lo deja invariante. Al igual que bits, el valor de tasa_errores debe ser reiniciado en el flanco de reloj que el contador bits sea igual a max_bit; es decir, si existe una inconcordanca el valor de tasa_errores deber ser '1' sino debe ser '0'.

Debido a que el contador tasa_errores es un acumulador y no determina el valor real del BER hasta el final de la muestra, es necesario indicar al sistema cuando se tiene un valor válido en el contador: esta es la función de la señal fin_testing. Esta salida solo esta activa en el flanco de reloj positivo en el cual se tiene que bits es igual a max_bit – 1, es decir se activa en el mismo instante en el cual el valor max_bit de la muestra es evaluado. De aquí se infiere que la variable fin_testing tiene un valor de '1' solamente cuando se tiene un valor válido en el acumulador tasa_errores. Así tenemos que la señal fin_testing tiene las características necesarias para funcionar como habilitador de un registro de salida, que tenga como entrada al contador tasa_errores y que funcione con el flanco negativo del reloj del bloque Ber_tester. Finalmente, el BER corresponde a la salida de dicho registro que se actualizaría cada max_bit valores muestreados, es decir al final de cada muestra.

5. Funcionamiento del Sistema Implementado

En este capítulo se revisarán las características del sistema implementado en el FPGA. Como se indicó en el capítulo II para la etapa de implementación y pruebas se ha hecho uso de diferentes herramientas de hardware y Software. Sin embargo, quizás las herramientas fundamentales para comprender el funcionamiento del sistema implementado son las del ChipScopePro. Estas herramientas integran componentes claves para el análisis lógico de hardware con el diseño del sistema en el interior del FPGA. De esta forma, el software del CSP se comunica con estos componentes internos y proveen un completo análisis lógico al diseñador [a]. Así, teniendo en cuenta las características del CSP, podríamos definir el esquema del sistema implementado para las pruebas mediante la Fig. 5.1.

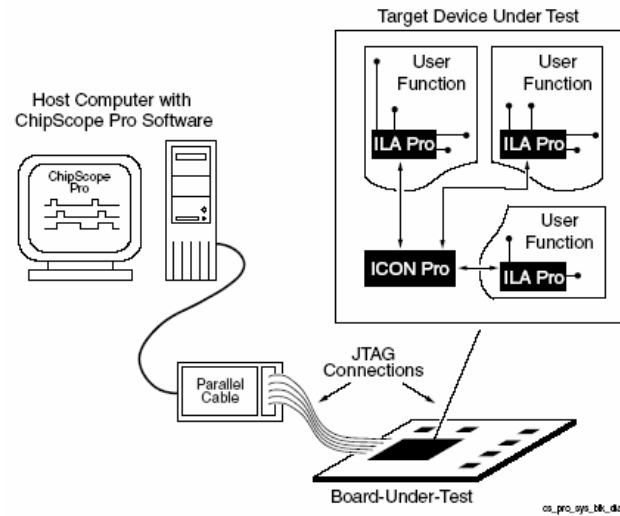


Fig. 5. 1 Esquema de las pruebas del sistema mediante el CSP

Como podemos observar en la figura superior, el esquema consta de una PC, la tarjeta del FPGA y un cable que funciona como interfase entre el FPGA y el Software del CSP en la PC. En nuestro caso el cable utilizado corresponde al Paralel IV de Xilinx (fig. 5.2), mientras que el FPGA corresponde al FPGA principal del KIT XTREME DSP VIRTEX II PRO (fig. 5.6).



Fig. 5. 2 Cable Paralel IV de Xilinx

El paquete del CSP esta formado por 3 herramientas: CSP Generator, GSP Inserter y el CSP Analyzer. Las dos primeras herramientas tienen el objetivo de agregarle al sistema original los componentes del CSP (CSP CORE) a ser implementados en el FPGA, mientras que la última permite controlar dichos componentes una vez que el sistema ha sido implementado. De esta forma tenemos que el software ilustrado en la PC de la fig. 5.1 corresponde específicamente al Analyzer. Por otro lado, la diferencia entre las dos primeras herramientas radica en el momento en que se agrega el CSP CORE y en la disponibilidad de COREs que cada herramienta provee (fig. 5.3). En nuestra implementación la herramienta utilizada a este nivel es el CSP Inserter ya que representa menores tiempos de respuesta en la etapa de implementación.

En este proyecto, existen dos tipos de CSP Cores insertados mediante el CSP Inserter: CSP Core de control (ICON) y CSP Core de usuario (ILA). El ICON tiene la función de controlar al resto de CSP CORE's (en nuestro caso los ILA) y en la etapa de pruebas su acción es prácticamente transparente para el diseñador. Esto se debe a que las interfaces gráficas del CSP Analyzer simulan una interacción directa con los CORE de usuario.

Debido a que se ha escogido la utilización del CSP Inserter, las señales a observarse mediante el ILA deben ser conectadas al CORE luego de la

Síntesis (fig. 5.4) y variaran según sea el experimento a realizarse durante este capítulo. Asimismo, al ILA también debe indicársele la señal de reloj y la señal o señales de disparo (trigger) y sus opciones de evaluación.

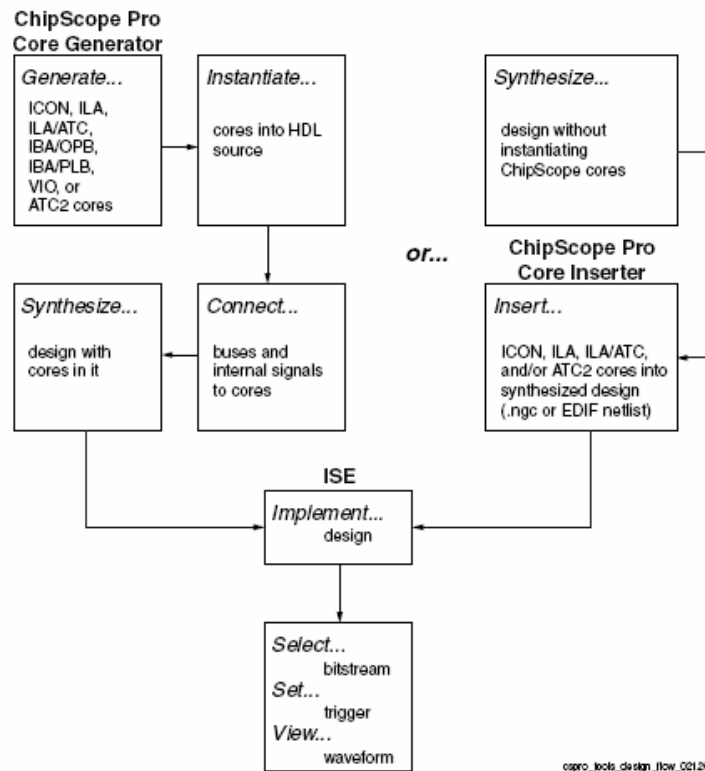


Fig. 5.3 Diagrama de procesos para las pruebas mediante el CSP

Estas señales permiten la flexibilidad de ejecutar la carga de los datos internos una vez que cierta condición se haya cumplido. En este caso la señal de disparo siempre corresponde al reloj ya que nuestro objetivo es describir el sistema en su totalidad y no ante la presencia de cierto evento. En lo referente al reloj del ILA, para una correcta visualización de

los datos en el CSP Analyzer, es siempre preferible que las señales conectadas al ILA estén en sincronía con el reloj.

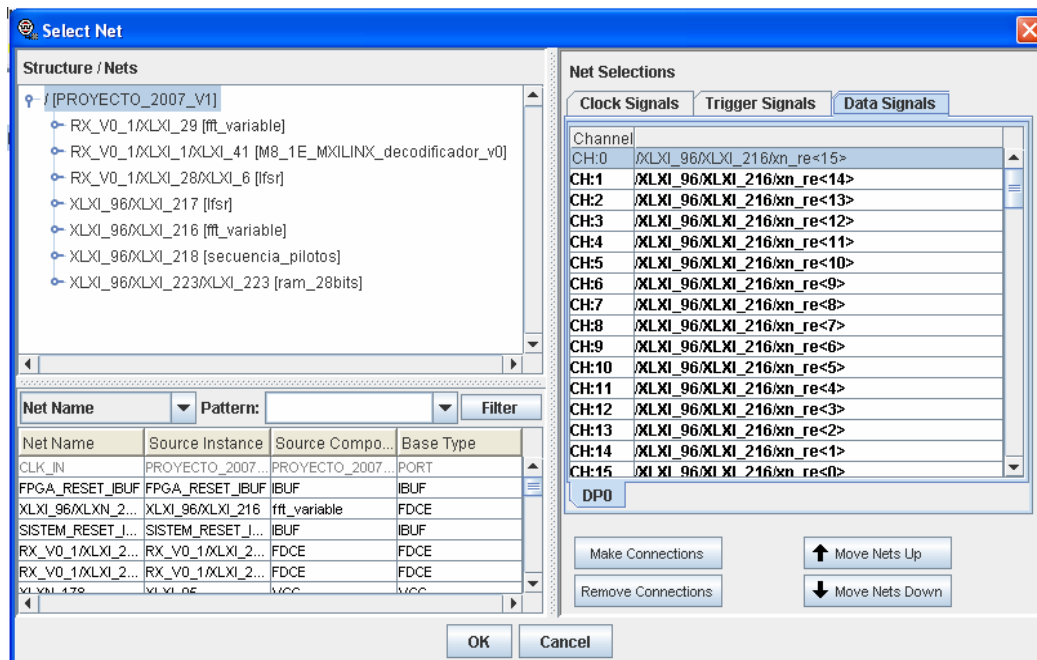


Fig. 5.4 Mapeo de señales mediante el CSP Inserter

Una vez que los CSP CORE sean agregados, el proceso normal de implementación mediante el ISE (Fig. 5.3) continuará hasta la obtención del archivo tipo bit. En nuestro caso, la implementación de este archivo es realizada mediante el Software FUSE en el FPGA principal del KIT. Este software interactúa con la tarjeta mediante el puerto USB de la PC y no solo permite la implementación del archivo tipo bit, sino también el control del sistema ya implementado. Como se puede observar en la Fig. 5.5 mediante el uso de este esquema es posible alimentar a los FPGAs

(FPGA de reloj y FPGA principal) del KIT con dos señales manipuladas desde el FUSE. En esta configuración, el FUSE propone utilizar estas señales como dos RESETs. En nuestro caso se ha acogido dicha sugerencia y dichas señales corresponden a las entradas de reset FPGA_Reset (Puerto V29) y Sistem_Reset (Puerto P28).

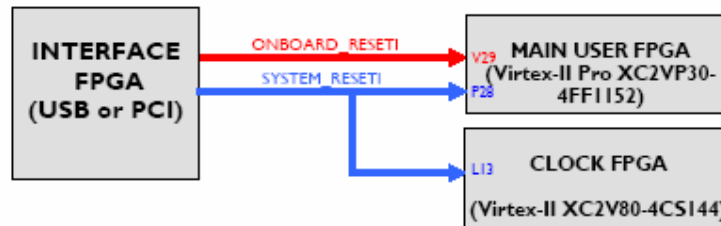


Fig. 5.5 Estructura de Reseteo sugerida por el XTREME DSP

En este aspecto es necesario tener en cuenta que el FPGA de reloj no es utilizado en nuestro diseño. Esto se debe a que en este proyecto no se utiliza los convertidores DAC y ADC por ende no tiene sentido tener una sincronía externa al FPGA. Como se explico en el capítulo II, la señal de reloj del sistema es generada a partir de un DCM que recibe una señal de reloj conectada directamente al FPGA. En las pruebas descritas durante todo este capítulo, esta señal de reloj corresponde al oscilador programable conectado en el puerto AJ17 (fig. 5.6).

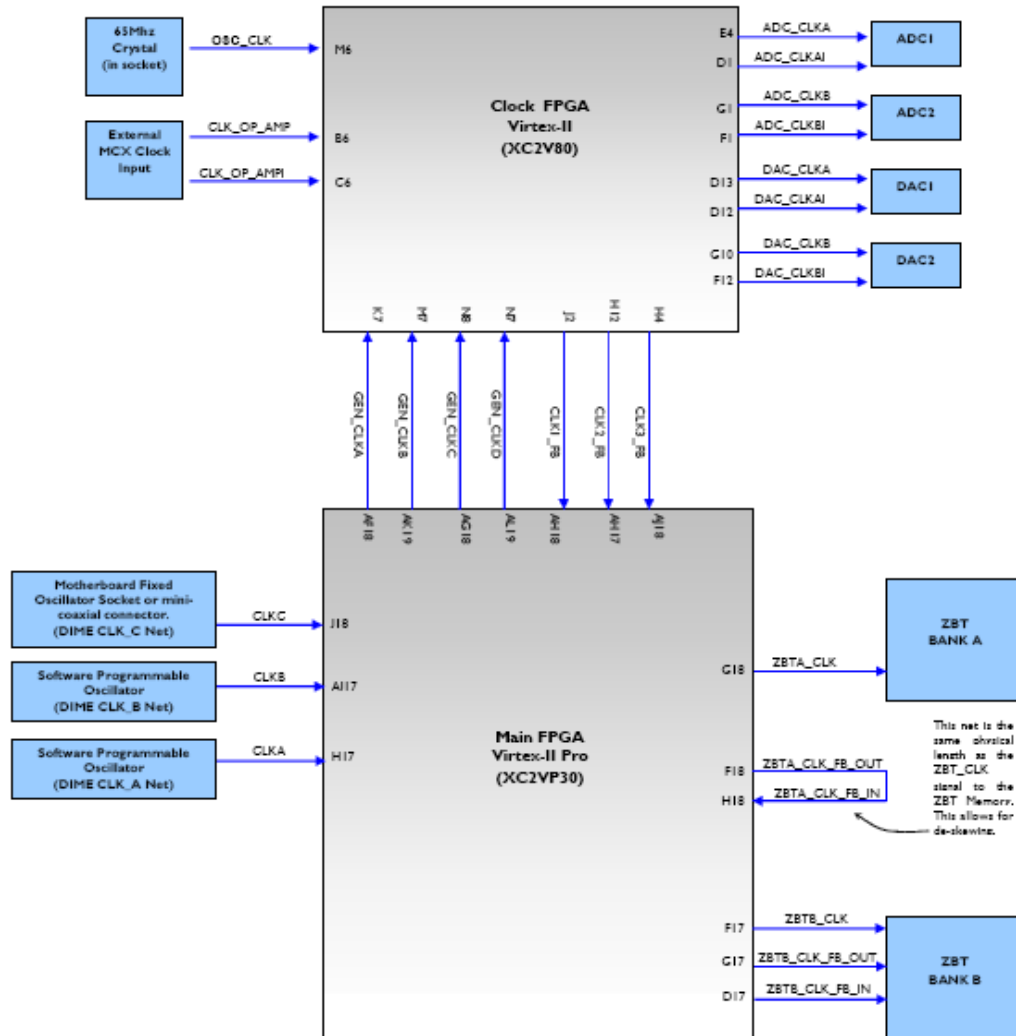


Fig. 5.6 Clocking propuesto por el XTREME DSP

Al igual que los Resets del sistema la frecuencia de este reloj es controlada mediante el Software FUSE. De esta manera, es posible definir y variar la frecuencia entre los diferentes valores soportados por el KIT, es decir:

20 MHz; 25 MHz; 30 MHz; 33.33 MHz; 40 MHz; 45 MHz; 50 MHz; 60 MHz; 66.66 MHz; 70 MHz; 75 MHz; 80 MHz; 90MHz; 100 MHz; 120 MHz.

Este detalle es de suma importancia ya que permite la evaluación del sistema implementado en diferentes valores de frecuencia. Esto implica que mediante el uso del CSP Analyzer es posible observar el comportamiento de las señales internas en función de cada frecuencia utilizada.

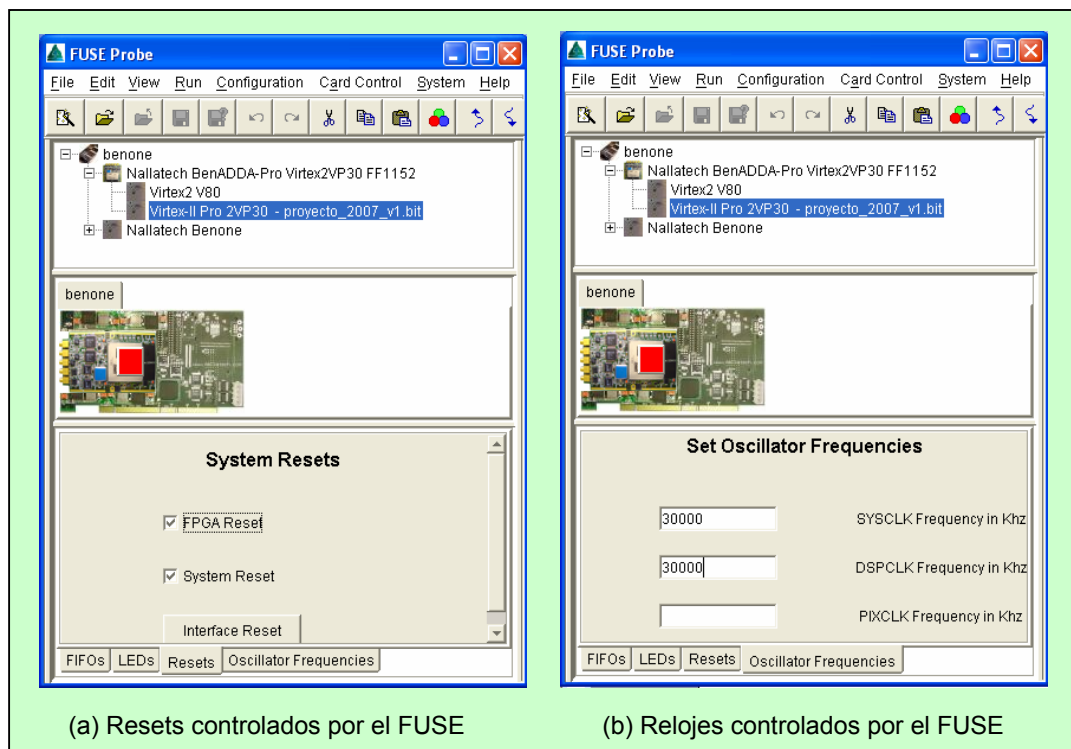


Fig. 5.7 Ventanas del Fuse para el control de los Resets y el reloj de entrada.

El CSP Analyzer permite analizar los datos muestreados en el interior del FPGA de tres formas: Diagrama de tiempo, listado en forma de tablas y Graficas de los Buses de datos. Las dos primeras herramientas son aplicables para cualquier tipo de señal, mientras que la última es exclusiva de los buses. En realidad en el proceso de mapeo solo es posible conectar señales binarias, los denominados buses consisten en agrupaciones de señales binarias según sea la necesidad del evaluador. Tanto la estructura de estos buses como su formato (Entero, real, escalado, etc) son definidos en el CSP Analyzer. Otra opción permitida por el CSP, es la de exportar en un archivo de texto los datos muestreados. Así, en el caso de necesitarse, dichos valores pueden ser analizados con otro software que permita una mayor flexibilidad. Todas estas opciones del CSP Analyzer han sido utilizadas en el desarrollo del actual proyecto y se verán plasmadas durante las pruebas del presente capítulo.

En fin, de manera general es posible dividir las pruebas realizadas en tres partes: Frecuencia de Operación del Sistema implementado, generación de los símbolos y bloques OFDM y evaluación de los datos recuperados. La primera parte se encargará de evaluar el rango de frecuencia de operación del sistema en función de los diferentes parámetros del sistema. En estas pruebas se alimentará al sistema con las diferentes frecuencias soportadas por el FUSE. La segunda parte, explorara el proceso de generación de los símbolos y el bloque OFDM. Para esto se

utilizará pruebas modulares que evaluará el proceso en cada etapa. Finalmente, en la tercera parte, se evaluarán los datos recuperados en el receptor. Es decir, se examinarán las constelaciones, la secuencia binaria y por último el BER en diferentes escenarios provocados. Cada una de estas secciones será cubierta en detalle a continuación.

5.1. Frecuencia de Operación del Sistema Implementado

En esta sección se analizará el rango de frecuencias válidas para el sistema implementado. La metodología a utilizar consiste en variar los parámetros del sistema y evaluar la correcta operación del mismo en función de las frecuencias de prueba soportadas por el KIT XTREME DSP. Debido a que el sistema implementado consta de un lazo entre el TX y el RX, se espera que no exista distorsión alguna en los datos recuperados. Esto implica que las constelaciones recuperadas, así como la correspondencia entre los datos transmitidos y recibidos y por ende el BER deben ser perfectas.

Es posible dividir los parámetros del sistema en dos grupos: los parámetros de sincronización y los parámetros de operación. Los parámetros de sincronización serían aquellos que participan en la conformación de los habilitadores de reloj en el sistema, es decir:

Div y Codec_bit. Por otro lado, los parámetros de operación serían aquellos que definen la conformación de los símbolos y bloques OFDM. Entre estos parámetros tenemos a: longitud_training, longitud_bloque, dim_guarda y Codec_bit, siendo este último considerado un caso especial ya que es tanto variable de operación como de sincronización.

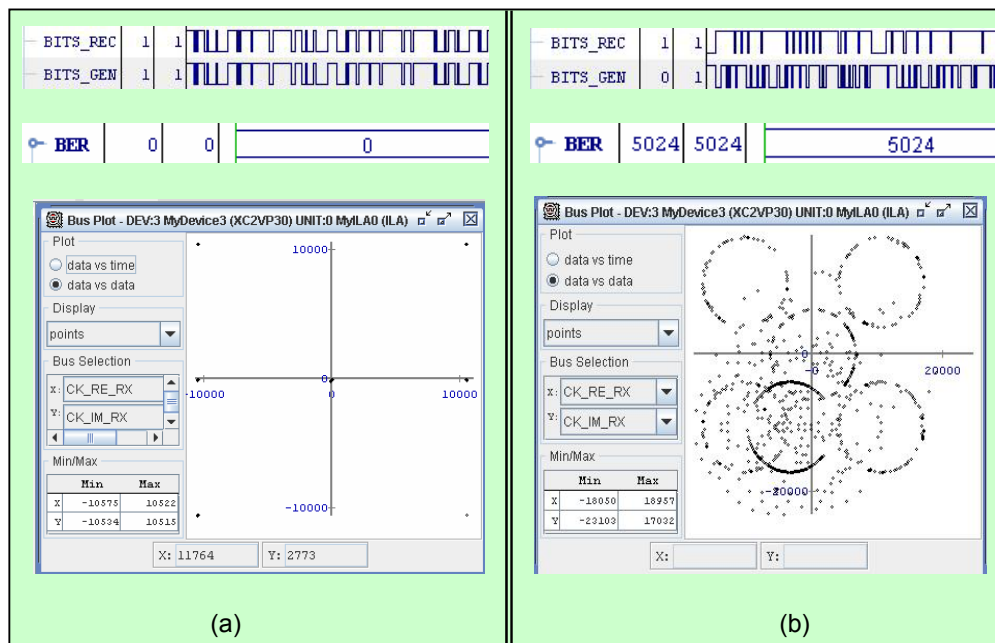


Fig. 5.8 Datos recuperados: (a) Sistema en perfecta sincronía, (b) Sistema desincronizado.

La primera hipótesis que se podría plantear a partir de esto es que el rango de frecuencia válido solamente depende de las variables de sincronización y no de operación. No obstante, esto es solo una hipótesis y por ende necesitaría ser evaluada. Dadas las

características de las pruebas realizadas, se tomará en cuenta esta presunción en el análisis desarrollado en esta sección.

Las primeras pruebas consisten en la evaluación del rango de frecuencias válido ante la variación de la longitud de la guarda y el tipo de codificación. Para estas pruebas se ha escogido una estructura de bloque definida por un símbolo de training y dos de datos, además el parámetro DIV definido corresponde a 8. La tabla IV resume el resultado de estas pruebas y como podemos observar se han evaluado todas las combinaciones posibles de Codec_bit, mientras que para el caso de dim_guarda se han escogido el valor nulo y los valores correspondientes a los parámetros G soportados por el IEEE 802.16 (tabla I).

Es importante tener en cuenta que el valor expuesto como límite superior en la prueba no es exacto, ya que solamente se pueden probar las frecuencias soportados por el KIT XTREME DSP. Debido a esto, lo que realmente estamos obteniendo es un rango válido para la frecuencia máxima, dicho rango estaría limitado por la mayor frecuencia con resultados sincronizados y la menor frecuencia con resultados erróneas. Este criterio es el que se expresa en la columna Rango de la tabla IX.

Tabla IX
Pruebas de Rango de Frecuencias ante la variación de la Guarda y
La Codificación

Codec_bit	dim_guarda	Frecuencia máxima observada (MHz)	Rango (MHz)
1	64(G=1/4)	70	$70 \leq f_{MAX} < 75$
	32(G=1/8)	70	$70 \leq f_{MAX} < 75$
	16(G=1/16)	70	$70 \leq f_{MAX} < 75$
	8(G=1/32)	70	$70 \leq f_{MAX} < 75$
	0(G=0)	70	$70 \leq f_{MAX} < 75$
2	64(G=1/4)	66.66	$66.66 \leq f_{MAX} < 70$
	32(G=1/8)	66.66	$66.66 \leq f_{MAX} < 70$
	16(G=1/16)	66.66	$66.66 \leq f_{MAX} < 70$
	8(G=1/32)	66.66	$66.66 \leq f_{MAX} < 70$
	0(G=0)	66.66	$66.66 \leq f_{MAX} < 70$
4	64(G=1/4)	66.66	$66.66 \leq f_{MAX} < 70$
	32(G=1/8)	66.66	$66.66 \leq f_{MAX} < 70$
	16(G=1/16)	66.66	$66.66 \leq f_{MAX} < 70$
	8(G=1/32)	66.66	$66.66 \leq f_{MAX} < 70$
	0(G=0)	66.66	$66.66 \leq f_{MAX} < 70$
6	64(G=1/4)	60	$60 \leq f_{MAX} < 66.66$
	32(G=1/8)	60	$60 \leq f_{MAX} < 66.66$
	16(G=1/16)	60	$60 \leq f_{MAX} < 66.66$
	8(G=1/32)	60	$60 \leq f_{MAX} < 66.66$
	0(G=0)	60	$60 \leq f_{MAX} < 66.66$
8	64(G=1/4)	50	$50 \leq f_{MAX} < 60$
	32(G=1/8)	50	$50 \leq f_{MAX} < 60$
	16(G=1/16)	50	$50 \leq f_{MAX} < 60$
	8(G=1/32)	50	$50 \leq f_{MAX} < 60$
	0(G=0)	50	$50 \leq f_{MAX} < 60$

Más haya de esta incertidumbre, en estas pruebas se encuentra una relación directa entre la frecuencia máxima observada y el tipo de codificación. Conforme aumenta el número de bits codificados, disminuye el máximo de frecuencia soportado. Por otra parte, en base de lo observado, no se verifica injerencia alguna de la dimensión de la guarda en el rango de la frecuencia soportada. Es decir, si se mantiene constante el tipo de codificación y se modifica G no se observa variación alguna en nuestra evaluación.

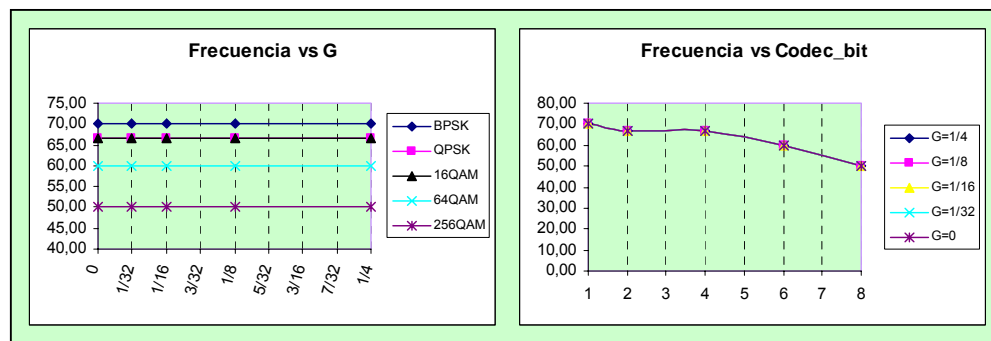


Fig. 5.9 Pruebas de Rango de Frecuencias ante la variación de la Guarda y la Codificación.

Los resultados obtenidos hasta este momento fortalecen la hipótesis planteada ya que `dim_guarda` es un parámetro de operación y `codec_bit` uno de sincronización. Sin embargo, debido a la incertidumbre introducida, las pruebas no son concluyentes. En base de esto, el siguiente paso consistió en variar la estructura de los bloques OFDM y observar si existía alguna modificación en los

resultados. Si la estructura del bloque es independiente de la frecuencia de operación, entonces el rango resultante debe ser el mismo siempre y cuando se mantengan fijos el resto de parámetros. Los resultados de estas pruebas están resumidos en la tabla X, que muestra el rango de la frecuencia máxima ante 9 diferentes estructuras de bloque OFDM con codificación 16QAM y $G=1/8$.

Tabla X
Pruebas de Rango de Frecuencias ante la variación de la Estructura del Bloque

Longitud de bloque	Longitud de training	Frecuencia máxima observada (MHz)	Rango (MHz)
64	0	66,66	$66.66 \leq f_{MAX} < 70$
	10	60	$60 \leq f_{MAX} < 66.66$
	15	60	$60 \leq f_{MAX} < 66.66$
128	0	66,66	$66.66 \leq f_{MAX} < 70$
	10	60	$60 \leq f_{MAX} < 66.66$
	15	50	$50 \leq f_{MAX} < 60$
255	0	75	$75 \leq f_{MAX} < 80$
	10	60	$60 \leq f_{MAX} < 66.66$
	15	70	$70 \leq f_{MAX} < 75$

Como podemos observar en la tabla X, en este caso la evidencia muestra que la hipótesis planteada es incorrecta: el rango de la frecuencia máxima si varía ante la variación de la estructura del bloque. Es más, la variación es de tal naturaleza que no se puede

determinar una correspondencia independiente para cada parámetro de bloque. Esto se evidencia en la fig. 5.10 que muestra una correlación entre ambos parámetros de bloque. Este resultado es de suma importancia, ya que es probable que exista una correlación entre todos los parámetros analizados en esta sección.

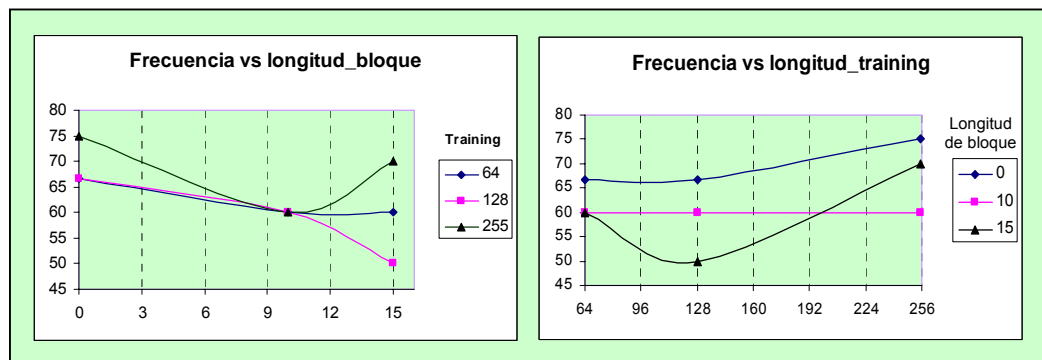


Fig. 5.10 Pruebas de Rango de Frecuencias ante la variación de la Estructura del Bloque

El último parámetro a analizar es la variable de sincronización Div. Para estas pruebas se ha mantenida fija una configuración de bloque determinada por un símbolo de training y dos de datos. Por otro lado, el esquema de codificación de los símbolos de datos ha sido variado entre todos los valores posibles para cada Div, mientras que la dimensión de la guarda corresponde a un $G=1/8$. Los resultados de estas pruebas han sido resumidos en la tabla XI.

Tabla XI
Pruebas de Rango de Frecuencias ante la variación de los
Parámetros de Sincronización

Div	Codec_bit	Frecuencia máxima observada (MHz)	Rango
1	1	70	$70 \leq f_{MAX} < 75$
2	1	66.66	$66.66 \leq f_{MAX} < 70$
	2	66.66	$66.66 \leq f_{MAX} < 70$
4	1	70	$70 \leq f_{MAX} < 75$
	2	60	$60 \leq f_{MAX} < 66.66$
	4	66.66	$66.66 \leq f_{MAX} < 70$
6	1	50	$50 \leq f_{MAX} < 60$
	2	50	$50 \leq f_{MAX} < 60$
	4	66.66	$66.66 \leq f_{MAX} < 70$
	6	50	$50 \leq f_{MAX} < 60$
8	1	70	$70 \leq f_{MAX} < 75$
	2	66.66	$66.66 \leq f_{MAX} < 70$
	4	66.66	$66.66 \leq f_{MAX} < 70$
	6	60	$60 \leq f_{MAX} < 66.66$
	8	50	$50 \leq f_{MAX} < 60$

En realidad esto es de esperarse ya que ambas señales definen los habilitadores de reloj y por ende están muy relacionadas entre sí en lo referente a la frecuencia máxima. Sin embargo, es importante tener en cuenta que no se puede observar un patrón definido para esta correlación, lo cual es en cierta forma inesperado.

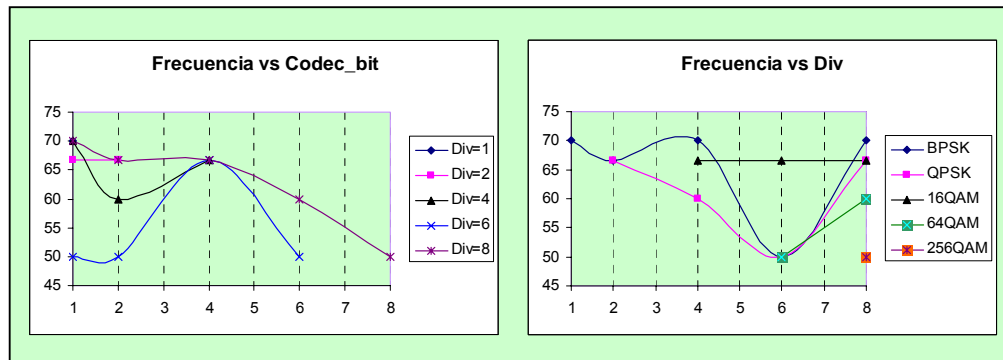


Fig. 5. 11 Pruebas de Rango de Frecuencias ante la variación de los parámetros Div y Codec_bit.

En base de todo lo expuesto, con los experimentos realizados no se puede concluir a ciencia cierta una regla de correspondencia para la conducta de la frecuencia máxima en función de los parámetros del sistema. Lamentablemente, la incertidumbre obtenida por la limitante de frecuencias, el tiempo de generación de cada caso de prueba (para cada escenario es necesario repetir todo el proceso de síntesis e implementación) y la cantidad de combinaciones de las mismas complican sobremanera este análisis.

No obstante, dados los objetivos del actual proyecto de tesis, esto en realidad no es una limitante alarmante. Hay que recordar que este proyecto se orienta a ser la etapa inicial de investigaciones más específicas y por ende intentar obtener este tipo de

conclusiones va más allá de los límites impuestos. En cierta forma, la elevada cantidad de casos a probar se debe al carácter modular y parametrizable que potencializa las futuras implementaciones. Además, la limitante del tiempo de generación de las pruebas puede ser reducida variando dinámicamente los parámetros a analizar. Por otro lado, el aspecto de las frecuencias de prueba, dado el actual hardware es infranqueable, por lo cual se debería de analizar una solución a nivel de hardware. En todo caso, todas estas opciones y análisis corresponden a futuras etapas de la actual implementación. Lo importante de todo esto es que a partir de estas pruebas es posible obtener conclusiones que sirven de premisas para futuras implementaciones.

5.2. Generación de los Símbolos y bloques OFDM

En esta sección se explorará el proceso de generación de los símbolos y bloques OFDM. Para esto se expondrán los resultados de diferentes pruebas enfocadas en las diferentes fases de conformación del símbolo y el bloque OFDM.

Las primeras dos pruebas examinan el comportamiento de los bloques que forman las subportadoras del símbolo OFDM de datos: Codificación de Datos y formación de las subportadoras de

piloto y guarda. Por otro lado, la tercera prueba evalúa la conformación del símbolo de Training. Una vez analizado el proceso de formación de las portadoras de training y de datos, se procederá a analizar la formación de los símbolos y el bloque OFDM en el dominio de la frecuencia y el tiempo. En estas pruebas últimas dos se evaluará de forma global el transmisor y el sistema completo respectivamente, por esta razón no se necesita simular señal alguna.

A pesar de esto, para la realización de las tres primeras pruebas es necesario simular algunas señales de control. Para este objetivo, se utiliza el bloque VHDL “Simulador Control Portadoras”. Como podemos observar en la tabla XII, los puertos de este bloque tienen diferentes funciones según sea la prueba a realizar.

El bloque “Simulador Control Portadoras” posee de una máquina de estados interna (fig. 5.12) que consta de dos estados: El estado de reseteo E0 y el estado de funcionamiento E1. Como su nombre lo indica en el estado E0 el reset del sistema será habilitado (Reset_out) y todas las variables internas serán inicializadas. La máquina de estado funcionará en E1 siempre y cuando el reset de la entrada no sea habilitado.

Tabla XII
Puertos del bloque “Simulador Control Portadoras”

Nombre	Longitud	Tipo	Descripción / Prueba		
			Codificación de Datos	Generación de Portadoras Pilotos y Guarda	Generación de Portadoras de Training
Clock	1	entrada	Reloj del Sistema	Reloj del Sistema	Reloj del Sistema
Reset_in	1	entrada	Reset de Entrada	Reset de Entrada	Reset de Entrada
Modo	1	entrada	Debe de tener un valor de '0' (GND)	Debe de tener un valor de '1' (VCC)	Debe de tener un valor de '1' (VCC)
Reset_out	1	salida	Reset del sistema	Reset del sistema	Reset del sistema
En_Pilotos	1	salida	No aplica	Habilitador de la secuencia ω_k para la formación de pilotos	No aplica
Codec_bit	4	salida	Número de bits codificados	Número de bits codificados	Número de bits codificados
Indice_Ck	8	salida	No aplica	Indice k del componente C_k	Indice k del componente Ck

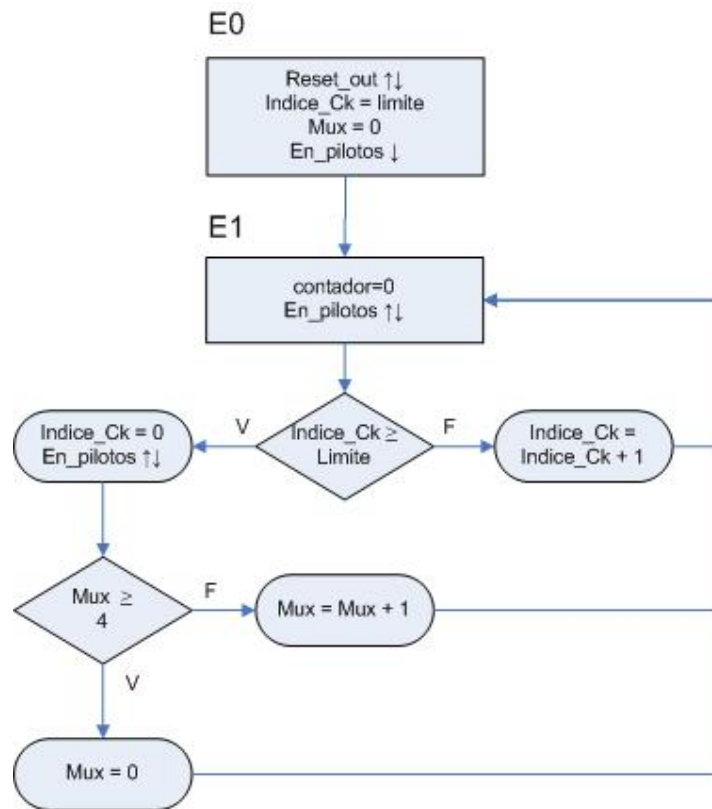


Fig. 5. 12 Diagrama ASM de la máquina secuencial del bloque “Simulador Control Portadoras”

En E1 se generaran dos secuencias dadas por las variables Índice_Ck y Mux. La secuencia Índice_Ck dependerá del valor del parámetro “límite”, que a su vez obedece al modo de operación del sistema. La secuencia de la salida Índice_Ck variara cada flanco negativo de reloj (Clock) y corresponderá a $\{0, 1, \dots \text{límite}\}$.

Tabla XIII
Valor de Variable Limite en funcion del Modo del bloque
“Simulador_Control_Portadoras”

Modo	Límite	Tipo de Prueba
'0'	23	Codificación de datos
'1'	255	Generación de portadoras de Training / Generación de portadoras Pilotos y Guarda

Por otra parte, la secuencia de la salida Mux variará cada vez que se cumpla un ciclo de la secuencia de Indice_Ck. Como se puede observar en la fig. 5.12 el valor de Mux variara cada vez que Indice_Ck llegue al valor "límite". La secuencia de la salida Mux también es sincrónica con el flanco negativo del reloj (Clock) pero corresponderá a $\{0, 1, \dots 4\}$. De esta manera, si tenemos en cuenta el Codificador interno de la señal Codec_bit (Tabla XIV), Mux seleccionara un tipo de codificación para cada secuencia de Indice_Ck.

Tabla XIV
Codificador de Salida Codec_bit ("Simulador_Control_Portadoras")

Mux	Codificación	Codec_bit
0	BPSK	1
1	QPSK	2
2	16-QAM	4
3	64-QAM	6
4	256-QAM	8

De manera similar a Mux, la salida En_pilotos solamente se habilitará cuando la secuencia de Indice_Ck haya culminado. Por

ende, esta señal también es sincrónica con el flanco negativo del reloj. La justificación de este esquema será expuesto en el desarrollo de los tres primeros experimentos dependiendo del caso.

5.2.1. Codificación de Datos

El objetivo de esta prueba es evaluar el correcto funcionamiento del bloque de codificación. Para esto, al igual que en el proyecto final, se utilizará un secuenciador binario pseudoaleatorio para generar la entrada de datos. La señal de reloj del sistema esta dada por un bloque DCM y las pruebas han sido realizadas a una frecuencia de reloj de 30 MHz. Por otra parte, como se indicó al inicio de esta sección, el esquema de codificación estará dado por el bloque “Simulador Control Portadoras” que debe tener un valor ‘0’ en su entrada modo. El esquema realizado para estas pruebas puede ser visualizado en la fig. 5.13.

La explicación de definir la entrada modo con un ‘0’ es que parámetro “limite” del “Simulador Control Portadoras” debe ser inicializado con un valor de 23(Tabla XIII) para las pruebas de codificación. De esta forma, el valor Mux

cambiará justamente cada 24 flancos de reloj, es decir podemos definir infinitas secuencias de codificación conformadas por 24 bits.

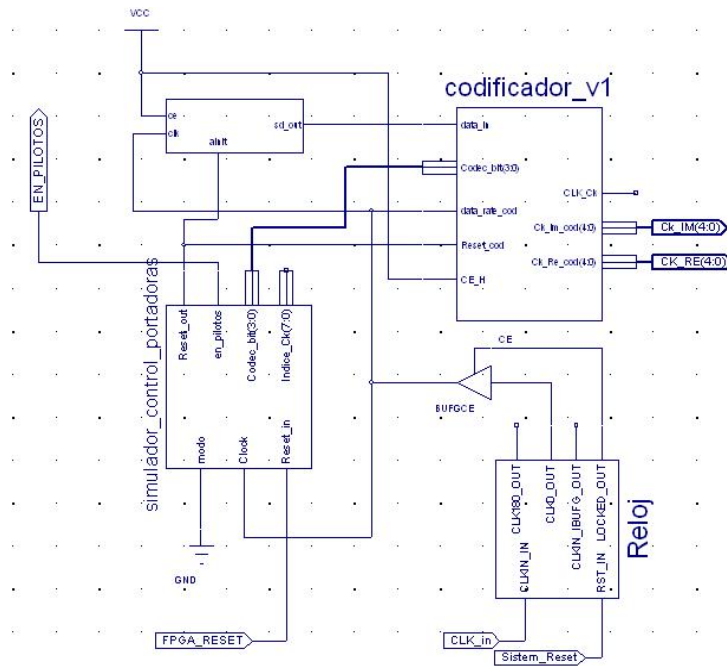


Fig. 5. 13 Esquemático de las pruebas del bloque de Codificación de Datos.

El valor 24 ha sido escogido para las pruebas debido a que es múltiplo de todos los posibles valores Codec_bit. Así, sin importar el tipo de codificación utilizada, al finalizar una secuencia de la variable Indice_Ck simultáneamente se tendrá el fin de un componente c_k . Si recordamos el capítulo II, esta propiedad es necesaria ya que si se cambia de

codificación en medio del procesamiento de un componente c_k este cambio es ignorado. En fin, dado este esquema, no solo permite evaluar la correcta codificación de forma estática sino también ante la variación dinámica de la entrada Codec_bit.

Para la evaluación del correcto funcionamiento de este esquema, las variables a analizar serían Codec_bit, la entrada binaria, el componente c_k codificado y el indicador de datos válidos FF_COD (Que en este experimento corresponde a la salida del codificador CLK_DEC en la fig. 3.3).

El primer factor a evaluar sería el correcto funcionamiento del esquema planteado en la parte superior. Acorde a este, debemos de tener diferentes secuencias de Codificación dadas por 24 flancos de reloj en los cuales tengamos un número exacto de componentes c_k . La manera más simple de evaluar esto es mediante un diagrama de tiempo del CSP Analyzer.

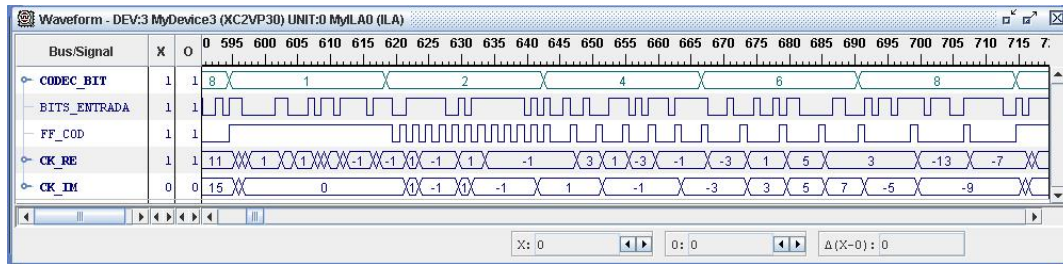


Fig. 5. 14 Diagrama de Tiempo de las Pruebas de Codificación de Datos

Como se puede observar en la fig. 5.12, tal como se espera se tiene un conjunto periódico de secuencias contiguas de Codificación, es decir: BPSK, QPSK, 16QAM, 64QAM y 256QAM. Por otro lado, si tomamos en cuenta que la salida FF_COD mostrara un '1' cada vez que se tenga un complejo c_k a la salida, se esperaría tener exactamente $24/\text{Codec_bit}$ pulsos en cada secuencia de codificación. Esta característica puede también ser observada en la fig. 5.14 y con mayor claridad en la fig. 5.15 que muestra el mismo diagrama de tiempo pero desglosando cada caso de codificación. Por otra parte, en ambas figuras se puede apreciar una línea continua en FF_COD para el caso de BPSK. Esto se debe a que en este caso la señal FF_COD no es deshabilitada ya que se produce un nuevo valor c_k cada flanco de reloj.

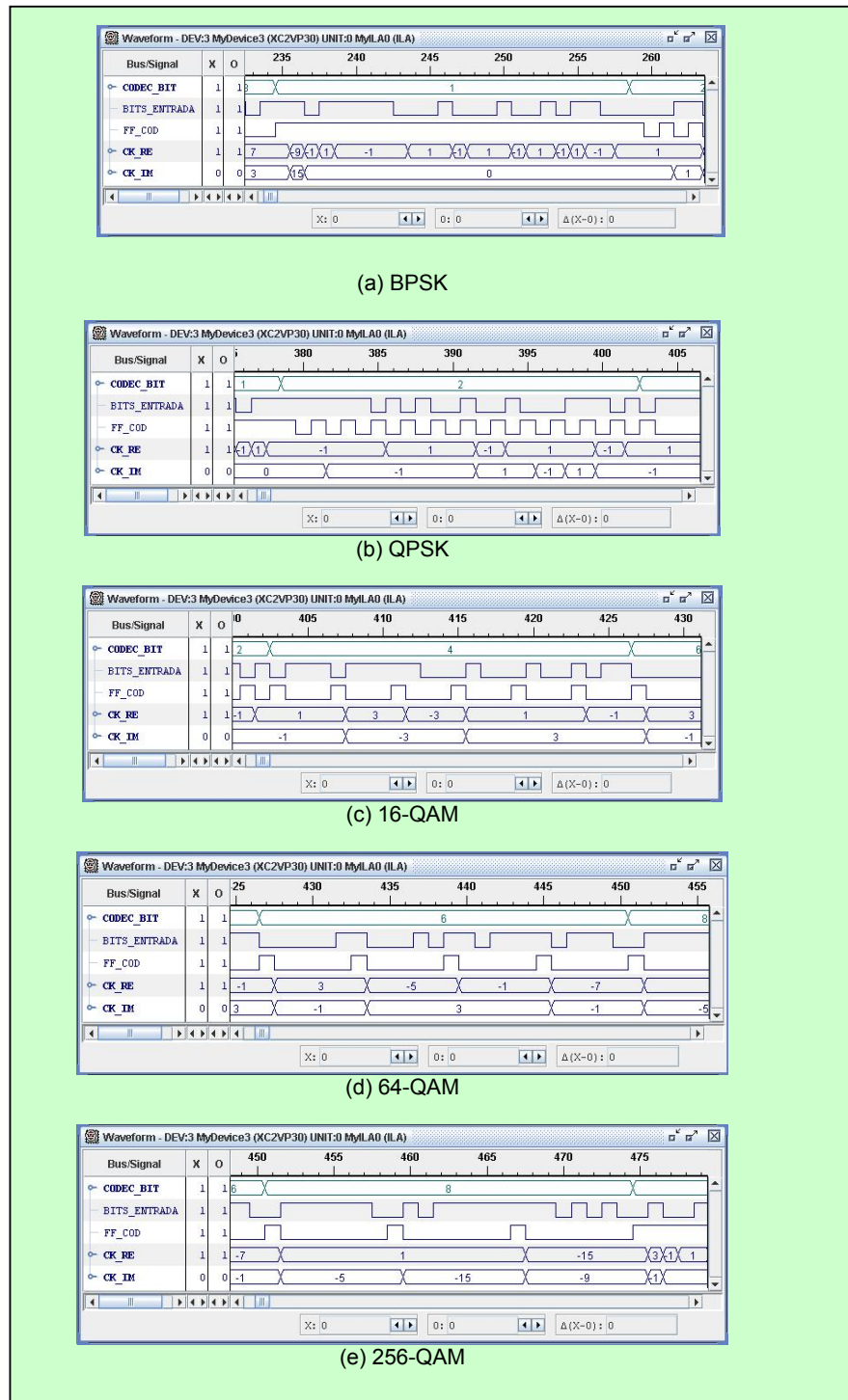


Fig. 5.15 Diagramas de tiempo de Codificación de Datos para diferentes esquemas de codificación.

Una vez comprobado que el bloque Codificador si procesa la cantidad de valores especificados en cada secuencia de codificación, es necesario validar que la modulación realizada por el codificador corresponda a las constelaciones definidas en el capítulo II. La mejor manera de visualizar esto es mediante la utilización de un listado en forma de tabla del CSP Analyzer.

Sample	CODEC_BIT	FF_COD	BITS EN...	CK_RE	CK_IM
951	8	0	0	-5	-11
952	8	0	1	-5	-11
953	8	0	1	-5	-11
954	8	0	1	-5	-11
955	1	1	1	-5	-11
956	1	1	0	-15	13
957	1	1	1	1	0
958	1	1	1	-1	0
959	1	1	1	-1	0
960	1	1	0	-1	0
961	1	1	0	1	0
962	1	1	1	1	0
963	1	1	1	-1	0
964	1	1	1	-1	0
965	1	1	1	-1	0
966	1	1	1	-1	0
967	1	1	1	-1	0
968	1	1	0	-1	0
969	1	1	0	1	0
970	1	1	1	1	0
971	1	1	0	-1	0
972	1	1	1	1	0
973	1	1	1	-1	0
974	1	1	1	-1	0
975	1	1	1	-1	0
976	1	1	1	-1	0
977	1	1	1	-1	0
978	1	1	1	-1	0
979	2	1	1	-1	0
980	2	0	0	-1	0
981	2	1	1	-1	0
982	2	0	0	-1	1
983	2	1	1	-1	1
984	2	0	0	-1	1

Fig. 5. 16 Tabla del CSP Analyzer para las pruebas de Codificación.

Si observamos la fig. 5.17 y analizamos las constelaciones correspondientes al caso de BPSK podemos notar que la regla de correspondencia del bloque si se cumple: Luego Codec_bit flancos de reloj se tiene un complejo c_k correspondiente a la constelación determinada por Codec_bit. Sin embargo, si observamos la transición de la salida c_k cuando la secuencia de codificación cambia (Se puede detectar la transición cuando Ck_IM deja de ser cero), podemos notar que existe un desfase de aparente de un flanco en la salida FF_COD. Por otra parte, si analizamos el comportamiento de la señal Codec_bit (Correspondiente a el esquema de Codificación muestreado en el bloque Codificador, es decir la entrada Codec_bit_c del controlador interno) esta debería de cambiar 2 flancos de reloj antes del primer c_k de la secuencia BPSK y no 3 flancos como se observa en la figura. Los 2 flancos se deben a que el valor Codec_bit de la secuencia j+1 es cargado cada vez que el último valor c_k de una secuencia j es mostrado, es decir la distancia en flancos entre el siguiente c_k (Correspondiente a la secuencia j+1) y la transición de Codec_bit debe ser dada por el tiempo de procesamiento, que en el caso de QPSK es

2. En definitiva, tanto en Codec_bit como en FF_COD se observa un retardo adicional de un flanco de reloj.

La explicación de esto es que tanto FF_COD como Codec_bit son síncronos con CLK.L (Ver sección 3.3) mientras que el CSP Core funciona con CLK.H. Como indicamos al principio de este capítulo el CSP Core solamente captará con exactitud las señales que son síncronas con su reloj, en el caso de esta prueba serían la entrada binaria y la salida compleja c_k .

Con todo, este aparente desfase complica un poco la comprobación mediante el CSP Analyzer. Por otra parte, si las listas muestreadas fuesen exportadas a un archivo de texto sería posible analizar estos datos mediante un programa como Excel. Así, teniendo en cuenta los retardos aparentes se podría evaluar el correcto comportamiento del bloque Codificador. Por esta razón, los datos del CSP han sido exportados en un archivo tipo ASCII, el cual ha sido visualizado mediante EXCEL. De esta manera, tomando en cuenta los retardos de las señales Codec_bit y FF_COD, es

posible observar las constelaciones resultantes para cada esquema de Codificación (fig. 5.17).

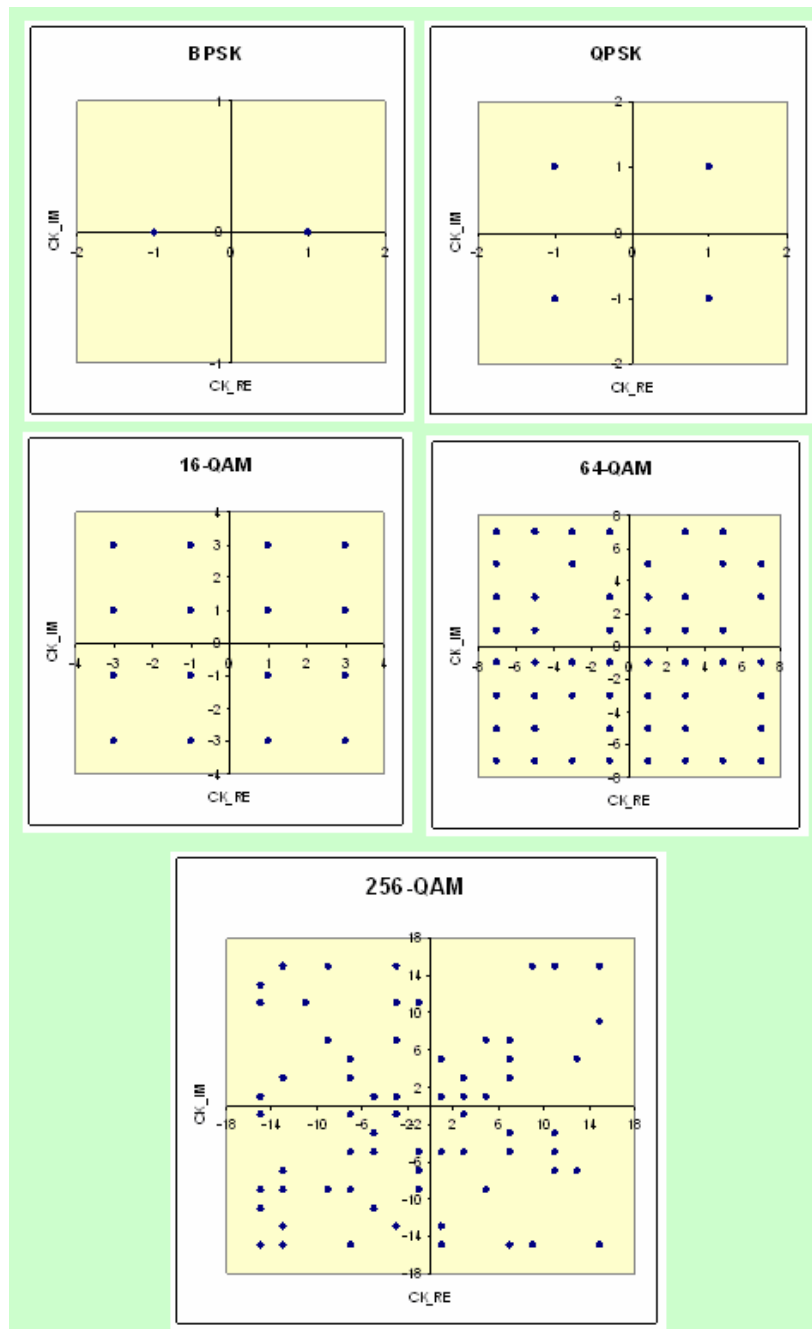


Fig. 5.17 Constelaciones obtenidas a partir de las pruebas de Codificación de datos.

El primer resultado positivo que arrojan estas pruebas es que las constelaciones observadas tienen exactamente la misma forma que las constelaciones primitivas definidas en el capítulo II (fig. 1.17 y fig. 2.1).

Sin embargo, esta prueba no es concluyente ya que existe la posibilidad de que las cadenas de bits estén siendo mapeadas a un c_k diferente que el especificado por su constelación. Por esta razón es necesario calcular los valores c_k para cada secuencia binaria de codificación y comparar los valores recuperados por el bloque. Para realizar este análisis, los datos han sido copiados a un libro de Excel diseñado para realizar el análisis descrito. Este análisis es ilustrado en la fig. 5.18.

El libro de Excel utilizado para la evaluación consta de dos hojas de cálculo, en la primera hoja se tiene el análisis de los datos recopilados por el CSP mientras que en la segunda hoja se tiene la base de datos de las constelaciones implementadas. El proceso del análisis (fig. 5.18a), consiste en primeramente detectar la secuencia (Mediante el cambio de la señal Codec_bit) y agrupar los

datos en cadenas de Codec_bit bits según sea la codificación.

	A	B	C	D	E	F	G	H	I	J	K
1	CODEC_BIT	BITS_ENTRADA	FF_COD	CK_PE	CK_IM	CAMBIO DE CODEC_BIT	INDICE DE SUBCADEN	CADENA A CODIFICA	I	Q	ES
41	2	1	1	-1	1	FALSO	20				
42	2	1	0	-1	-1	FALSO	21 11		-1		Orden ascendente Orden descendente
43	2	1	1	-1	-1	FALSO	22				(Todos)
44	2	1	0	-1	-1	FALSO	23 11		-1		(Diez mejores...) (Personalizar...)
45	4	1	1	-1	-1	VERDADERO	24				VERDADERO (Vacías) (No vacías)
46	4	1	0	-1	-1	FALSO	1 11		-1		
47	4	0	0	-1	-1	FALSO	2				
48	4	1	0	-1	-1	FALSO	3				
49	4	0	1	-1	-1	FALSO	4				
50	4	1	0	3	3	FALSO	5 0101		3	3	VERDADERO
51	4	0	0	3	3	FALSO	6				
52	4	0	0	3	3	FALSO	7				
53	4	1	1	3	3	FALSO	8				
54	4	0	0	-1	3	FALSO	9 1001		-1	3	VERDADERO
55	4	0	0	-1	3	FALSO	10				
56	4	0	1	1	2	FALSO	11				

(a) Hoja de Análisis de Datos de las pruebas de codificación.

	A	B	C	D	E	F	G	H	I	J	K	L
2	CODEC_BIT	B7	B6	B5	B4	B3	B2	B1	B0	ENTRADA	I	Q
3	1								0	0	1	0
4	1								1	1	-1	0
5	2							0	0	00	1	1
6	2							0	1	01	1	-1
7	2							1	0	10	-1	1
8	2							1	1	11	-1	-1
9	4					0	0	0	0	0000	1	1
10	4					0	0	0	1	0001	1	3
11	4					0	0	1	0	0010	1	-1
12	4					0	0	1	1	0011	1	-3
13	4					0	1	0	0	0100	3	1
14	4					0	1	0	1	0101	3	3
15	4					0	1	1	0	0110	3	-1
16	4					0	1	1	1	0111	3	-3
17	4					1	0	0	0	1000	-1	1
18	4					1	0	0	1	1001	-1	3
19	4					1	0	1	0	1010	-1	-1
20	4					1	0	1	1	1011	-1	-3
21	4					1	1	0	0	1100	-3	1
22	4					1	1	0	1	1101	-3	3
23	4					1	1	1	0	1110	-3	-1
24	4					1	1	1	1	1111	-3	-3
25	6			0	0	0	0	0	0	000000	3	3
26	6			0	0	0	0	0	1	000001	3	1
27	6			0	0	0	0	1	0	000010	3	5
28	6			0	0	0	0	1	1	000011	3	7
29	6			0	0	0	1	0	0	000100	3	-3

(b) Tabla de las Constelaciones utilizadas

Fig. 5. 18 Análisis en EXCEL de los datos reexportados en la prueba de Codificación de datos.

Estas cadenas son agrupadas justamente en las celdas que la salida del codificador presenta un nuevo valor. Así, el

siguiente paso consiste en buscar esta cadena en la base de datos de constelaciones (fig. 5.18b) y finalmente comparar los resultados obtenidos con los muestreados.

El uso de este archivo nos permite analizar datos de más de 16.000 bits obteniendo siempre resultados satisfactorios. Si repetimos el proceso exportando otras muestras desde el CSP, el resultado positivo se repite. En definitiva, se cumplen todas las expectativas planteadas y se comprueba que el bloque funciona correctamente y es capaz de soportar un cambio dinámico del valor de Codificación, siempre y cuando se termine de procesar el componente anterior.

5.2.2. Generación de las Portadoras de Piloto y Guarda

El objetivo de esta prueba es evaluar el correcto funcionamiento del bloque generador de las portadoras de piloto y guarda. La sincronización del sistema es idéntica al experimento anterior y estas pruebas también han sido realizadas con una frecuencia de reloj de 30MHz. El bloque “Simulador Control Portadoras” también es utilizado en este experimento, pero su entrada modo debe ser definida con

un valor '1'. El esquema realizado para estas pruebas puede ser visualizado en la fig. 5.19.

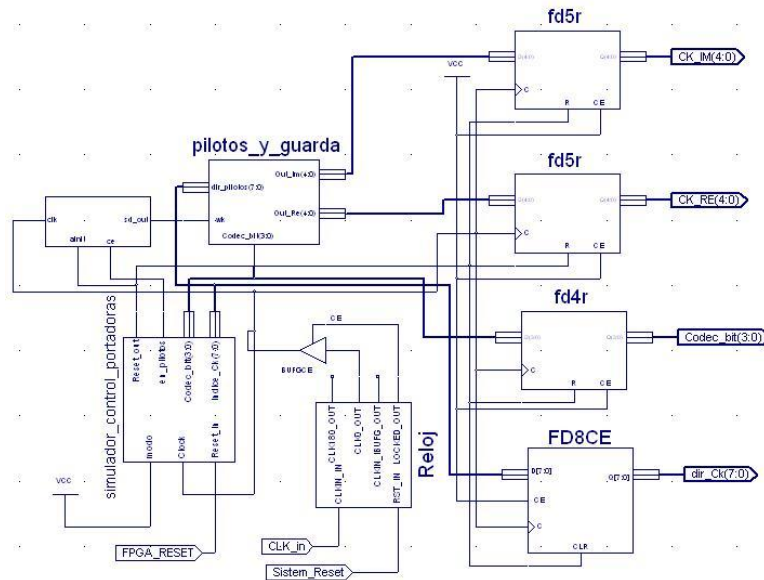


Fig. 5. 19 Esquemático de las pruebas de Pilotos y Guarda.

En estas pruebas, es necesario que la entrada modo sea definida con '1' ya que se desea "limite" del "Simulador Control Portadoras" sea definido como 255(Tabla XIII). Así, el valor Mux cambiará justamente cada 256 flancos de reloj, mientras que la salida Indice_Ck variará de 0 a 255 indefinidamente cada flanco negativo de reloj. Gracias a este esquema es posible interpretar a cada secuencia de Codificación como si correspondiera a un símbolo OFDM de 256 portadoras. Es decir, la secuencia Indice_Ck correspondería a Dir_Ck y la señal En_pilotos cumpliría las

restricciones para ser el habilitador de la secuencia ω_k ya que solamente se habilita durante la primera portadora. De esta manera es posible evaluar la generación de las portadoras pilotos y guarda en diferentes esquemas de codificación.

Dado el actual escenario las señales a evaluar son Codec_bit, Indice_Ck, la secuencia ω_k y la salida c_k . Como podemos observar en la fig. 5.19 estas señales han sido enrutadas a Flip Flops. La explicación de esto se debe a que en el proceso de selección de señales mediante el uso del CSP Inserter, algunas variables no aparecían en el listado de selección. Esto se debe a que el sintetizador de Xilinx optimiza el sistema según su criterio y en algunos casos codifica las señales internas diseñadas por el usuario, por esta razón algunas señales originales desaparecen. Al sincronizar dichas señales mediante registros, se obliga al sintetizador la utilización de las mismas y por ende no pueden ser manipuladas. Otra ventaja de los registros es que evita cualquier propagación de espurias o desincronización al CORE del CSP, en este aspecto hay

que tener en cuenta que el bloque “Pilotos y Guarda” es netamente combinatorial.

La evaluación del bloque de Pilotos y Guarda consiste en corroborar que el mismo ejecute la función desarrollada en (3.6) y la tabla III. Como se puede observar en estas especificaciones la salida del bloque variara entre 0, max y max_negado en función de ω_k y dir_Ck, mientras que los valores de max y max_negado serán función de Codec_bit. Esto significa que en cada secuencia de codificación (correspondiente a un símbolo OFDM en este experimento) se debe tener una oscilación de valores con una amplitud máxima determinada por la variable Codec_bit. Esta característica es ilustrada en la fig. 5.20a que muestra las señales c_k y Codec_bit en función del tiempo.

Por otra parte, en el diagrama de tiempo de la fig. 5.20b es posible observar que la señal CK_IM es siempre cero tal como se espera ya que la salida del bloque siempre tiene que ser BPSK. En esta última figura también es posible apreciar la variación de la señal ω_k en cada símbolo o secuencia de Codificación.

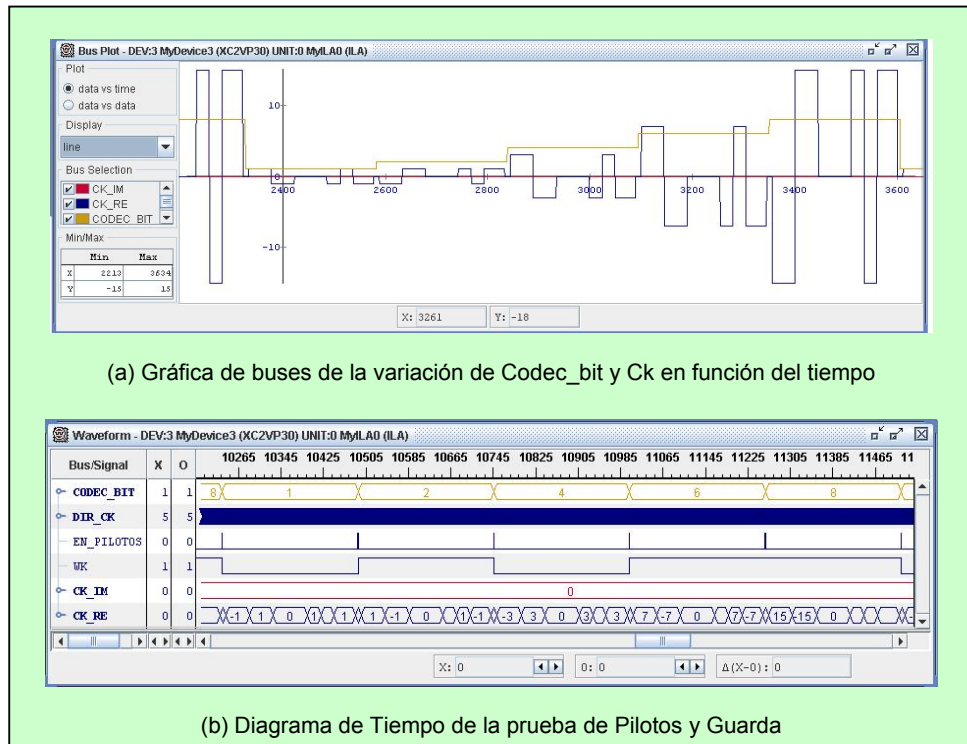


Fig. 5. 20 Pruebas de Pilotos y Guarda mediante el CSP Analyzer.

No obstante, la forma más precisa para evaluar el correcto funcionamiento del bloque es prediciendo la salida mediante el análisis de las entradas, para finalmente comparar estos valores con la salida muestreada por el CSP. Como hemos visto el CSP Analyzer no permite la realización de este tipo de análisis de manera automática. Por esta razón, al igual que en el anterior experimento, se ha procedido a exportar los datos muestreados por el CSP a un archivo tipo ASCII. Así, mediante la utilización de una herramienta como Excel,

no solo es posible realizar el análisis descrito sino también, mediante el uso de filtros, mostrar la constelaciones resultantes en función de los diferentes valores de Codec_bit (fig. 5.21).

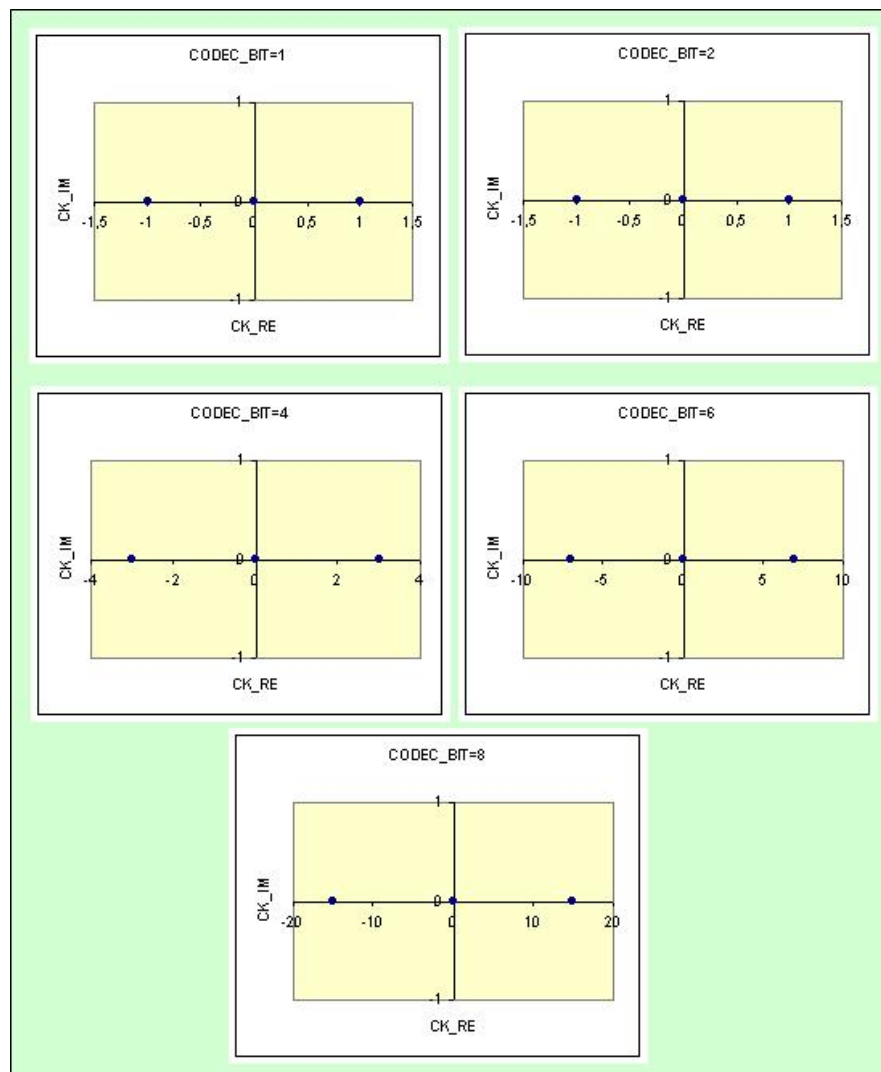


Fig. 5. 21 Constelaciones resultantes del bloque Pilotos y Guarda.

Como podemos observar en la fig. 5.21 las constelaciones resultantes se apegan a lo especificado en el capítulo III. Es decir, siempre se tiene una codificación BPSK o nula y existe una variación en el valor máximo en función de Codec_bit.

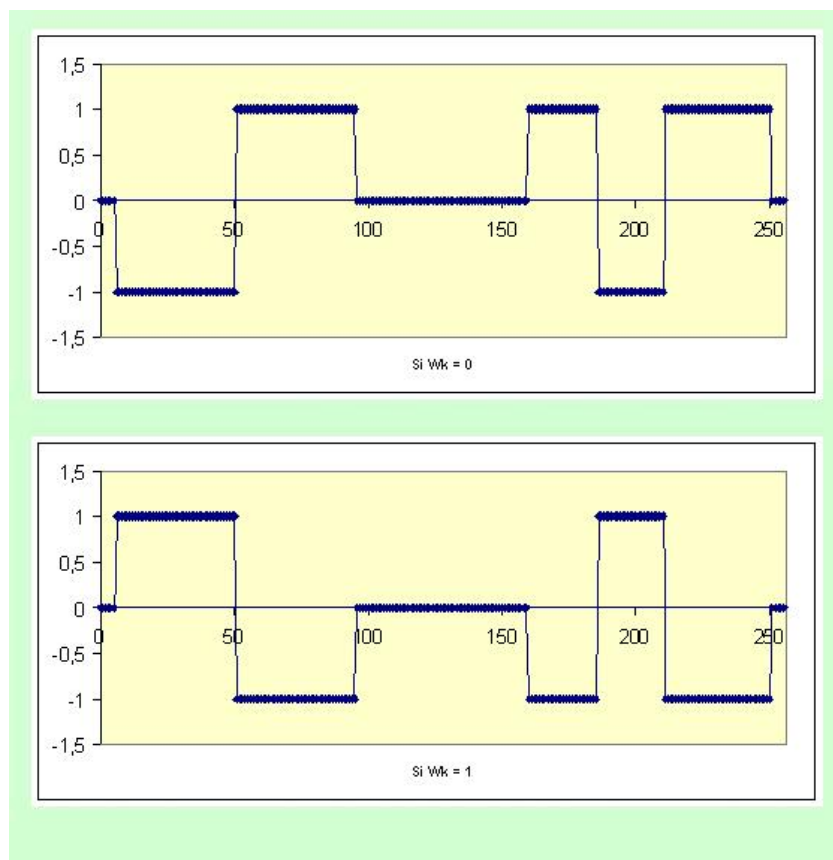


Fig. 5.22 Salida del bloque Pilotos y Guarda en función de ω_k

Otro gráfico de evaluación que se puede generar gracias a los filtros en Excel, corresponde a la respuesta de la salida

Ck_Re en función de $|\omega_k|$. Debido a la definición de los pilotos, el gráfico esperado para un valor ω_k de '1' debe ser el mismo que el correspondiente a '0' pero invertido con respecto al eje horizontal. Tal como se puede observar en la fig. 5.22, este comportamiento se apega a lo obtenido en las pruebas.

Por otra parte, para poder predecir las señales resultantes del bloque y compararlas con las recuperadas, se ha hecho uso de un libro de Excel con dos hojas de cálculo. En la primera hoja se tiene el análisis de los datos recopilados por el CSP mientras que en la segunda hoja se tiene la base de datos de los valores resultantes en función de (3.6) y la tabla III. Así, las señales de entrada ω_k , dir_Ck y Codec_bit (fig. 5.23a) son correlacionadas con las tablas de la regla de Correspondencia (fig. 5.23b) para obtener la señal CK_RE recuperada (Columna G en la fig. 5.23a). Posteriormente, este valor es comparado con el componente muestreado por el CSP (Columna D en la fig. 5.23a) para la evaluación del bloque.

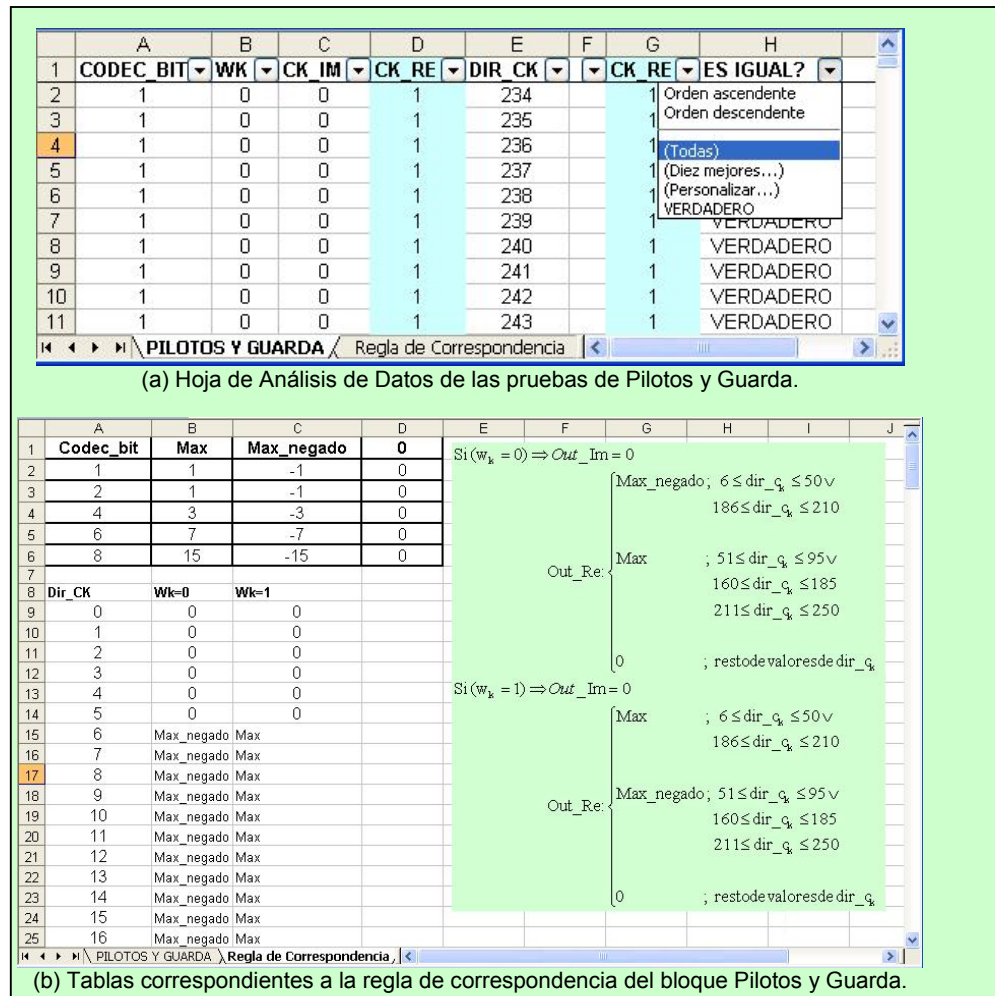


Fig. 5. 23 Análisis en EXCEL en las pruebas de Pilotos y Guarda.

De esta manera, gracias a la utilización de este archivo, es posible evaluar el correcto funcionamiento del bloque para más de 16.000 transiciones en la entrada. Además, si repetimos este procedimiento de manera recursiva podemos observar los mismos resultados. En definitiva, en base a la evidencia se concluye que el correcto funcionamiento del bloque generador de las componentes de Piloto y Guarda.

5.2.3. Generación del Símbolo de Training

El objetivo de esta prueba es evaluar el correcto funcionamiento del bloque de Training. Debido a la similitud del experimento anterior, tanto la sincronización como la configuración del bloque de control “Simulador Control Portadoras” son idénticas. La fig. 5.24 ilustra el esquema utilizado en estas pruebas.

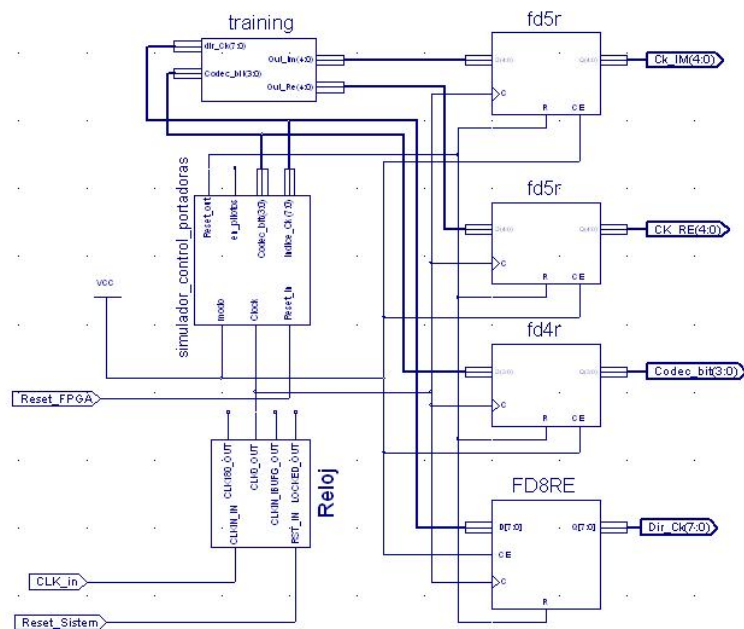


Fig. 5. 24 Esquemático de las pruebas del Training.

Puesto que la configuración del bloque “Simulador Control Portadoras” es la misma que la anterior, la secuencia `Indice_Ck` también corresponderá a `Dir_Ck` y repetirá de

manera periódica la secuencia de enteros entre 0 y 255. Es decir, en estas pruebas es posible evaluar la generación de las portadoras de Training en diferentes esquemas de codificación, donde cada secuencia de codificación corresponderá a un Símbolo de Training.

Dado el actual escenario las señales a evaluar son Dir_Ck, Codec_bit y la salida Ck. De manera similar al experimento anterior, las señales a evaluar han sido conectadas a Flip Flops. La explicación de esto es exactamente la misma que la expuesta anteriormente. La evaluación del bloque de Training consiste en corroborar que el mismo ejecute las especificaciones desarrolladas en la sección 3.2. Así, se debe mostrar la estructura del preámbulo de la fig. 1.9 multiplicado por el valor Max (tabla III) en función del tipo del tipo de codificación. Esto implica que en este experimento se debe dar una oscilación similar al experimento anterior. En este caso hay que tener en cuenta que la componente imaginaria de la salida c_k ya no debe ser cero debido a que la codificación del preámbulo utilizado es QPSK y no BPSK. La variación de la salida c_k y Codec_bit es ilustrada en la fig. 5.25.

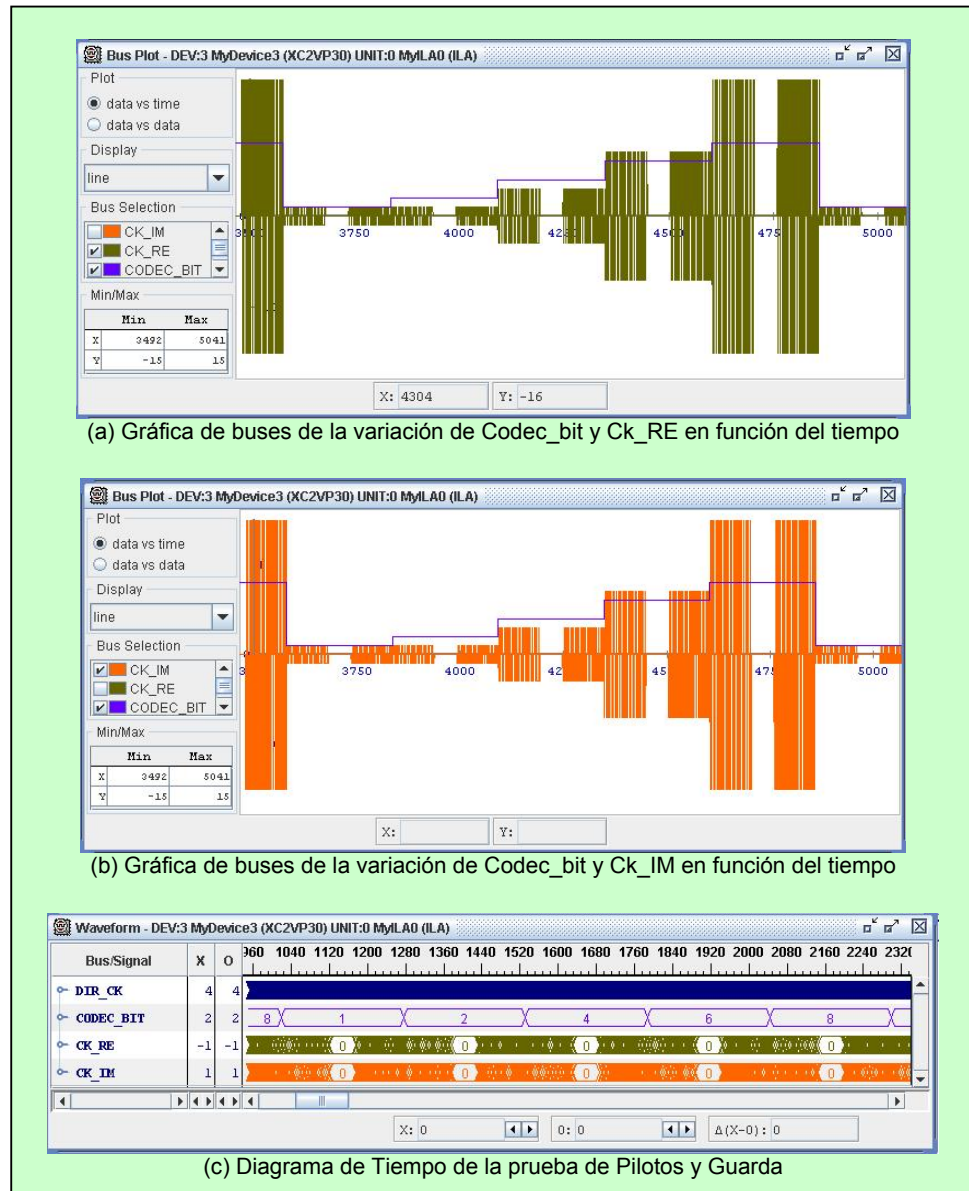


Fig. 5.25 Pruebas del Training mediante el CSP Analyzer.

Por otro lado, al igual que en los dos experimentos anteriores, la evaluación final se ha realizado exportando los datos muestreados por el CSP a un archivo tipo ASCII, para

una posterior evaluación mediante Excel. Así, gracias a la utilización de filtros en Excel, es posible observar las constelaciones resultantes del bloque de Training en función de la variable Codec_bit (fig. 5.26).

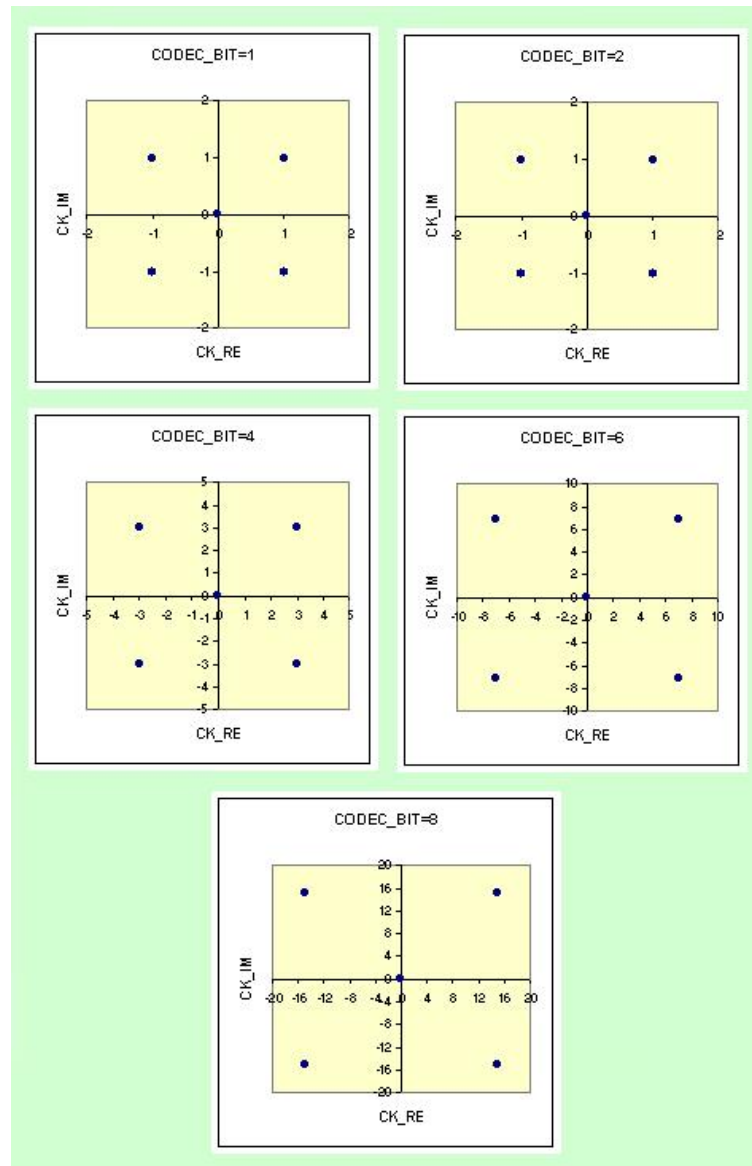


Fig. 5. 26 Constelaciones resultantes del bloque Training.

Como se evidencia en dicho gráfico las constelaciones resultantes se apegan a sus especificaciones. Es decir, siempre se tiene una codificación QPSK o nula y existe una variación en el valor máximo en función de Codec_bit.

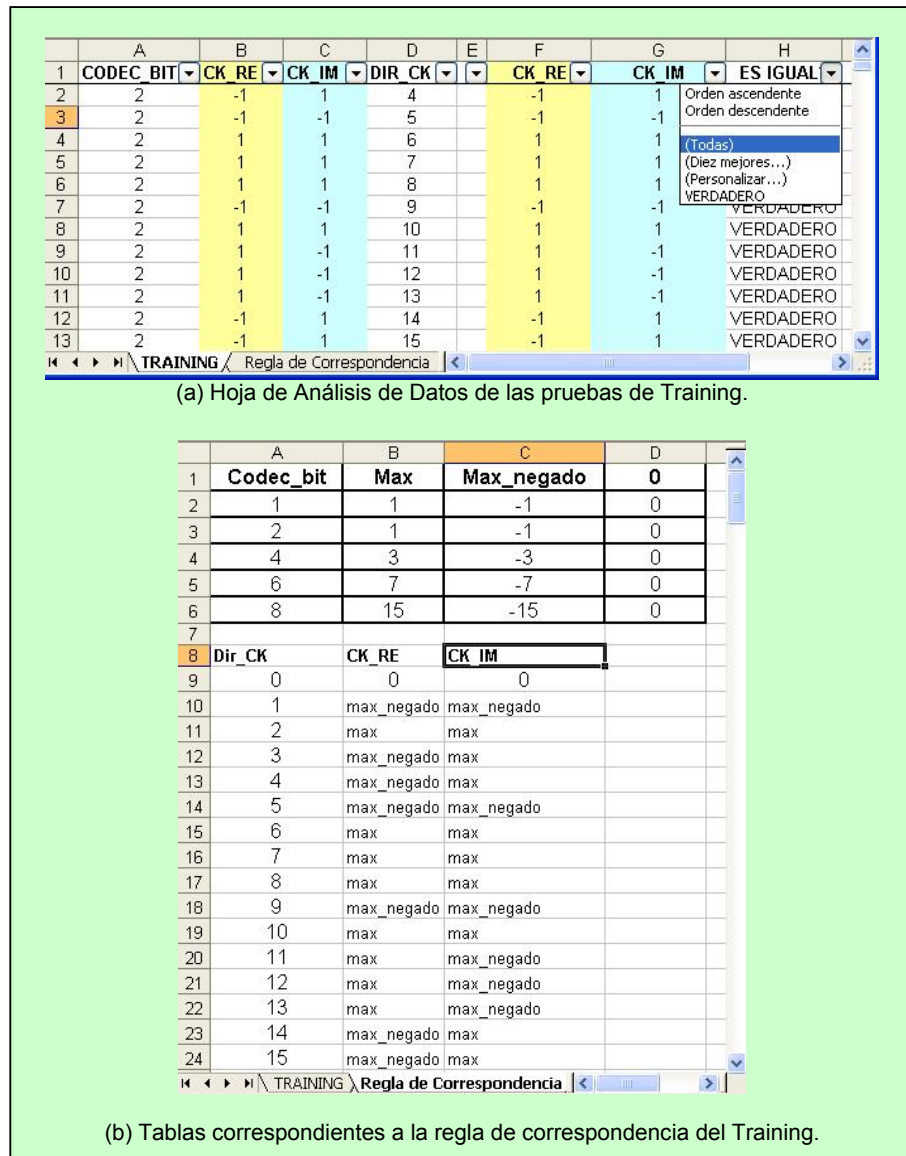


Fig. 5. 27 Análisis en EXCEL de los datos exportados en las pruebas de Training.

De manera idéntica a los dos experimentos anteriores, la prueba final del experimento radica en predecir los valores resultantes en función de la entrada y compararlos con los obtenidos. Por eso la ejecución de esta prueba sigue la misma metodología, es decir se utiliza un archivo en Excel con dos hojas de cálculo: la primera para el análisis de datos y la segunda como base de datos de la regla de correspondencia a evaluar. Así, una vez más se tiene que los análisis realizados de manera repetitiva mediante el uso de este archivo han sido satisfactorios. En definitiva, se demuestra la correcta operación del bloque de Training.

5.2.4. Generación del Bloque OFDM en el dominio de la Frecuencia

El objetivo de esta pruebas es observar el proceso de conformación de los símbolos y bloques OFDM en el dominio de la frecuencia. Para esto se analizará las señales de control y las constelaciones a transmitirse en la entrada de la IFFT. Específicamente las señales a analizarse son MUX, En_Datos, En_Pilotos, Dir_Ck (provenientes del Controlador contador TX), EN_entramador (proveniente del

bloque de inserción de Guarda) y la constelación C_k a transmitirse (proveniente del bloque multiplicador de escala).

Para la realización de este experimento se ha utilizado prácticamente el mismo sistema implementado en el proyecto final, la diferencia radica en que no se incluye al receptor en la implementación. Los parámetros definidos en el bloque Settings para el actual experimento son especificados en la tabla XV. Además, el parámetro div en el bloque de Clocking fue definido internamente con un valor de 8. Para la realización de estas pruebas se utilizó una frecuencia de operación de 30MHz.

Tabla XV
Parámetros del bloque Settings definidos para los experimentos de generación de los bloques OFDM en el tiempo y la frecuencia

Nombre del Parámetro	Valor del Parámetro
Dim_guarda	64
NFFT	255
Longitud_training	1
Logintud_bloque	3
Codec_bit	4

Como se explico en el capitulo III, la conformación de los símbolos OFDM se logra mediante la utilización del contador Dir_Ck, la multiplexación del banco de selectores y las

señales habilitadoras. La señal Dir_Ck se encarga de definir el índice de la portadora a modular, mientras que la multiplexación escoge el tipo de portadora a ser muestreada a la entrada de la IFFT. De esta forma, se espera que la señal Dir_Ck genere una secuencia desde 0 hasta 255 indefinidamente, mientras que la señal Mux debe generar un 1 durante los símbolos de training y oscilar entre 0 y 2 en los símbolos de datos. Esto se ilustra en la fig. 5.28 que muestra el comportamiento de las señales Mux y Dir_Ck en función del tiempo.

Como podemos observar, la señal Mux tiene un '1' durante un periodo de Dir_Ck mientras que oscila entre 0 y 2 para las 2 oscilaciones siguientes. Este comportamiento se repite indefinidamente y se ajusta a las especificaciones del bloque definidas en estas pruebas (longitud_bloque=3 y longitud_training=1).

Por otra parte, si analizamos la señal Dir_Ck podremos notar que efectivamente genera una secuencia entre 0 y 255, sin embargo también se observa que la señal se

congela justamente en un valor determinado y su periodo no corresponde a 256 periodos de muestreos sino a 320.

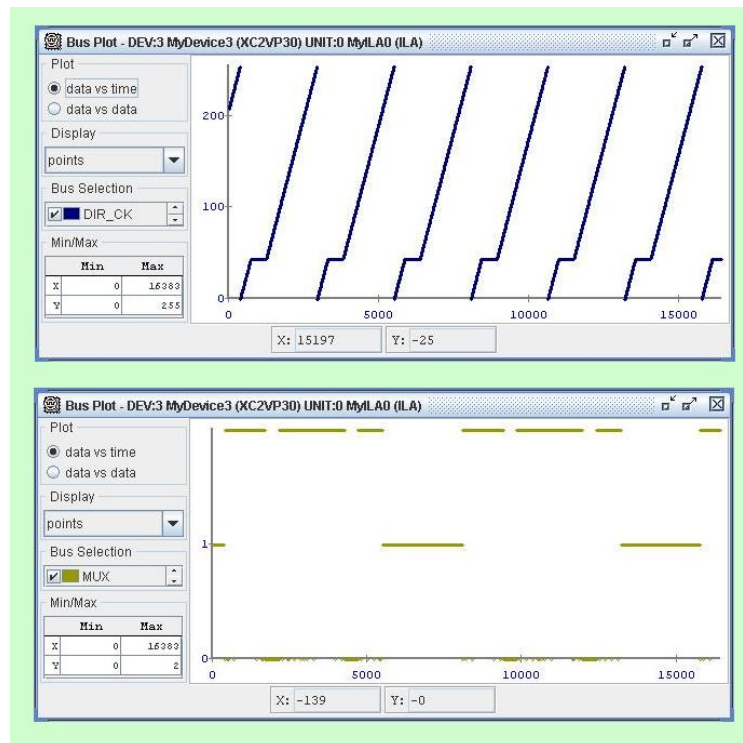


Fig. 5. 28 Gráfica de buses Dir_Ck y Mux en función del tiempo.

La justificación de esto es la acción de la señal `en_entrador` que detiene la generación de los símbolos en frecuencia durante `dim_guarda` flancos de reloj para que el prefijo cíclico sea insertado en el tiempo. De esta manera, el periodo de `Dir_Ck` debe ser $NFFT + dim_guarda$ que para el caso de nuestras pruebas corresponde a 320. Asimismo, el efecto de `en_entrador` no solo debe de reflejarse en

Dir_Ck sino también en el resto de señales escogidas para la evaluación de estas pruebas. Esto se debe a que todas estas señales corresponden al proceso de generación del símbolo en frecuencia. Este comportamiento se ilustra en la fig. 5.29 y fig. 5.30 para el caso de C_k y los otros dos habilitadores respectivamente.

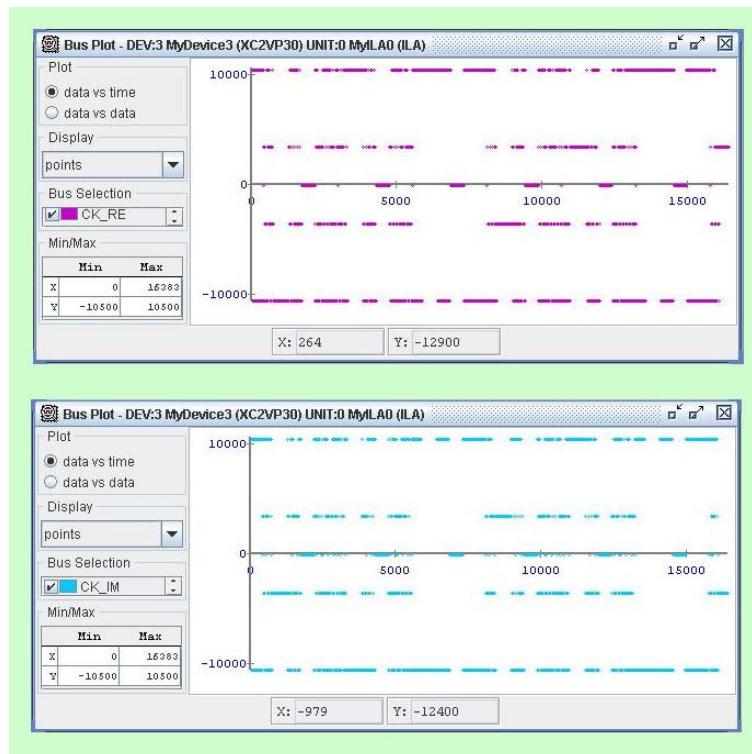


Fig. 5. 29 Componentes de la señal C_k procesada en el tiempo.

Al igual que el selector Mux y el contador Dir_Ck los habilitadores en_pilotos y en_datos son fundamentales para

la formación de los símbolos de datos. El habilitador en_pilotos tiene la función de habilitar la secuencia ω_k al principio de cada símbolo de datos. Esto significa, que debemos esperar observar a en_pilotos como un pulso cuyo valor de subida se presente mientras Dir_CK sea cero durante un símbolo de datos. Por otra parte, la señal en_Datos tiene la función de habilitar al codificador durante Codec_bit flancos de reloj cada vez que Dir_Ck corresponda a una portadora de datos dentro de un símbolo de datos. De manera similar, se debe observar que en_Datos debe oscilar entre '1' y '0' solamente durante los símbolos de datos, ya que durante el training siempre es '0'.

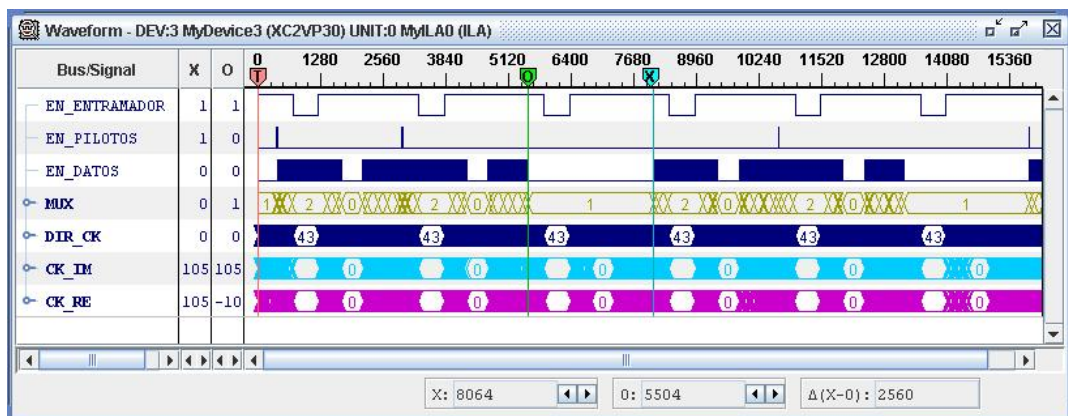


Fig. 5.30 Diagrama de tiempo de las pruebas en frecuencia.

Acorde a lo observado en el diagrama de tiempo de la Fig. 5.30, parecería posible expresar que todo lo expuesto sobre los habilitadores en_datos y $en_pilotos$ se cumple. El primer aspecto que salta a la vista es la correlación entre el selector MUX y estos habilitadores. Como MUX tiene un valor de 1 cuando se está procesando el training, ambos habilitadores deben tener un valor de 0. Por otro lado, justamente en el momento en que MUX comienza a oscilar entre 0 y 2 $en_Pilotos$ presenta un '1' siempre y cuando Dir_Ck sea 0. De manera similar, en_Datos presenta pulsos de subida solamente cuando MUX corresponde a datos.

Sin embargo, si observamos en detalle las señales analizadas, se puede notar que existe un desfase de un flanco en retraso para que la señal MUX corresponda al comportamiento especificado anteriormente para los habilitadores. Por otra parte, la salida C_k también presenta un desfase, pero en este caso exactamente igual a un periodo de muestreo.

Sample	DIR_CK	MUX	CK_RE	CK_IM
8056	255	1	-10500	-10500
8057	255	1	-10500	-10500
8058	255	1	-10500	-10500
8059	255	1	-10500	-10500
8060	255	1	-10500	-10500
8061	255	1	-10500	-10500
8062	255	1	-10500	-10500
8063	0	1	10500	10500
8064	0	0	10500	10500
8065	0	0	10500	10500
8066	0	0	10500	10500
8067	0	0	10500	10500
8068	0	0	10500	10500
8069	0	0	10500	10500
8070	0	0	10500	10500
8071	1	0	0	0
8072	1	2	0	0
8073	1	2	0	0
8074	1	2	0	0
8075	1	2	0	0
8076	1	2	0	0
8077	1	2	0	0
8078	1	2	0	0

Fig. 5. 31 Captura de la ventana Listing de las señales Dir_Ck, Mux y Ck.

El aparente desfase de la señal MUX se explica al hecho de que esta no es mostrada en el listado del CSP Inseter y, al igual que en anteriores experimentos, se ha utilizado un registro para poder observarla. Es decir, lo que se ve en el CSP Analyzer en realidad no corresponde a MUX sino a una versión de esta misma señal desfasada un flanco de reloj. Por otra parte, el retraso de la salida C_k se justifica a la acción del registro interno en el bloque de escalamiento. Como se puede observar en la fig. 3.2 este bloque es

síncrono con Simbol_rate por la cual se explica el retraso correspondiente a un periodo de muestreo.

Con las pruebas realizadas hasta el momento, se ha podido demostrar de forma general el correcto funcionamiento del contador Dir_Ck , el selector MUX y la correlación de este con los habilitadores en_Datos y en_pilotos . Sin embargo, la prueba definitiva consiste en determinar el comportamiento de estas señales discriminando que portadoras corresponden a datos y cuales a piloto o guarda dentro de un símbolo de datos (En el Training solo basta demostrar que se genere la secuencia Dir_Ck y que MUX sea síncrono con esta, lo cual ya se demostró). Por esta razón, como en casos anteriores, se ha optado por exportar los datos y analizarlos en EXCEL tomando en cuenta los retrasos aparentes especificados.

Para las pruebas en Excel primeramente las señales MUX y C_k han sido sincronizadas. Además, se ha añadido una columna que evalúa si Dir_Ck corresponde a una portadora de Datos en función de la tabla I y (3.1). Gracias a esta variable lógica y al uso de filtros en Excel es posible

determinar de forma exhaustiva el correcto funcionamiento del sistema en el caso de los símbolos de Datos.

Así, si por ejemplo filtramos los valores correspondientes a datos en la columna MUX SINCRONIZADO, se debe tener un valor verdadero en la variable Dir_Ck=Dato para todos los casos. De manera análoga si MUX SINCRONIZADO corresponde a pilotos o guarda, esta variable lógica siempre debe ser falsa.

(a) Filtro MUX SINCRONIZADO=2 (Datos)

	A	B	C	D	E	F	G	H	I	J	K
1	CK_MA	CK_DE	MUX		MUX SINCRONIZADO	CK_RE SINCRONIZADO	CK_IM SINCRONIZADO	DIR_CK	EN_DATOS	EN_PILOTOS	DIR_Ck = DATO?
1696	0	0	0		2	-10500	-3500	1	1		VERDADERO
1697	0	0	2		2	-10500	-3500	1	1		VERDADERO
1698	0	0	2		2	-10500	-3500	1	1		VERDADERO
1699	0	0	2		2	-10500	-3500	1	1		VERDADERO
1700	0	0	2		2	-10500	-3500	1	0		VERDADERO
1701	0	0	2		2	-10500	-3500	1	0		VERDADERO
1702	0	0	2		2	-10500	-3500	1	0	0	VERDADERO
1703	0	0	2		2	-10500	-3500	1	0	0	VERDADERO
1704	-3500	-10500	2		2	3500	-10500	2	1	0	VERDADERO
1705	-3500	-10500	2		2	3500	-10500	2	1	0	VERDADERO
1706	-3500	-10500	2		2	3500	-10500	2	1	0	VERDADERO
1707	-3500	-10500	2		2	3500	-10500	2	1	0	VERDADERO
1708	-3500	-10500	2		2	3500	-10500	2	0	0	VERDADERO

(b) Filtro MUX SINCRONIZADO=0 (Pilotos o Guarda)

	A	B	C	D	E	F	G	H	I	J	K
1	CK_MA	CK_DE	MUX		MUX SINCRONIZADO	CK_RE SINCRONIZADO	CK_IM SINCRONIZADO	DIR_CK	EN_DATOS	EN_PILOTOS	DIR_Ck = DATO?
1688	10500	10500	1		0	0	0	0	0		VERDADERO
1689	10500	10500	0		0	0	0	0	0		VERDADERO
1690	10500	10500	0		0	0	0	0	0		VERDADERO
1691	10500	10500	0		0	0	0	0	0		VERDADERO
1692	10500	10500	0		0	0	0	0	0		VERDADERO
1693	10500	10500	0		0	0	0	0	0		VERDADERO
1694	10500	10500	0		0	0	0	0	0		VERDADERO
1695	10500	10500	0		0	0	0	0	0	1	FALSO
1792	-3500	-10500	2		0	10500	0	13	0	0	FALSO
1793	-3500	-10500	0		0	10500	0	13	0	0	FALSO
1794	-3500	-10500	0		0	10500	0	13	0	0	FALSO
1795	-3500	-10500	0		0	10500	0	13	0	0	FALSO
1796	-3500	-10500	0		0	10500	0	13	0	0	FALSO

Fig. 5. 32 Evaluación del selector de MUX durante un símbolo de datos, mediante filtros en Excel.

El caso de En_Datos es muy similar al anterior. En esta ocasión el factor a verificar es que si En_datos es '1', MUX debe ser 2 y Dir_Ck=Datos siempre debe ser Verdadero. Obviamente, si la prueba anterior fue satisfactoria, bastaría simplemente con evaluar que MUX corresponda a datos.

Finalmente, para el caso de En_pilotos, se debe validar que si esta señal esta habilitada, el valor de Dir_Ck debe ser cero y MUX no puede ser 1.

(a) Filtro En_Datos=1

	A	B	C	D	E	F	G	H	I	J	K
1	CK_IM	CK_RE	MUX	MUX	CK_RE	CK_IM	DIR_CK	EN_DATOS	EN_PILOTOS	DIR_Ck = DATO?	
1696	0	0	0	Orden ascendente	-10500	-3500	1	1	0	VERDADERO	
1697	0	0	2	Orden descendente	-10500	-3500	1	1	0	VERDADERO	
1698	0	0	2	(Todas)	-10500	-3500	1	1	0	VERDADERO	
1699	0	0	2	(Diez mejores...)	-10500	-3500	1	1	0	VERDADERO	
1704	-3500	-10500	2	(Personalizar...)	3500	-10500	2	1	0	VERDADERO	
1705	-3500	-10500	2	2	3500	-10500	2	1	0	VERDADERO	
1706	-3500	-10500	2	2	3500	-10500	2	1	0	VERDADERO	
1707	-3500	-10500	2	2	3500	-10500	2	1	0	VERDADERO	
1712	-10500	3500	2	2	3500	-3500	3	1	0	VERDADERO	
1713	-10500	3500	2	2	3500	-3500	3	1	0	VERDADERO	
1714	-10500	3500	2	2	3500	-3500	3	1	0	VERDADERO	

(b) Filtro En_pilotos=1

	A	B	C	D	E	F	G	H	I	J	K
1	CK_IM	CK_RE	MUX	MUX	CK_RE	CK_IM	DIR_CK	EN_DATOS	EN_PILOTOS	DIR_Ck = DATO?	
1688	10500	10500	1	0	0	0	0	0	1	FALSO	
1689	10500	10500	0	0	0	0	0	0	1	FALSO	
1690	10500	10500	0	0	0	0	0	0	1	FALSO	
1691	10500	10500	0	0	0	0	0	0	1	FALSO	
1692	10500	10500	0	0	0	0	0	0	1	FALSO	
1693	10500	10500	0	0	0	0	0	0	1	FALSO	
1694	10500	10500	0	0	0	0	0	0	1	FALSO	
1695	10500	10500	0	0	0	0	0	0	1	FALSO	
4248	10500	-10500	2	0	0	0	0	0	1	FALSO	
4249	10500	-10500	0	0	0	0	0	0	1	FALSO	
4250	10500	-10500	0	0	0	0	0	0	1	FALSO	

Fig. 5. 33 Evaluación de los habilitadores En_Datos y En_pilotos mediante filtros en Excel.

Así, tenemos que mediante el uso de este archivo, todas estas condiciones se han cumplido para más de 2000 periodos de muestreo. Además, si se repite de manera recurrente la prueba el resultado es el mismo, por lo cual claramente se puede concluir que el sistema implementado funciona conforme a lo diseñado.

Por otra parte, este archivo no solo nos permite evaluar la correcta multiplexación, también nos es posible evaluar las constelaciones resultantes. Mediante los filtros implementados, es viable observar las constelaciones enrutadas en la entrada de la IFFT. Así, por ejemplo para el caso de que `Codec_bit` sea 4, se esperaría una constelación tipo 16-QAM para las portadoras de datos (`MUX=2`), mientras que para el caso del training (`MUX=1`) y los pilotos o guarda (`MUX=1`) deben ser siempre QPSK y BPSK respectivamente.

En este aspecto es importante recordar que la resultante de las portadoras de Pilotos y Training a la salida del bloque multiplicador, no debe variar en función de la variable `Codec_bit`. Es decir, el aspecto de estas constelaciones es

siempre como el especificado en la fig. 5.34 sin importar Codec_bit.

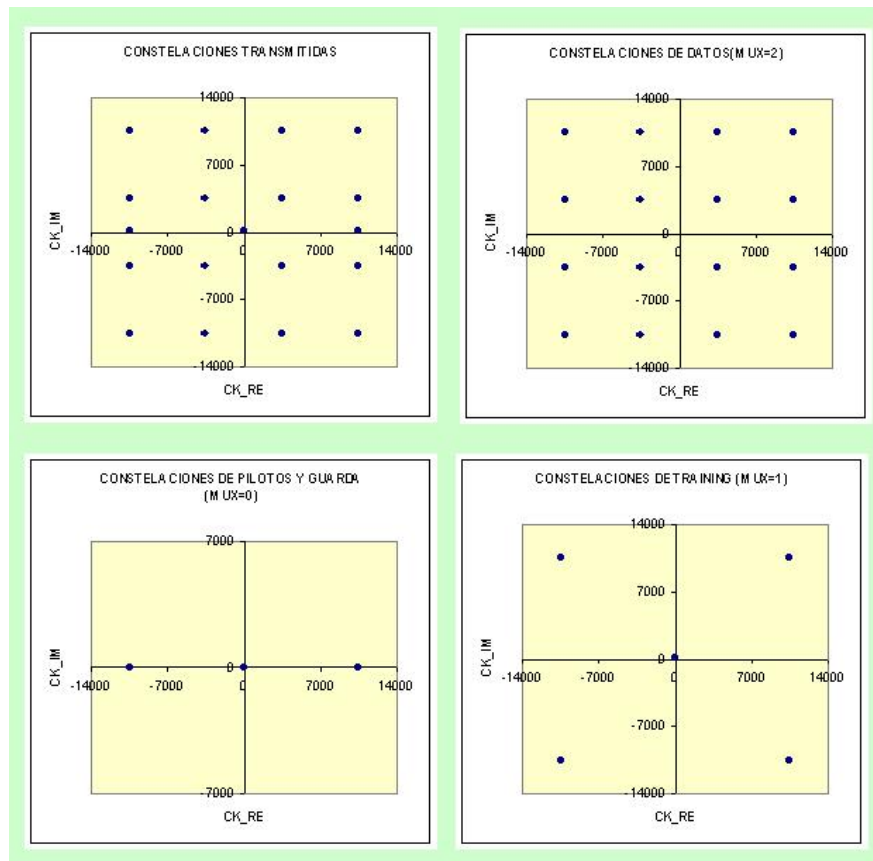


Fig. 5.34 Constelaciones transmitidas en función del selector MUX.

5.2.5. Generación del Bloque OFDM en el dominio del Tiempo

El objetivo de estas pruebas es observar la conformación de los símbolos y bloques en el dominio del tiempo. Debido a que la transformación del símbolo en frecuencia al símbolo útil en el tiempo es realizada automáticamente por la IFFT,

en este nivel no existe mucho que corroborar. Por esta razón, el aspecto más importante a analizar es la inserción del prefijo cíclico mediante el bloque de Inserción de Guarda.

Debido a esto, las señales a evaluar en este experimento son las correspondientes al símbolo útil (X_n_{Re} , X_n_{Im}), al símbolo extendido ($I[n]$, $Q[n]$) y sus correspondientes índices (Dir_{Xn} , Dir_{RAM}). Los datos del símbolo útil son mapeados a partir de las salidas de la IFFT, mientras que los del símbolo extendido a partir del controlador y el registro de salida en el bloque de inserción de Guarda. Por otra parte, debido a que las señales Inicio_Símbolo y En_entrador del bloque de Inserción de Guarda determinan el origen del símbolo útil y la duración del prefijo cíclico respectivamente, también han sido seleccionadas para estas pruebas.

El esquema utilizado en este experimento es exactamente igual al anterior. Es decir, se utiliza el mismo diseño del proyecto final pero no se incluye al receptor. De igual forma, los parámetros de los bloques de Settings y Clocking así como la frecuencia de operación son idénticos.

Como se vio en el capítulo III, podríamos dividir el procesamiento de un símbolo OFDM por parte del bloque de Inserción de Guarda en tres partes:

1. Captura del símbolo útil (Se esta a la espera que $Dir_Xn=255$).
2. Muestreo del prefijo cíclico (En esta etapa la formación de los símbolos en frecuencia es detenida por la señal $En_estramador$).
3. Muestreo del símbolo útil para la culminación del símbolo extendido (Paralelamente se captura el siguiente símbolo útil).

De este proceso, deberíamos esperar que una vez que Dir_Ck genere un ciclo completo (Desde 0 a 255), dicha señal quedase “congelada” en dicho estatus mientras el prefijo cíclico sea muestreado. Este comportamiento se explica al hecho de que $en_entramador$ detiene a la FFT durante la inserción del prefijo. Por otro lado, la señal Dir_RAM debe ser exactamente igual a Dir_Ck mientras se esta capturando el símbolo útil, pero debe de mostrar la sucesión entre $(NFFT -G)$ y $NFFT-1$ mientras se esta

mostrando el prefijo cíclico. Este comportamiento se justifica ya que en el proceso de captura el componente Xn debe de ser almacenado en la dirección n, mientras que en la inserción del prefijo se deben mostrar de manera ascendente los últimos Dim_guarda componentes almacenados en la RAM.

Si observamos el diagrama de buses de la fig. 5.35 podemos notar que el comportamiento obtenido es exactamente al esperado para las señales Dir_Xn y Dir_Ck.

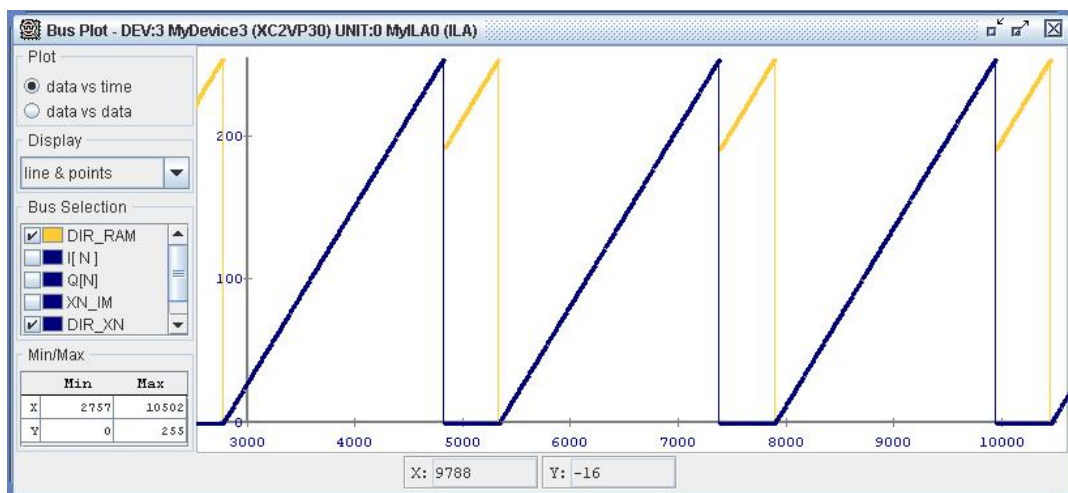


Fig. 5. 35 Diagrama de buses de Dir_Ck y Dir_Xn

En base de esto, se puede concluir que la sincronía entre la pausa en el procesamiento del símbolo útil y la inserción de

la guarda es realizada satisfactoriamente. Sin embargo, si se desea ilustrar con más detalle esto, se puede analizar el comportamiento de En_entrador. Si analizamos el diagrama de tiempo de la fig. 5.36 podremos comprobar que efectivamente cuando Dir_Xn llega a 255 En_entrador se deshabilita y dura justamente dim_guarda periodos de muestreo.

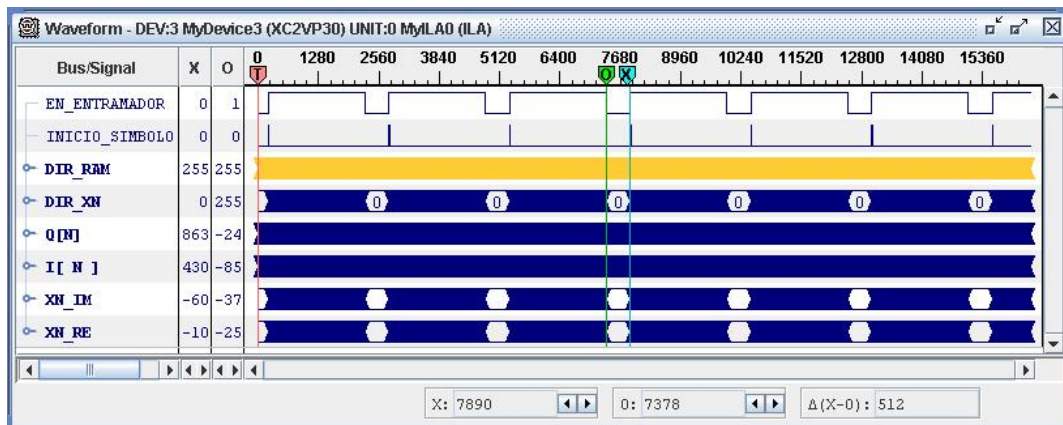


Fig. 5.36 Diagrama de tiempo de las pruebas de Bloque y Símbolo en el tiempo.

Sin embargo, es necesario tener en cuenta que en realidad la señal En_entrador no se deshabilita en el mismo instante que el prefijo cíclico es insertado. Esto se debe a que la señal habilitadora debe de variar instantes antes para que se pueda detener o reanudar la generación del símbolo útil cuando el prefijo es insertado. Por esta razón, habría que tener en cuenta este aparente retardo si se quiere realizar

cualquier análisis. Si observamos la fig. 5.37 podemos determinar que para las pruebas realizadas el retardo corresponde exactamente a 8 flancos. Por otra parte, tanto en la fig. 5.36 como en la fig. 5.37 se puede observar el correcto comportamiento de la señal Inicio_simbolo, ya que esta se habilita justamente cuando el símbolo útil debe ser mostrado en las salidas I[n] y Q[n].

<->	Sample	DIR_XN	EN_ENTRAMADOR	INICIO SIMBOLO
	2767	0	0	0
	2768	0	0	0
	2769	0	0	0
	2770	0	0	0
0	2771	0	1	0
	2772	0	1	0
	2773	0	1	0
	2774	0	1	0
	2775	0	1	0
	2776	0	1	0
	2777	0	1	0
	2778	0	1	0
X	2779	1	1	1
	2780	1	1	1
	2781	1	1	1
	2782	1	1	1
	2783	1	1	1
	2784	1	1	1
	2785	1	1	1
	2786	1	1	1
	2787	2	1	0
	2788	2	1	0
	2789	2	1	0
	2790	2	1	0
	2791	2	1	0

X: 2779 0: 2771 Δ(X-0): 8

Fig. 5.37 Tablas del CSP Analyzer para las pruebas de generación de símbolo y bloque en el tiempo.

A pesar de todo esto, la única prueba concluyente consiste en comparar los datos generados por la IFFT y los mostrados por el bloque de inserción de guarda. Para realizar este tipo de análisis definamos la variable N de la fig. 5.38 como el número de muestreos transcurridos en un instante t . Si tomamos en cuenta el flanco de retardo extra en las salidas de la RAM (ver sección 3.8), el símbolo extendido j comienza a ser mostrado justamente dos periodo de muestreo después de que el símbolo útil j haya sido procesado por completo por la IFFT. Así, si N pertenece al muestreo del prefijo cíclico, es decir al rango $\Delta N1$, se debe cumplir que la salida $I[N]$ y $Q[N]$ debe ser igual a los valores $X_{RE}[N-\text{dim_guarda}-1]$ y $X_{IM}[N-\text{dim_guarda}-1]$. Esto se explica por el hecho de que el prefijo cíclico corresponde a la copia de los últimos dim_guarda valores del símbolo útil resultante de la IFFT. Por otra parte, para los N pertenecientes al rango de símbolo $\Delta N2$, se debe cumplir que $I[N]$ y $Q[N]$ debe ser igual a $X_{RE}[N-\text{NFFT}-\text{dim_guarda}-1]$ y $X_{IM}[N-\text{NFFT}-\text{dim_guarda}-1]$. En definitiva, si fuese posible discriminar los periodos de muestreo correspondientes al prefijo cíclico y al símbolo útil, bastaría

con evaluar estas condiciones para asegurar el correcto comportamiento del sistema.

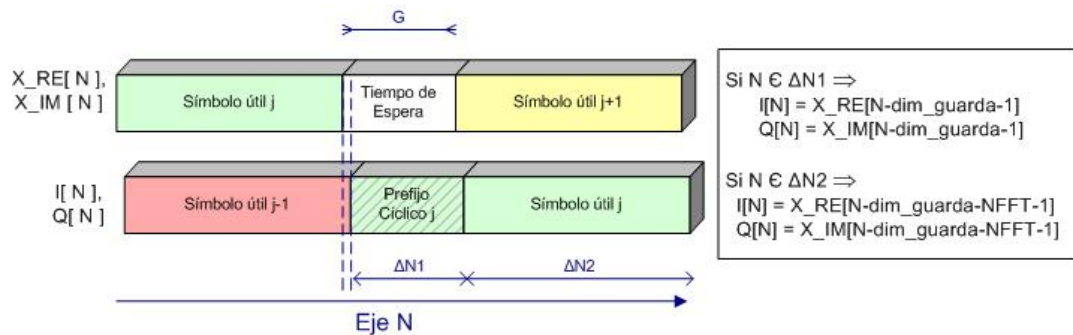


Fig. 5. 38 Condiciones para X_RE , X_IM , $I[N]$ y $Q[N]$ en función del periodo de muestreo N .

Lamentablemente, este tipo de pruebas son imposibles de realizar con las herramientas del CSP Analyzer. Sin embargo, si se exportan los datos y se analizan mediante EXCEL, el asunto se reduciría a tener una señal que posea la información de cuando se muestra el prefijo o el símbolo útil. Como vimos anteriormente, la señal en_entrador posee exactamente las características solicitadas.

En fin, si se toma en cuenta el retardo aparente de esta señal, mediante filtros en Excel se podría fácilmente discriminar que datos corresponden al símbolo útil y al prefijo cíclico. El resto de la evaluación consistiría en verificar que las condiciones especificadas sean siempre

verdaderas en cada caso. Este análisis se ilustra en la fig. 5.39.

(a) Evaluación de los periodos correspondientes al símbolo útil

	A	B	C	D	E	F	G	H
1	EN_ENTRAMADOR	I[N]	Q[N]	XN_RE	XN_IM	SIMBOLO UTIL	I[N] = X_RE[N-G] Q[N] = X_IM[N-G]	I[N] = X_RE[N-NFFT-G] Q[N] = X_IM[N-NFFT-G]
3082	1	-958	-466	-1040	-1013	1	FALSO	Orden ascendente
3083	1	-958	-466	-1040	-1013	1	FALSO	Orden descendente
3084	1	-958	-466	-1040	-1013	1	FALSO	(Todas)
3085	1	-958	-466	-1040	-1013	1	FALSO	(Diez mejores...)
3086	1	-958	-466	-1040	-1013	1	FALSO	(Personalizar...)
3087	1	-958	-466	-1040	-1013	1	FALSO	VERDADERO
3088	1	-958	-466	-1040	-1013	1	FALSO	(Vacías)
3089	1	-958	-466	-1040	-1013	1	FALSO	(No vacías)
3090	1	-196	-21	-340	-450	1	FALSO	VERDADERO

(b) Evaluación de los periodos correspondientes al prefijo cíclico

	A	B	C	D	E	F	G	H
1	EN_ENTRAMADOR	I[N]	Q[N]	XN_RE	XN_IM	SIMBOLO UTIL	I[N] = X_RE[N-G] Q[N] = X_IM[N-G]	I[N] = X_RE[N-NFFT-G] Q[N] = X_IM[N-NFFT-G]
5130	0	-739	-547	-225	-293	0	Orden ascendente	FALSO
5131	0	-739	-547	-225	-293	0	Orden descendente	FALSO
5132	0	-739	-547	-225	-293	0	(Todas)	FALSO
5133	0	-739	-547	-225	-293	0	(Diez mejores...)	FALSO
5134	0	-739	-547	-225	-293	0	(Personalizar...)	FALSO
5135	0	-739	-547	-225	-293	0	VERDADERO	FALSO
5136	0	-739	-547	-225	-293	0	(Vacías)	FALSO
5136	0	-739	-547	-225	-293	0	(No vacías)	FALSO
5137	0	-739	-547	-225	-293	0	VERDADERO	FALSO
5138	0	-153	-337	-225	-293	0	VERDADERO	FALSO

Fig. 5.39 Análisis del proceso de inserción de Guarda mediante Excel.

Durante las pruebas realizadas, se pudo demostrar que para cada caso de en_entramador las condiciones especificadas se cumplían siempre. De igual forma, si estas pruebas se realizan de manera repetida, los resultados son los mismos. Ante esto, queda clara evidencia del correcto funcionamiento de formación del símbolo extendido en el dominio del tiempo.

5.3. Evaluación de los datos recuperados

Debido a que en la anterior sección se observó que el TX funcionaba tal y como se diseñó, bastaría con evaluar la correcta recuperación de los datos para concluir que el sistema funciona fielmente a lo diseñado. Específicamente, los datos a observar en este experimento corresponden a las constelaciones recuperadas por la FFT, la secuencia binaria recuperada y por último el BER.

Como ya se especificó, el esquema implementado en estos experimentos corresponde al sistema final ilustrado en la fig. 2.3. Para estas pruebas se ha definido un valor de 8 para el parámetro Div en el bloque de Clocking. Por otra parte, todos los parámetros del bloque de Settings, a excepción de Codec_bit, han sido definidos con los valores de la tabla XV. Esto se debe a que en estas pruebas se evaluará al sistema con todos los tipos de codificaciones soportadas, es decir se utilizarán todos los valores posibles de Codec_bit.

Debido a que el sistema implementado consiste en un lazo entre el TX y RX, se espera que los datos recuperados sean prácticamente perfectos. En este esquema la única distorsión se debe de dar a causa del procesamiento implícito en la modulación y

demodulación digital. En definitiva, se espera observar constelaciones casi perfectas mientras que los datos recuperados deben ser perfectos y por ende el BER debe ser cero. A pesar de que estas condiciones son validas para verificar el correcto funcionamiento del esquema, nos proporciona muy poca información sobre la respuesta del mismo ante cualquier distorsión. Debido a esto, sería interesante evaluar el sistema OFDM ante la presencia de diferentes tipos de canal. No obstante, esto va más haya de los objetivos del actual proyecto de tesis y además se estarían obviando los errores de sincronización insertados por los efectos del ruido. Por esta razón, sería de gran importancia realizar este tipo de análisis una vez que se añada una fase de sincronización como trabajo futuro.

Por otra parte, lo que si es posible estimar con el actual esquema es el impacto de los errores de sincronización. Para esto, se ha provocado un TOE mediante la manipulación de la señal Inicio_simbolo. Así, si definimos a la variable TOE como la diferencia entre el inicio del símbolo útil transmitido y la habilitación de Inicio_simbolo, el codificador de la señal Inicio_simbolo corresponderá a los valores especificados en la tabla II.

Tabla XVI
Codificador de la señal Inicio_simbolo en función de la señal TOE, en
simbolos OFDM con Guarda diferente de cero

TOE	Inicio_simbolo
-2	Eactual=E1 AND Gvar=1
-1	Eactual=E2 AND dir_Xn=0
0	Eactual=E2 AND dir_Xn=1
1	Eactual=E2 AND dir_Xn=2
2	Eactual=E2 AND dir_Xn=3

En este aspecto, es importante tener en cuenta que el TOE esta expresado en función de periodos de símbolos. Por ejemplo, si se indica que TOE = -1, esto significa que la señal Inicio_simbolo es habilitada un periodo de símbolo antes que el inicio del símbolo útil. Gracias a esto es posible evaluar el sistema en perfecta sincronía (TOE=0) y ante la presencia de errores de sincronización (TOE≠0). Es más, si tomamos en cuenta el carácter serial de los símbolos, es posible evaluar indirectamente los efectos del ISI en el sistema. Esto se debe a que si TOE>0 una porción de los datos de un símbolo j+1 serán interpretados por el RX como parte del símbolo j.

Teniendo en cuenta todo esto, la metodología utilizada en estas pruebas consistió en implementar todas las codificaciones para cada TOE y capturar las constelaciones recuperadas y el BER. Las constelaciones resultantes de estas pruebas son mostradas desde la fig. 5.40 hasta la fig. 5.44 para cada esquema de codificación.

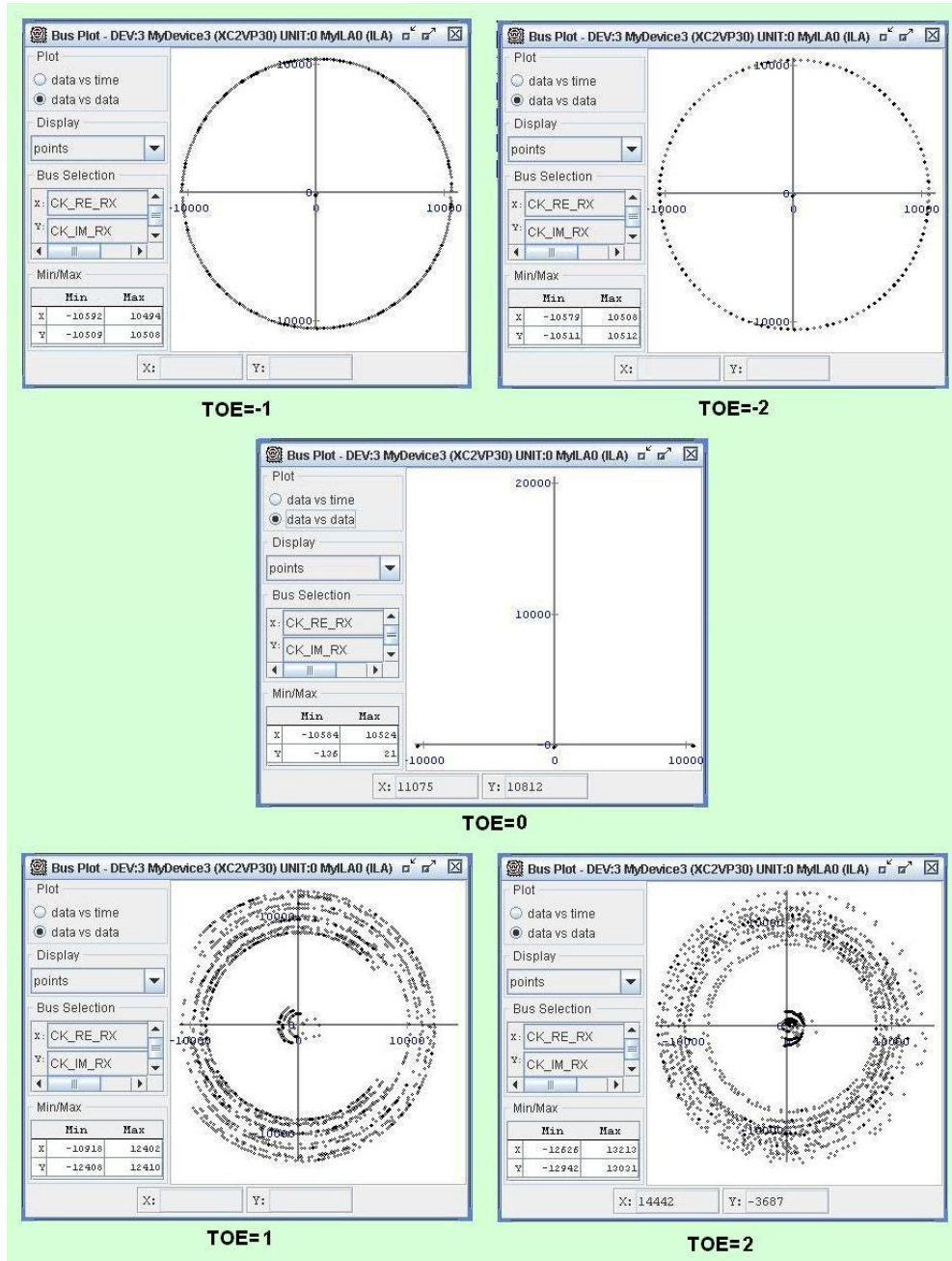


Fig. 5. 40 Constelaciones recuperadas en codificación BPSK.

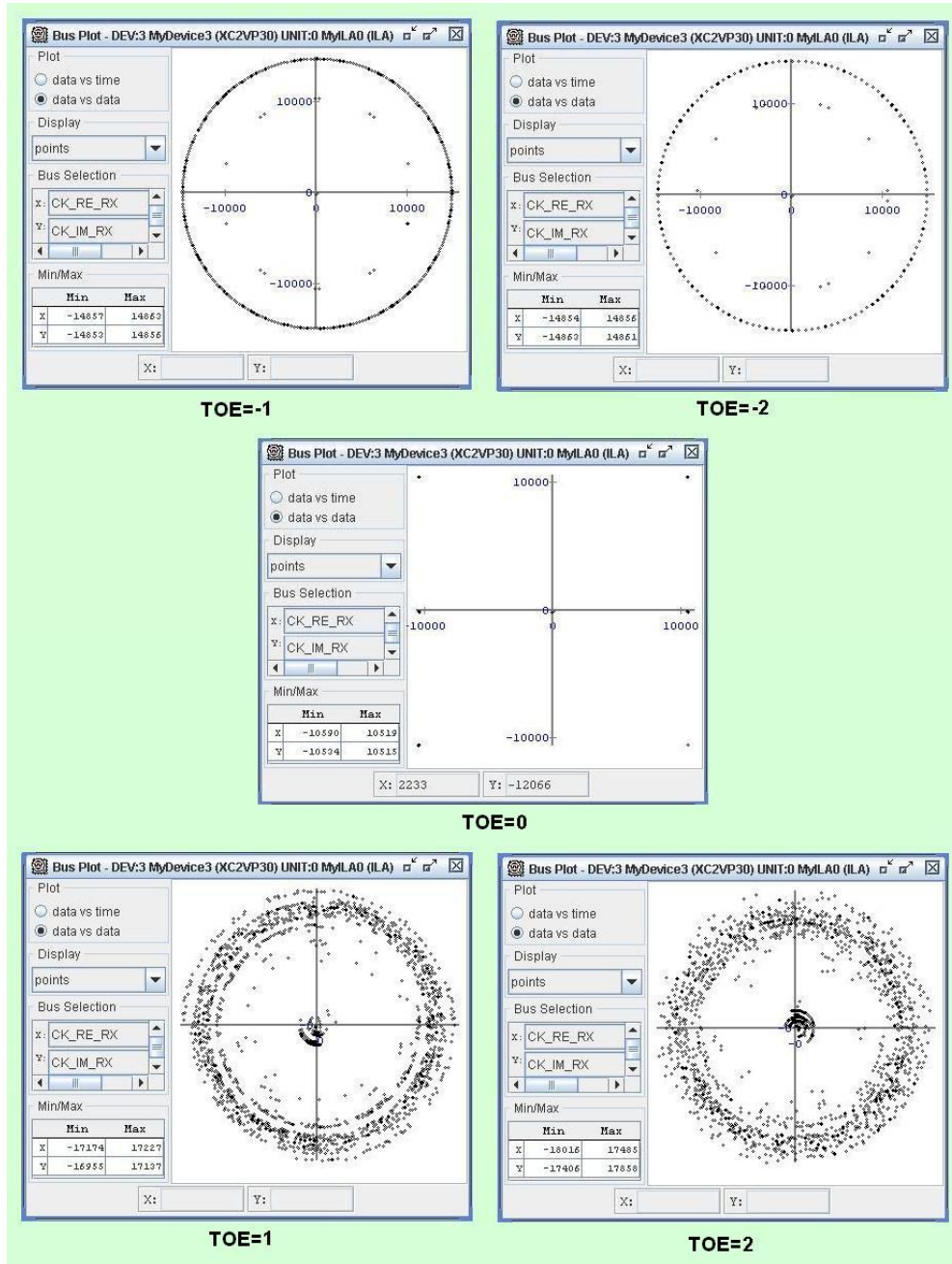


Fig. 5. 41 Constelaciones recuperadas en codificación QPSK.

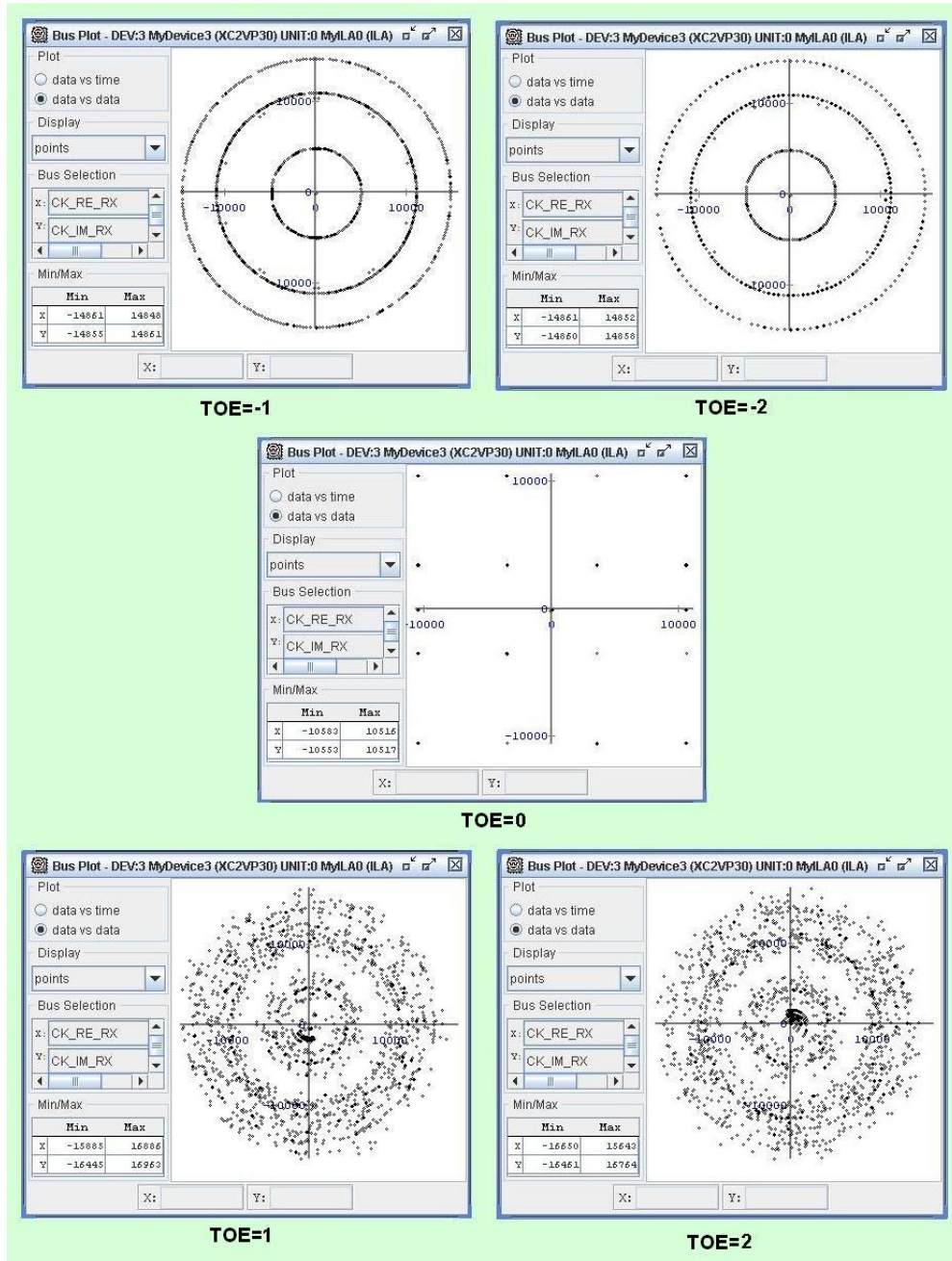


Fig. 5. 42 Constelaciones recuperadas en codificación 16QAM.

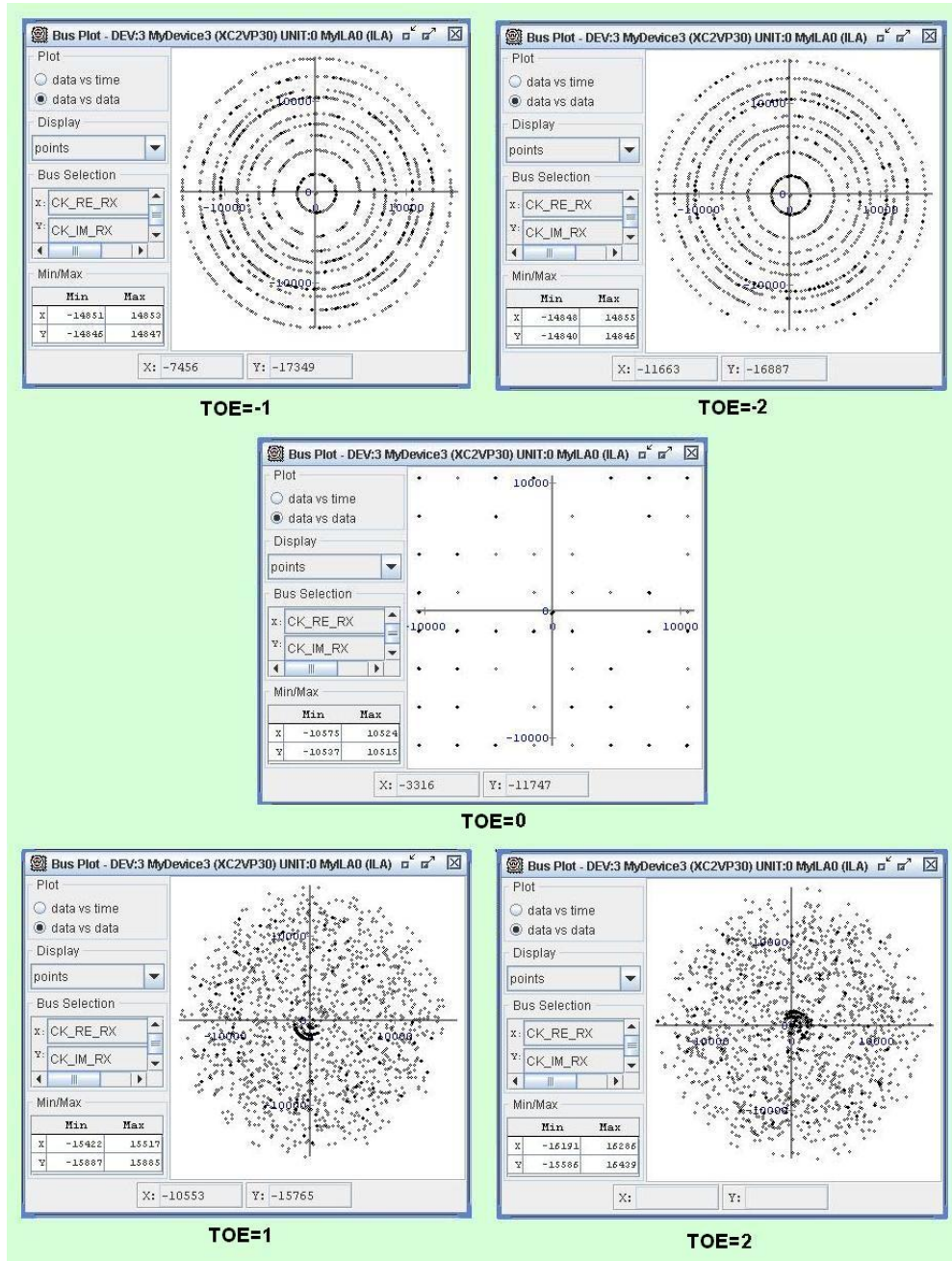


Fig. 5. 43 Constelaciones recuperadas en codificación 64QAM.

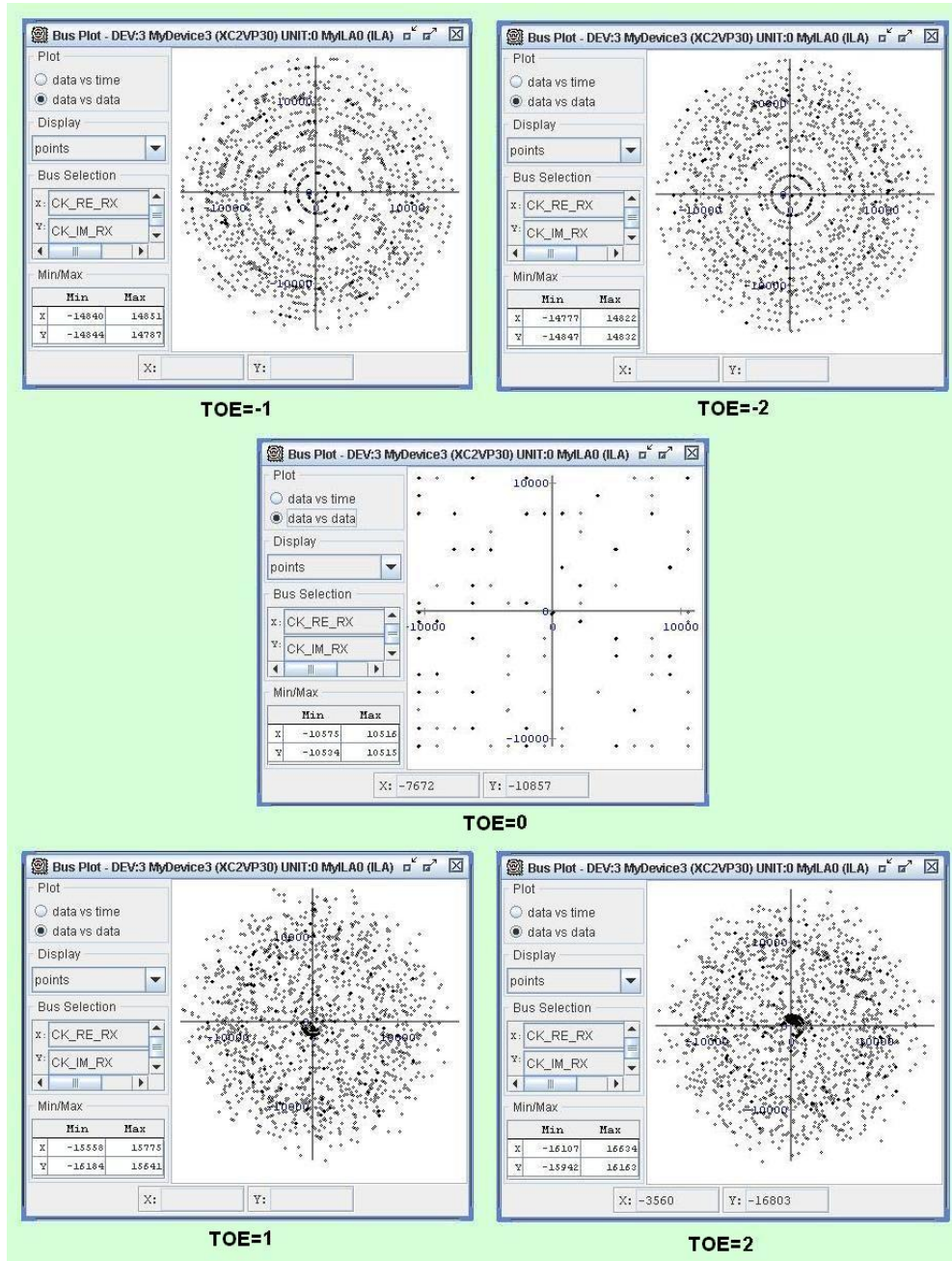


Fig. 5. 44 Constelaciones recuperadas en codificación 256QAM.

El primer aspecto a observar en estos gráficos es que cuando no existe TOE, las constelaciones recuperadas no sufren distorsión alguna. Este resultado nos garantiza que la FFT está transformando correctamente el símbolo útil desde el dominio del tiempo hasta el de la frecuencia.

Por otro lado, en todos los casos en que se ha inducido un TOE se puede observar que las constelaciones tienden a experimentar oscilaciones alrededor de su centro. Además, estas oscilaciones son mucho más nítidas para los casos en que TOE es menor a cero. Si analizamos las características del sistema OFDM implementado, se puede demostrar que este comportamiento se debe a dos factores:

- La implementación del prefijo cíclico.
- El ISI insertado para los $TOE > 0$

Para poder comprender esto de mejor manera, analicemos primeramente el caso de que TOE sea menor a cero y su valor absoluto no es mayor que la dimensión de la guarda, es decir: $dim_guarda \leq TOE \leq 0$. Si analizamos la fig. 5.45 y las propiedades del prefijo cíclico, se puede concluir que la ventana de la FFT en el RX toma al mismo símbolo útil pero con un desfase en el tiempo de TOE puntos. Si tomamos en cuenta que un desfase en el tiempo

implica una variación en la fase de cada uno de los componentes en frecuencias, podemos concluir que en este caso el desplazamiento en el tiempo produce una variación en la fase ϕ_k que depende del índice k y del TOE. Esto implica que en este caso no debe haber repercusión alguna en la magnitud de cada uno de los componentes C_k . En base de todo esto, como las graficas de las constelaciones agrupan a todos los C_k , es completamente lógico esperar oscilaciones circulares perfectas que dependan de la geometría de cada tipo de modulación.

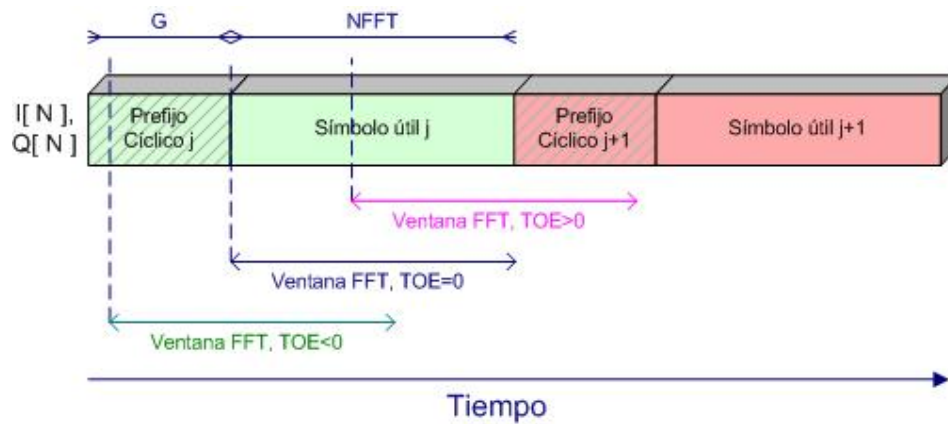


Fig. 5. 45 Efectos del TOE en el procesamiento de la FFT en el RX.

Por otro lado, si TOE es mayor a cero, la ventana de la FFT en el RX capturaría un símbolo híbrido conformado por $NFFT - TOE$ componentes del símbolo j , más TOE componentes del símbolo $j+1$. Es decir, por un lado existe una similitud con caso anterior ya

que se tienen NFFT-TOE componentes del símbolo útil desfasado, pero por otro lado se tiene la presencia de ISI debido a los TOE componentes del símbolo $j+1$ insertados al final. Esto implica que estos últimos componentes pueden ser tomados como procesos aleatorios. Si para efectos de análisis asumimos que este proceso resulta de la suma entre los valores del símbolo útil desfasados TOE flancos y una señal de ruido, podríamos por superposición dividir la señal resultante como la suma del símbolo útil desfasado TOE unidades y una señal de ruido provocada por el ISI insertado por los últimos componentes. Así, tendríamos que la resultante en frecuencia estaría determinada por la misma oscilación descrita anteriormente (cuando $-dim_guarda \leq TOE \leq 0$), pero en este caso afectada por dicho ruido. Esto claramente justifica el porque las oscilaciones observadas en el caso anterior son mucho más nítidas.

El caso final a analizar, sería cuando el $TOE < -dim_guarda$. A pesar de que en nuestras pruebas no se incluye este caso, en base de lo expuesto se esperarían resultados similares a los obtenidos cuando el TOE es mayor a cero. En definitiva, el efecto del TOE puede ser dividido en dos casos generales: cuando no hay ISI y solo existe una variación determinística de la fase, y ante la

presencia del ISI donde también se presenta una variación aleatoria tanto de la fase como la amplitud. Conclusiones similares a estas han sido extraídas en la literatura actual [18].

En este aspecto, es importante recordar que el objetivo principal del prefijo cíclico es evitar el ISI causado por los componentes multitrayectoria. Si bien es cierto que en nuestro caso no hay canal, en aplicaciones reales y análisis futuros se debe tomar en cuenta el esparcimiento del canal para poder analizar que tan efectivo es el prefijo cíclico. Debido a esto, tampoco sería correcto concluir que si el TOE esta en el rango $-dim_guarda \leq TOE \leq 0$ no hay ISI. En fin, si tomamos en cuenta los efectos de canal y las características analizadas en esto resultados, se podría concluir que la verdadera zona libre del ISI estaría dada por (5.1), donde T_m es el tiempo de esparcimiento del canal y $T_{Sampling}$ es el periodo de muestreo.

$$(T_m - dim_guarda \cdot T_{Sampling}) \leq (TOE \cdot T_{Sampling}) \leq 0 \quad (5.1)$$

Una vez examinadas las constelaciones resultantes, la prueba definitiva consiste en analizar el BER en cada uno de los casos. Los valores de BER utilizados para nuestros análisis corresponden al promedio de 10 muestreos de la variable BER mediante el CSP

Analizer. El ANEXO A muestra dichos valores muestreados, mientras que la tabla XVII resume los promedios obtenidos en cada uno de los casos.

Tabla XVII
Valores Promedio Resultantes de las pruebas de BER

Codificación	BER				
	TOE=-2	TOE=-1	TOE=0	TOE=1	TOE=2
BPSK	6354×10^{-4}	3707×10^{-4}	0	3654×10^{-4}	6341×10^{-4}
QPSK	5862×10^{-4}	3647×10^{-4}	0	3656×10^{-4}	5856×10^{-4}
16QAM	4635×10^{-4}	3848×10^{-4}	0	3869×10^{-4}	4661×10^{-4}
64QAM	4483×10^{-4}	4028×10^{-4}	0	4094×10^{-4}	4584×10^{-4}
256QAM	4533×10^{-4}	4205×10^{-4}	0	4310×10^{-4}	4627×10^{-4}

El primer aspecto a evaluar en estas pruebas es que no se produzcan errores cuando el sistema este en perfecta sincronía (TOE=0). Como se puede evidenciar en la fig. 5.45 esto se cumple a cabalidad, ya que cuando TOE es cero se tiene un BER perfecto sin importar el tipo de codificación utilizada.

En este mismo gráfico podemos apreciar la magnitud de la degradación del TOE. Un solo periodo de muestreo de desfase causa una degradación superior al 30%. Como todos sabemos, este valor es inaceptable en aplicaciones reales.

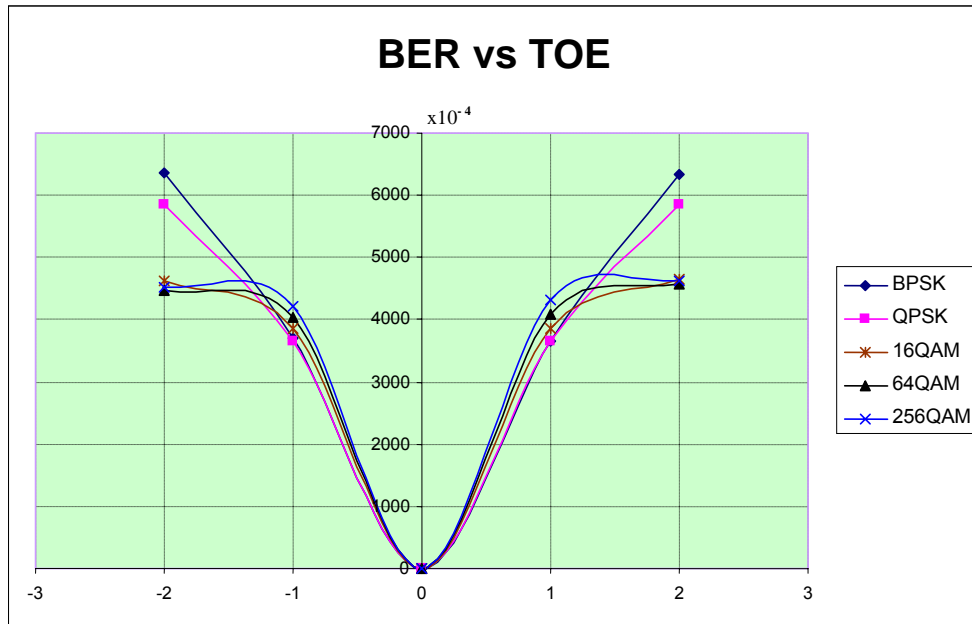


Fig. 5. 46 Resultado de las pruebas de BER en función del TOE.

Otro aspecto que podemos evaluar en un gráfico de este tipo es el efecto del ruido insertado por el ISI, que debe diferenciar los ramales positivo y negativo de la variable TOE. Se esperaría que mientras más positivo sea el TOE en el ramal derecho mayor sea la diferencia con su correspondiente a la izquierda. Esto se justifica al hecho que el ISI es insertado por los últimos TOE valores cuando $TOE > 0$; es decir a mayor TOE, mayor ISI.

Sin embargo, el efecto del TOE por si solo es tan degradante (mayor a 30%), que los demás análisis sobre la degradación del ISI pierden importancia. Sin embargo, como vimos durante el

transcurso de esta sección, el desplazamiento en el tiempo por si solo provoca una oscilación en fase, que es función de la frecuencia k y el TOE. Esto implica que existe la posibilidad de recuperar los valores de C_k mediante esquemas de ecualización en frecuencia, que calculen dicho desfase e implementen factores de corrección de la fase ϕ_k degradada por la oscilación. De analizarse este aspecto y lograrse corregir, resurgiría la importancia del análisis del ISI ya que esta degradación tiene un comportamiento aleatorio y no se vislumbra forma de eliminar su efecto. Ante esto, si tomamos en cuenta la posibilidad de implementar una ecualización en frecuencia y analizamos a (5.1), podemos concluir la existencia de un rango de tolerancia para el TOE. Es más, podría resultar conveniente inducir un desplazamiento negativo del TOE en el sincronizador. Así, se minimizaría la probabilidad caer en la zona de ISI provocada por un TOE positivo. Sin embargo, este análisis debe de ser estudiado en detalle ya que de ser muy alto este desplazamiento, aumentaría la probabilidad de que se inserte ISI debido al efecto multitrayectoria. En fin, ante la inducción de un TOE negativo, existe un compromiso entre la robustez de la sincronización y la tolerancia al desvanecimiento multitrayectoria[19][20], el cual depende netamente del canal.

CONCLUSIONES Y RECOMENDACIONES

Al final de este proyecto podemos concluir que:

- En todo sistema OFDM, existe un compromiso entre la pérdida de Ortogonalidad y la presencia del desvanecimiento selectivo en frecuencia en función del número de portadoras NFFT y el tipo de canal utilizado.
- En todo sistema OFDM, existe un compromiso entre la reducción del ISI y la pérdida de E_b / N_0 y SNR en función de la dimensión del prefijo cíclico y el tipo de canal utilizado.
- De los dos puntos anteriores se concluye que los valores óptimos para los parámetros T_g y NFFT en un sistema OFDM, variaran en función del tipo de canal utilizado.
- El uso del prefijo cíclico tiene la característica de transformar la convolución lineal con el canal en una convolución circular, esta propiedad permite la implementación de ecualizadores que tienen la función de acortar la respuesta efectiva al impulso del canal, de esta forma reduce el valor requerido para T_g , ganando así eficiencia en el sistema.

- Gracias a la implementación de los bloques multiplicador de escala en el TX y mapeador de constelación en el RX, el resto de bloques del sistema han podido ser diseñado de manera independiente y apegada al estándar, facilitando así su posible reutilización.
- Debido a que el factor multiplicativo determina el porcentaje de excursión de la señal de salida, la correcta elección de dicho factor es de vital importancia para el desempeño del sistema.
- De acuerdo a los experimentos realizados en lo referente a la frecuencia de operación se tiene una frecuencia máxima de al menos 50MHz para todos los casos y se induce una correlación entre todos los parámetros del sistema. Sin embargo, con las pruebas realizadas, no se puede concluir a ciencia cierta una regla de correspondencia para la conducta de la frecuencia máxima en función de estos parámetros.
- De las pruebas de codificación de datos se concluye que el bloque codificador funciona correctamente y es capaz de soportar un cambio dinámico del valor de Codificación, siempre y cuando se termine de procesar el componente anterior
- De las pruebas de pilotos y guarda, training y generación del bloque OFDM en la frecuencia, se concluye que el sistema multiplexado diseñado para la generación de los bloques y símbolo útil OFDM funciona correctamente.

- De las pruebas de generación del bloque en el dominio del tiempo se concluye el correcto funcionamiento del proceso de inserción de guarda y por ende de la generación de los símbolos extendidos.
- Debido a que las constelaciones recuperadas y el BER no sufren distorsión alguna en todos los casos de sincronía perfecta ($TOE=0$), se concluye la correcta operación del sistema Modulador/Demodulador diseñado e implementado.
- En el actual esquema OFDM la presencia de TOE genera una degradación de BER superior al 30%, lo cual en términos prácticos es inaceptable.
- Gracias a la implementación del prefijo cíclico, el efecto del TOE el efecto del TOE puede ser dividido en dos casos generales: cuando no hay ISI y solo existe una variación determinística de la fase, y ante la presencia del ISI donde también se presenta una variación aleatoria tanto de la fase como la amplitud.
- Gracias a la implementación de un sistema de ecualización en frecuencia, podría resultar conveniente inducir un desplazamiento negativo del TOE en el sincronizador. En este aspecto, hay que tomar en cuenta que existe un compromiso entre la robustez de la sincronización y la tolerancia al desvanecimiento multitraectoria que depende del canal utilizado.

Para las futuras implementaciones y/o análisis se recomienda tomar en cuenta los siguientes puntos:

- Para la implementación de trabajos futuros en lo referente a sincronizadores OFDM, se recomienda tomar como referencia el preámbulo completo desarrollado por el estándar. En este aspecto el bloque generador de Training y las características parametrizables de la estructura del bloque OFDM en el actual proyecto pueden ser de gran utilidad.
- Las características académicas del actual sistema pueden ser potencializadas con la posibilidad de variar dinámicamente los parámetros de operación. En este aspecto se recomienda analizar las propiedades del CORE VIO mediante el uso del CSP Generator. Las características del VIO permiten simular señales de entrada y salida como pulsadores, switches o incluso cadenas de bits. Otra posible solución sería la implementación de una mesa de prueba mediante el uso de otro hardware.
- Para poder analizar en mayor detalle las características de OFDM y sus parámetros es necesario futuras implementaciones de modelos de canal. Para su posible implementación en FPGAs, se recomienda la realización del modelo matemático del canal a evaluarse en Simulink y

mediante la utilización del paquete System Generator obtener el modelo VHDL en XILINX.

- La posibilidad de poder evaluar el sistema ante la presencia de diferentes tipos de canales abriría las puertas a futuros estudios tales como el análisis del NFFT y el T_g óptimo, así como los efectos de un ecualizador OFDM en la eficiencia del sistema.
- El actual proyecto presenta las facilidades de realizar adecuaciones relativas a la codificación de canal. Entre los futuros proyectos es posible citar: modulación adaptativa, aleatorización, entrelazado o técnicas de FEC como Reed Solomon.
- Para la implementación de trabajos futuros en lo referente a Modulación adaptativa se recomienda prestar vital importancia a las pruebas de codificación de la sección 5.2.1.
- Para cualquier implementación futura en FPGA's es de vital importancia tener en mente la optimización de dos factores: el porcentaje de ocupación del FPGA y la frecuencia máxima soportada.
- Para la optimización del porcentaje de utilización del FPGA es importante mantener un diseño modular. De esta manera, mediante un reporte de síntesis se puede estimar los recursos necesarios para la implementación de un nuevo módulo. Si el porcentaje de ocupación de cierto recurso es excedido, se debería de analizar la posibilidad de realizar cambios en el diseño del nuevo módulo o incluso del actual

sistema. Otra opción, sería la implementación del sistema en otro FPGA con los recursos suficientes; sin embargo, esto no siempre es posible y no es lo más óptimo.

- En lo referente a la sincronización de un sistema digital en VHDL, se debe evitar a toda costa que las señales de reloj sean tratadas como señales combinatoriales ya que esto afecta al rendimiento del sistema.
- Para la optimización de la tasa de datos, se recomienda futuras implementaciones que se basen en el procesamiento paralelo de cadenas de Codec_bit bits en los bloques de codificación/decodificación. De esta manera, se podría aprovechar digitalmente las propiedades de codificación obteniendo una tasa de datos Codec_bit veces mayor.
- Para la implementación de cadenas paralelas en la entrada de datos, se recomienda la implementación del core LFSR configurado con una salida paralela correspondiente al mínimo común múltiplo de los valores Codec_bit a implementarse. Así, mediante la utilización de un registro de desplazamiento de Codec_bit bits, sería posible predecir el número de desplazamientos a necesitarse entre cada transición del reloj del LFSR.

BIBLIOGRAFÍA

- [1] NAIHUA YUAN, "AN EQUALIZATION TECHNIQUE FOR HIGH RATE OFDM SYSTEMS", UNIVERSITY OF SASKATCHEWAN, SASKATOON, SASKATCHEWAN, CANADA, DICIEMBRE 2003, PAG 36, 37.

- [2] ALAN V. OPPENHEIM, ALAN S. WILLSKY, S. HAMID NAWAB, "SEÑALES Y SISTEMAS", PRENTICE HALL, MEXICO, 1998.

- [3] <http://www.arrakis.es/~ppriego/fourier/fourier.htm>

- [4] <http://www.cmlab.csie.ntu.edu.tw/cml/dsp/training/coding/transform/fft.html>

- [5] DAVID GESBERT, LUC HAUMONTÉ, HELMUT BÖLCSKEI, RAJEEV KRISHNAMOORTHY, AROGYASWAMI J. PAULRAJ, "TECHNOLOGIES AND PERFORMANCE FOR NON-LINE-OF-SIGHTBROADBAND WIRELESS ACCESS NETWORKS". IEEE COMMUNICATIONS MAGAZINE, ABRIL 2002, PAG 86-95.

- [6] SKLAR B., "RAYLEIGH FADING CHANNELS IN MOBILE DIGITAL COMMUNICATION SYSTEMS, PART I: CHARACTERIZATION", IEEE COMMUN. MAG., VOL. 35, NO. 9, JULIO 1997, PAG. 90–100.

- [7] SKLAR BERNARD, "RAYLEIGH FADING CHANNELS IN MOBILE DIGITAL COMMUNICATION SYSTEMS PART II: MITIGATION", IEEE COMMUNICATIONS MAGAZINE, PP 102-109, JULIO 1997.

- [8] DURGIN GREGORY D., "THEORY OF STOCHASTIC LOCAL AREA CHANNEL MODELING FOR WIRELESS COMMUNICATIONS", FACULTY OF THE VIRGINIA

POLYTECHNIC INSTITUTE AND STATE UNIVERSITY,
BLACKSBURG, VIRGINIA, DICIEMBRE 2000, PAG 20-25.

- [9] IEEE STD 802.16 -2004, AIR INTERFACE FOR FIXED BROADBAND WIRELESS ACCESS SISTEM, PAG 427-449
- [10] ETSI TS 102 177 V1.2.1 (2005-1) BRAN HIPERMAN; PHY LAYER
- [11] DS083 (V4.5) VIRTEX-II PRO AND VIRTEX-II PRO X PLATFORM FPGAS: COMPLETE DATA SHEET, XILINX, OCTUBRE 2005, DISPONIBLE EN:
<http://www.xilinx.com/bvdocs/publications/ds083.pdf>
- [12] NT107-0246: XTREMEDSP DEVELOPMENT KIT PRO USER GUIDE, XILINX, DICIEMBRE 2004. DISPONIBLE EN:
http://direct.xilinx.com/bvdocs/userguides/xtremedsp_dev_kit_user_guide.pdf
- [13] VTT010 (V1.2) VIRTEX-II DIGITAL CLOCK MANAGER, XILINX, JUNIO 2003 DISPONIBLE EN:
<http://www.xilinx.com/products/virtex/techtopic/vtt010.pdf>
- [14] DS257 (V 1.0) LINEAR FEEDBACK SHIFT REGISTER V3.0, XILINX, MARZO 2003. DISPONIBLE EN:
<http://www.xilinx.com/ipcenter/catalog/logicore/docs/lfsr.pdf>
- [15] DS260 FAST FOURIER TRANSFORM V3.1, XILINX, NOVIEMBRE 2004. DISPONIBLE EN:
<http://www.xilinx.com/ipcenter/catalog/logicore/docs/xfft.pdf>
- [16] DS230 DISTRIBUTED MEMORY V7.1 XILINX, MAYO 2004. DISPONIBLE EN:
http://www.xilinx.com/ipcenter/catalog/logicore/docs/dist_mem.pdf

- [17] UG029 (V7.1) CHIPSCOPE PRO SOFTWARE AND CORES USER GUIDE (CHIPSCOPE PRO SOFTWARE V7.1I), XILINX, FEBRERO 2005.
DISPONIBLE EN:
http://www.xilinx.com/ise/verification/chipscope_pro_sw_cores_7_1i_ug029.pdf
- [18] BO AI, ZHI-XING YANG, CHANG-YONG PAN, JIAN-HUA GE, YONG WANG, AND ZHEN LU, "ON THE SYNCHRONIZATION TECHNIQUES FOR WIRELESS OFDM SYSTEMS", IEEE TRANSACTIONS ON BROADCASTING, VOL. 52, NO. 2, JUNIO 2006
- [19] WEIDONG XIANG AND THOMAS PRATT, XUDONG WANG, "A SOFTWARE RADIO TESTBED FOR TWO-TRANSMITTER TWO-RECEIVER SPACE-TIME CODING OFDM WIRELESS LAN", IEEE RADIO COMMUNICATIONS, JUNIO 2004
- [20] ALFONSO LUÍS TROYA CHINCHILLA, "SYNCHRONIZATION AND CHANNEL ESTIMATION IN OFDM: ALGORITHMS FOR EFFICIENT IMPLEMENTATION OF WLAN SYSTEMS", VON DER FAKULTÄT FÜR MATHEMATIK, NATURWISSENSCHAFTEN UND INFORMATIK DER BRANDENBURGISCHEN TECHNISCHEN UNIVERSITÄT COTTBUS, ALEMANIA, JULIO 2004

ANEXOS

ANEXO A

HOJA DE DATOS PARA LAS MEDICIONES DEL BER EN LAS PRUEBAS DE LOS DATOS RECUPERADOS

Codificación	#medicion	MM=-2	MM=-1	MM=0	MM=1	MM=2
BPSK	1	6348	3703	0	3655	6349
	2	6366	3694	0	3654	6329
	3	6358	3711	0	3657	6352
	4	6359	3720	0	3650	6337
	5	6349	3712	0	3649	6352
	6	6362	3704	0	3652	6339
	7	6337	3696	0	3655	6356
	8	6351	3705	0	3654	6322
	9	6347	3718	0	3646	6332
	10	6363	3703	0	3667	6339
QPSK	1	5860	3654	0	3655	5853
	2	5867	3656	0	3656	5847
	3	5852	3648	0	3660	5857
	4	5851	3649	0	3661	5859
	5	5867	3641	0	3655	5850
	6	5859	3637	0	3651	5862
	7	5876	3654	0	3650	5837
	8	5863	3639	0	3661	5864
	9	5859	3647	0	3651	5853
	10	5864	3648	0	3656	5880
16QAM	1	4666	3845	0	3899	4653
	2	4642	3833	0	3847	4654
	3	4630	3849	0	3858	4613
	4	4640	3859	0	3883	4677
	5	4632	3826	0	3861	4698
	6	4622	3856	0	3885	4628
	7	4663	3855	0	3851	4674
	8	4623	3844	0	3876	4694
	9	4619	3862	0	3861	4627
	10	4613	3846	0	3868	4694

64QAM	1	4415	4074	0	4030	4572
	2	4512	3976	0	4130	4577
	3	4438	4001	0	4065	4553
	4	4458	3984	0	4082	4531
	5	4584	4071	0	4070	4644
	6	4419	4050	0	4082	4566
	7	4522	4125	0	4158	4561
	8	4428	3992	0	4151	4599
	9	4552	3959	0	4099	4586
	10	4505	4046	0	4073	4652
256QAM	1	4536	4319	0	4372	4639
	2	4377	4133	0	4222	4539
	3	4519	4211	0	4203	4684
	4	4631	4152	0	4409	4640
	5	4487	4333	0	4274	4665
	6	4562	4155	0	4357	4552
	7	4594	4259	0	4274	4676
	8	4541	4087	0	4357	4671
	9	4624	4079	0	4304	4574
	10	4462	4322	0	4328	4632