UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

DEPARTAMENTO DE TEORÍA DE LA SEÑAL Y COMUNICACIONES

PROYECTO FIN DE CARRERA

PROTOTIPADO DE UN SISTEMA WIMAX MIMO 2X2 (I)

INGENIERÍA DE TELECOMUNICACIÓN

Autor: DAVID DÍAZ MARTÍN

Tutor: Dr. VÍCTOR P. GIL JIMÉNEZ

ENERO 2010

A todos los que me han acompañado en este largo e intenso viaje no libre de dificultades y sacrificios…

…Especialmente a mis padres, a mi hermana y a Jennifer…

RESUMEN

Este proyecto surgió como respuesta al creciente interés que se ha generado en el mundo de las telecomunicaciones alrededor de las **comunicaciones inalámbricas de alta capacidad**, las técnicas de diversidad **MIMO** (Múltiples Entradas Múltiples Salidas) y las plataformas de desarrollo basadas en **FPGA** (Matrices de Puertas Programables).

El desarrollo de las comunicaciones inalámbricas de alta capacidad se ha producido, principalmente y según la cobertura de las redes, debido a la evolución de la familia de estándares IEEE 802.11 (Wi-Fi, es decir, Fidelidad Inalámbrica) y los sistemas HSDPA (Acceso Descendente de Paquetes a Alta Velocidad) introducidos por el 3GPP (Proyecto Conjunto de Tercera Generación). Sin embargo, la incipiente revolución en estas comunicaciones de alta capacidad viene dada por la existencia de dos familias de estándares inalámbricos diseñados para tal fin: IEEE 802.16e-2005 (WiMAX Móvil, donde WiMAX significa Interoperabilidad Mundial para Acceso por Microondas) e IEEE 802.20 (Mobile-Fi, es decir, Fidelidad Móvil). Ambos cuentan con características similares, si bien WiMAX Móvil posee la ventaja de una mayor experiencia en el mercado y apoyo de la industria.

Las técnicas MIMO proporcionan una gran mejora en el canal de comunicación, esto es, incremento del radio de la celda y mayor capacidad. Por esta razón, cada vez juega un papel más importante en los últimos estándares de comunicación inalámbricos (IEEE 802.11n, IEEE 802.16e-2005, etc.) siendo unas técnicas fundamentales en los proyectados a futuro.

Las plataformas de desarrollo basadas en FPGA han experimentado una gran evolución en los últimos años gracias la mejora de la velocidad de proceso y capacidad de almacenamiento de dichas FPGA. Estos factores, unido a la posibilidad de reprogramación y a la existencia de entornos de desarrollo cada vez más sencillos y flexibles, han permitido su incursión en campos de aplicación típicos de los ASIC (Circuitos Integrados para Aplicaciones Específicas).

Este proyecto cuenta con el objetivo principal de aunar los tres puntos anteriores: comunicaciones inalámbricas de alta capacidad, técnicas MIMO y desarrollo sobre FPGA. Para ello, se ha diseñado un **receptor para el estándar IEEE 802.16d-2004 (WiMAX Fijo) utilizando MIMO 2x2, OFDM (Multiplexado por División en Frecuencias Ortogonales) en la capa física y BPSK (Modulación Binaria por Salto de Fase) para la modulación de datos**. El desarrollo de dicho receptor se ha realizado empleando una **plataforma de desarrollo sobre FPGA basada en Simulink®**, la cual proporciona un alto nivel de abstracción en el diseño de los modelos.

ABSTRACT

This project arose in response to the growing interest generated in the telecommunications industry around **high-capacity wireless communications**, **MIMO** (Multiple Input Multiple Output) diversity techniques and development platforms based on **FPGA** (Field Programmable Gate Array).

The development of high-capacity wireless communications has occurred, mainly and depending on networks coverage, due to the evolution of the family of standards IEEE 802.11 (Wi-Fi, i.e., Wireless Fidelity) and HSDPA (High Speed Downlink Packet Access) systems introduced by the 3GPP (3rd Generation Partnership Project). Nevertheless, the incipient revolution in these high-capacity communications comes from the existence of two families of wireless standards designed for such purpose: IEEE 802.16e-2005 (Mobile WIMAX, where WIMAX stands for Worldwide Interoperability for Microwave Access) and IEEE 802.20 (Mobile-Fi, that is, Mobile Fidelity). Both standards have similar features, although Mobile WIMAX has the advantage of greater market experience and industry support.

MIMO techniques provide a great improvement in the communication channel, that is to say, an increase in cell radius and capacity. For this reason, they play a more and more important role in the latest wireless communications standards (IEEE 802.11n, IEEE 802.16e-2005), being key techniques for future projected standards to come.

Development platforms based on FPGA have in recent years due to improvements in process and storage capacities of the FPGA. These factors, coupled with the possibility of reprogramming and the appearance of ever simpler and more flexible development environments have allowed FPGA their incursion in application fields typical of ASIC (Application Specific Integrated Circuit).

This project has the main objective of joining the above three points: high-capacity wireless communications, MIMO techniques and development on FPGA. With such aim, **a receiver has been designed for IEEE 802.16d-2004 standard (Fixed WiMAX), using 2x2 MIMO, OFDM (Orthogonal Frequency Division Multiplexing) in the physical layer and BPSK (Binary Phase Shift Keying) for data modulation**. This receiver has been developed using a **development platform on FPGA based in Simulink®**, which provides a high level of abstraction in model designing.

GLOSARIO

3GPP	3rd Generation Partnership Project (Proyecto Conjunto de Tercera Generación)
AAS	Adaptative Antenna System (Sistema de Antena Adaptativa)
ACI	Adjacent Channel Interference (Interferencia de Canal Adyacente)
AES	Advanced Encryption Standard (Estándar Avanzado de Encriptación)
ADC	Analogic to Digital Converter (Conversor Analógico Digital)
ADOA	Amplitude Difference Of Arrival (Diferencia en Amplitud de Llegada)
All-IP	All Internet Protocol (Todo Protocolo de Interconexión de Redes)
ASIC	Application Specific Integrated Circuit (Circuito Integrado para Aplicaciones Específicas)
ATM	Asynchronous Transfer Mode (Modo de Transferencia Asíncrono)
BER	Bit Error Rate (Tasa de Error de Bit)
BPSK	Binary Phase Shift Keying (Modulación Binaria por Salto de Fase)
BS	Base Station (Estación Base)
CID	Connection Identifier (Identificador de Conexión)
CLB	Configurable Logic Block (Bloque Lógico Configurable)
CP	Cyclic Prefix (Prefijo Cíclico)
cPCI	Compact Peripheral Component Interconnect (Interconexión de Componentes Periféricos Compactos)
CPE	Customer Premises Equipment (Equipo Local del Cliente)
CPLD	Complex Programmable Logic Device (Dispositivo Lógico Programable Complejo)
DAC	Digital to Analogic Converter (Conversor Digital Analógico)
DCM	Digital Clock Manager (Gestor del Reloj Digital)
DDR	Double Data Rate (Tasa Doble de Transferencia de datos)
DES	Data Encryption Standard (Estándar de Encriptación de Datos)
DFT	Discrete Fourier Transform (Transformada Discreta de Fourier)
DOA	Direction Of Arrival (Dirección de Llegada)
DSP	Digital Signal Processor (Procesador Digital de Señales)
FBWA	Fixed Broadband Wireless Access (Acceso Inalámbrico de Banda Ancha Fijo)
FDD	Frequency Division Duplex (Duplexado por División en Frecuencia)
FFT	Fast Fourier Transform (Transformada Rápida de Fourier)
FIFO	First In, First Out (Primero en Entrar, Primero en Salir)
FIR	Finite Impulse Response (Respuesta al Impulso Finita)
FPGA	Field Programmable Gate Array (Matriz de Puertas Programables)
GBps	Gigabyte Per Second (Gigabyte Por Segundo)
GMACS	Giga Multiply Accumulate Operations per Second (Gigaoperaciones Acumuladas de Multiplicación Por Segundo)

GSM	Global System for Mobile communications (Sistema Global para las Comunicaciones Móviles)
HDL	Hardware Description Language (Lenguaje de Descripción de Soportes Físicos)
HFDD	Half-Duplex Frequency Division Duplex (Duplexado Mitad por División en Frecuencia)
HSDPA	High Speed Downlink Packet Access (Acceso Descendente de Paquetes a Alta Velocidad)
HSUPA	High Speed Uplink Packet Access o (Acceso Ascendente de Paquetes a Alta Velocidad)
IFFT	Inverse Fast Fourier Transform (Transformada Rápida Inversa de Fourier)
IOB	Input Output Banks (Bancos de Entrada Salida)
IP	Internet Protocol (Protocolo de Interconexión de Redes)
ISI	Inter-Symbols Interference (Interferencia entre Símbolos)
ISP	Internet Service Provider (Proveedor de Servicios de Internet)
LAN	Local Area Network (Red de Área Local)
LOS	Line Of Sight (Línea de Visión Directa)
LTE	Long Term Evolution (Evolución a Largo Plazo)
MAC	Medium Access Control (Control de Acceso al Medio)
MAN	Metropolitan Area Network (Red de Área Metropolitana)
MB	Megabyte (Megabyte)
MBWA	Mobile Broadband Wireless Access (Acceso Inalámbrico de Banda Ancha Móvil)
MIMO	Multiple Input Multiple Output (Múltiples Entradas Múltiples Salidas)
MMCX	Micro-Miniature Coaxial (Coaxial Micro Miniaturizado)
Mobile-Fi	Mobile Fidelity (Fidelidad Móvil)
NAS	Network Attached Storage (Almacenamiento Conectado a Red)
NLOS	Non Line Of Sight (Sin Línea de Visión Directa)
OFDM	Orthogonal Frequency Division Multiplexing (Multiplexado por División en Frecuencias Ortogonales)
OFDMA	Orthogonal Frequency Division Multiple Access (Acceso Múltiple por División en Frecuencias Ortogonales)
PHY	Physical Layer (Capa Física)
QAM	Quadrature Amplitude Modulation (Modulación en Amplitud por Cuadratura)
QoS	Quality of Service (Calidad de Servicio)
QPSK	Quadrature Phase Shift Keying (Modulación en Cuadratura por Salto de Fase)
RAM	Random Access Memory (Memoria de Acceso Aleatorio)
RF	Radio Frequency (Radio Frecuencia)
ROM	Read Only Memory (Memoria de Sólo Lectura)
SAN	Storage Area Network (Red de Área de Almacenamiento)
SDR	Software-Defined Radio (Equipos de Radio Definidos por Software)

SDRAM	Synchronous Dynamic Random Access Memory (Memoria Dinámica de Acceso Aleatorio Síncrona)
SFID	Service Flow Identifier (Identificador de Flujo de Servicio)
SRAM	Static Random Access Memory (Memoria Estática de Acceso Aleatorio)
SS	Subscriber Station (Estación Suscriptora)
STC	Space Time Coding (Codificación Espacio Tiempo)
TDD	Time Division Duplex (Duplexado por División en Tiempo)
TDM	Time Division Multiplexing (Multiplexado por División en Tiempo)
TDOA	Time Difference Of Arrival (Diferencia en Tiempo de Llegada)
UL	Uplink (Enlace Ascendente)
UMTS	Universal Mobile Telecomunication System (Sistema Universal de Telecomunicaciones Móviles)
VHDL	Very High Speed Integrated Circuit Hardware Description Language (Lenguaje de Descripción de Soportes Físicos para Circuitos Integramos Muy Rápidos)
VLAN	Virtual Local Area Network (Red de Área Local Virtual)
VoIP	Voice over IP (Voz Sobre IP)
WAN	Wide Area Network (Red de Área Amplia)
WCDMA	Wideband Code Division Multiple Access (Acceso múltiple por División de Código de Banda Ancha)
WiBro	Wireless Broadband (Banda Ancha Inalámbrica)
Wi-Fi	Wireless Fidelity (Fidelidad Inalámbrica)
WiMAX	Worldwide Interoperability for Microwave Access (Interoperabilidad Mundial para Acceso por Microondas)
xDSL	Digital Subscriber Line (Línea de Suscripción Digital)

INDICE

1. Introducción

Las líneas de trabajo del presente proyecto surgieron debido al creciente interés que se ha generado en el actual panorama de las telecomunicaciones por las **comunicaciones inalámbricas de alta capacidad, las técnicas MIMO (Múltiples Entradas Múltiples Salidas) y el potencial de las plataformas de desarrollo basadas en FPGA (Matrices de Puertas Programables)**. Además, este proyecto, concretamente, cuenta con un importante antecedente realizado en forma de Estudio Tecnológico.

Así, en los siguientes apartados se detallan los antecedentes de este proyecto, los objetivos con los cuales se ha realizado y la descripción de los contenidos del presente texto que ilustran su consecución.

1.1 Antecedentes del proyecto

Este proyecto describe uno de los dos elementos, el receptor, de un **proyecto realizado conjuntamente** entre el presente autor (David Díaz Martín) y Roberto Prieto Alonso. Dicho proyecto conjunto consiste en el diseño, simulación y validación de un sistema de comunicación completo (emisor y receptor) para el estándar IEEE 802.16d-2004 [1] utilizando MIMO 2x2, OFDM (Multiplexado por División en Frecuencias Ortogonales) en la capa física y BPSK (Modulación Binaria por Salto de Fase) para la modulación de datos. Todo ello mediante una plataforma de desarrollo para FPGA basada en Simulink®: ***VHS-ADC Virtex-4* de Lyrtech** junto *a* ***Xilinx System Generator for DSP*** (Procesador Digital de Señales) [2] [3].

El proyecto conjunto surge de la realización previa de un **Estudio Tecnológico** por parte de los mismos autores. Este estudio, cuyo título es **"Estudio práctico sobre el prototipado de un sistema MIMO 2x2 para WiMAX"** (Interoperabilidad Mundial para Acceso por Microondas) [4] contaba con los siguientes objetivos principales:

- Caracterización de la plataforma de desarrollo sobre FPGA basada en Simulink® denominada *VHS-ADC Virtex-4* de Lyrtech y de la herramienta software *Xilinx System Generator for DSP*.

- Desarrollo de un sencillo sistema de comunicación básico empleando la plataforma anterior que sirviese como base para un posible diseño referido al estándar IEEE 802.16d-2004 (WiMAX Fijo).

La consecución del primer objetivo incluyó el diseño, simulación, síntesis, ejecución y validación sobre la propia FPGA de sencillos modelos que permitieron ilustrar las diferentes

características de esta plataforma de desarrollo. Dicho modelos fueron realizados mediante la herramienta software *Xilinx System Generator for DSP*.

Por otra parte, una vez caracterizada la plataforma, el segundo objetivo se cumplió a través del diseño, simulación y validación de un sistema de comunicación elemental. Dicho modelo se encuentra formado por un emisor y un receptor, en los cuales destaca el empleo un esquema IFFT/FFT (Transformada Rápida Inversa de Fourier / Transformada Rápida de Fourier).

Finalmente, comentar que tanto este proyecto conjunto como otros dos previos, que seguían la misma línea de trabajo (basados en M ATLAB® y Simulink®), forman parte de un proyecto de la CICYT (Comisión Interministerial de Ciencia y Tecnología) denominado MACAWI (Modelado de canal, Algoritmos y Capacidad para comunicaciones WiMAX).

1.2 Objetivos del proyecto

El presente proyecto cuenta con una serie de objetivos asociados a sus características más destacadas: estándar IEEE 802.16d-2004 (WiMAX Fijo), técnicas MIMO y plataforma de desarrollo sobre FPGA basada en Simulink®.

Así, los principales objetivos son los siguientes:

- Estudio y análisis del **estándar IEEE 802.16d-2004 (WiMAX Fijo)** en su definición OFDM para la capa física.
- Estudio y análisis de las técnicas **MIMO 2x2** en referencia al estándar IEEE 802.16d-2004 (WiMAX Fijo).
- Aplicación de los objetivos anteriores para el diseño, simulación y validación de un receptor para el estándar IEEE 802.16d-2004 (WiMAX Fijo) utilizando MIMO 2x2, OFDM en la capa física y BPSK para la modulación de datos. Para ello se emplea una **plataforma de desarrollo para FPGA basada en Simulink®: *VHS-ADC Virtex-4* de Lyrtech junto a la herramienta *Xilinx System Generator for DSP*.**

Además de los anteriores objetivos principales, existen una serie de objetivos secundarios:

- Caracterización de la plataforma de desarrollo para FPGA basada en Simulink® (*VHS-ADC Virtex-4* de Lyrtech junto a la herramienta *Xilinx System Generator for DSP*) haciendo especial hincapié en las limitaciones de las mismas respecto a los bloques empleados en el diseño.

- Análisis de la situación del estándar posterior al IEEE 802.16d-2004 (WiMAX Fijo) que representa su evolución hacia la movilidad, es decir, el IEEE 802.16e-2005 (WiMAX Móvil) [5] en el panorama actual y futuro de las comunicaciones inalámbricas de alta capacidad.

1.3 Descripción de contenidos

Los contenidos que conforman este documento tienen el fin de ilustrar el cumplimiento de los objetivos del proyecto desarrollado y que fueron descritos en el apartado anterior.

Así, el detalle de los contenidos de cada uno de los siguientes apartados de este proyecto es el siguiente:

- **Estándar IEEE 802.16 (WiMAX)**. Visión global de las características y aplicaciones de las principales versiones existentes de WiMAX: WiMAX Fijo (IEEE 802.16d-2004), WiMAX Móvil (IEEE 802.16e-2005) y WiBro (IEEE 802.16e-2005). Además se compara el estándar IEEE 802.16e-2005 con sus más directos competidores (Wi-Fi, 3G HSDPA y Mobile-Fi, es decir: Fidelidad Inalámbrica, Acceso Descendente de Paquetes a Alta Velocidad y Fidelidad Móvil) y se explica el proceso de implantación inicial de WiMAX en España.

- **Caracterización de la plataforma empleada**. Descripción de la plataforma de desarrollo para FPGA basada en Simulink® utilizada, tanto a nivel de hardware como de software. Además, se realiza una caracterización detallada de cada unos de los bloques empleados en el receptor diseñado.

- **Diseño del receptor**. Caracterización del receptor diseñado indicando, para cada etapa, los requisitos según el estándar IEEE 802.16d-2004 (WiMAX Fijo) y el modelo correspondiente desarrollado convenientemente justificado.

- **Validación, simulación y resultados**. Verificación del correcto funcionamiento del receptor diseñado a través de su simulación empleando diferentes fuentes de datos. Así, se analiza y verifica en diferentes puntos de dicho receptor la correspondencia de la señal con su equivalente en el emisor.

- **Conclusiones y trabajos futuros**. Descripción de las principales conclusiones obtenidas tras la realización de este proyecto, así como las posibles mejoras y líneas de trabajo en el futuro en base al mismo.

- **Presupuesto**. Detalle del coste de la elaboración del presente proyecto, tanto desde el punto de vista personal como material.

- **Bibliografía**. Origen de la documentación empleada para la realización de este proyecto.

- **Anexos**. Información ampliada referente a varios aspectos del proyecto: código de las funciones MATLAB® implementadas y fundamentos de FPGA, MATLAB® y Simulink®.

2. Estándar IEEE 802.16 (WiMAX)

El presente proyecto se basa en uno de los estándares que forman la familia de estándares de acceso inalámbrico IEEE 802.16, generalmente conocidos como WiMAX (Interoperabilidad Mundial para Acceso por Microondas). Dicho estándar es el **IEEE 802.16d-2004** [1] y es conocido como **WiMAX Fijo**.

WiMAX, realmente, es una certificación privada proporcionada por el **WiMAX Forum** [6] para asegurar la conformidad de diferentes dispositivos con la familia de estándares de acceso inalámbrico IEEE 802.16 (principalmente IEEE 802.16d-2004 [1] e IEEE 802.16e-2005 [5], como se muestra en la figura 1) y la interoperabilidad entre ellos. WiMAX Forum define perfiles [6] que determinan las diferentes opciones que se contemplan en la familia de estándares (modulación, ancho de banda, frecuencia de utilización, etc.), y los complementa en áreas no cubiertas, como sistemas de pruebas, arquitectura de red, etc.

Figura 1: Definición del perfil Mobile WiMAX Release 1
(Fuente: Fundación Telefónica [7])

La familia de estándares en la cual se basa WiMAX [8] definen un acceso a **redes inalámbricas de banda ancha**, comprendiendo tanto la capa MAC (Control de Acceso al Medio) como la capa PHY (Capa Física), de tal forma que dichos estándares permiten mediante una única torre de distribución, conexiones inalámbricas para miles de usuarios con una velocidad similar a la obtenida mediante tecnología xDSL (Línea de Suscripción Digital) o cablemódem en entornos NLOS (Sin Línea de Visión Directa) y hasta a una distancia con LOS (Línea de Visión Directa) de unos 50 km, en el caso del estándar IEEE 802.16d-2004 y de 10 km en el caso de IEEE 802.16e-2005.

Esta familia de estándares surge debido a las expectativas y realidades sucedidas en los últimos tiempos en torno al acceso de banda ancha de “última milla”. El cableado empleado para esta última milla tiene unos altos costes para los operadores que no siempre justifican su instalación en zonas rurales o geográficamente inaccesibles. Además, la tecnología celular, pese a sus avances en capacidad (HSDPA en los sistemas 3G), todavía tiene lejos la transmisión en tiempo real de aplicaciones multimedia. Por estas razones aparece WiMAX, que permite llevar la

banda ancha a un mayor número de usuarios y cuyo impacto puede ser extraordinario debido a una serie de factores: bajo coste de implantación, gran alcance sin necesidad de línea de visión directa, alta velocidad de transmisión y gran versatilidad como tecnología portadora.

Aunque puedan parecer a primera vista similares, **WiMAX y Wi-Fi** poseen características diferentes, haciendo más propicia una tecnología u otra según el escenario. Wi-Fi, incluida en la familia de estándares IEEE 802.11, se creó para ambientes inalámbricos internos como alternativa a las tradicionales redes internas de cable, es decir, como "conexiones Ethernet inalámbricas", con la limitación de distancia máxima de pocos metros sin visión directa (NLOS) y de cientos de metros con línea de visión directa [9]. Aunque actualmente hay desarrollos de nuevos estándares dentro de la familia IEEE 802.11 (como el IEEE 802.11n que incorpora técnicas MIMO), los más extendidos son el IEEE 802.11b e IEEE 802.11g. Sin embargo, WiMAX, fue diseñado principalmente como tecnología de "última milla" y se puede utilizar en enlaces de acceso MAN (Red de Área Metropolitana) o incluso WAN (Red de Área Amplia).

WiMAX destaca por su capacidad como tecnología portadora, sobre la cual se pueden transportar servicios paquetizados como IP (Protocolo de Interconexión de Redes) y VoIP (Voz Sobre IP), servicios conmutados (TDM o Multiplexado por División en Tiempo), E1/T1, voz tradicional (Clase-5), e interconexiones ATM (Modo de Transferencia Asíncrono) y *Frame Relay*, soportando múltiples servicios simultáneamente con QoS (Calidad de Servicio), lo que la hace adecuada para grandes redes corporativas de voz y datos, así como para operadores de telecomunicaciones.

Así pues se podría definir WiMAX como aquella tecnología que toma lo mejor de la tecnología celular, permitiendo una gran cobertura, y de la tecnología Wi-Fi, en cuanto a la alta velocidad de transmisión.

En el desarrollo de WiMAX es básico el papel desempeñado por el WiMAX Forum (creado en Diciembre de 2001) y por Intel, principal propulsor del mismo. El motivo de la implicación de Intel responde a su deseo de hacerse un hueco en la telefonía móvil, siendo el primer paso para ello la expansión del estándar IEEE 802.16e-2005 (WiMAX Móvil).

Tal y como se comentó anteriormente, WiMAX Forum certifica y fomenta la compatibilidad e interoperabilidad entre diferentes marcas de productos basados en la familia de estándares IEEE 802.16 con el objetivo de acelerar la introducción de esta tecnología en el mercado. El certificado WiMAX es expedido por el WiMAX Forum en base a los resultados de un proceso de certificación realizado en un laboratorio oficial.

Actualmente, el WiMAX Forum se encuentra formado por más de 500 organizaciones dedicadas a la fabricación de chips y equipos, proveedores de servicios, desarrolladores de software, etc. [6]. La importancia de esta tecnología queda patente con el compromiso de compañías como Acer, Agilent Technologies, Alcatel-Lucent, Alvarion, AT&T, British Telecom, Cisco Systems, Dell, Deutsche Telecom AG, EADS Secure Networks, Ericsson, France Telecom, Fujitsu, Google, HTC Corporation, Intel Corporation, Microsoft Corporation, Motorola, NTT (Nippon Telegraph and Telephone), Nokia, Nortel, RIM (Research in Motion Limited), Samsung, Siemens AG, Sony Electronics, Symbian, Texas Instruments, Toshiba, VeriSign, Vodafone, Xilinx, etc.

Además, el WiMAX Forum trabaja conjuntamente con los gobiernos, pues para la expansión de WiMAX es básico el conseguir los necesarios acuerdos internacionales de asignación del espectro, ya que esta asignación, en diferentes países, no está tan madura como la tecnología.

Los contenidos que conforman este apartado tienen el objetivo de ofrecer una visión global de las características y aplicaciones de los perfiles de la familia de estándares IEEE 802.16 definidos por WiMAX. Asimismo se pretende hacer reflexionar al lector al respecto de las posibilidades futuras de esta tecnología, teniendo en cuenta los desafíos a los que se enfrenta.

2.1 Principales versiones de WiMAX: características y aplicaciones

La familia de estándares IEEE 802.16 a partir de la cual surgió WiMAX es muy amplia, encontrándose en constante evolución y contando cada una de ellos con diferentes características a nivel físico que determinan sus aplicaciones. Un resumen de las mismas se ofrece en la tabla 1:

Estándar / Característica	**IEEE 802.16**	**IEEE 802.16a**	**IEEE 802.16d-2004 (WiMAX Fijo)**	**IEEE 802.16e-2005 (WiMAX Móvil)**	**WiBro (IEEE 802.16e-2005)**
Fecha de aprobación	Diciembre 2001	Enero 2003	Junio 2004	Diciembre 2005	Noviembre 2004
Espectro	10 - 66 GHz	2 - 11 GHz	2 - 11 GHz	2 - 6 GHz	2.3 GHz – 2.4 GHz (Corea, Europa y EEUU). Actualmente se negocian más bandas para Europa y EEUU
Funcionamiento	Visión directa (LOS)	Sin visión directa (NLOS)	Revisión del anterior con modificaciones de la capa MAC. Sin visión directa (NLOS)	Sin visión directa (NLOS)	Sin visión directa (NLOS)
Tasa de bit	Hasta 134 Mbit/s con canales de 28 MHz	Hasta 75 Mbit/s con canales de 20 MHz	Hasta 75 Mbit/s con canales de 20 MHz	Hasta 15 Mbit/s con canales de 5 MHz	Hasta 30 Mbit/s
Modulación	QPSK, 16QAM y 64 QAM	OFDM con 256 subportadoras QPSK, 16QAM, 64QAM	OFDM con 256 subportadoras BPSK, QPSK, 16QAM, 64QAM	OFDMA con hasta 2048 subportadoras BPSK, QPSK, 16QAM, 64QAM	OFDMA con 1024 subportadoras BPSK, QPSK, 16QAM, 64QAM
Movilidad	Sistema fijo	Sistema fijo	Sistema fijo	Movilidad de hasta 120 km/h	Movilidad de hasta 120 km/h
Anchos de banda	20, 25 y 28 MHz	Seleccionables entre 1,25 y 20 MHz	Seleccionables entre 1,25 y 20 MHz	Seleccionables entre 1,25 y 20 MHz	9 MHz
Radio de celda típico	2 - 5 km	5 - 10 km aprox. (alcance máximo de unos 50 km con LOS)	5 - 10 km aprox. (alcance máximo de unos 50 km con LOS)	2 - 5 km aprox. (alcance máximo de unos 10 km con LOS)	2 km

Tabla 1: Comparativa a nivel físico de la familia de estándares IEEE 802.16 (incluyendo a WiBro)

WiMAX dispone de **2 versiones principales** en base al estándar en el cual se basa, esto es: **WiMAX Fijo para IEEE 802.16d-2004** [1] y **WiMAX Móvil para IEEE 802.16e-2005** [5]. La principal diferencia entre ambos [10], tal y como se muestra en la figura 2, se encuentra en que el primero se trata de un estándar de acceso inalámbrico fijo mientras que el segundo añade la posibilidad de movilidad en los terminales.

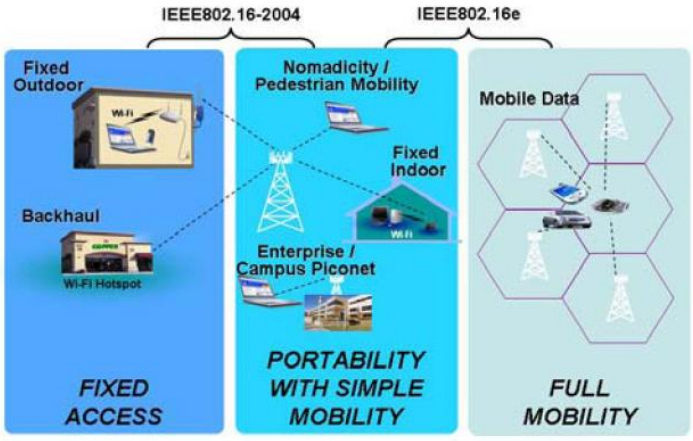

Figura 2: Concepto de WiMAX Fijo vs WiMAX Móvil
(Fuente: Universidad Galileo de Guatemala [10])

2.1.1 WiMAX Fijo (IEEE 802.16d-2004)

El estándar en el cual se basa WiMAX Fijo, IEEE 802.16d-2004, se trata de un estándar de **acceso inalámbrico fijo** (no ofrece movilidad, tal y como se muestra en la figura 3) aprobado en junio de 2004 que ofrece una velocidad de hasta 75 Mbps con un alcance típico de entre 5 y 10 km sin visión directa y 50 km como máximo con visión directa [11]. Además, emplea OFDM para operar de manera fiable en entonos NLOS a una distancia de kilómetros sirviendo a múltiples usuarios de forma que tengan la sensación de que siempre están transmitiendo o recibiendo.

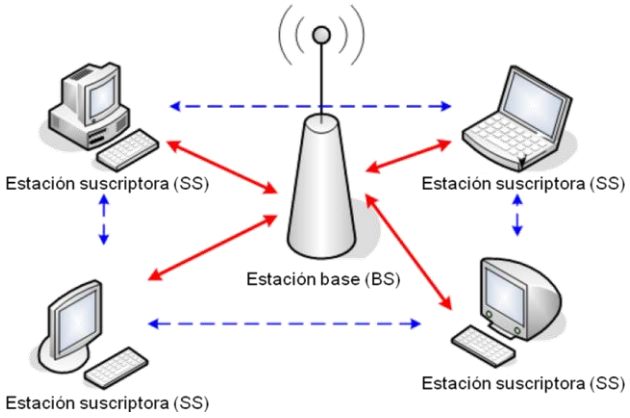

Figura 3: Concepto de WiMAX Fijo
(Fuente: Universidad Carlos III de Madrid [11])

Una de las principales características de WiMAX Fijo a nivel físico es que permite modulación TDD (Duplexado por División en Tiempo), FDD (Duplexado por División en Frecuencia) en la BS (Estación Base), y HFDD (Duplexado Mitad por División en Frecuencia) en la SS (Estación Suscriptora).

En cuanto a la capa MAC, es de destacar que este estándar puede ser usado como portador de ATM, Ethernet e IP, siendo diseñado para acomodar fácilmente futuros protocolos aún no implementados. Se tiene especial interés en proporcionar distintas **QoS** a capas superiores y tener la máxima eficiencia de ancho de banda en el UL (Enlace Ascendente), asignando slots de frecuencia dinámicamente en función de peticiones de ancho de banda.

El estándar IEEE 802.16d-2004 se encuentra **orientado a conexión**, de tal forma que cada estación suscriptora tiene asignada por la estación base un único CID (Identificador de Conexión), existiendo un CID básico, primario, de transporte y de *broadcast*. Además, se proporciona seguridad empleando el protocolo PKM para la gestión de claves que utiliza certificados digitales X.509 y el algoritmo RSA (PKCS#1) para cifrado. También existe la posibilidad de usar triple DES (Estándar de Encriptación de Datos) y AES (Estándar Avanzado de Encriptación).

La **principal aplicación** de este estándar es competir con los proveedores de cable o xDSL para proporcionar servicio de voz (sobre IP) y banda ancha en aquellas regiones donde no es rentable emplear otra tecnología de acceso, pues se tiene un gran radio de cobertura. Se podrían diseñar CPE (Equipos Locales del Cliente) autoinstalables que cumpliesen con este estándar, a fin de abaratar los costes para el cliente. Sin embargo, la industria no ha apostado por ello, utilizándose unidades exteriores. Además, su empleo podría extenderse como *backhaul* inalámbrico (interconexión de redes) para puntos de acceso Wi-Fi o redes celulares. Todas estas aplicaciones se muestran en la estructura de red siguiente (figura 4):

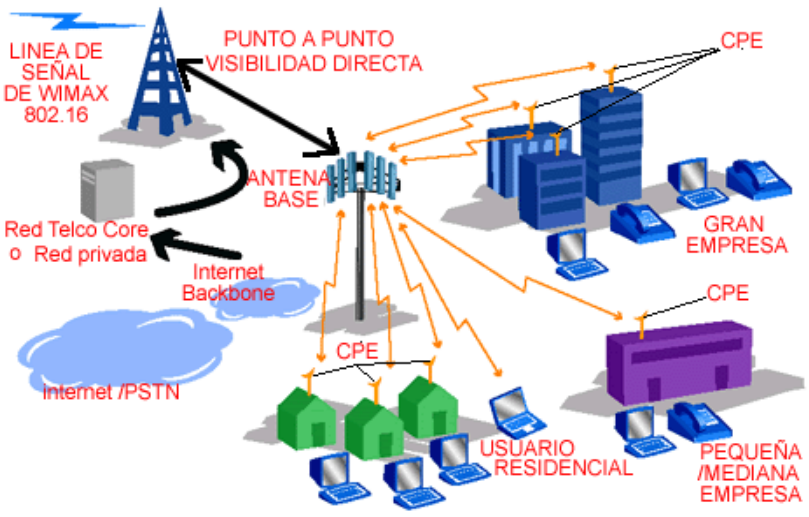

Figura 4: Red WiMAX Fijo

(Fuente: Observatorio Tecnológico del Ministerio de Educación [9])

2.1.2 WiMAX Móvil (IEEE 802.16e-2005)

El estándar en el cual se basa WiMAX Móvil, IEEE 802.16e-2005, amplía las posibilidades del estándar anterior, IEEE 802.16d-2004 mediante la **inclusión de movilidad y *roaming*** (como se muestra en la figura 5) [11]. Se aprobó en diciembre de 2005 y ofrece una velocidad de hasta 20 Mbps con un alcance típico de entre 2 y 5 km sin visión directa (10 km como máximo con visión directa).

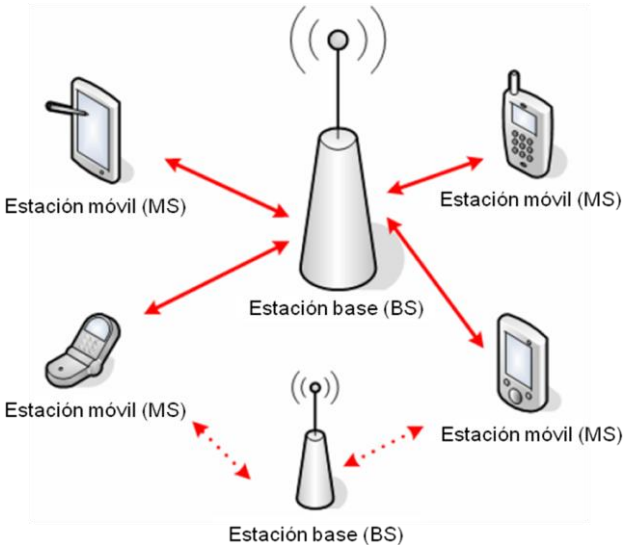

Figura 5: Concepto de WiMAX Móvil
(Fuente: Universidad Carlos III de Madrid [11])

La principal novedad es la **incorporación de OFDMA** (Acceso Múltiple por División en Frecuencias Ortogonales) cuyo concepto se muestra en la figura 6 y su objetivo es mejorar la asignación de ancho de banda a los usuarios. Además, se agregó tecnología de antenas inteligentes MIMO y el uso de AAS (Sistemas de Antenas Adaptativas). Todas estas características se encuentran orientadas a permitir la comunicación móvil en entornos de altas interferencias con múltiples obstáculos, como entornos urbanos.

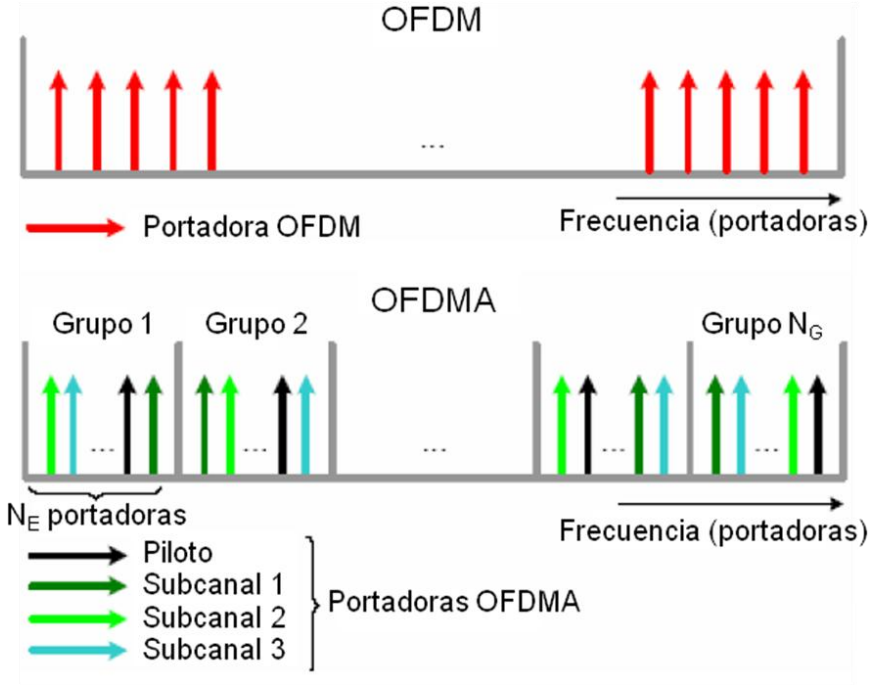

Figura 6: Concepto de OFDM vs OFDMA

(Fuente: Xilinx [3])

De manera análoga al estándar anterior (IEEE 802.16d-2004), WiMAX Móvil cuenta como principal característica a nivel físico que permite modulación TDD, FDD (en la estación base) y HFFD (en la estación móvil).

Además, a nivel de capa MAC, el estándar IEEE 802.16e-2005 tiene también casi las mismas características que WiMAX Fijo (IEEE 802.16d-2004) con aspectos intrínsecos a la movilidad, soportando cientos de usuarios por canal, con un gran ancho de banda y siendo adecuado para tráfico continuo y a ráfagas independientemente del protocolo (IP, Ethernet, ATM…). También soporta simultáneamente diferentes tipos de servicios, ofreciendo QoS mediante los identificadores CID y SFID (Identificador de Flujo de Servicio).

Respecto a la seguridad, mantiene la misma estructura de WiMAX Fijo (IEEE 802.16d-2004) sustituyendo PKM por PKM-II.

La red WiMAX basada en el estándar IEEE 802.16e-2005 que permite **movilidad** se articula de una manera similar a las redes de telefonía celular. La cobertura de un área geográfica se divide en una seria de zonas solapadas llamadas celdas, proporcionando cobertura cada una de ellas para los usuarios que se encuentren dentro. Al moverse un usuario, se desplazará de celda, siendo traspasada la conexión de manera transparente al usuario.

Uno de los problemas del desarrollo de esta tecnología es que los proveedores de servicios WiMAX emplean bandas de frecuencias con licencia, lo cual permite la reducción de interferencias, con una mayor predicción en el comportamiento y estabilidad. Sin embargo esto supone un sobrecoste para el servicio y la imposibilidad regulatoria en muchos países.

Así pues, pese a las dificultades intrínsecas al despliegue de una nueva tecnología, el impacto del estándar WiMAX Móvil ha sido mayor que el de WiMAX Fijo. La versión fija de WiMAX atrajo fundamentalmente a nuevas empresas o especialistas en comunicaciones fijas inalámbricas como Alvarion, mientras que la versión la móvil sí parece despertar el interés de los grandes suministradores de tecnología para redes celulares debido a su abanico de **aplicaciones inherentes a la movilidad**:

- Servicios de telefonía celular de voz mediante VoIP.
- Conectividad de datos de gran ancho de banda para dispositivos móviles.
- Servicios basados en la localización, obteniendo información en tiempo real relacionada con el lugar y momento en que se está.

Además de estas nuevas aplicaciones, WiMAX Móvil hereda las propias de WiMAX Fijo:

- Servicio de voz (sobre IP) y banda ancha en aquellas regiones donde no es rentable emplear otra tecnología.
- *Backhaul* inalámbrico (interconexión de redes) para punto de acceso Wi-Fi o redes celulares.

Todas estas aplicaciones se muestran en la estructura de red siguiente (figura 7):

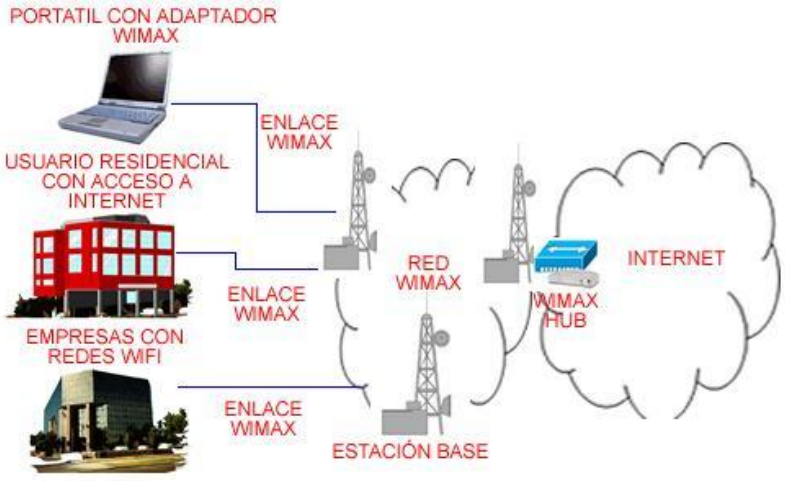

Figura 7: Red WiMAX Móvil

(Fuente: Observatorio Tecnológico del Ministerio de Educación [9])

2.1.3 WiBro (IEEE 802.16e-2005)

WiBro**,** Banda Ancha Inalámbrica, se trata de un estándar creado en **Corea del Sur** a finales de 2004 para proporcionar acceso inalámbrico a la banda ancha a dispositivos fijos y móviles [11]. Se basa en WiMAX Móvil (IEEE 802.16e-2005), siendo reconocido como IEEE 802.16e a finales de 2005 y tratándose actualmente de un **perfil de WiMAX Móvil**. Su creación se vio apoyada por WiMAX Forum e Intel para adelantarse a HSDPA en el mercado asiático.

Como principales características a nivel físico destacan:

- Banda de frecuencias de funcionamiento en 2,3 – 2,4 GHz.
- Ancho de banda del canal de 9 MHz.
- Modulación TDD con el objetivo de proporcionar la máxima eficiencia espectral posible y permitir el mayor número de usuarios por celda.

2.2 WiMAX Móvil (IEEE 802.16e-2005) vs otras tecnologías

Desde un punto de vista puramente técnico, las tecnologías que sirven de base para WiMAX, especialmente en su versión móvil (OFDM, MIMO y All-IP, esto es, Todo Protocolo de Interconexión de Redes), parecen las más adecuadas para llegar a proporcionar servicios de muy alta tasa binaria en entornos inalámbricos.

La posición natural de WiMAX [12] se sitúa entre las tecnologías **Wi-Fi** (teniendo WiMAX mayor cobertura y movilidad) y **3G HSDPA** (teniendo WiMAX mayor tasa de transmisión), Sin embargo, esto no significa que pueda ser una amenaza a corto plazo importante para servicios de datos inalámbricos basados en estas tecnologías.

Además, WiMAX Móvil cuenta con la competencia de una tecnología orientada a cubrir el mismo tipo de comunicaciones móviles de banda ancha. Se trata de la novedosa **Mobile-Fi** (IEEE 802.20), cuyo objetivo en el comienzo de su desarrollo (diciembre 2002) era proporcionar la movilidad que el precursor de WiMAX Fijo (IEEE 802.16a) no podía aportar.

Por tanto, aunque cada una de estas tecnologías cuenta con diferentes características, haciéndolas idóneas para diferentes situaciones, esto no evita que se solapen en cuanto a ciertos aspectos de alcance, movilidad y tasa máxima (figuras 8 y 9), produciéndose una fuerte **competencia por el éxito como estándar de acceso inalámbrico en el camino hacia las redes 4G.**

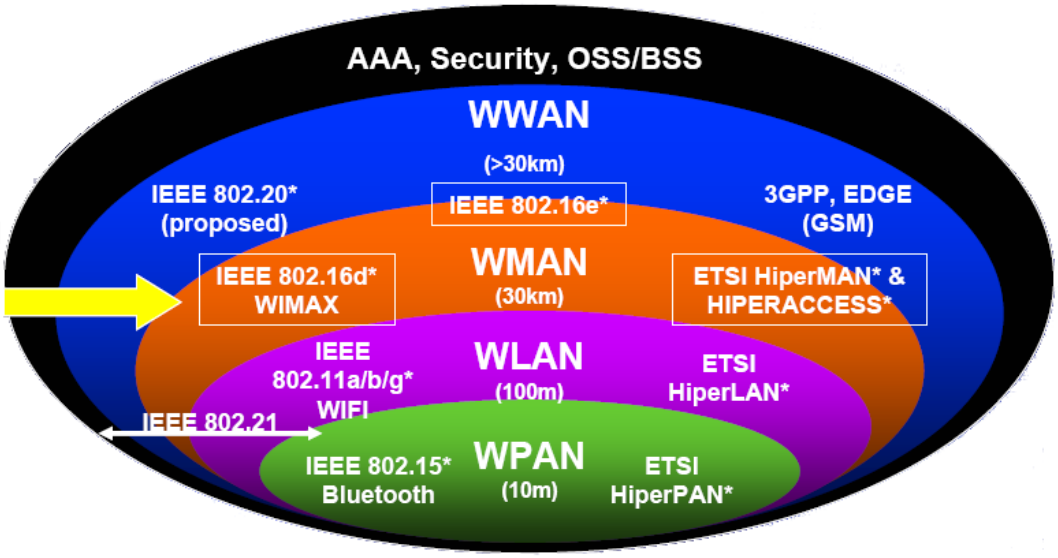

Figura 8: Cobertura de estándares inalámbricos

(Fuente: Intel [12])

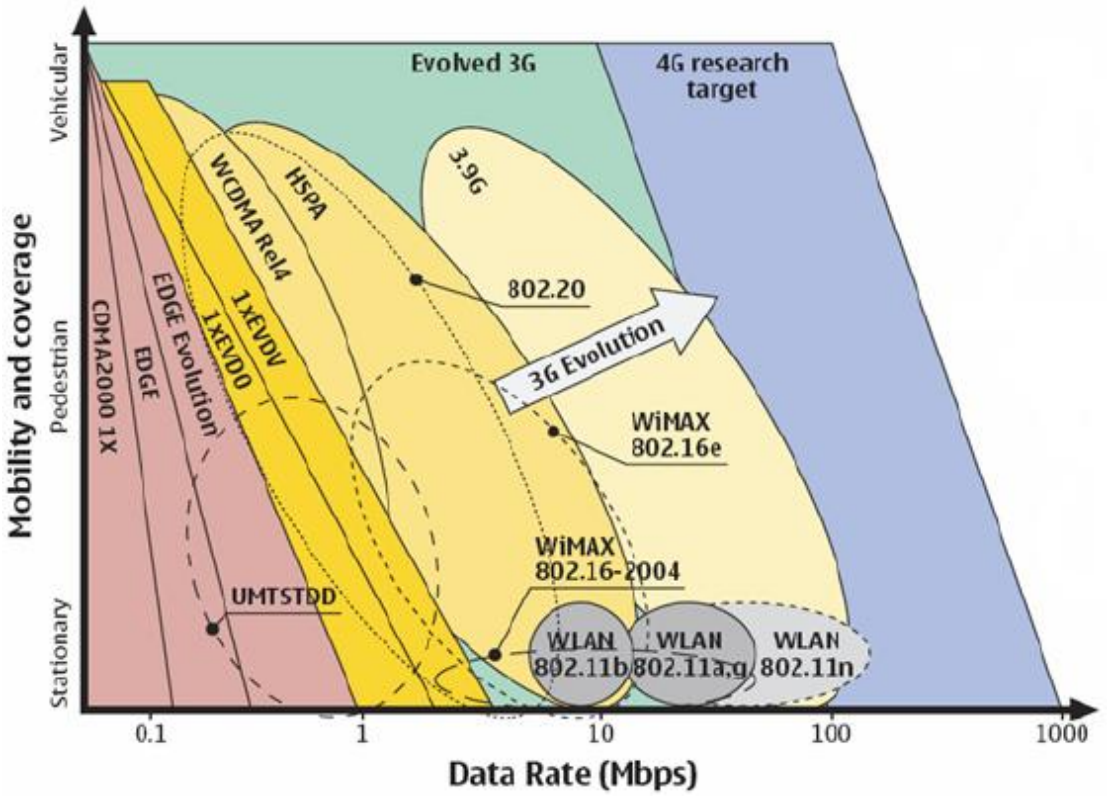

Figura 9: Tasa máxima y movilidad de estándares inalámbricos

(Fuente: Intel [12])

2.2.1 WiMAX Móvil vs 3G HSDPA

La aparición de WiMAX estuvo muy propiciada por el incumplimiento de plazos de los sistemas de telefonía móvil global 3G y sus bajas prestaciones iniciales [7]. Estos sistemas 3G son introducidos por el 3GPP (Proyecto Conjunto de Tercera Generación) y como

solución a esas bajas capacidades iniciales se incorporó HSDPA, el cual consiste en un avance en WCDMA (Acceso múltiple por División de Código de Banda Ancha) para obtener mayores velocidades de transmisión de datos con una mínima de inversión.

El 3GPP identifica **3 fases en la evolución de HSDPA**. La primera se conoce como "HSDPA básico" y se definió en la Release 5 permitiendo tasas de datos típicas de 10.8Mbit/s y máximas de 14.4Mbit/s. La segunda fase agrega HSUPA (Acceso Ascendente de Paquetes a Alta Velocidad) y antenas inteligentes, mientras que la tercera fase ya incluye la combinación de OFDM y MIMO. Esta última fase, desarrollada en el grupo de estudio LTE (Evolución a Largo Plazo), pronostica tasas de transmisión de datos de 100 Mbps en el enlace descendente mediante el uso de múltiples antena con multiplexado espacial.

En la actualidad nos encontramos con un dominio de la primera fase de HSDPA, el llamado "HSDPA básico", encontrándose en despliegue HSUPA y en pruebas piloto LTE.

Así, las ventajas de HSDPA son la **madurez de la tecnología** y el previsible alto coste de las licencias para WiMAX asociado a las decisiones regulatorias sobre el uso espectro. Este último se trata de un asunto básico, tanto es así, que WiMAX necesita de una banda de frecuencia inferior a las que emplea actualmente para conseguir la ubicuidad de los sistemas 3G sin tener un coste de despliegue de red excesivamente elevado. Además, en el caso de que se permita a WiMAX emplear el espectro asignado a los actuales sistemas 3G por razones de neutralidad tecnológica, dicha decisión debería sustentarse en estudios que asegurasen que dichas tecnologías podrían compartir emplazamientos y frecuencias adyacentes sin causar interferencias mutuas.

Estos factores han provocado que ningún gran operador haya invertido en WiMAX tanto como en el año 2000 hicieron los operadores móviles en la tecnología UMTS (Sistema Universal de Telecomunicaciones Móviles) cuando ésta se encontraba en un estado de madurez similar. Incluso en Corea del Sur, donde se ha potenciado WiMAX a través de WiBro, la presencia de HSDPA como tecnología prioritaria es clara.

Pese a esta situación, WiMAX mantiene sus **ventajas tecnológicas** respecto a HSDPA: tasa máxima, All-IP y técnicas inalámbricas punteras como OFDM y MIMO. Sin embargo, y tal y como se comentó anteriormente, 3GPP se encuentra evolucionando continuamente HSDPA con el objetivo de alcanzar las ventajas de WiMAX.

Por tanto, aunque no parece que WiMAX vaya a sustituir a las redes 3G, no se pueden despreciar sus características y su gran cantidad de aplicaciones a nivel comercial, tales

como servicios de datos en entornos urbanos (mediante *hotspots* gigantes) o accesos a banda ancha en entornos rurales donde la tecnología xDSL no llega.

2.2.2 WiMAX Móvil vs Wi-Fi

WiMAX y Wi-Fi no son estándares enfrentados [9] [12], pues Wi-Fi (familia de estándares IEEE 802.11) vino a sustituir a las tradicionales LAN por cable (Redes de Área Local) por conexiones locales inalámbricas, mientras que WiMAX fue diseñado para **redes metropolitanas** (MAN) en la "última milla", tal y como se muestra en la figura 10:

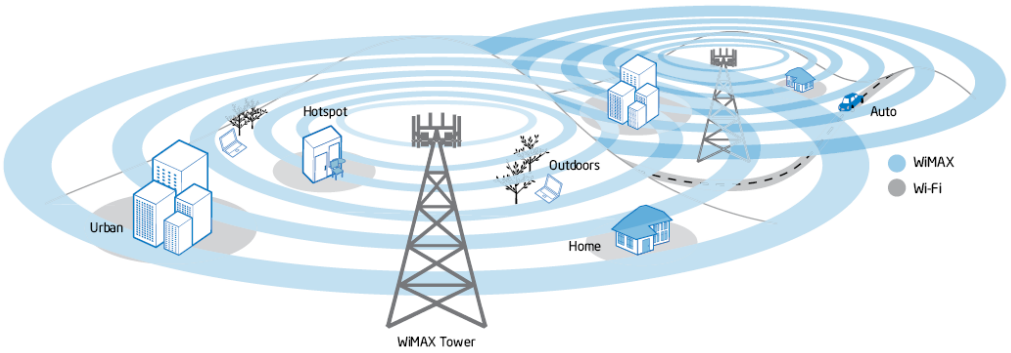

Figura 10: Cobertura WiMAX vs Wi-Fi
(Fuente: Intel [12])

El diseño de Wi-Fi se orienta a ambientes inalámbricos internos donde la capacidad sin línea de visión directa es de pocos metros, destacando IEEE 802.11b e IEEE 802.11g, los cuales usan la banda de 2.4 GHz (banda sin licencia y por ello sujeta a interferencias). Pese a su diseño, muchos ISP (Proveedor de Servicios de Internet) implementan sistemas Wi-Fi en la "última milla", con lo que la capacidad del servicio es limitada. Para adaptarse a las nuevas necesidades, Wi-Fi introdujo varias mejoras: seguridad, VLAN (Red de Área Local Virtual), soporte básico para servicios de voz (con QoS) y técnicas MIMO en el reciente IEEE 802.11n.

WiMAX, por el contrario, se diseño específicamente como solución de última milla en redes metropolitanas (MAN) y orientada tanto a los proveedores de servicio de Internet como a los suscriptores finales. El principal negocio es ofrecer a los ISP y telefónicas acceso para que éstos, a su vez, proporcionen banda ancha a sus clientes sin la necesidad de tener cableado en la última milla. Sin embargo, habrá situaciones en las cuales el acceso sea directamente al suscriptor a través de *hotspots* WiMAX.

Un aspecto interesante es el hecho de que Wi-Fi se encuentra diseñado para trabajar en rangos de frecuencia saturados o con interferencias al emplear espectro libre sin licencia. Sin embargo, WiMAX fue diseñado para operar en frecuencias de banda con licencia y

relativamente libre de interferencias, encontrándose dichas bandas en proceso de regulación. Además, WiMAX, a diferencia de Wi-Fi, necesita una planificación de radiofrecuencia compleja para ubicar las antenas, como en GSM (Sistema Global para las Comunicaciones Móviles).

Por tanto, tal y como se comentó anteriormente, Wi-Fi no será totalmente reemplazada por WiMAX, sino que constituirá una **extensión o superposición de las redes WiMAX** de tal forma que los usuarios podrán conectarse a la red que les proporcione mejores prestaciones en cada caso gracias a los dispositivos duales Wi-Fi/WiMAX que se están desarrollando.

2.2.3 WiMAX Móvil vs Mobile-Fi

En diciembre de 2002 comenzó el desarrollo del estándar IEEE 802.20 (Mobile-Fi) para la especificación de una interfaz radio (capa física y de control de acceso al medio) de transmisión eficiente de paquetes optimizada para servicios IP. El objetivo es el desarrollo de redes MBWA (Acceso Inalámbrico de Banda Ancha Móvil). Este estándar nació con el fin de complementar al estándar IEEE 802.16a que inicialmente estaba destinado a enlaces fijos FBWA (Acceso Inalámbrico de Banda Ancha Fijo). La evolución del estándar IEEE 802.16 hacia enlaces móviles (IEEE 802.16e-2005) implica que ambos estándares están orientados a cubrir el **mismo tipo de comunicaciones móviles**.

El estándar IEEE 802.20 está destinado a operar en bandas bajo licencia a velocidades de hasta 250 Km/h, empleando anchos de canal escalables entre 5-20 MHz e incorporando el uso de múltiples antenas tanto para realizar *beamforming* (conformación de haces) como para conformar canales MIMO. A diferencia de IEEE 802.16e-2005 (WiMAX Móvil), que es una evolución del IEEE 802.16d-2004 (WiMAX Fijo), IEEE 802.20 está creado desde cero y **orientado desde su inicio a comunicaciones móviles**. Este hecho permite que sea un estándar más eficiente y de mayor rendimiento, centrando sus ventajas en una mayor tasa para móviles con una velocidad de hasta 250 km/h (WiMAX Móvil soporta hasta 120 km/h) y un mayor radio de celda.

Sin embargo, su principal inconveniente frente a WiMAX son los continuos retrasos y polémicas en la publicación del estándar final, mientras que WiMAX Móvil lleva ya varios años en el mercado siendo **apoyado por la industria** y con unas características similares a Mobile-Fi [12].

Por tanto, no parece claro que Mobile-Fi suponga una competencia directa para WiMAX Móvil, aunque se deben apreciar sus cualidades en entornos de alta movilidad (más de 120 km/h).

2.3 Primeras implantaciones de WiMAX en España

El interés de las operadoras de telecomunicaciones por la tecnología WiMAX se demostró desde el momento de la aprobación del estándar IEEE 802.16d-2004 (WiMAX Fijo), siendo creciente a partir de la aprobación de IEEE 802.16e-2005 (WiMAX Móvil). Una muestra es el hecho de que se realizaron unas 150 pruebas piloto desde 2005 hasta 2007 a nivel mundial [7] (principalmente referidas a WiMAX Fijo), tal y como se muestra en la siguiente figura 11:

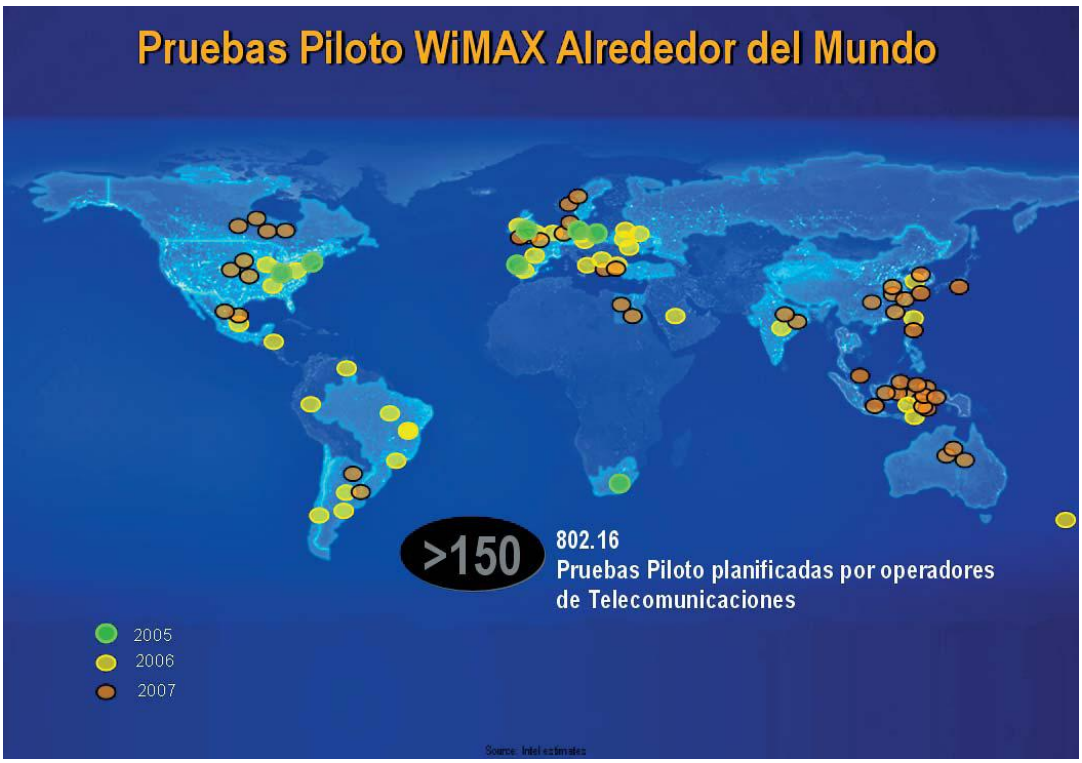

Figura 11: Primeras pruebas piloto WiMAX a nivel mundial
(Fuente: Fundación Telefónica [7])

Las operadoras e instituciones españolas no han sido ajenas a esta tecnología, realizándose **muchas de las primeras pruebas piloto a nivel mundial en el territorio nacional** [9]. Así pues, los principales programas, pruebas o explotaciones comerciales de WiMAX Fijo en España, por orden cronológico, fueron:

- Cataluña: a finales del año 2003, el Centre de Telecomunications de la Generalitat de Catalunya adjudicó a Iberbanda el concurso público para proporcionar cobertura de banda ancha a zonas rurales de las provincias de Lleida y Tarragona, siendo considerada la solución WiMAX presentada por Iberbanda como la más adecuada para dar una respuesta eficaz a las necesidades de conectividad de estas zonas rurales.

- Andalucía: en **Julio y Agosto de 2005**, Iberbanda comenzó en Andalucía la **primera experiencia comercial de WiMAX con tecnología Intel en Europa** instalando equipos

de Alvarion con chip Rosedale de Intel para unos cincuenta clientes de diferentes localidades de Almería (Canjayar, Instinción, Ohanes, Padules y Rágol). De especial interés es el chip Rosedale de Intel, presentado en septiembre de 2004, pues se trató del primer chip de bajo coste desarrollado y puesto a la venta por Intel para conexiones WiMAX compatible con el estándar IEEE 802.16d-2004 (WiMAX Fijo).

- Valencia: en Septiembre de 2005, el instituto ITACA (Instituto de Aplicaciones de las Tecnologías de la Información y de las Comunicaciones Avanzadas) de la Universidad Politécnica de Valencia instaló una antena de emisión WiMAX, alcanzando una cobertura de 20 kilómetros a la redonda con una velocidad media de 10 Mbps.

- Baleares: en **Septiembre de 2005**, Telefónica realizó la **primera experiencia a nivel mundial de navegación simultánea por el mar e Internet**. La estación base fue situada en la Sierra de Na Burguesa, Palma de Mallorca, dando cobertura a toda la bahía de Palma. El equipo receptor WiMAX (CPE) se instaló en la parte superior del palo de mesana (el trasero) del velero Rafael Verdera, de 23 metros de eslora, consiguiendo navegar en todo momento con tasas medias de 2 Mbps.

- Castilla y León: en Noviembre de 2005 fue presentado el programa de Banda Ancha 2005 – 2007 con el objetivo de que todos los municipios de la comunidad, fueran de zonas rurales o urbanas, contasen con las mismas oportunidades de acceder a los servicios de banda ancha. Iberbanda fue seleccionado para llevar WiMAX a las zonas rurales de esta comunidad autónoma.

- Galicia: el operador galleo R ofrece servicios basados en WiMAX desde el año 2005.

- Navarra: Iberbanda, a principios del año 2006, recibió del Gobierno de Navarra el encargo de desplegar la tecnología WiMAX en 855 municipios rurales para permitir el acceso a la banda ancha a más de 75.000 ciudadanos con una velocidad de entre 512 kbps y 4 Mbps.

- Aragón: Telefónica y Embou (operadora aragonesa de banda ancha mediante tecnología inalámbrica), con financiación del Gobierno de Aragón, desplegaron en la primera mitad de 2006 una red WiMAX piloto con el objetivo de hacer llegar la banda ancha a las zonas rurales (concretamente en las comarcas de Sobrarbe y Ribagorza). Estas pruebas piloto contaban con 19 postes que proporcionaban una cobertura del 90% de la población a 27 municipios de estas comarcas.

- País Vasco: en esta comunidad, Euskaltel comenzó en el año 2006 a ofrecer conexiones WiMAX de manera comercial.

3. Caracterización de la plataforma empleada

La plataforma empleada para la realización de este proyecto consta de dos elementos fundamentales: ***VHS-ADC Virtex-4* de Lyrtech** (hardware) [2] y ***System Generator for DSP* de Xilinx** (software) [3].

La *VHS-ADC Virtex-4* consiste, básicamente, en una placa de adquisición de datos analógicos con una FPGA de alta capacidad integrada y plena compatibilidad con *System Generator for DSP*. Éste se trata de un software que, **funcionando conjuntamente con MATLAB® y Simulink®** [13] y, mediante unos bloques propios, permite construir sistemas DSP empleando FPGA, realizando las simulaciones previas necesarias a la síntesis del modelo, esto es, la **generación del código VHDL** (Lenguaje de Descripción de Soportes Físicos para Circuitos Integramos Muy Rápidos).

Por tanto, mediante el empleo de estos dos elementos se puede realizar el proceso completo de prototipado de un sistema real DSP: diseño, simulación y síntesis.

En los siguientes apartados se ofrece una descripción del *VHS-ADC Virtex-4*, el *System Generator for DSP* y de los bloques de dicho software empleados en el modelo. Además, **en los Anexos B, C y D se ofrece más información sobre FPGA, MATLAB® y Simulink®**.

3.1 VHS-ADC Virtex-4 de Lyrtech

Se denomina *VHS-ADC Virtex-4* [2] a una placa de adquisición de datos analógicos, comercializada por Lyrtech, con una FPGA de alta capacidad integrada y plena **compatibilidad con el software MATLAB®, Simulink® y *System Generator for DSP*** (figura 12). Esta compatibilidad permite diseñar modelos basados en bloques que se pueden compilar automáticamente en la FPGA sin pérdida de prestaciones respecto a diseños realizados en VHDL.

Figura 12: VHS-ADC Virtex-4
(Fuente: Lyrtech [2])

Esta placa cuenta con las siguientes **características**:

- Conexión al PC mediante Compact PCI (Interconexión de Componentes Periféricos Compactos) del tipo 6U, lo cual le permite instalarse en un chasis cPCI estándar.

- 8 ADC (Conversores Analógico Digital) integrados, con una capacidad máxima, por cada uno, de 105 megamuestras por segundo (regulable entre 1 MHz y 105 MHz) y una resolución de 14 bits por muestra (2 muestras por cada palabra de 32 bits). Además, la ganancia es independiente para cada canal y programable por software, siendo la entrada del tipo 50 Ω MMCX (Coaxial Micro Miniaturizado).

- FPGA Virtex-4 de bajo consumo integrada en la placa con 256 GMACS (Gigaoperaciones Acumuladas de Multiplicación Por Segundo) y 152.000 células lógicas.

- Memoria SDRAM (Memoria Dinámica de Acceso Aleatorio Síncrona) integrada de 128 MB (Megabytes) de capacidad para almacenar y reproducir datos mediante una herramienta software incluida.

- Puertos digitales RapidCHANNEL (uno de transmisión y otro de recepción) que alcanzan hasta 1 GBps (Gigabyte Por Segundo) en full-dúplex.

- Posibilidad de utilizar la placa sin una conexión cPCI mediante el empleo de la memoria flash que posee integrada para la FPGA y el puerto externo I^2C/JTAG.

- Conector de expansión que permite agregar 8 canales adiciones de entrada/salida (ADC/DAC) y varios gigabytes de memoria DDR2 SDRAM (SDRAM de Tasa Doble de Transferencia de Datos).

- Control de los parámetros de la placa mediante una herramienta software.

- Kit software de desarrollo para la placa basado en los lenguajes de programación VHDL y C.

- Integración completa con MATLAB®, Simulink® y *System Generator for DSP*, lo cual permite un alto nivel de abstracción en el diseño de modelos y su compilación automática en la FPGA, generándose el código VHDL correspondiente.

A modo de resumen se muestran en la figura 13 los diferentes bloques que conforman la *VHS-ADC Virtex-4*:

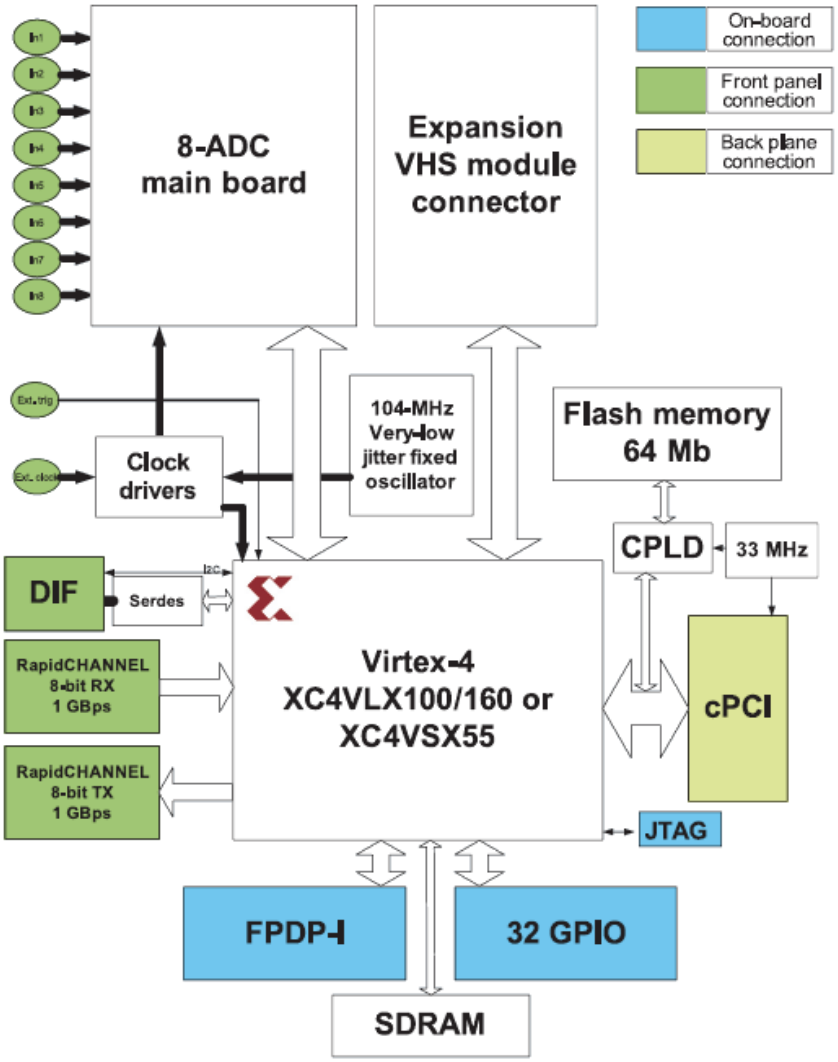

Figura 13: Diagrama de bloques VHS-ADC Virtex-4

(Fuente: Lyrtech [2])

Las **aplicaciones** más importantes en las cuales destaca la *VHS-ADC Virtex-4* debido a sus características son las siguientes:

- Equipos SDR (Equipos de Radio Definidos por Software).
- Antenas inteligentes, sistemas a frecuencia intermedia multicanal y conformadores de haces (*beamformers*).
- Grabación y reproducción multicanal de alta velocidad.
- Sistemas de medida y test de alta velocidad.
- Geolocalización basada en DOA (Dirección de Llegada), TDOA (Diferencia en Tiempo de Llegada) y ADOA (Diferencia en Amplitud de Llegada).

- *Routers* inalámbricos.
- Comunicaciones por satélite, radar y radar basado en una agrupación de antenas controladas por fase (*phased-array radar*).
- Medicina, tomografías y ultrasonidos.

Además de las aplicaciones en las cuales la *VHS-ADC Virtex-4* se encuentra consolidada, existen una serie de **áreas potenciales de aplicación** que están siendo impulsadas actualmente:

- STC (Codificación Espacio Tiempo) para sistemas MIMO.
- Diversidad de antenas para OFDM.
- Triangulación multicanal para el estándar IEEE 802.11n (Wi-Fi).
- Conformado de haces para la familia de estándares IEEE 802.16 (WiMAX) basado en DOA.
- Procesamiento multicanal basado en FPGA.

3.2 System Generator for DSP

System Generator for DSP [3] se trata de una herramienta software, comercializada por Xilinx, cuya función es ofrecer una visión de alto nivel de un sistema DSP e implementarlo en una FPGA, **generando el código VHDL correspondiente**. Para ello **se integra con el software MATLAB® y Simulink®**, tal y como se muestra en la figura 14:

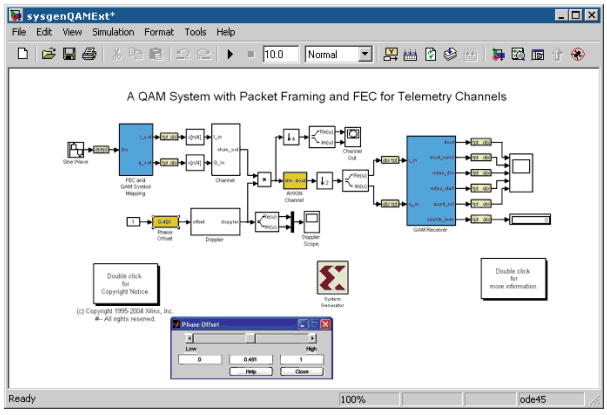

Figura 14: Ejemplo de diseño mediante System Generator for DSP
(Fuente: Xilinx [3])

Esta herramienta cuenta con las siguientes **características**:

- Soporte para diferentes familias de dispositivos:
  - Virtex-5 LX, LXT, SXT, FXT.
  - Virtex-4 FX, LX, SX.
  - Virtex-II Pro.
  - Virtex-II.
  - Virtex-E.
  - Spartan-3A DSP.
  - Spartan-3A, AN.
  - Spartan-3, 3E.
  - Spartan-II, IIE.
- Diseño, simulación y generación de código VHDL o Verilog para sistemas DSP modelados en Simulink®. Para ello se suministra la biblioteca, *Xilinx Blockset*, cuyos bloques contienen funciones para:
  - Operaciones aritméticas y lógica digital.
  - Procesamiento de señal: filtros FIR (Respuesta al Impulso Finita), FFT...
  - Corrección de errores: decodificador Viterbi, codificador/decodificador Reed-Solomon.
  - Memorias: FIFO (Primero en Entrar, Primero en Salir), RAM (Memoria de Acceso Aleatorio), ROM (Memoria de Sólo Lectura)...
  - Importación de funciones de MATLAB® y módulos HDL (Lenguaje de Descripción de Soportes Físicos).
- Simulación del sistema sobre la propia FPGA (co-simulación) mediante una opción de generación de código. Esto permite realizar una simulación del modelo ya implementado en la FPGA empleando Simulink® como interfaz, el cual gestiona la entrada y salida de datos de la tarjeta.

3.3 Descripción de los bloques empleados

La finalidad de este apartado es ofrecer una explicación de los diferentes bloques empleados en el receptor diseñado pertenecientes al ***Xilinx Blockset* de *Xilinx System Generator for DSP*** [3]. Con este objetivo se ofrecerá, para cada uno de ellos, una descripción de sus características generales, peculiaridades de funcionamiento y un modelo de demostración en el cual se podrá verificar todo lo anterior.

Para la realización de estos modelos demostrativos se han empleado, en la medida de lo posible, bloques Xilinx pertenecientes al receptor diseñado y, por tanto, explicados es este apartado. No obstante existen dos bloques no presentes en el receptor y que sí son empleados en los modelos de ejemplo: *Xilinx Counter* y *Xilinx Convert*:

- ***Xilinx Counter***:

  El bloque *Xilinx Counter* (figura 15) permite implementar un contador libre o limitado con un número de bits configurable. Además, para cada uno de ellos, puede elegirse el sentido de la cuenta (ascendente o descendente), el tipo de dato de salida (con signo o sin signo), el valor inicial y el paso (valor del incremento o decremento).

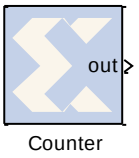

Figura 15: Bloque Xilinx Counter

- ***Xilinx Convert***:

  El bloque *Xilinx Convert* (figura 16) tiene la finalidad de modificar el tipo de dato que recibe a la entrada (con signo o sin signo) y su número de bits, permitiendo emplear diferentes técnicas para la cuantificación. Este bloque es útil para adaptar las tipologías de los datos entre aquellas etapas que tengan diferentes requisitos.

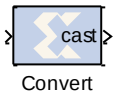

Figura 16: Bloque Xilinx Convert

3.3.1 Bloque *Lyrtech VHS-ADAC Board Configuration*

a) Descripción

El bloque *Lyrtech VHS-ADAC Board Configuration* (figura 17) permite configurar el hardware sobre el cual será sintetizado el modelo diseñado mediante *System Generator for DSP*.

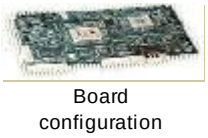

Figura 17: Bloque Lyrtech VHS-ADAC Board Configuration

b) Configuración básica

La configuración básica del bloque se muestra mediante la siguiente captura (figura 18):

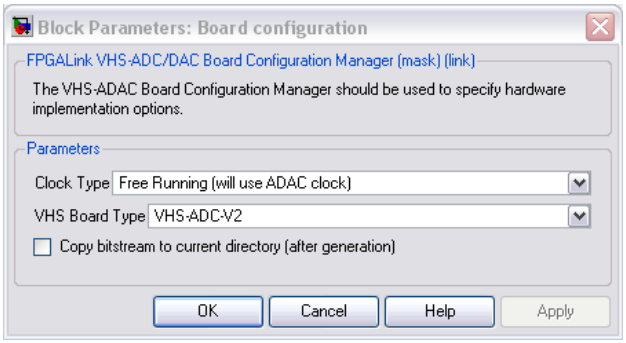

Figura 18: Configuración básica del bloque Lyrtech VHS-ADAC Board Configuration

- `Clock Type` (tipo de reloj): reloj que empleará el diseño basado en *System Generator for DSP*. Este reloj puede ser `Single Step` (empleado para la co-simulación o simulación sobre la FPGA) o `Free Running` (utilizado para el resto de operaciones con el modelo).

- `VHS Board Type` (tipo de placa VHS): placa empleada por el modelo basado en *System Generator for DSP* (`VHS-ADC-V2`, `VHS-DAC-V2` ó `VHS-ADC-V4`).

- `Copy bitstream to current directory` (copiar flujo de bits al directorio actual): indica si se desea copiar el flujo de bits de la FPGA al directorio de trabajo actual de MATLAB®.

c) Observaciones

- Este bloque es proporcionado por el fabricante del hardware, en este caso Lyrtech, y no por Xilinx. Se trata del bloque que acompaña al *Xilinx System Generator*.

- Las restricciones del reloj en el diseño se especifican mediante el parámetro FPGA `Clock Period` (periodo de reloj de la FPGA) en la configuración del bloque *Xilinx System Generator*.

- Este bloque no es necesario para aquellos modelos realizados mediante bloques pertenecientes al *Xilinx Blockset* y que únicamente se quieran simular. Si se desea realizar cualquier otra operación que implique hardware, será necesario que este bloque esté presente.

3.3.2 Bloque *Xilinx System Generator*

a) Descripción

El bloque *Xilinx System Generator* (figura 19) proporciona el control del sistema y de los parámetros de simulación a nivel general del modelo diseñado. Además, mediante este bloque se puede indicar cómo se realizará la generación de código, así como invocar al propio generador.

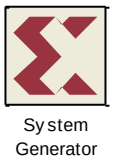

Figura 19: Bloque Xilinx System Generator

b) Configuración básica

La configuración básica del bloque se muestra mediante la siguiente captura (figura 20):

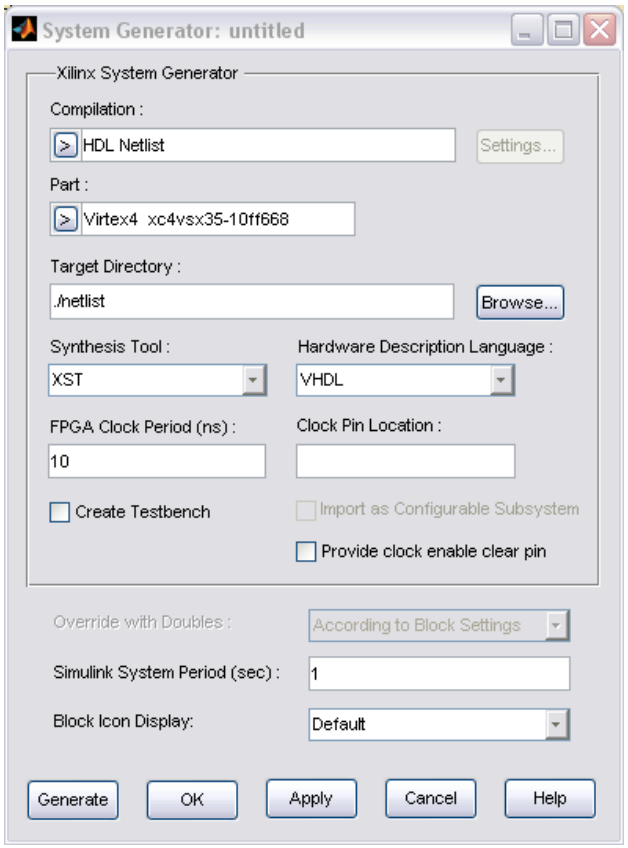

Figura 20: Configuración básica del bloque Xilinx System Generator

- `Compilation` (compilación): tipo de compilación resultante tras invocar al generador de código: `HDL Netlist`, `NGC Netlist`, `Bitstream`, `EDK Export Tool`, `Hardware Co-Simulation`, `Lyrtech` o `Time Analysis`.

- `Part` (componente): tipo de FPGA empleada: `Spartan2`, `Spartan2E`, `Spartan3`, `Spartan3E`, `Virtex`, `VirtexE`, `Virtex2`, `Virtex2P` o `Virtex4`.

- `Target directory` (directorio objetivo): directorio en el cual *System Generator* escribirá los resultados de la compilación.

- `Synthesis Tool` (herramienta de síntesis): herramienta empleada para sintetizar el modelo diseñado: `Synplify`, `Synplify Pro` (ambas de Synplicity) o `XST` (de Xilinx).

- `Hardware Description Language` (lenguaje de descripción de hardware): lenguaje HDL utilizado para compilar el diseño: `VHDL` o `Verilog`.

- `FPGA Clock Period` (periodo de reloj de la FPGA): valor, en nanosegundos, del período del reloj del hardware. Este valor (no necesariamente un entero) es enviado a las herramientas de implementación de Xilinx mediante un fichero de restricciones y es usado como el periodo global, empleado el resto de elementos múltiplos enteros del mismo.

- `Clock Pin Location` (localización del pin del reloj): posición del pin del reloj hardware. Esta información es enviada a las herramientas de implementación de Xilinx mediante un fichero de restricciones.

- `Create Testbench` (crear banco de pruebas): ordena a *System Generator* la creación de un banco de pruebas en HDL, comparando los resultados de la simulación en Simulink® con los obtenidos mediante la versión compilada del modelo.

- `Import as Configurable Subsystem` (importar como un subsistema configurable): indica a *System Generator* que se desea crear un subsistema configurable.

- `Provide clock enable clear pin` (proporcionar pin de borrado de habilitado del reloj): ordena a *System Generator* a proveer de un puerto para manejar la lógica de funcionamiento del reloj.

- `Override with Doubles` (anular con precisión doble): especifica que todos los cálculos realizados en el ámbito del bloque deberían ser realizados con aritmética de precisión doble.

- `Simulink System Period` (período del sistema Simulink): este periodo, en segundos, se define como el máximo común divisor de todos los periodos de muestreo que aparecen en el modelo.

- `Block Icon Display` (visualización del icono de bloque): tipo de información mostrada en el icono de los bloques, siendo actualizada tras la compilación del modelo. Las opciones de visualización incluyen: vista por defecto, tasas de muestreo (periodos de muestreo normalizados al valor de `Simulink System Period`), nombres en HDL de los puertos, tipos de datos a la entrada y tipos de datos a la salida.

c) Observaciones

- Este bloque debe estar presente siempre que en un modelo Simulink® aparezca un bloque del *Xilinx Blockset*, aunque sólo se desee simular dicho modelo.

3.3.3 Bloque *Xilinx WaveScope*

a) Descripción

El bloque *Xilinx WaveScope* (figura 21) permite observar, tras la simulación del modelo, los valores de las señales en las líneas que conectan los bloques.

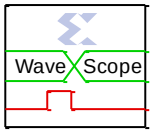

Figura 21: Bloque Xilinx WaveScope

Este bloque permite ver simultáneamente varias señales y configurar esta visualización de la manera más adecuada en cada caso.

b) Configuración básica

La configuración básica del bloque se muestra mediante la siguiente captura (figura 22):

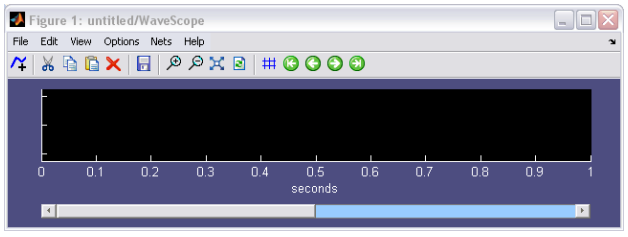

Figura 22: Configuración básica del bloque Xilinx WaveScope

Este bloque, al tratarse de un visualizador de señales, consta de una ventana en la cual se podrán observar dichas señales de diferentes maneras. Por tanto, la visualización y configuración se realizan en la misma ventana.

Básicamente, esta ventana permite:

- Seleccionar las señales a visualizar: el bloque detecta todas las señales del modelo para poder agregarlas, aunque también es posible añadirlas de forma gráfica. Dichas señales no tendrán valores asociados hasta que se realice la simulación del modelo. Además, siempre que exista una señal, aparecerá una señal de reloj que representa la mayor tasa de muestreo del sistema.

- Configurar la presentación de la señal: pulsando con el botón derecho del ratón sobre cualquiera de las señales agregadas, será posible modificar su formato (lógico o analógico), su base (binaria, decimal o hexadecimal; con o sin complemento a dos), su color y nombre.

- Ver las señales: las señales representadas se puede observar con diferentes niveles de zoom. Debido a que cuanto mayor sea la cantidad de datos representados, menor será el rendimiento de este bloque, se ofrece la posibilidad de seleccionar los valores que se grabarán y mostrarán (por defecto todos los de la simulación).

c) Observaciones

- La ventana de visualización de este bloque, en el caso de que estuviera cerrada, se abrirá automáticamente al final de la simulación del modelo.

- Es posible visualizar la misma señal varias veces, con el objetivo de observar sus valores de diferentes maneras.

3.3.4 Bloque *Xilinx Delay*

a) Descripción

El bloque *Xilinx Delay* (figura 23) consiste en una línea de retardo que permite agregar la latencia deseada a los modelos diseñados.

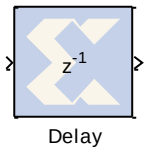

Figura 23: Bloque Xilinx Delay

Los datos obtenidos a la salida serán los mismos que a la entrada tras el retardo indicado.

b) Puertos de entrada y salida básicos

- `Input` (entrada): datos sobre cuales se desee aplicar un retardo.
- `Output` (salida): datos retardados.

c) Configuración básica

La configuración básica del bloque se muestra mediante la siguiente captura (figura 24):

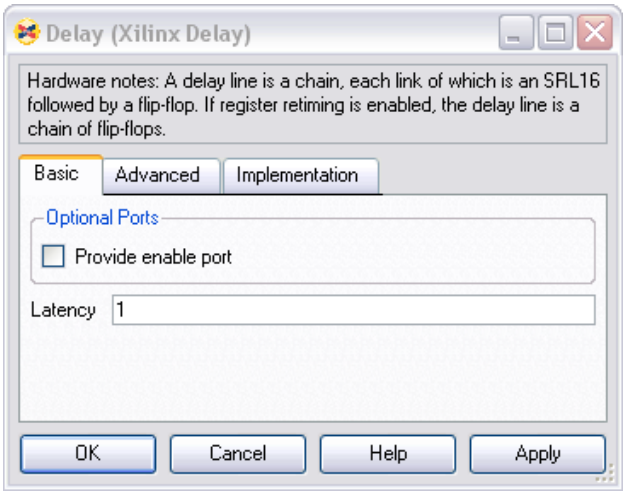

Figura 24: Configuración básica del bloque Xilinx Delay

- `Latency` (latencia): número de períodos de muestreo que se desea retardar los datos a la entrada.

d) Observaciones

- Existe la posibilidad de proporcionar un puerto de habilitado (`enable port`) a la entrada del bloque. Su señal de entrada debe ser de tipo booleano. Este puerto se proporciona en la configuración del bloque.

e) Ejemplo de funcionamiento

El funcionamiento del bloque *Xilinx Delay* se muestra con el siguiente modelo (figura 25):

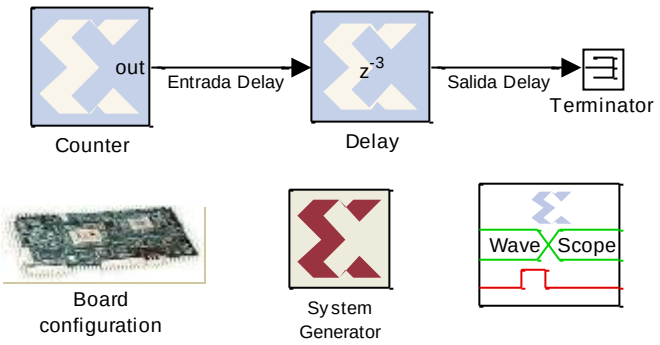

Figura 25: Ejemplo funcionamiento del bloque Xilinx Delay. Modelo

Este modelo consta de:

- Una fuente de datos que consiste en un bloque *Xilinx Counter* configurado como contador libre de 4 bits sin signo.
- Un bloque *Xilinx Delay* con un retardo de tres períodos de muestreo.
- Un bloque *Xilinx WaveScope* para observar la forma de las señales a la entrada y salida del bloque.

Este sistema realiza la sencilla operación de retrasar tres períodos de muestra los datos de entrada, tal y como se verifica en las figuras 26 y 27:

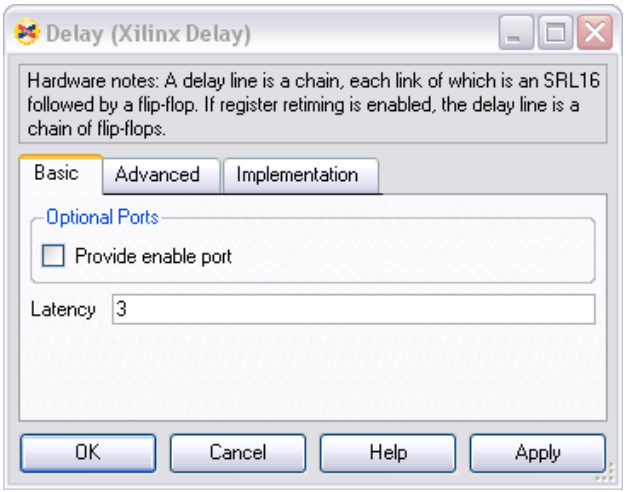

Figura 26: Ejemplo funcionamiento del bloque Xilinx Delay. Configuración básica del bloque

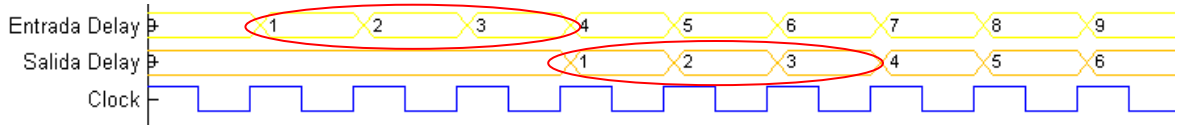

Figura 27: Ejemplo funcionamiento del bloque Xilinx Delay. Forma de la señales

3.3.5 Bloque *Xilinx Serial to Parallel*

a) Descripción

El bloque *Xilinx Serial to Parallel* (figura 28) toma una serie de entradas de cualquier tamaño y proporciona una única salida múltiplo del tamaño de la entrada.

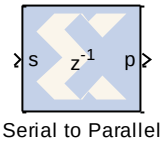

Figura 28: Bloque Xilinx Serial to Parallel

La serie de entradas puede ser ordenada con la palabra más significativa primero o la menos significativa.

b) Puertos de entrada y salida básicos

- `Input` (entrada): serie de datos de cualquier tamaño a partir de los cuales se desea obtener una palabra única múltiplo de su tamaño.
- `Output` (salida): datos agrupados con un tamaño múltiplo de los de entrada.

c) Configuración básica

La configuración básica del bloque se muestra mediante la siguiente captura (figura 29):

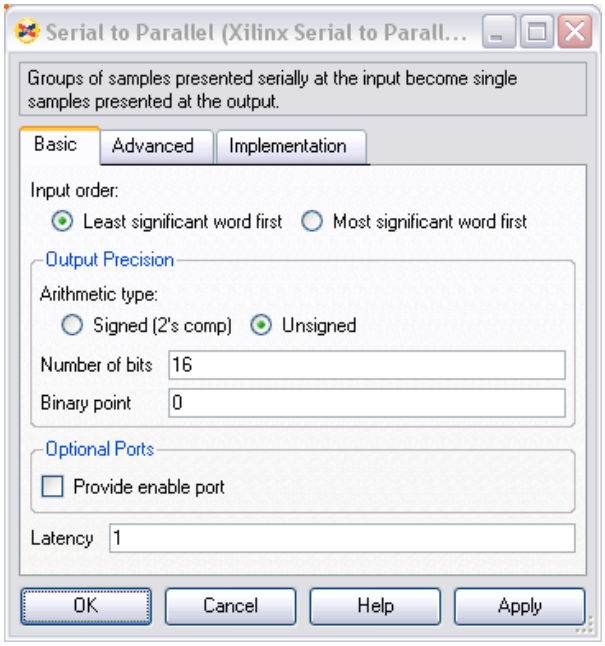

Figura 29: Configuración básica del bloque Xilinx Serial to Parallel

- `Input order` (orden de entrada): orden que se desea aplicar a los datos de entrada, esto es, primero la palabra (que no bit) menos significativa o la más significativa.

- `Arithmetic type` (tipo de aritmética): utilización de signo (complemento a dos) o no para los datos de salida.

- `Number of bits` (número de bits): tamaño de la salida, la cual debe ser múltiplo del tamaño de la entrada.

- `Binary point` (punto binario): posición del punto binario en los datos de salida.

d) Observaciones

- Existe la posibilidad de proporcionar un puerto de habilitado (`enable port`) a la entrada del bloque. Su señal de entrada debe ser de tipo booleano. Este puerto se proporciona en la configuración del bloque.

- El retardo mínimo de este bloque es el necesario para que todos los bits que se desean agrupar entren en el bloque, proporcionándose la salida en momento en el cual se dispone de todos los bits necesarios. Además, el valor del retardo se puede aumentar en la configuración del bloque.

e) Ejemplo de funcionamiento

El funcionamiento del bloque *Xilinx Serial to Parallel* se ilustra con el siguiente sistema (figura 30):

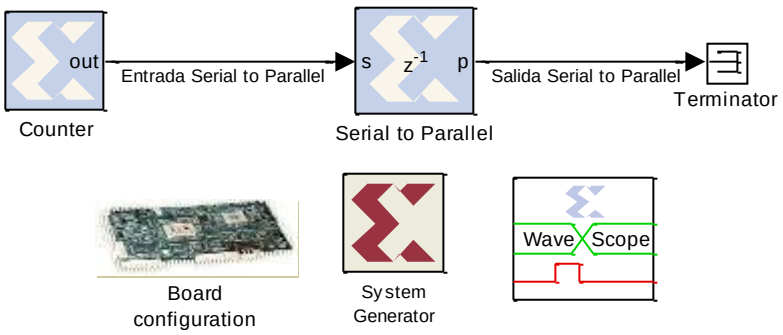

Figura 30: Ejemplo funcionamiento del bloque Xilinx Serial to Parallel. Modelo

Este esquema consta de los siguientes elementos:

- Una fuente de datos que consiste en un bloque *Xilinx Counter* configurado como contador libre de 2 bits sin signo. Por tanto la entrada serie al bloque *Xilinx Serial to Parallel* será de 2 bits.
- Un bloque *Xilinx Serial to Parallel* que ordena la entrada con la palabra menos significativa primero y una palabra de salida de 4 bits.
- Un bloque *Xilinx WaveScope* para observar la forma de las señales a la entrada y salida del bloque.

Este modelo tiene una fuente de datos de 2 bits y una salida de 4 bits, siendo agrupados por el bloque *Xilinx Serial to Parallel*. Dicha salida, tal y como se aprecia en las figuras 31 y 32, se ordena con la palabra menos significativa de entrada primero y se proporciona en el momento en el cual se disponen de todos los datos necesarios.

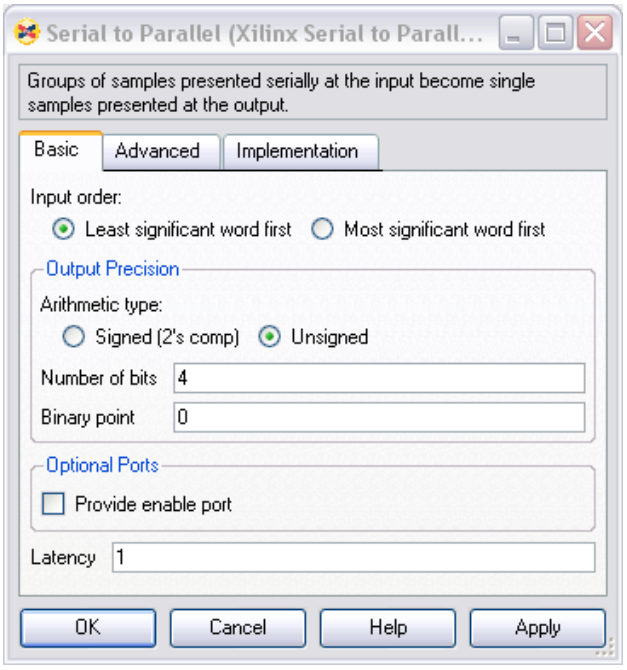

Figura 31: Ejemplo funcionamiento del bloque Xilinx Serial to Parallel. Configuración básica del bloque

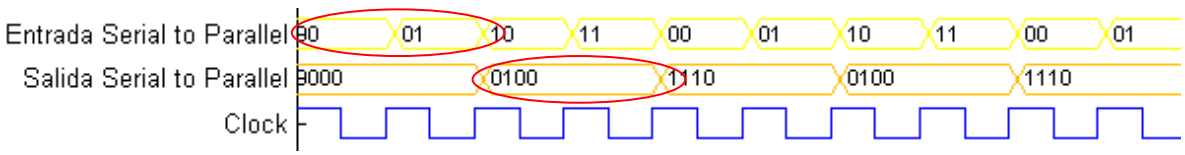

Figura 32: Ejemplo funcionamiento del bloque Xilinx Serial to Parallel. Forma de la señales

3.3.6 Bloque *Xilinx Parallel to Serial*

a) Descripción

El bloque *Xilinx Parallel to Serial* (figura 33) toma una palabra a la entrada de cualquier tamaño y la divide en N palabras de salida multiplexadas en tiempo, siendo N la relación entre el número de bits de la palabra de entrada y la de salida.

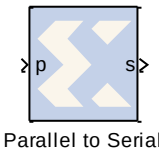

Figura 33: Bloque Xilinx Parallel to Serial

La salida puede ser ordenada con la palabra más significativa primero o la menos significativa.

b) Puertos de entrada y salida básicos

- `Input` (entrada): palabra de cualquier tamaño a partir de la cual se desea obtener palabras de un tamaño inferior.
- `Output` (salida): palabra obtenida con el número de bits deseado.

c) Configuración básica

La configuración básica del bloque se muestra mediante la siguiente captura (figura 34):

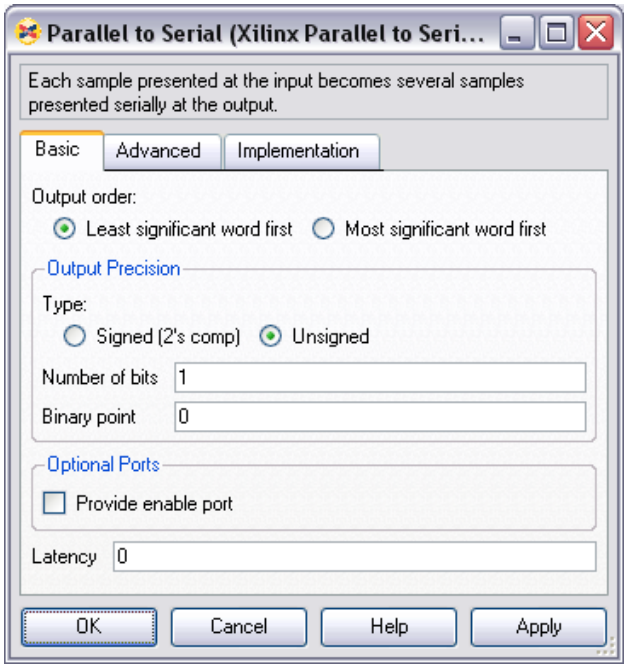

Figura 34: Configuración básica del bloque Xilinx Parallel to Serial

- `Output order` (orden de salida): orden que se desea aplicar a los datos de salida, esto es, primero la palabra (que no bit) menos significativa o la más significativa.
- `Type` (tipo de aritmética): utilización de signo (complemento a dos) o no para los datos de salida.
- `Number of bits` (número de bits): tamaño de la salida, la cual debe cumplir que el número de bits de la palabra de entrada sea múltiplo de los de la palabra de salida.
- `Binary point` (punto binario): posición del punto binario en los datos de salida.

d) Observaciones

- Existe la posibilidad de proporcionar un puerto de habilitado (`enable port`) a la entrada del bloque. Su señal de entrada debe ser de tipo booleano. Este puerto se proporciona en la configuración del bloque.
- Este bloque no produce retardo. Sin embargo, se puede agregar un valor de retardo en la configuración de bloque.

e) Ejemplo de funcionamiento

El funcionamiento del bloque *Xilinx Parallel to Serial* se muestra mediante el siguiente modelo de ejemplo (figura 35):

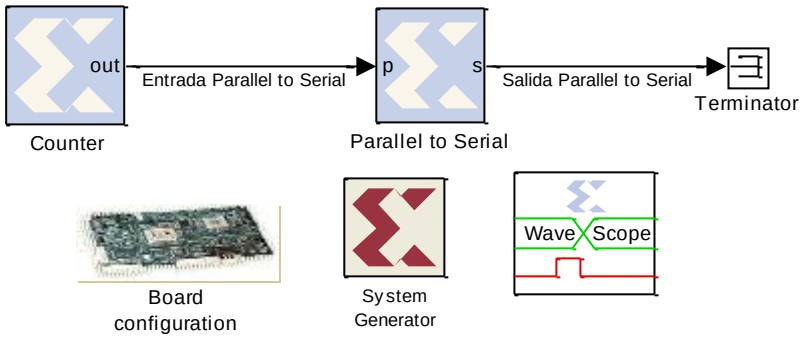

Figura 35: Ejemplo funcionamiento del bloque Xilinx Parallel to Serial. Modelo

Este modelo está formado por:

- Una fuente de datos que consiste en un bloque *Xilinx Counter* configurado como contador libre de 4 bits sin signo. Por tanto, la entrada al bloque *Xilinx Parallel to Serial* será de 4 bits.

- Un bloque *Xilinx Parallel to Serial* ordenando la entrada con la palabra menos significativa primero y una palabra de salida de 2 bits.

- Un bloque *Xilinx WaveScope* para observar la forma de las señales a la entrada y salida del bloque.

Este modelo consta de una fuente de datos de 4 bits y una salida de 2 bits, siendo divididos por el bloque *Xilinx Parallel to Serial* (tal y como se observa en las figuras 36 y 37). Además, dicha salida se ordena con la palabra de entrada menos significativa primero sin producirse ningún retardo.

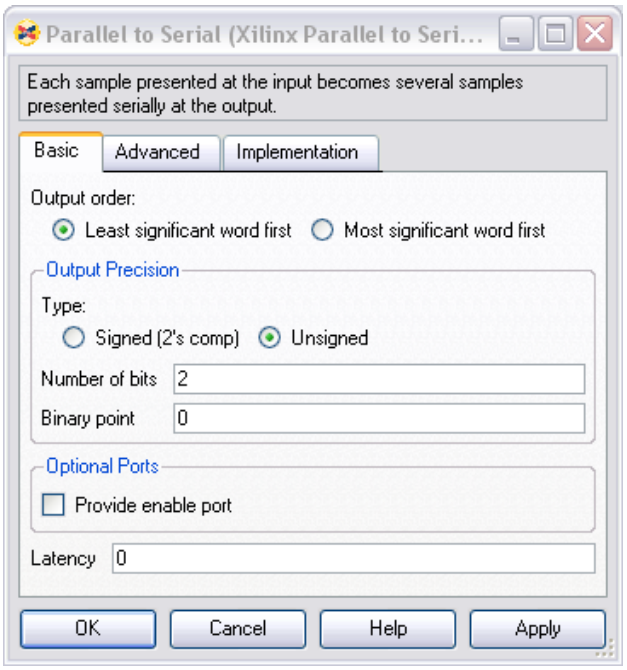

Figura 36: Ejemplo funcionamiento del bloque Xilinx Parallel to Serial. Configuración básica del bloque

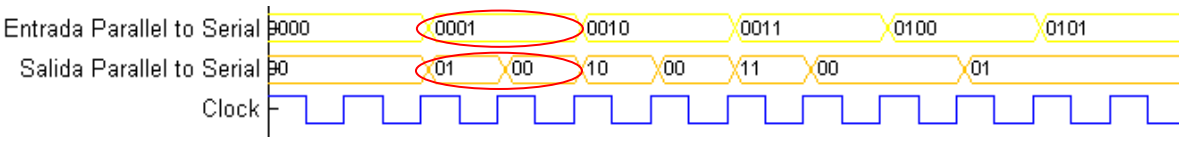

Figura 37: Ejemplo funcionamiento del bloque Xilinx Parallel to Serial. Forma de la señales

3.3.7 Bloque *Xilinx BitBasher*

a) Descripción

El bloque *Xilinx BitBasher* (figura 38) permite realizar operaciones a nivel de bit: extraer, concatenar y repetir. Estas operaciones se describen mediante sintaxis Verilog.

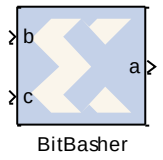

Figura 38: Bloque Xilinx BitBasher

El número de puertos de salida que proporciona el bloque es igual al número de expresiones, con un máximo de cuatro.

b) Puertos de entrada y salida básicos

- `Input` (entrada): datos sobre los cuales se desean realizar operaciones de extracción, concatenación y repetición a nivel de bit.
- `Output` (salida): datos resultantes de las operaciones definidas. Se pueden definir hasta cuatros puertos de salida.

c) Configuración básica

La configuración básica del bloque se muestra mediante la siguiente captura (figura 39):

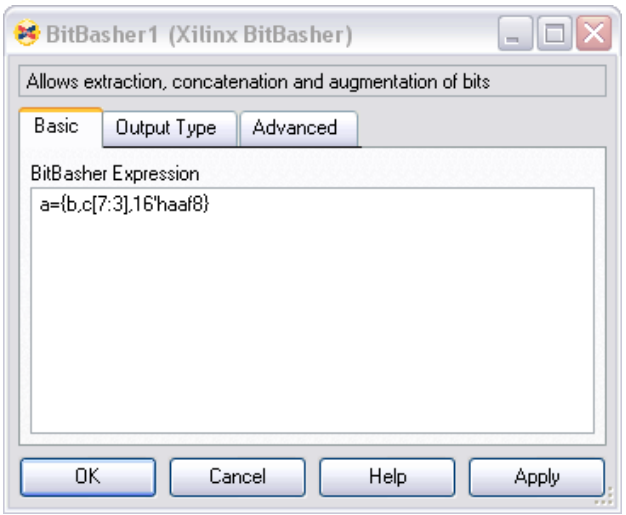

Figura 39: Configuración básica del bloque Xilinx BitBasher (1)

- `BitBasher Expression` (expresión del BitBasher): expresión de manipulación de los bits de entrada basada en la sintaxis Verilog. Es posible especificar múltiples

expresiones (hasta un límite de 4) empleando una nueva línea como separador entre las mismas.

La configuración del tipo de salida del bloque se muestra mediante la siguiente captura (figura 40):

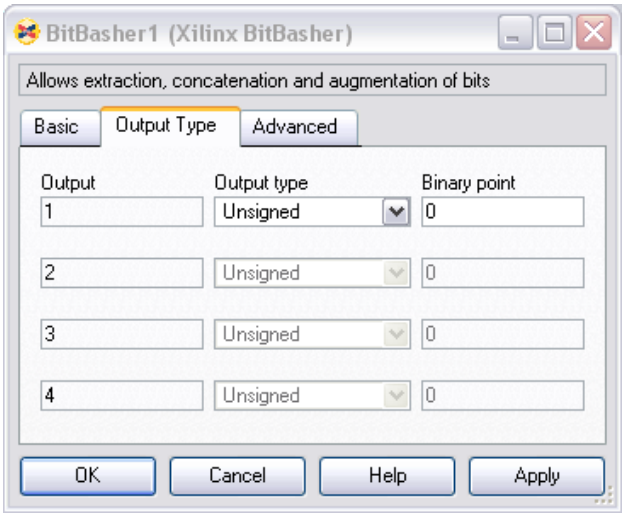

Figura 40: Configuración básica del bloque Xilinx BitBasher (2)

- `Output` (salida): puerto de salida del cual se especificará el tipo y el punto binario.
- `Output type` (tipo de salida): utilización de signo (complemento a dos) o no para los datos de salida.
- `Binary point` (punto binario): posición del punto binario en los datos de salida.

d) Observaciones

- Este bloque únicamente soporta un subconjunto de las expresiones Verilog que permiten realizar operaciones a nivel de bit, incluyendo operadores para extraer, concatenar, repetir y definir constantes.
- Una expresión no puede contener únicamente constantes, es decir, deben aparecer involucrados datos de entrada. Por tanto, el bloque *Xilinx BitBasher* no puede ser empleado como una fuente de contantes de manera análoga al bloque *Xilinx Constant*.

e) Ejemplo de funcionamiento

El funcionamiento del bloque *Xilinx BitBasher* se ilustra con el siguiente diseño (figura 41):

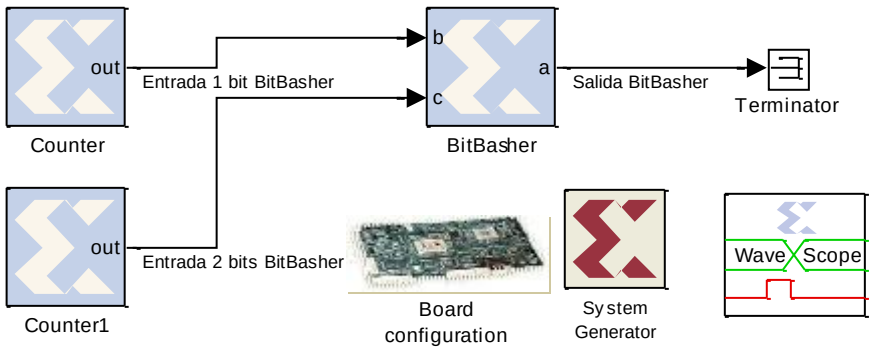

Figura 41: Ejemplo funcionamiento del bloque Xilinx BitBasher. Modelo

Dicho sistema consta de:

- Dos fuentes de datos que consisten en dos bloques *Xilinx Counter* configurados como contadores libres de 1 y 2 bits sin signo. Dichas fuentes serán las entradas al bloque *Xilinx BitBasher*.

- Un bloque *Xilinx BitBasher* que concatena y reordena las entradas situando el punto binario en el bit 1.

- Un bloque *Xilinx WaveScope* para observar la forma de las señales a la entrada y salida del bloque.

En las figuras 42, 43 y 44 se puede observar cómo se concatenan y reordenan los datos de entrada de la manera indicada en la expresión del bloque *Xilinx BitBasher*. Además, el punto binario de la salida se encuentra en el bit 1 y sólo hay una salida, pues únicamente se ha formulado una expresión en el bloque.

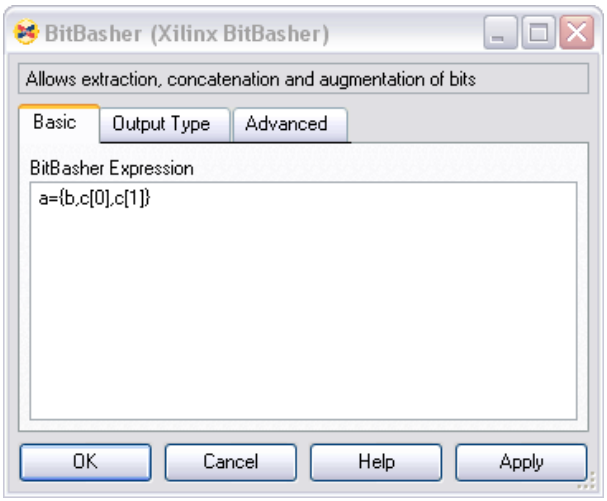

Figura 42: Ejemplo funcionamiento del bloque Xilinx BitBasher. Configuración básica del bloque (1)

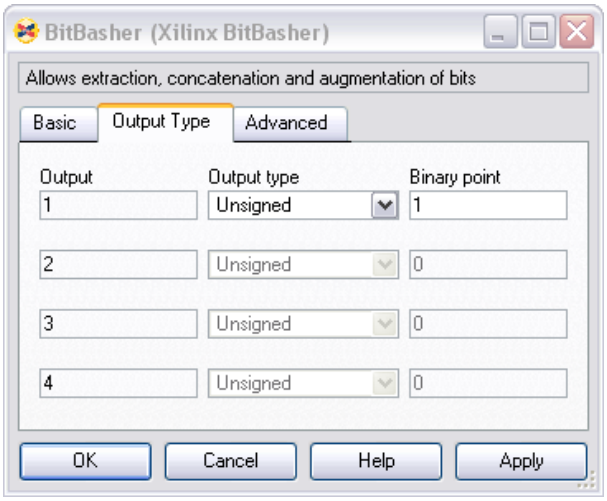

Figura 43: Ejemplo funcionamiento del bloque Xilinx BitBasher. Configuración básica del bloque (2)

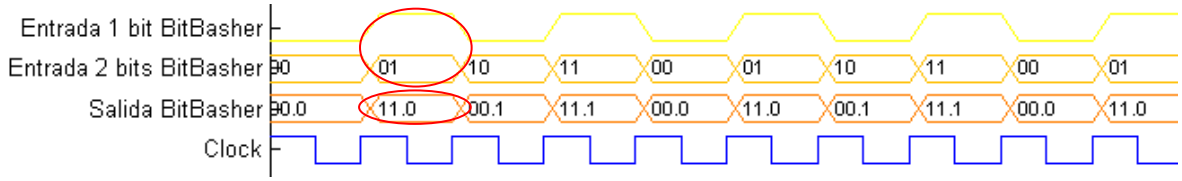

Figura 44: Ejemplo funcionamiento del bloque Xilinx BitBasher. Forma de la señales

3.3.8 Bloque *Xilinx MCode*

a) Descripción

El bloque *Xilinx MCode* (figura 45) permite ejecutar funciones MATLAB® diseñadas por el usuario, llamadas m-code. Así, al generar el hardware, el comportamiento del código es traducido a un equivalente en VHDL/Verilog.

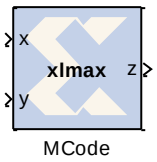

Figura 45: Bloque Xilinx MCode

Mediante este boque se puede construir de manera sencilla funciones aritméticas, máquinas de estados finitos y máquinas de control lógico.

Dicho bloque proporciona un puerto de entrada por cada parámetro de la función y uno de salida por cada uno de los valores que devuelve. Aunque pueden existir puertos de entrada opcionales.

b) Puertos de entrada y salida básicos

- `Input` (entrada): cada uno de los parámetros de la función.
- `Output` (salida): cada uno de los valores que devuelve la función.

c) Configuración básica

La configuración básica del bloque se muestra mediante la siguiente captura (figura 46):

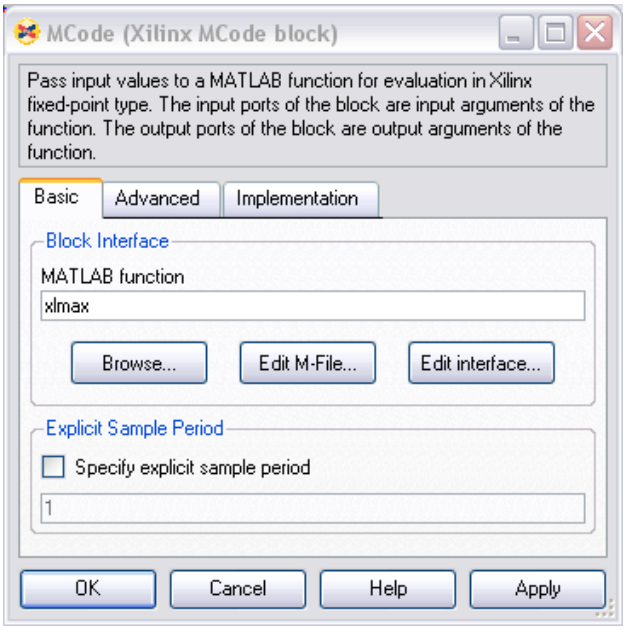

Figura 46: Configuración básica del bloque Xilinx MCode

- `MATLAB function` (función MATLAB): nombre del código de extensión .m que debe ejecutar el bloque.
- `Browse` (navegar): permite localizar el fichero que contiene el código.
- `Edit M-File` (editar fichero .m): accede al código de la función para su modificación mediante el editor de MATLAB®.
- `Edit interface` (editar interfaz): muestra la lista de entradas y salidas de la función con sus valores constantes, si los tuvieran.

d) Observaciones

- Este bloque no soporta el lenguaje MATLAB® por completo, sino un subconjunto del mismo. Además, cuenta con una serie de restricciones importantes en cuanto a su

uso, como que todas las entradas y salidas deben ser del tipo punto fijo y el bloque debe tener al menos una salida.

- La configuración del bloque permite activar dos herramientas para realizar la depuración del código: la impresión de mensajes por pantalla desde el código mediante la función `disp ()` y el depurador de MATLAB® (`Enable printing with disp` y `Enable MATLAB debugging`). Además, existe la posibilidad de establecer el periodo de muestro a un valor diferente del que se detecta a la entrada del bloque y se proporciona a la salida. Su valor se especifica en la configuración del bloque.

- Se puede usar la misma función en diferentes bloques *Xilinx MCode*, empleando diferentes parámetros en dicha función para que se comporte de forma diferente.

e) Ejemplo de funcionamiento

El funcionamiento del bloque *Xilinx MCode* se muestra mediante el siguiente modelo (figura 47):

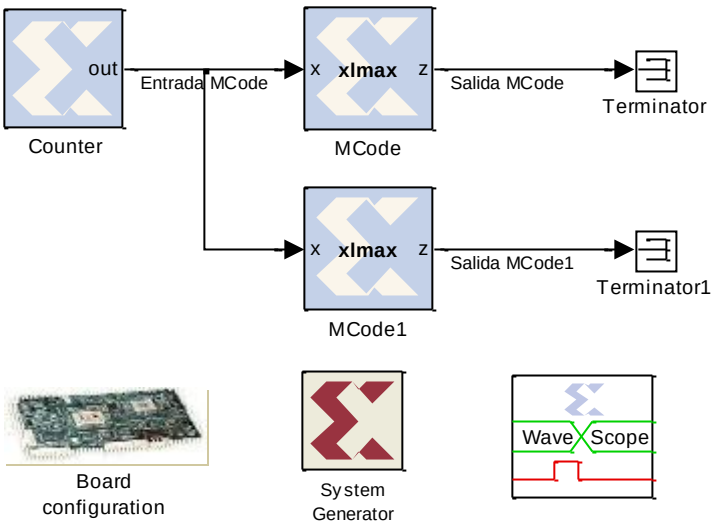

Figura 47: Ejemplo funcionamiento del bloque Xilinx MCode. Modelo

Este diseño se encuentra formado por:

- Una fuente de datos que consiste en un bloque *Xilinx Counter* configurado como contador libre de 3 bits sin signo. Esta fuente es la entrada a ambos bloques *Xilinx MCode*.

- Dos bloques *Xilinx MCode* que ejecutan la misma función (comparación de la entrada con un valor fijo mostrando a la salida el máximo) con diferentes parámetros (umbral de valor 2 y 5).

- Un bloque *Xilinx WaveScope* para observar la forma de las señales a la entrada y salida de los bloques.

Este modelo consta de dos bloques *Xilinx MCode* que ejecutan la misma función (la salida es el máximo de las entradas) aunque con diferentes parámetros. Para cada bloque, si se observa la función implementada, se han especificado 2 parámetros de entrada y un valor de salida. Sin embargo, una de las entradas se ha fijado, con lo cual no se ofrece como puerto externo, haciendo la función de umbral fijo. Por tanto, los bloques *Xilinx MCode* se encuentran actuando como comparadores con diferentes umbrales establecidos. En las figuras 48, 49, 50, 51 y 52 se aprecian estas características:

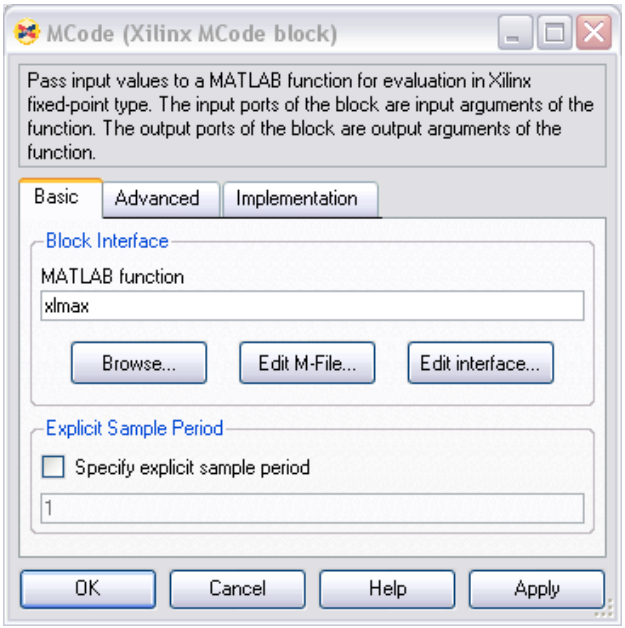

Figura 48: Ejemplo funcionamiento del bloque Xilinx MCode. Configuración básica del bloque (1)

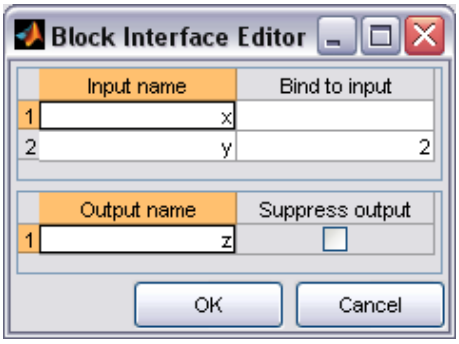

Figura 49: Ejemplo funcionamiento del bloque Xilinx MCode. Configuración básica del bloque (2)

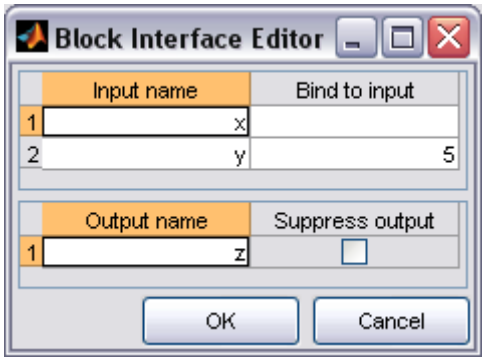

Figura 50: Ejemplo funcionamiento del bloque Xilinx MCode. Configuración básica del bloque (3)

```
function z = xlmax(x, y)
  if x > y
    z = x;
  else
    z = y;
  end
```

Figura 51: Ejemplo funcionamiento del bloque Xilinx MCode. Código implementado

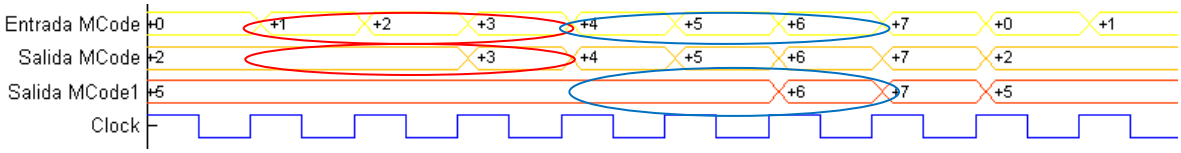

Figura 52: Ejemplo funcionamiento del bloque Xilinx MCode. Forma de la señales

3.3.9 Bloque *Xilinx Logical*

a) Descripción

El bloque *Xilinx Logical* (figura 53) permite realizar operaciones lógicas a nivel de bit con datos que contengan punto binario. Por defecto, previamente a realizar la operación lógica deseada, los operandos son completados con ceros y extendidos en signo tanto como sea necesario hasta que las posiciones de los puntos binarios coincidan.

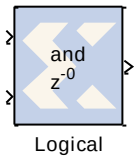

Figura 53: Bloque Xilinx Logical

b) Puertos de entrada y salida básicos

- `Input` (entrada): datos sobre los cuales se desea realizar una operación lógica. Se admiten desde 1 hasta 1024 entradas.
- `Output` (salida): resultado de la operación lógica realizada sobre la entrada.

c) Configuración básica

La configuración básica del bloque se muestra mediante la siguiente captura (figura 54):

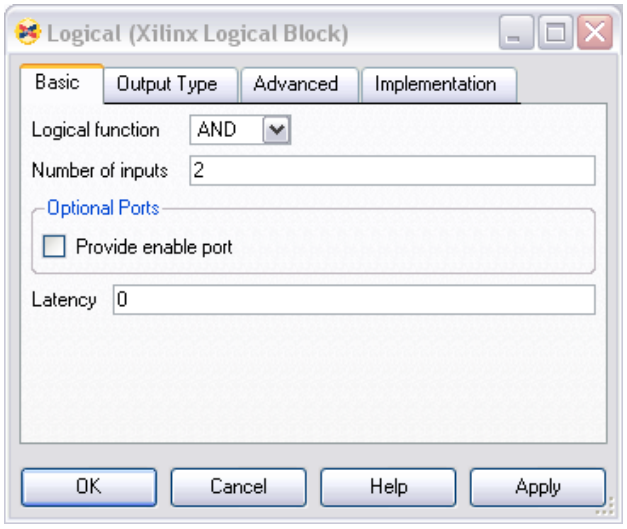

Figura 54: Configuración básica del bloque Xilinx Logical (1)

- `Logical function` (función lógica): operación lógica a realizar sobre los datos de entrada, esto es: `AND, NAND, OR, NOR, XOR, XNOR`.
- `Number of inputs` (número de entradas): número de entradas al bloque (desde 1 hasta 1024).

La configuración del tipo de salida del bloque se muestra mediante la siguiente captura (figura 55):

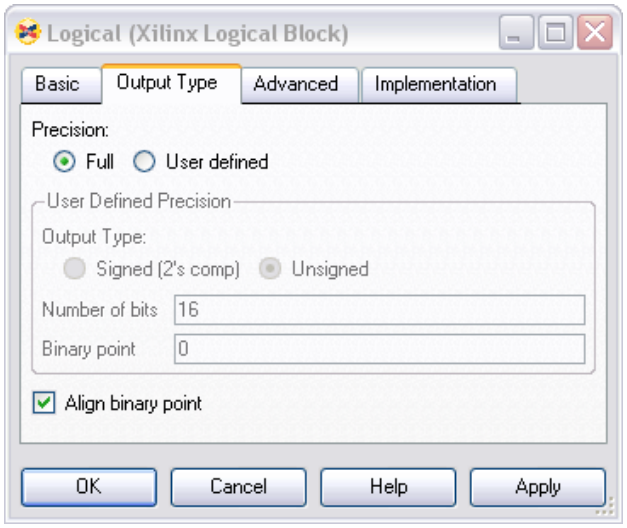

Figura 55: Configuración básica del bloque Xilinx Logical (2)

- `Precision` (precisión): precisión de los datos salida. En el caso de seleccionar `full`, la precisión es la suficiente para representar el resultado sin error.

- `Align binary point` (alinear punto binario): indica si se desea que el bloque alinee automáticamente los puntos binarios de cada entrada. En caso contrario, todas las entradas deben tener la misma posición para el punto binario.

d) Observaciones

- Existe la posibilidad de proporcionar un puerto de habilitado (`enable port`) a la entrada del bloque. Su señal de entrada debe ser de tipo booleano. Este puerto se proporciona en la configuración del bloque.

- Este bloque no produce retardo. Sin embargo, se puede agregar un valor de retardo en la configuración de bloque.

e) Ejemplo de funcionamiento

El funcionamiento del bloque *Xilinx Logical* se ilustra mediante este diseño (figura 56):

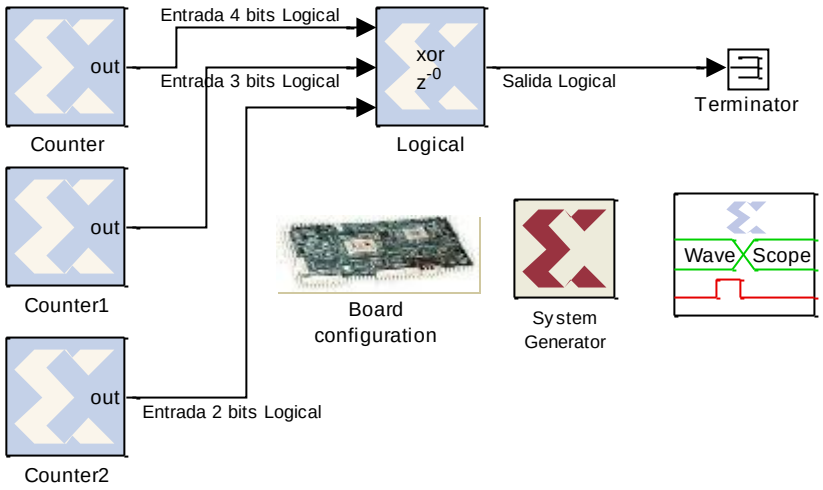

Figura 56: Ejemplo funcionamiento del bloque Xilinx Logical. Modelo

Este sistema consta de:

- Tres fuentes de datos consistentes en tres bloques *Xilinx Counter* configurados como contadores libres de 4, 3 y 2 bits, los cuales comienzan a contar desde valores diferentes al cero y tienen establecido el punto binario en diferentes posiciones. Estos contadores serán la entrada al bloque *Xilinx Logical*.

- Un bloque *Xilinx Logical* configurado para realizar la función `XOR` (recordar que es asociativa) y emplear la máxima precisión a la salida.

- Un bloque *Xilinx WaveScope* para observar la forma de las señales a la entrada y salida del bloque.

Este modelo realiza la función `XOR` (OR exclusivo) de las tres fuentes de datos a nivel de bit, por lo que actúa como la función de paridad de las entradas (si el número de unos a la entrada es impar se tendrá a la salida un uno). Además, se ha configurado al bloque para que alinee los puntos binarios de las entradas, por lo que los operandos han sido completados con ceros y extendidos en signo tanto como sea necesario hasta que las posiciones de los puntos binarios han coincidido.

Finalmente, notar que la salida se ha configurado para que proporcione la máxima precisión sin error (coincide con la entrada de 4 bits, la de mayor tamaño). Todas estas características se muestran mediante las figuras 57, 58 y 59:

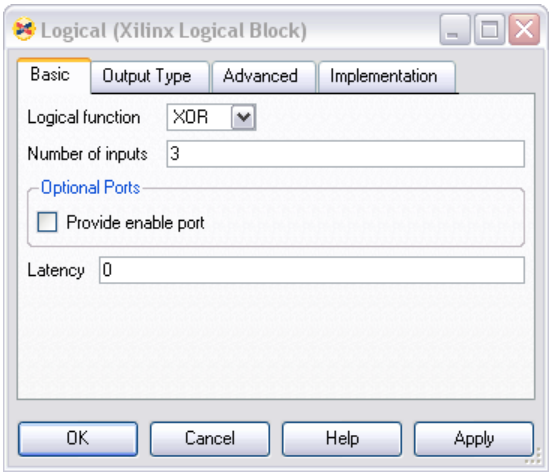

Figura 57: Ejemplo funcionamiento del bloque Xilinx Logical. Configuración básica del bloque (1)

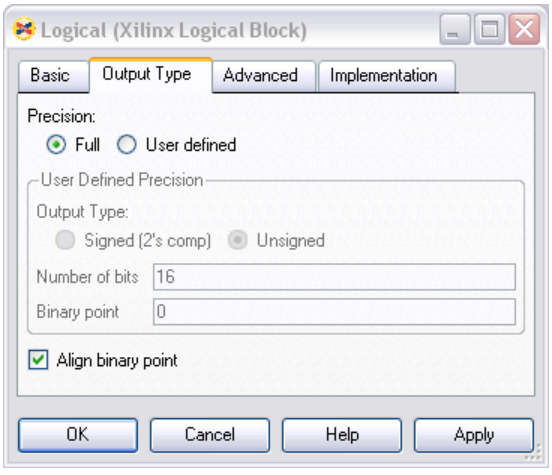

Figura 58: Ejemplo funcionamiento del bloque Xilinx Logical. Configuración básica del bloque (2)

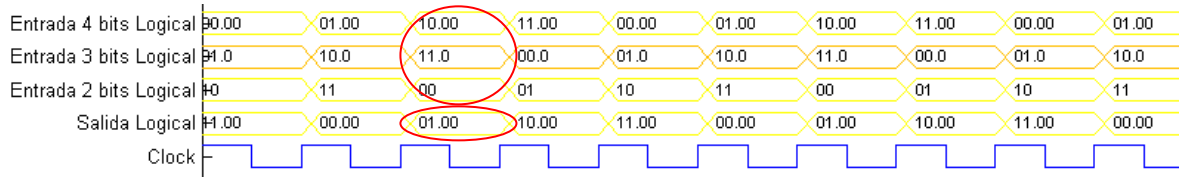

Figura 59: Ejemplo funcionamiento del bloque Xilinx Logical. Forma de la señales

3.3.10 Bloque *Xilinx Shift*

a) Descripción

El bloque *Xilinx Shift* (figura 60) permite realizar un desplazamiento a la izquierda o la derecha de los datos de entrada. La salida obtenida tiene el punto binario situado en la misma posición que en la entrada.

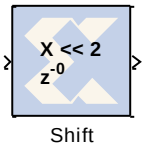

Figura 60: Bloque Xilinx Shift

b) Puertos de entrada y salida básicos

- `Input` (entrada): datos que se desean desplazar a la izquierda o derecha.
- `Output` (salida): datos de entrada desplazados tantos bits como se desea en el sentido indicado.

c) Configuración básica

La configuración básica del bloque se muestra mediante la siguiente captura (figura 61):

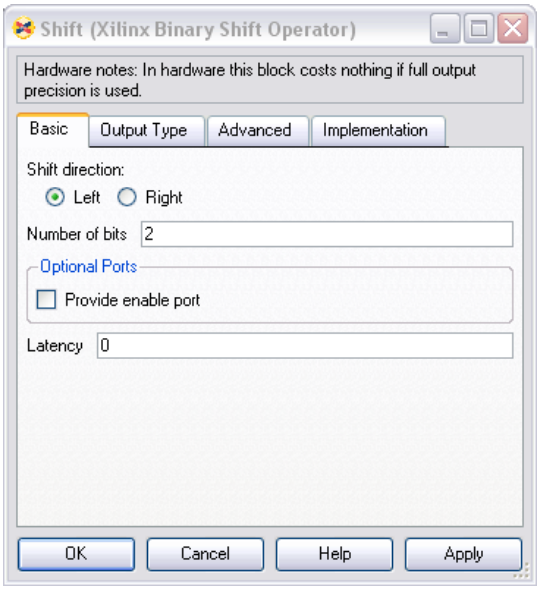

Figura 61: Configuración básica del bloque Xilinx Shift (1)

- `Shift Direction` (sentido del desplazamiento): especifica el sentido del desplazamiento, es decir, izquierda o derecha. El desplazamiento a la derecha desplaza los bits hacia el bit menos significativo descartando los que salen y rellenando con ceros por el bit más significativo. El desplazamiento a la izquierda desplaza los bits hacia el bit más significativo descartando los que salen y rellenando con ceros por el bit menos significativo.

- `Number of bits` (número de bits): indica el número de bits que serán desplazados, siendo el sentido del desplazamiento invertido si este valor es negativo.

La configuración del tipo de salida del bloque se muestra mediante la siguiente captura (figura 62):

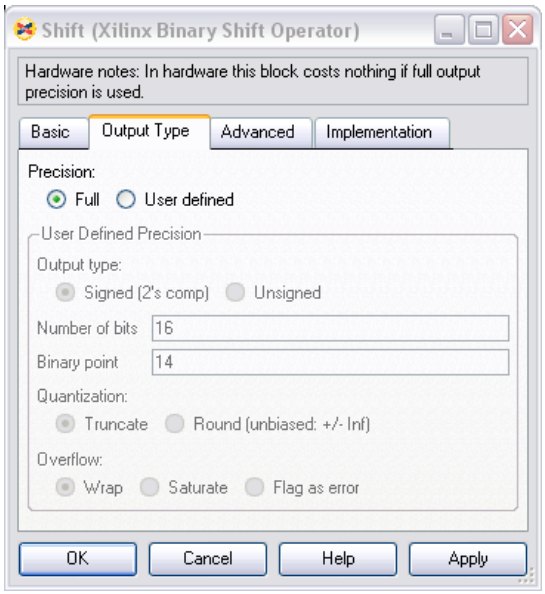

Figura 62: Configuración básica del bloque Xilinx Shift (2)

- `Precision` (precisión): precisión de los datos salida. En el caso de seleccionar `full`, la precisión es la suficiente para representar el resultado sin error.

d) Observaciones

- Existe la posibilidad de proporcionar un puerto de habilitado (`enable port`) a la entrada del bloque. Su señal de entrada debe ser de tipo booleano. Este puerto se proporciona en la configuración del bloque.

- Este bloque no produce retardo. Sin embargo, se puede agregar un valor de retardo en la configuración de bloque.

e) Ejemplo de funcionamiento

El funcionamiento del bloque *Xilinx Shift* se muestra con el siguiente sistema (figura 63):

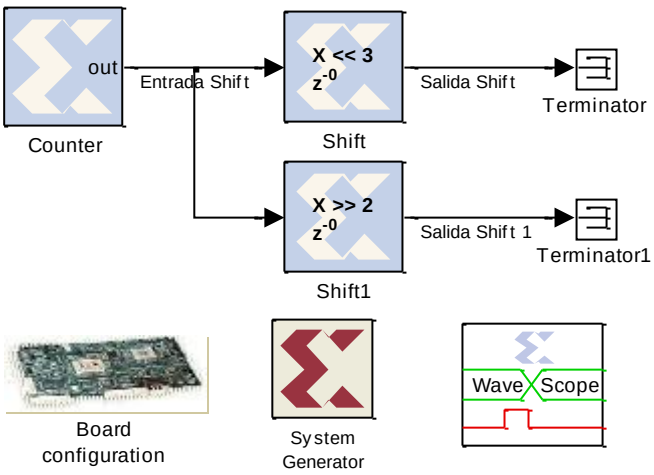

Figura 63: Ejemplo funcionamiento del bloque Xilinx Shift. Modelo

Este modelo está formado por:

- Una fuente de datos consistente en un bloque *Xilinx Counter* configurado como contador libre de 8 bits iniciando la cuenta en 16 y posicionando su punto binario en el bit 2. Esta fuente será la entrada de ambos bloques *Xilinx Shift*.
- Dos bloques *Xilinx Shift* configurados para realizar, respectivamente, un desplazamiento de 3 bits a la izquierda y de 2 bits a la derecha (empleando un valor negativo en la configuración del número de bits de este último).
- Un bloque *Xilinx WaveScope* para observar la forma de las señales a la entrada y salida de los bloques.

Este sistema desplaza un diferente número de bits de una fuente de datos común. Uno de los bloques *Xilinx Shift* desplaza 3 bits a la izquierda (hacia el bit más significativo) descartando los que salen y rellenando con ceros por la derecha (bit menos significativo). El otro bloque *Xilinx Shift* desplaza 2 bits a la derecha (descartando los que salen y rellenando con ceros por la izquierda), consiguiendo este efecto configurándolo con un desplazamiento a la izquierda con un número negativo (-2 bits). Para ambos bloques se observa cómo el punto binario de la entrada es conservado a la salida. Todas estas características se muestran mediante las figuras 64, 65, 66, 67 y 68:

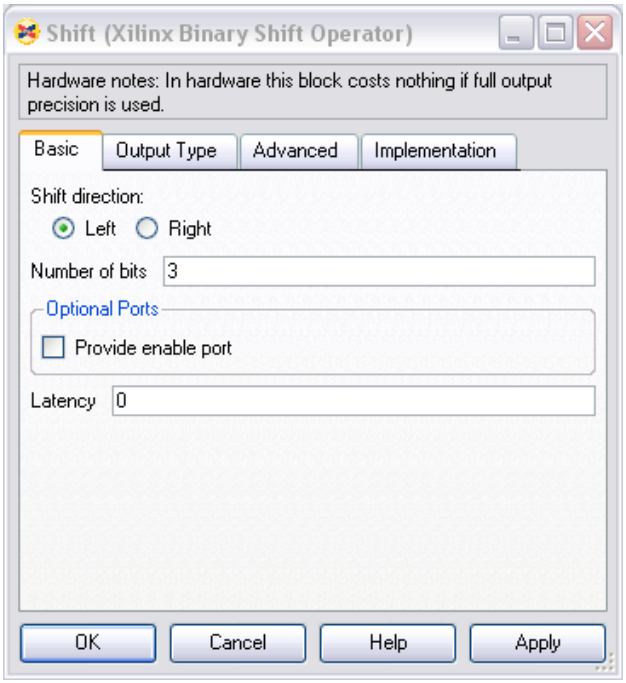

Figura 64: Ejemplo funcionamiento del bloque Xilinx Shift. Configuración básica del bloque (1)

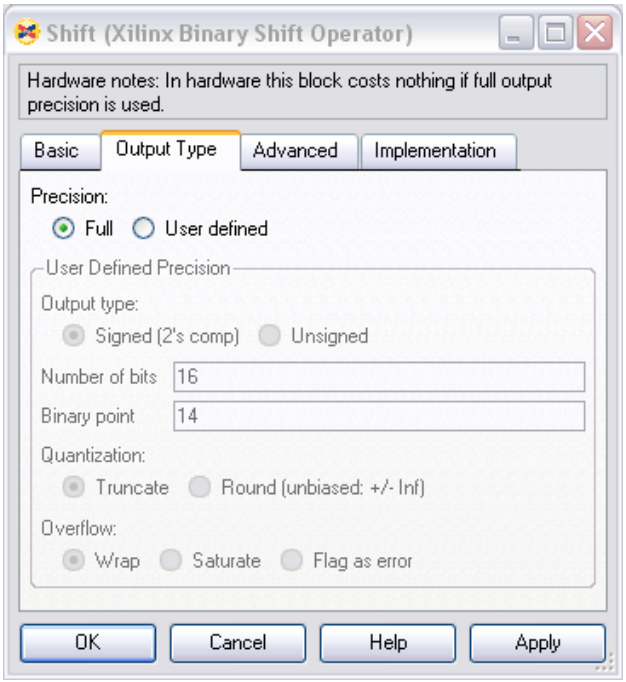

Figura 65: Ejemplo funcionamiento del bloque Xilinx Shift. Configuración básica del bloque (2)

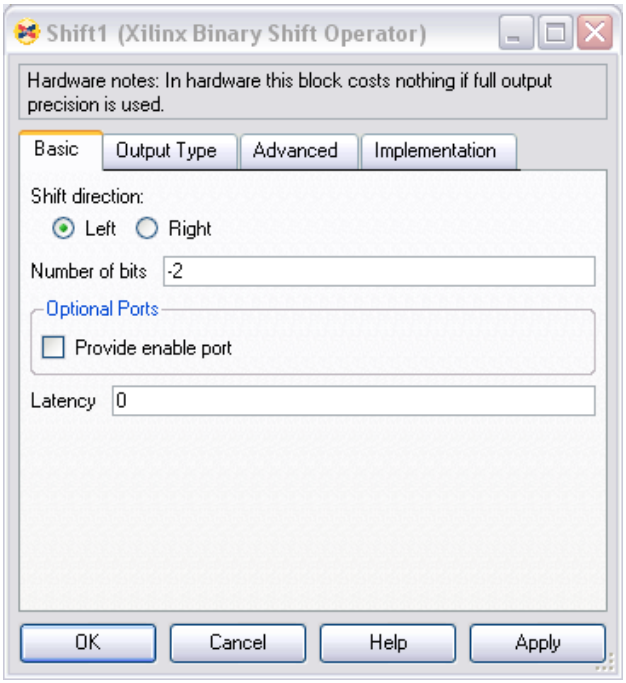

Figura 66: Ejemplo funcionamiento del bloque Xilinx Shift. Configuración básica del bloque (3)

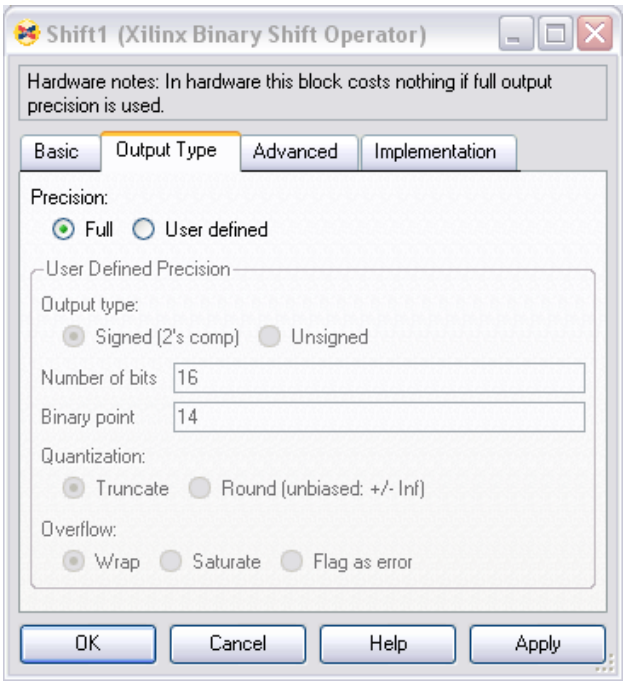

Figura 67: Ejemplo funcionamiento del bloque Xilinx Shift. Configuración básica del bloque (4)

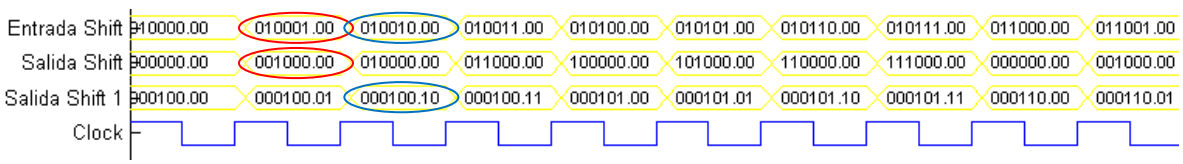

Figura 68: Ejemplo funcionamiento del bloque Xilinx Shift. Forma de la señales

3.3.11 Bloque *Xilinx Constant*

a) Descripción

El bloque *Xilinx Constant* (figura 69) genera un dato constante que puede ser un valor con un punto binario fijo, un valor booleano o una instrucción DSP48.

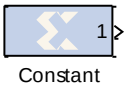

Figura 69: Bloque Xilinx Constant

Este bloque es similar al bloque de Simulink *Constant* pero con la cualidad de que puede ser usado para introducir entradas constantes en el resto de bloques Xilinx.

b) Puertos de entrada y salida básicos

- `Output` (salida): dato de valor constante en formato booleano, punto binario fijo (con o sin signo) o instrucción DSP.

c) Configuración básica

La configuración básica del bloque se muestra mediante la siguiente captura (figura 70):

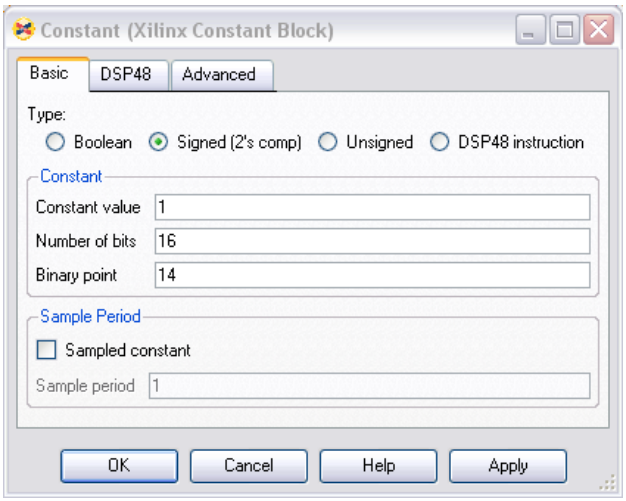

Figura 70: Configuración básica del bloque Xilinx Constant

- `Type` (tipo): especifica el tipo de constante. Puede ser booleana, punto fijo con signo, punto fijo sin signo o una instrucción DSP48.

- `Constant Value` (valor de la constante): especifica el valor de la constante. Cuando es modificado, el nuevo valor aparece en el icono del bloque.

- `Number of bits` (número de bits): en el caso de que la constante sea de tipo punto fijo, indica el número de bits de la constante de salida.

- `Binary point` (punto binario): en el caso de que la constate sea de tipo punto fijo, indica la posición del punto binario en la constante de salida.

- `Sampled constant` (constante muestreada): permite que un periodo de muestreo sea asociado a la constante de tal manera que sea heredado por los bloques a los cuales se dirige.

d) Observaciones

- En el caso de que se seleccione la constante del tipo instrucción DSP48, la pestaña DSP48 será activada en la configuración del bloque.

e) Ejemplo de funcionamiento

El funcionamiento del bloque *Xilinx Constant* se ilustra mediante el siguiente diseño (figura 71):

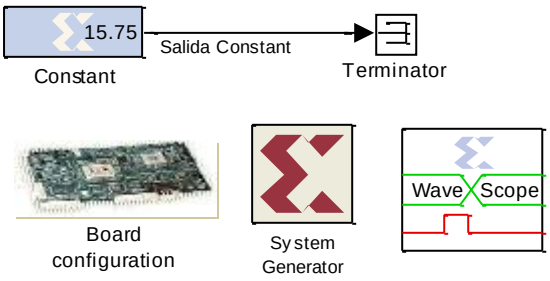

Figura 71: Ejemplo funcionamiento del bloque Xilinx Constant. Modelo

Este sistema consta de:

- Un bloque *Xilinx Constant* configurado para generar una constante de valor 15.75 del tipo punto fijo con signo.

- Un bloque *Xilinx WaveScope* para observar la forma de la señal a la salida del bloque.

Este modelo realiza la sencilla operación de proporcionar un valor con signo constante de 15,75 (1111.11 en binario) que es válido como entrada de los bloques Xilinx. Se han empleado 12 bits en total, situando el punto binario en el número 4, para asegurar tener

la resolución suficiente en la representación de la constante. Todas estas características se muestran mediante las figuras 72 y 73:

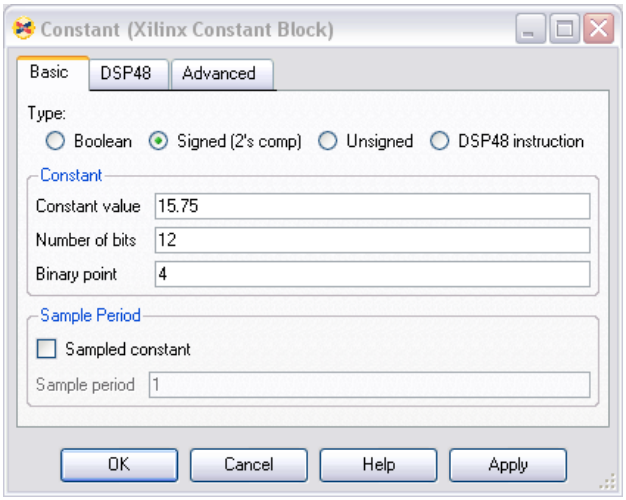

Figura 72: Ejemplo funcionamiento del bloque Xilinx Constant. Configuración básica del bloque

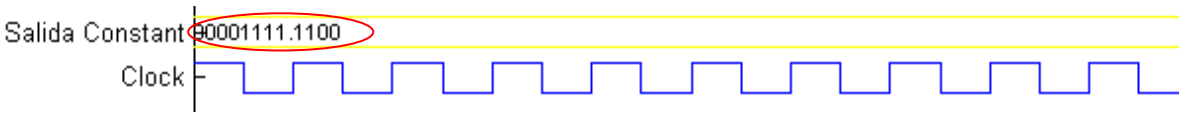

Figura 73: Ejemplo funcionamiento del bloque Xilinx Constant. Forma de la señales

3.3.12 Bloque *Xilinx FFT v1_0*

a) Descripción

El bloque *Xilinx FFT v1_0* (figura 74) realiza la DFT (Transformada Discreta de Fourier) empleado el algoritmo Cooley-Tukey de tipo radix-4, el cual divide la DFT en varias más pequeñas.

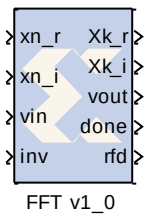

Figura 74: Bloque Xilinx FFT v1_0

Este bloque acepta un flujo de datos complejos de entrada representados como un par de líneas (real e imaginaria).

b) Puertos de entrada y salida básicos

- `Xn_r` (parte real entrada): componente real del flujo de datos de entrada.
- `Xn_i` (parte imaginaria entrada): componente imaginaria del flujo de datos de entrada.

- `Vin` (validación de entrada): marca los datos de entrada como válidos o inválidos. Esta señal puede ser empleada para alinear el comienzo de la FFT con las tramas de datos.

- `Inv` (inversa): selecciona si el bloque funcionará como FFT (0) o como IFFT (1).

- `Xk_r` (parte real salida): componente real del flujo de datos de salida.

- `Xk_i` (parte imaginaria salida): componente imaginaria del flujo de datos de salida.

- `Vout` (validación de salida): marca los datos de salida como válidos o inválidos. Si alguna de las N entradas de una trama es marcada como inválida, la trama correspondiente de salida será marcada también como inválida.

- `Done` (hecho): señala con un nivel alto la primera muestra de salida de una trama.

- `Rfd` (preparado para datos): señala con nivel alto cuando el bloque puede aceptar datos de entrada.

c) Configuración básica

La configuración básica del bloque se muestra mediante la siguiente captura (figura 75):

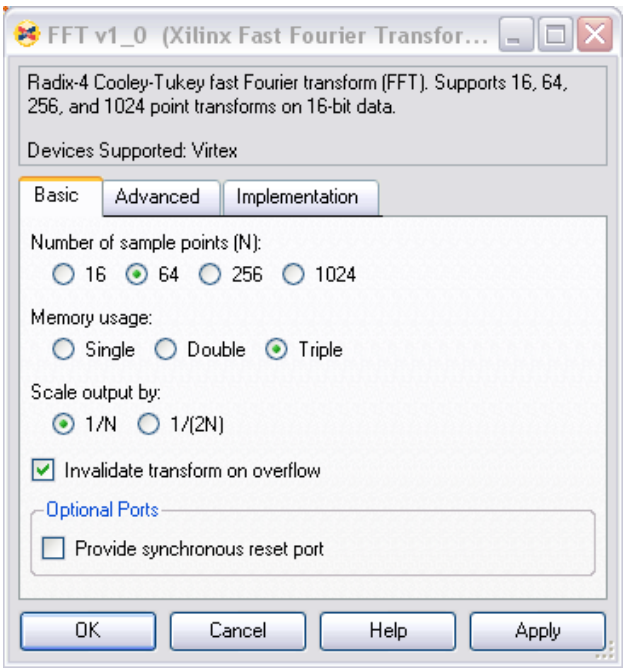

Figura 75: Configuración básica del bloque Xilinx FFT v1_0

- `Number of simple points` (número de puntos de muestreo): longitud (N) de la transformada de Fourier a realizar (16, 64, 256 ó 1024).

- `Memory usage`: (uso de memoria): número de bancos de memoria empelados para calcular la transformada (simple, doble o triple).

- `Scale output by` (salida escalada por): valor del escalado de los datos de salida, pudiendo ser 1/N o 1/(2N).

- `Invalidate transform on overflow` (invalidar transformada al producirse desbordamiento): define el comportamiento del bloque cuando se produce un desbordamiento interno. Es posible invalidar la salida o parar la simulación ante un evento de *overflow*.

d) Observaciones

- Existe la posibilidad de proporcionar un puerto de reinicio (`synchronous reset port`) a la entrada del bloque que permite retornarlo a su estado inicial en el caso de recibir una señal a nivel alto (tipo booleano). Este puerto se proporciona en la configuración del bloque.

- Se puede especificar el periodo de muestreo del flujo de datos sin emplear el detectado a la entrada.

- Este bloque escala los valores de salida, tanto reales como imaginarios, por un factor 1/N o 1/(2N). Por tanto, para no perder precisión durante su utilización como IFFT, es necesario adaptar los datos de entrada ampliando adecuadamente el número de bits. Además, debido a que el flujo de datos entrada a este bloque es continuo y a que hasta que no se tiene una trama completa no se pueden procesar dichos datos, los datos válidos de salida (correspondientes a la IFFT de los de entrada) únicamente se generan durante un 1/3 de la duración de la trama de entrada, siendo empleado los 2/3 restantes en procesar los datos. Esta ruptura en el flujo debe ser corregida eliminando los datos de salida correspondientes al procesamiento y ampliando la duración de los datos válidos hasta ocupar toda la trama. Gráficamente, para cualquiera de las entradas, este proceso es el siguiente (figura 76):

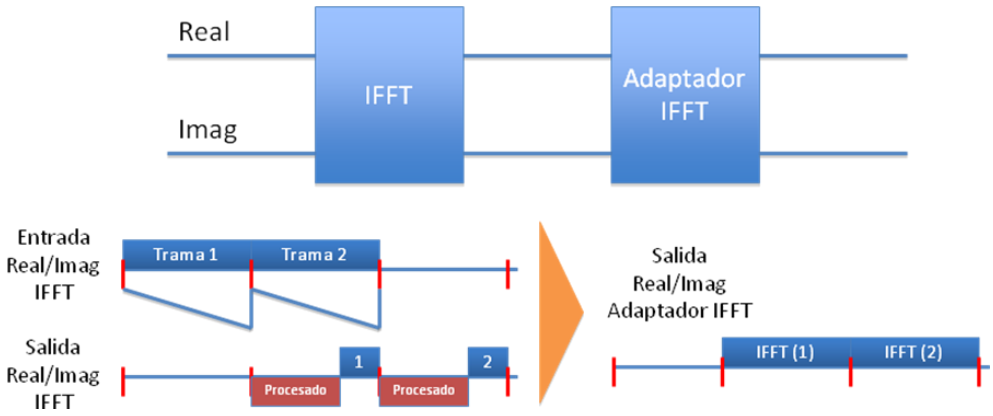

Figura 76: Procesamiento de datos de entrada en modo IFFT del bloque Xilinx FFT v1_0

En el caso de que se esté empleando este bloque en modo FFT para deshacer una anterior operación de IFFT, no será necesario adaptar los datos de entrada (ampliando el número de bits), sino escalar por N ó 2N la salida. Además, al igual que en el modo IFFT, el funcionamiento síncrono del bloque junto a su necesidad de un tiempo de procesamiento implica que los datos de salida ocupen 1/3 del tiempo total de la trama de entrada. Sin embargo, en este caso, hay que agregar el hecho de que los datos a la salida se encontrarán invertidos. Por tanto, deberá existir un bloque de adaptación que tenga en cuenta estas tres características. Gráficamente, para cualquiera de las entradas, este proceso es el siguiente (figura 77):

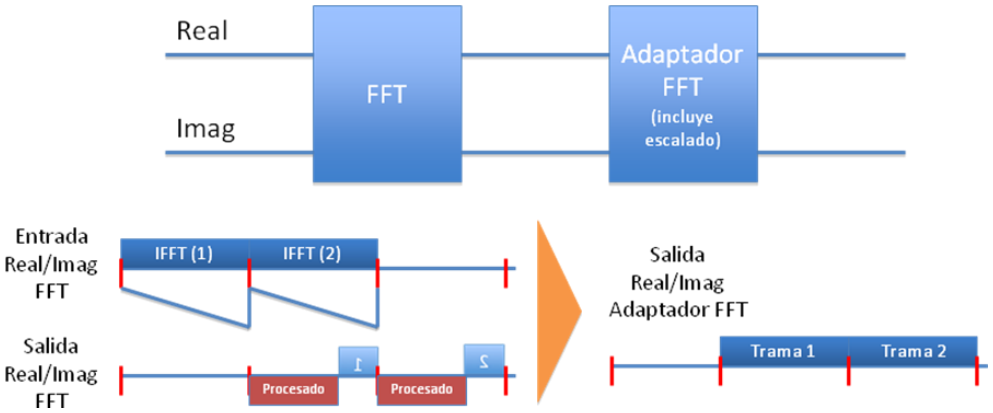

Figura 77: Procesamiento de datos de entrada en modo FFT del bloque Xilinx FFT v1_0

- Este bloque es soportado únicamente por los dispositivos Virtex, existiendo el bloque *Xilinx FFT v3_1* para el resto de familias de dispositivos.

e) Ejemplo de funcionamiento

El funcionamiento del bloque *Xilinx FFT v1_0* se ilustra mediante el siguiente modelo (figura 78):

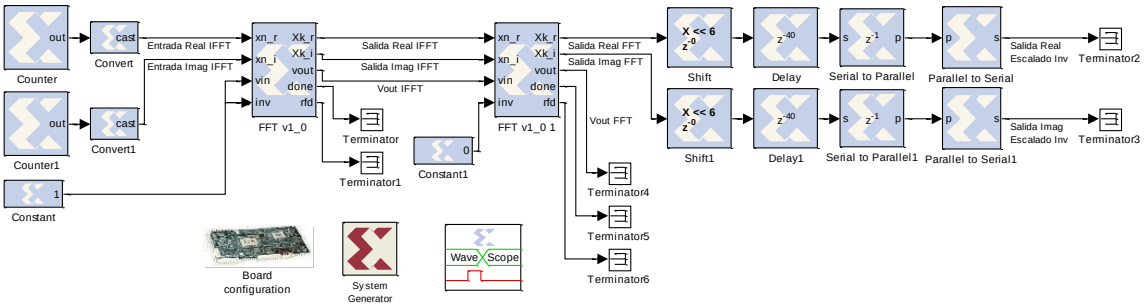

Figura 78: Ejemplo funcionamiento del bloque Xilinx FFT v1_0. Modelo

Dicho diseño está formado por:

- Dos fuentes de datos que proporcionan la entrada real e imaginaria al bloque *Xilinx FFT v1_0*. Estas fuentes consisten en dos bloques *Xilinx Counter* configurados como contadores libres de 4 con signo.

- Dos bloques *Xilinx Convert* para ampliar el número de bits de entrada al bloque *Xilinx FFT v1_0* y no perder precisión por efecto del escalado. En este caso amplía hasta 10 bits a la salida, estableciendo el punto binario en el sexto (pues N=64 en la FFT).

- Dos bloques *Xilinx FFT v1_0* consecutivos configurados como IFFT y FFT con N=64.

- Dos bloques *Xilinx Shift* para deshacer el escalado que realiza el bloque *Xilinx FFT v1_0*. En este caso, al emplear N=64, se desplazan 6 bits hacia la izquierda.

- Dos bloques *Xilinx Delay*, dos bloques *Xilinx Serial to Parallel* y dos bloques *Xilinx Parallel to Serial* que deshacen la inversión del orden de los símbolos obtenidos tras el segundo bloque *Xilinx FFT v1_0* (el cual realiza la FFT).

- Un bloque *Xilinx WaveScope* para observar la forma de la señal en los puntos de interés.

El modelo diseñado recompone casi totalmente la señal de entrada, exceptuando la cuestión de la ruptura del flujo (dicho problema se soluciona en modelos avanzados, como en el receptor diseñado en este proyecto). Además, este diseño permite observar las especiales características de funcionamiento comentadas previamente: escalado y

ruptura del flujo de los datos al realizar la IFFT (figura 79); inversión y ruptura del flujo de datos en el caso de la FFT (figura 80).

En la figura 81 se observa, mediante las señales “Vout IFFT” y “Vout FFT”, la necesidad de un tiempo de procesamiento para los datos igual a 2/3 del tiempo de trama, siendo los datos válidos durante 1/3 del mismo.

También es posible verificar que la salida de la FFT se encuentra escalada e invertida respecto a los datos de entrada (figura 82). Así como la corrección de este efecto (figura 83).

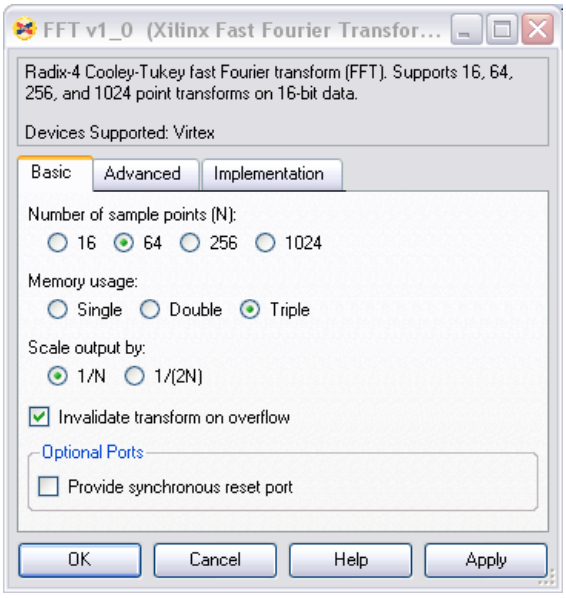

Figura 79: Ejemplo funcionamiento del bloque Xilinx FFT v1_0. Configuración básica del bloque (IFFT)

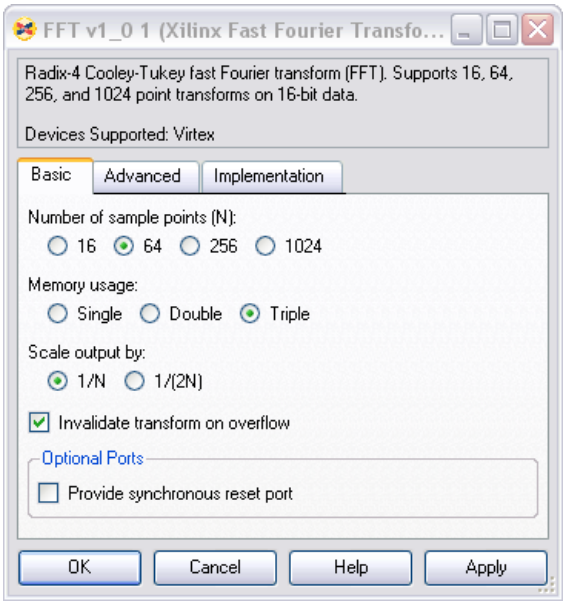

Figura 80: Ejemplo funcionamiento del bloque Xilinx FFT v1_0. Configuración básica del bloque (FFT)

Figura 81: Ejemplo funcionamiento del bloque Xilinx FFT v1_0. Forma de la señales (ruptura del flujo continuo de datos en IFFT y FFT)

Figura 82: Ejemplo funcionamiento del bloque Xilinx FFT v1_0. Forma de la señales (salida escalada e invertida de la FFT)

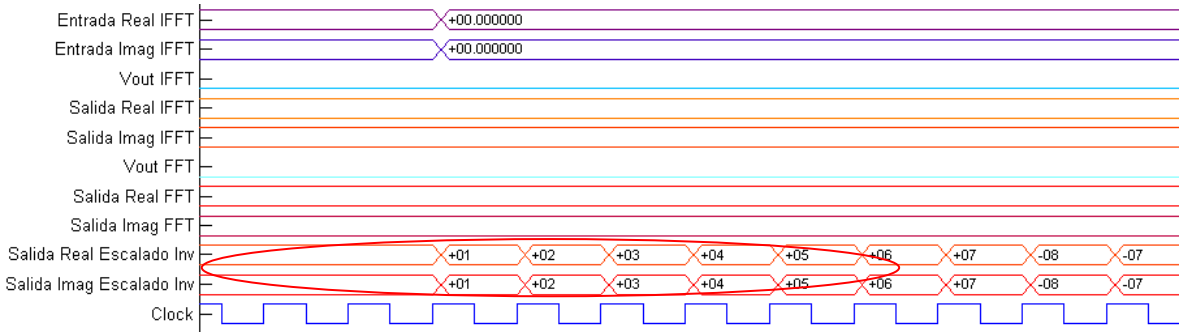

Figura 83: Ejemplo funcionamiento del bloque Xilinx FFT v1_0. Forma de la señales (salida final sin flujo continuo)

3.3.13 Bloque *Xilinx CMult*

a) Descripción

El bloque *Xilinx CMult* (figura 84) consiste en un bloque de ganancia en el cual la salida es el producto de la entrada por una constante. Dicha constante puede ser directamente un valor o una expresión de MATLAB® que proporcione un valor constante.

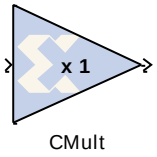

Figura 84: Bloque Xilinx CMult

b) Puertos de entrada y salida básicos

- `Input` (entrada): datos que se desean multiplicar por una constante. Sólo se admite un puerto de entrada.

- `Output` (salida): producto resultante. Sólo se admite un puerto a la salida.

c) Configuración básica

La configuración básica del bloque se muestra mediante la siguiente captura (figura 85):

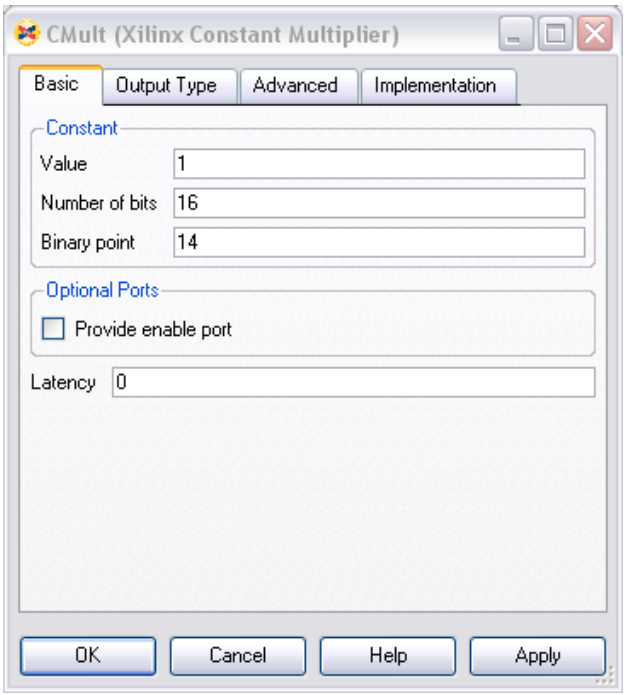

Figura 85: Configuración básica del bloque Xilinx CMult (1)

- `Value` (valor): valor de la constante de multiplicación, el cual debe ser una constante como tal o una expresión de MATLAB® que proporcione un valor constante. Si la constante no puede ser representada exactamente mediante el punto fijo indicado, su valor es redondeado y saturado tanto como sea necesario. Además, los valores positivos se implementan como números sin signo, mientras que los valores negativos con signo.

- `Number of bits` (número de bits): indica el número de bits de la constante de multiplicación.

- `Binary point` (punto binario): indica la posición del punto binario en la constante de multiplicación.

La configuración del tipo de salida del bloque se muestra mediante la siguiente captura (figura 86):

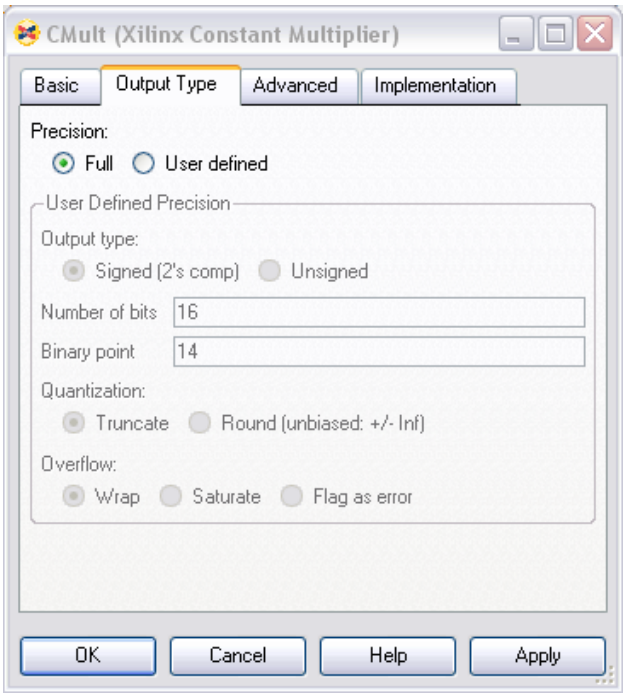

Figura 86: Configuración básica del bloque Xilinx CMult (2)

- `Precision` (precisión): precisión de los datos de salida. En el caso de seleccionar `full`, la precisión es la suficiente para representar el resultado sin error.

d) Observaciones

- Existe la posibilidad de definir ciertos parámetros avanzados de la implementación del bloque, a nivel hardware, en la pestaña *Implementation* de la configuración del bloque.

e) Ejemplo de funcionamiento

El funcionamiento del bloque *Xilinx CMult* se muestra con el siguiente sistema (figura 87):

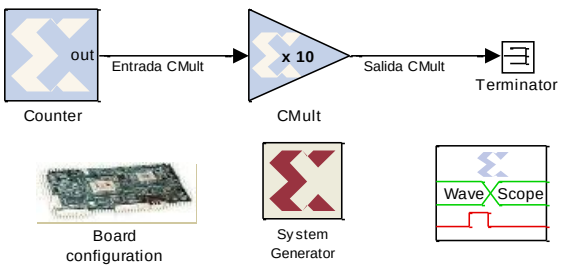

Figura 87: Ejemplo funcionamiento del bloque Xilinx CMult. Modelo

Este diseño consta de:

- Una fuente de datos consistente en un bloque *Xilinx Counter* configurado como contador libre de 4 bits iniciando la cuenta en 11. Este contador será la entrada al bloque *Xilinx CMult*.
- Un bloque *Xilinx CMult* configurado para proporcionar una ganancia de 10 empleando 7 bits y situando el punto binario en el segundo bit.
- Un bloque *Xilinx WaveScope* para observar la forma de las señales a la entrada y salida del bloque.

Este modelo proporciona una ganancia de 10 a los valores de la fuente de datos. Para representar esta ganancia se ha utilizado en la configuración del bloque una expresión de MATLAB® que proporcione su valor (5x2), a la cual se le han agregado 2 bits decimales. Además, la salida emplea la resolución necesaria para representar sus valores sin necesidad de ser redondeada. Todas estas características se muestran mediante las figuras 88, 89 y 90:

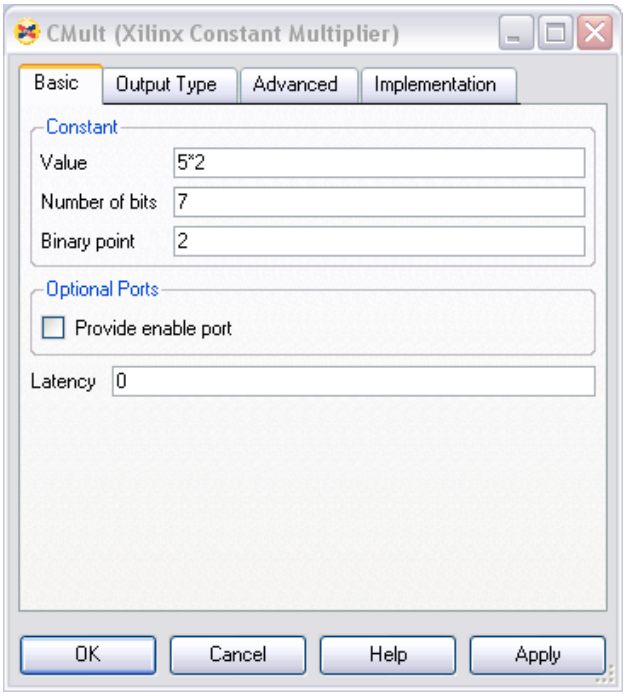

Figura 88: Ejemplo funcionamiento del bloque Xilinx CMult. Configuración básica del bloque (1)

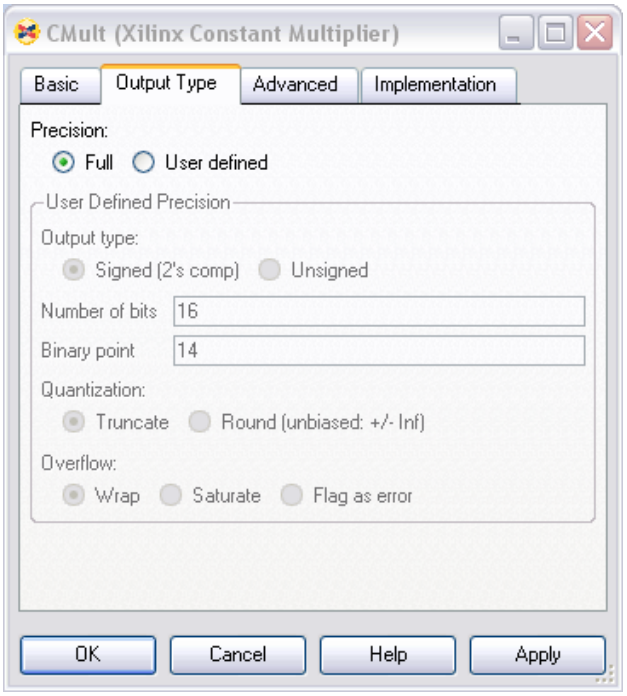

Figura 89: Ejemplo funcionamiento del bloque Xilinx CMult. Configuración básica del bloque (2)

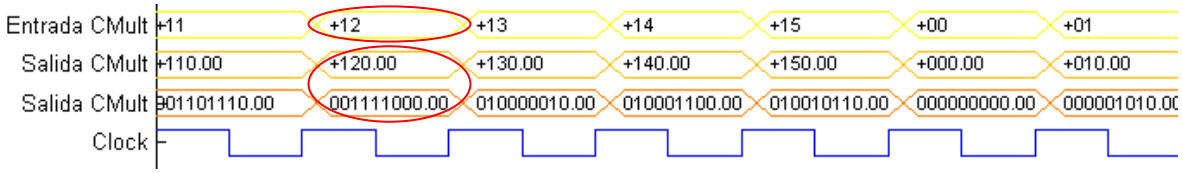

Figura 90: Ejemplo funcionamiento del bloque Xilinx CMult. Forma de la señales

3.3.14 Bloque *Xilinx Viterbi Decoder v5_0*

a) Descripción

El bloque *Xilinx Viterbi Decoder v5_0* (figura 91) se emplea para decodificar aquellos datos codificados mediante un codificador convolucional, soportando tasas desde 1/2 a 1/7.

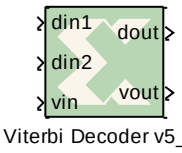

Figura 91: Bloque Xilinx Viterbi Decoder

El proceso de decodificación se realiza en dos pasos: en el primero se determina el coste de los datos de entrada en todas sus posibles combinaciones (empleado métrica Hamming o euclídea para ello); y en el segundo paso se rastrea su diagrama de rejilla (o diagrama de Trellis) y se determina la ruta óptima, siendo configurable la profundidad de truncado.

b) Puertos de entrada y salida básicos

- `Din` (datos de entrada): datos de entrada codificados mediante una codificador convolucional. Al soportar tasas desde 1/2 a 1/7, se podrán tener de 2 a 7 entradas. En el caso de emplear codificación dura, cada dato deberá ser de 1 bit. Sin embargo, si se emplea codificación blanda, los datos deberán tener un tamaño de entre 3 y 8 bits.

- `Vin` (validación de entrada): indica que los datos que se encuentran en la entrada son válidos.

- `Dout` (datos de salida): datos obtenidos a la salida tras la decodificación. Este puerto proporciona a la salida 1 bit.

- `Vout` (validación de salida): indica que el bit generado a la salida es válido.

c) Configuración básica

La configuración básica del bloque se muestra mediante la siguiente captura (figura 92):

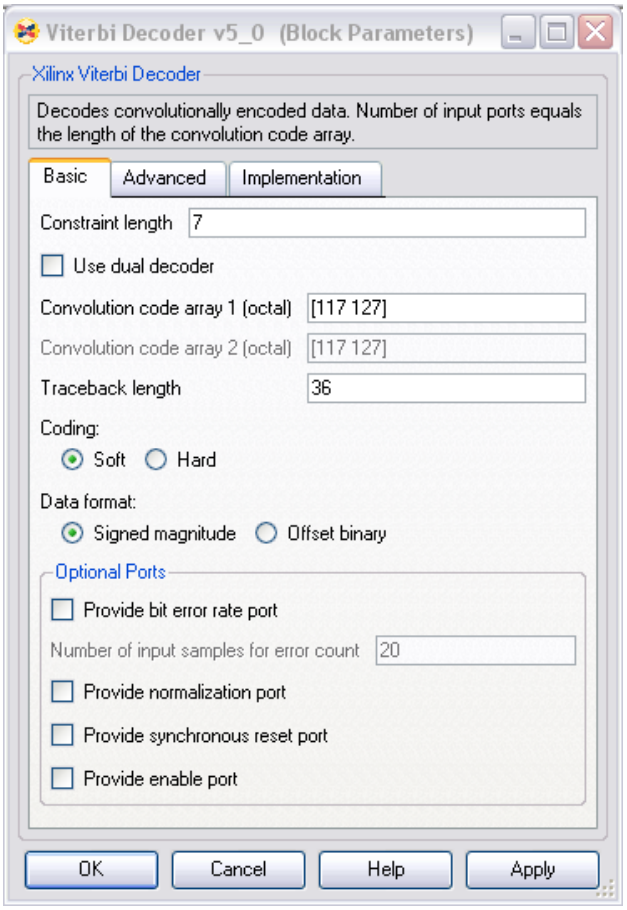

Figura 92: Configuración básica del bloque Xilinx Viterbi Decoder

- `Constraint length` (longitud restringida): es la longitud del registro de desplazamiento mediante el cual se realiza la codificación convolucional.

- `Use dual decoder` (usar decodificador dual): en el caso de emplear codificación blanda, permite que el bloque se comporte como un decodificador dual (emplear dos códigos de convolución diferentes para los datos de entrada). Mediante un puerto llamado `sel` (selección) se indica el código de convolución a emplear para los datos de entrada.

- `Convolution code array 1` (matriz del código de convolución 1): primer código de convolución en sistema octal. La tasa de salida depende de la longitud de este vector (entre 2 y 7). Para un funcionamiento correcto, los códigos de convolución usados para decodificar deben coincidir con los empleados para codificar. Al emplear un decodificador dual, un valor 0 en el puerto `sel` seleccionará este código.

- `Convolution code array 2` (matriz del código de convolución 2): segundo código de convolución en sistema octal. La tasa de salida depende de la longitud de este vector (entre 2 y 7). Para un funcionamiento correcto, los códigos de convolución usados para decodificar deben coincidir con los empleados para codificar. Al emplear un decodificador dual, un valor 1 en el puerto `sel` seleccionará este código.

- `Traceback length` (longitud de rastreo): profundidad de truncado al recorrer el diagrama de rejilla (o diagrama de Trellis). La longitud óptima es entre 5 y 7 veces la longitud restringida, según indicaciones del fabricante.

- `Coding` (codificación): método empleado para calcular el coste. Puede ser `Hard` (se emplea la distancia de Hamming y los datos de entrada son de 1 bit) o `Soft` (se emplea la distancia euclídea y los datos de entrada tienen un tamaño entre 3 y 8 bits). Además, la codificación blanda permite emplear decodificación dual y *puncturing* externo.

- `Data format` (formato de datos): permite indicar el formato de los datos en el caso de emplear codificación blanda.

d) Observaciones

- Existe la posibilidad de proporcionar dos puertos a la salida del bloque que indican la tasa de error de bit (*bit error rate*). El primer puerto es la `BER` (Tasa de Error de Bit) y

proporciona una medida de la tasa de error de bit del canal comparando la salida recodificada con la entrada retardada (el número de muestras de entrada sobre las cuales se calcula es configurable). El segundo puerto es el `ber_done`, que indica el momento en el cual han sido procesadas el número de muestras de entrada configuradas para contabilizar el error. Estos puertos se activan en la configuración del bloque.

- Se contempla la posibilidad de agregar un puerto a la salida que proporciona monitorización inmediata de los errores en el canal (`normalization port`). Este puerto se activa en la configuración del bloque.

- El bloque proporciona la opción de emplear *puncturing* externo (sólo con codificación blanda) mediante unos puertos a la entrada que se activan en la configuración del bloque. Dependiendo de la tasa del decodificador, habrá disponibles hasta 7 puertos de eliminación que permitirán tratar los datos como un símbolo nulo.

- Existe una opción en la configuración del bloque que provoca que el rastreo en el diagrama de rejilla comience desde el estado óptimo.

- El decodificador alcanza la mínima tasa de error al emplear códigos de convolución óptimos, los cuales se muestran en la tabla 2:

Longitud restringida	**Código de convolución óptimo para tasa 1/2 (octal)**	**Código de convolución óptimo para tasa 3/4 (octal)**
3	[7 5]	[7 7 5]
4	[17 13]	[17 13 15]
5	[37 33]	[37 33 25]
6	[57 65]	[57 65 71]
7	[117 127]	[117 127 155]
8	[357 233]	[357 233 251]
9	[755 633]	[755 633 447]

Tabla 2: Códigos de convolución óptimos

e) Ejemplo de funcionamiento

El funcionamiento del bloque *Xilinx Viterbi Decoder v5_0* se ilustra con este modelo (figura 93):

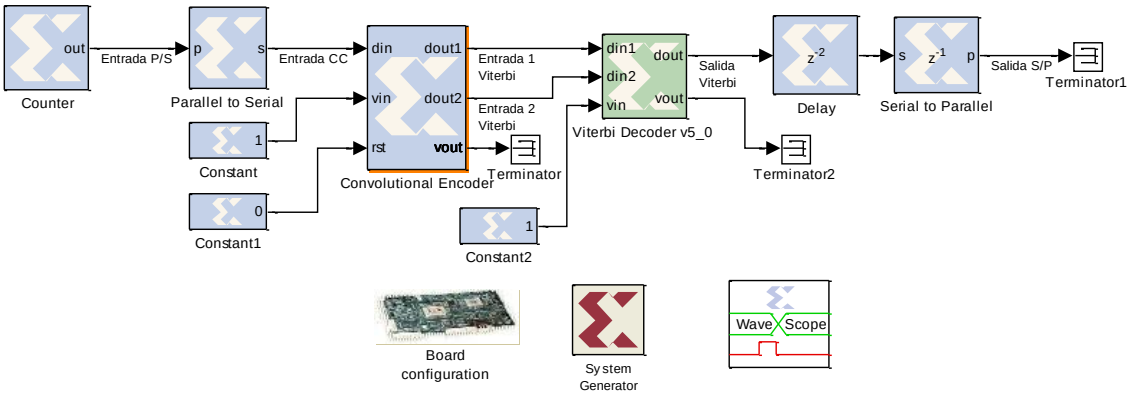

Figura 93: Ejemplo funcionamiento del bloque Xilinx Viterbi Decoder. Modelo

Este sistema se encuentra formado por:

- Una fuente de datos consistente en un bloque *Xilinx Counter* configurado como contador libre de 4 bits. Estos datos serán los que se desean recuperar mediante el bloque *Xilinx Viterbi Decoder v5_0*.

- Un bloque *Xilinx Parallel to Serial* ordenando la entrada con la palabra más significativa primero y una palabra de salida de 1 bit.

- Un codificador convolucional consistente en un bloque *Xilinx Convolutional Enconder* configurado para emplear un código [117 127] y una longitud restringida de 7.

- Un bloque *Xilinx Viterbi Decoder v5_0* configurado para codificación `Hard`, el mismo código que el codificador convolucional ([117 127]) y una longitud restringida de 7.

- Un bloque *Xilinx Delay* y un bloque *Xilinx Serial to Parallel* para adaptar la salida del decodificador al tamaño de palabra de la fuente de datos. Para ello se ordena la entrada con la palabra más significativa primero y una palabra de salida de 3 bits.

- Un bloque *Xilinx WaveScope* para observar la forma de las señales a la entrada y salida de los bloques de interés.

El modelo diseñado, tal y como se muestra en las figuras 94 y 95, recompone la señal suministrada por la fuente de datos tras su codificación por parte del codificador convolucional. Para ello se emplea el mismo código óptimo de tasa 1/2 en éste y en el bloque *Xilinx Viterbi Decoder v5_0*. Además, en dicho decodificador, se emplea una codificación Hard (entrada al decodificador de 1 bit) y una profundidad óptima de truncado de 36 (5,14 veces la longitud restringida).

Finalmente, notar cómo el bloque *Xilinx Viterbi Decoder v5_0* necesita un tiempo para procesar los datos de entrada (determinar las distancias y ruta óptima en el diagrama) antes de generar los datos decodificados.

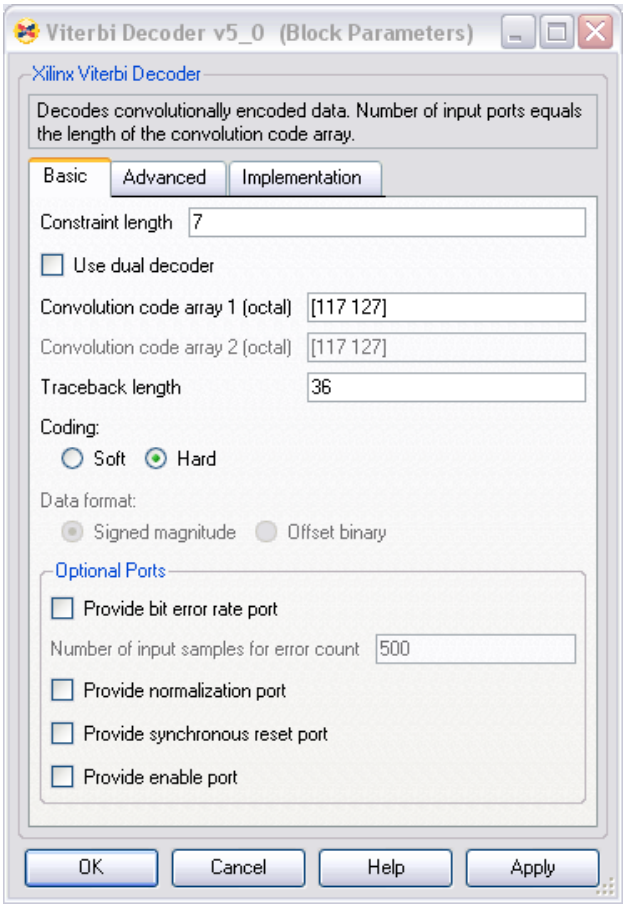

Figura 94: Ejemplo funcionamiento del bloque Xilinx Viterbi Decoder. Configuración básica del bloque

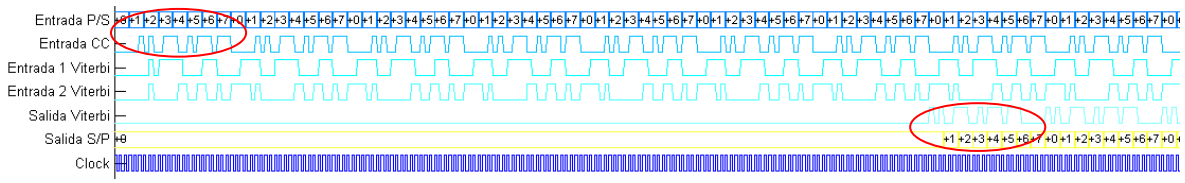

Figura 95: Ejemplo funcionamiento del bloque Xilinx Viterbi Decoder. Forma de la señales

3.3.15 Bloque *Xilinx Mux*

a) Descripción

El bloque *Xilinx Mux* (figura 96) implementa un multiplexor. Para ello cuenta con un número configurable de líneas de entrada (desde 2 a 32) y un puerto de selección de entrada (de tipo sin signo y con el número de bits óptimo para el número de líneas de entrada).

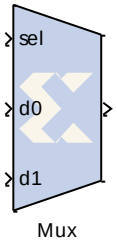

Figura 96: Bloque Xilinx Mux

b) Puertos de entrada y salida básicos

- `Input` (entrada): conjunto de datos que se desean seleccionar a la salida. Se admiten entre 2 y 32 líneas de entrada.
- `Sel` (selección): indica la línea de entrada que será mostrada a la salida. El número de bits de este puerto debe ser el óptimo para el número de líneas de entrada.
- `Output` (salida): línea de datos de entrada seleccionada.

c) Configuración básica

La configuración básica del bloque se muestra mediante la siguiente captura (figura 97):

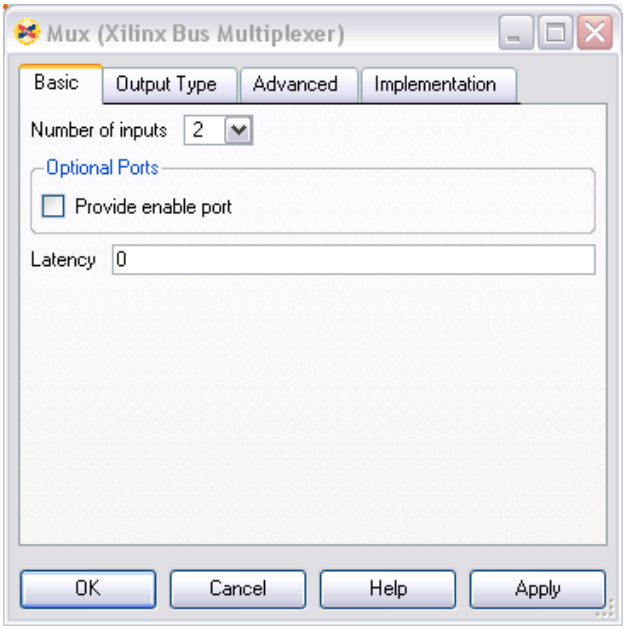

Figura 97: Configuración básica del bloque Xilinx Mux (1)

- `Number of inputs` (número de entradas): número de líneas de entrada al bloque multiplexor (entre 2 y 32).

La configuración del tipo de salida del bloque se muestra mediante la siguiente captura (figura 98):

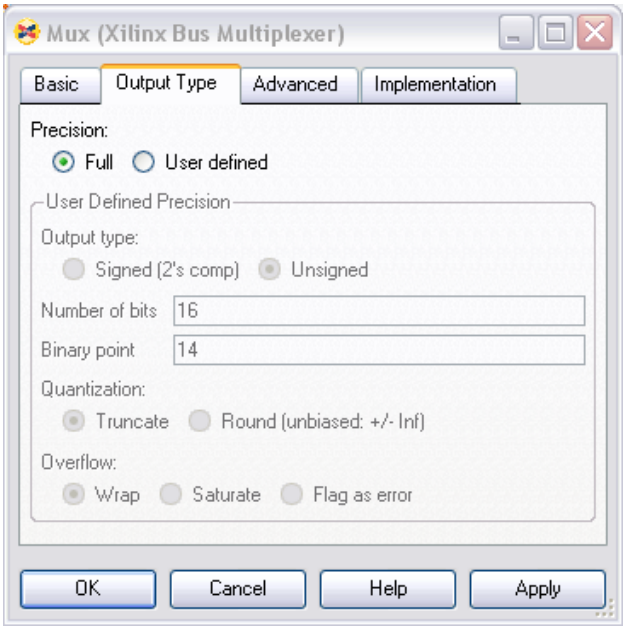

Figura 98: Configuración básica del bloque Xilinx Mux (2)

- `Precision` (precisión): precisión del punto fijo. En el caso de seleccionar `full`, la precisión es la suficiente para representar el resultado sin error.

d) Observaciones

- Existe la posibilidad de proporcionar un puerto de habilitado (`enable port`) a la entrada del bloque. Su señal de entrada debe ser de tipo booleano. Este puerto se proporciona en la configuración del bloque.

- Este bloque no produce retardo. Sin embargo, se puede agregar un valor de retardo en la configuración de bloque.

e) Ejemplo de funcionamiento

El funcionamiento del bloque *Xilinx Mux* se muestra con el siguiente diseño (figura 99):

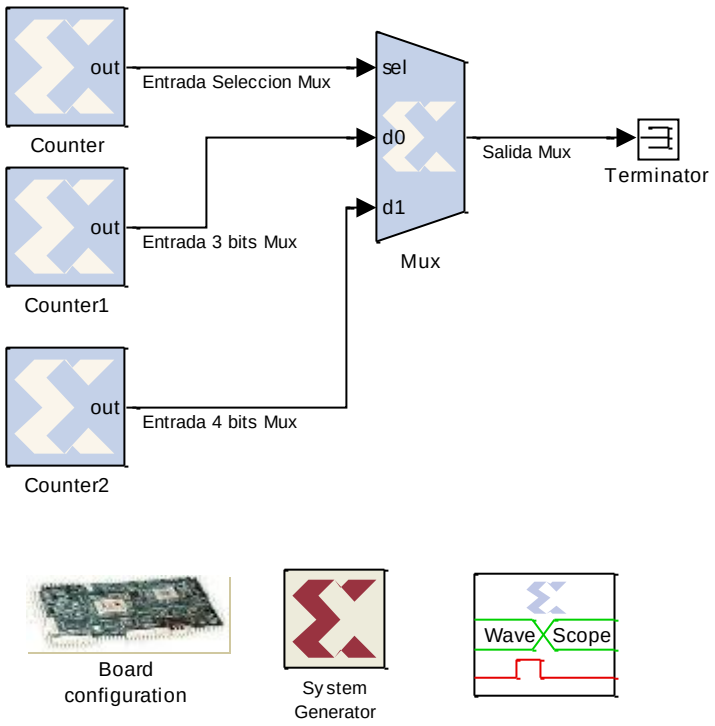

Figura 99: Ejemplo funcionamiento del bloque Xilinx Mux. Modelo

Este modelo consta de:

- Dos fuentes de datos consistentes en dos bloques *Xilinx Counter* configurados como contadores libres de 3 y 4 bits, lo cuales comienzan a contar desde posiciones diferentes al cero. Estos datos serán los seleccionados por el bloque *Xilinx Mux*.

- Un selector de entrada definido mediante un bloque *Xilinx Counter* y configurado como contador libre de 1 bit (pues se tienen 2 líneas de entrada).

- Un bloque *Xilinx Mux* configurado con 2 líneas de entrada y empleando la máxima precisión en los datos de salida.

- Un bloque *Xilinx WaveScope* para observar la forma de las señales a la entrada y salida del bloque.

Este sistema selecciona las 2 entradas de las que dispone de manera alternativa y sin retardo (figura 100). Se puede observar, en las figuras 101 y 102, cómo la precisión de la salida es la máxima requerida, correspondiéndose con la segunda entrada.

Además, notar cómo el número de bits de la línea de selección es 1, encontrándose ajustado al número de entradas del modelo.

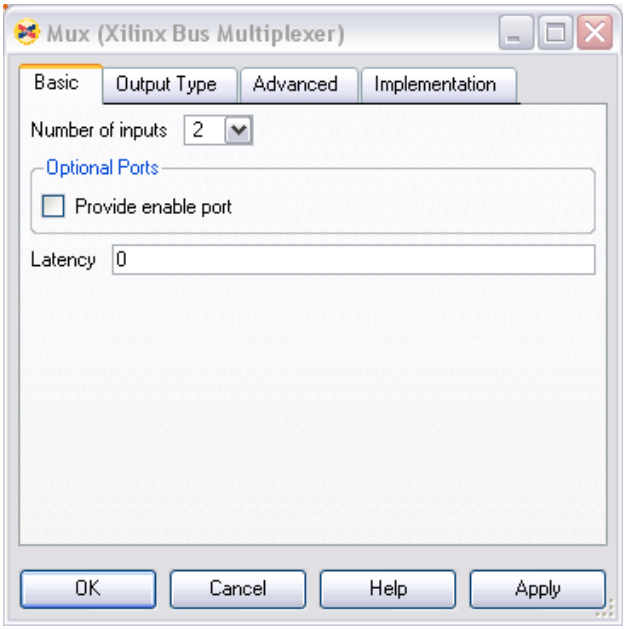

Figura 100: Ejemplo funcionamiento del bloque Xilinx Mux. Configuración básica del bloque (1)

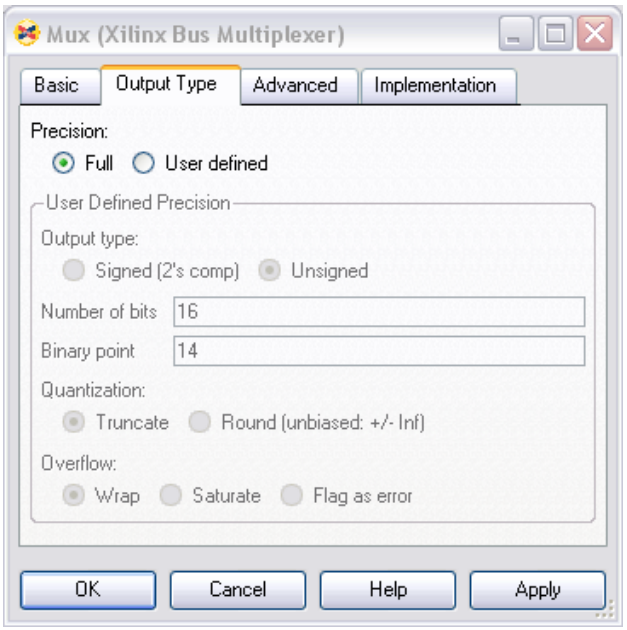

Figura 101: Ejemplo funcionamiento del bloque Xilinx Mux. Configuración básica del bloque (2)

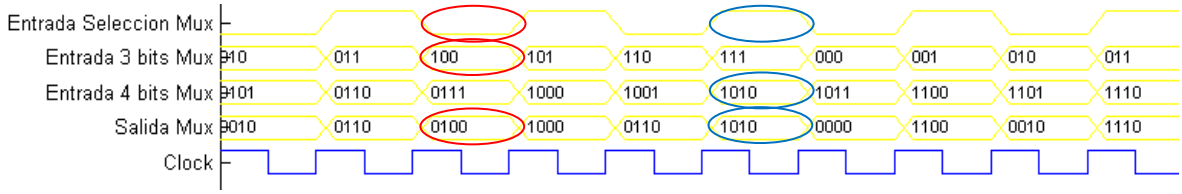

Figura 102: Ejemplo funcionamiento del bloque Xilinx Mux. Forma de la señales

4. Diseño del receptor

El receptor diseñado en este proyecto ha sido realizado mediante bloques pertenecientes al *Xilinx Blockset* de *Xilinx System Generator for DSP* [3], los cuales han sido caracterizados en el apartado "3.3 Descripción de los bloques empleados". Dicho diseño, cuyo diagrama de bloques se muestra en la figura 103, se corresponde con un **receptor para el estándar IEEE 802.16d-2004 utilizando MIMO 2x2, OFDM en la capa física y BPSK para la modulación de los datos.**

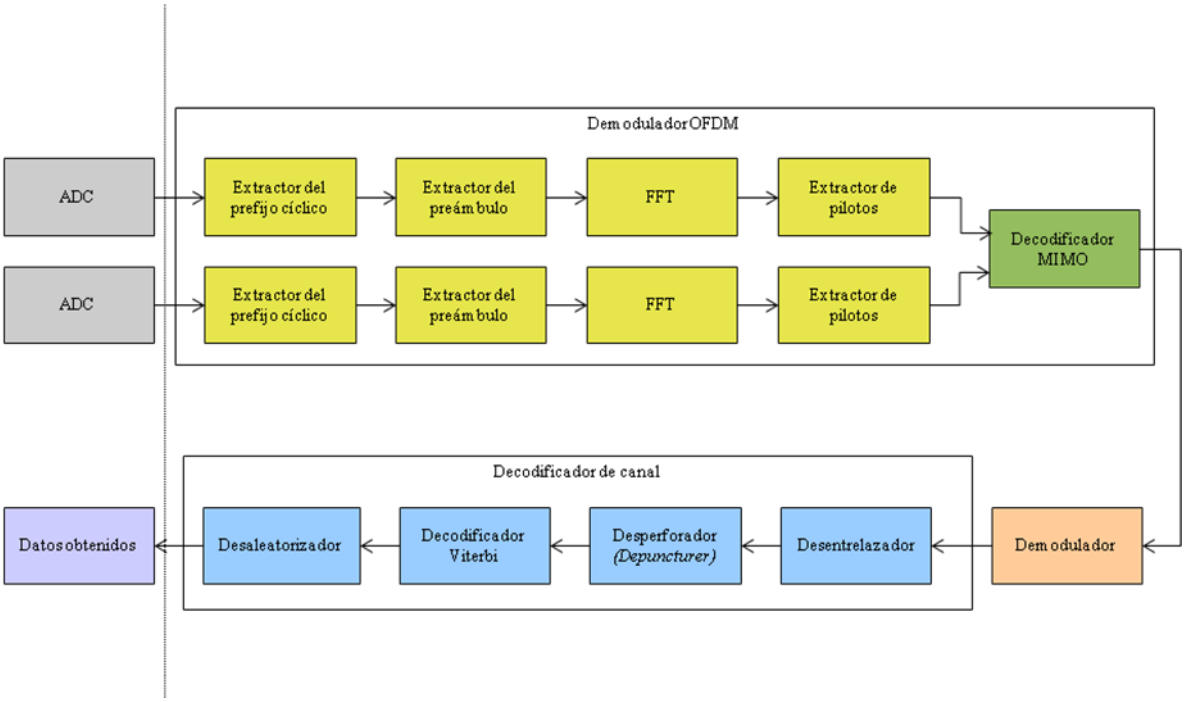

Figura 103: Diagrama de bloques del receptor diseñado para el estándar IEEE 802.16d-2004 utilizando MIMO 2x2, OFDM y BPSK

Inicialmente se trabajó en la realización de un receptor que, manteniendo MIMO 2x2 y OFDM en capa física, fuera capaz de emplear cualquier de las modulaciones de datos definidas en el estándar: BPSK, QPSK (Modulación en Cuadratura por Salto de Fase), 16-QAM y 64-QAM (donde QAM se refiere a Modulación en Amplitud por Cuadratura). Sin embargo, las pruebas iniciales demostraron que la complejidad del sistema crecía en base a dos factores inherentes a la plataforma empleada: sincronismo y espacio en memoria. Por ello, se decidió emplear la modulación BPSK, la cual permitía demostrar que la realización del receptor era posible utilizando los medios dispuestos.

Este proyecto **no considera la existencia de canal de comunicación**. Por tanto, las operaciones asociadas a su existencia (principalmente sincronización, estimación y compensación del canal) no son contempladas. Además, ciertos elementos del receptor se encuentran adaptados el tipo de **trama empleada**, la cual se repite cíclicamente y consta de **10 símbolos OFDM con modulación BPSK**.

Para finalizar los principios en los cuales se basa el receptor diseñado, se debe hacer notar la **naturaleza síncrona de los bloques empleados**. Para solucionar esta cuestión, en lugar de emplear señales de control que activen y desactiven los bloques, se ha procedido retardar los datos hasta el intervalo temporal adecuado para un funcionamiento correcto cada bloque.

Estas consideraciones y simplificaciones permiten disminuir la complejidad (principalmente computacional) del receptor diseñado. Si bien, éste mantiene los requisitos básicos recogidos en el estándar IEEE 802.16d-2004 [1].

Además, comentar que se sugiere como importante **línea de trabajo futura la síntesis, ejecución y validación del receptor diseñado sobre una FPGA**, pues el **alcance de este proyecto es la simulación y validación de dicho receptor sobre las herramientas software**.

El objetivo de este apartado es mostrar los bloques del *Xilinx Blockset* [3] empleados en cada etapa del receptor, justificando su estructura y configuración en base al estándar IEEE 802.16d-2004 (aplicado para MIMO 2x2, OFDM en la capa física y BPSK para la modulación de los datos) y otro tipo de consideraciones implícitas de la plataforma utilizada.

4.1 Extractor del prefijo cíclico

a) Requisitos IEEE 802.16d-2004

El estándar IEEE 802.16d-2004 define para el emisor, tras la conformación temporal de cada símbolo OFDM mediante la IFFT (tiempo T_b), la copia de un número determinado de sus muestras finales al principio del mismo (tiempo T_g), las cuales mantienen la ortogonalidad entre portadoras. Este proceso se muestra en la figura 104:

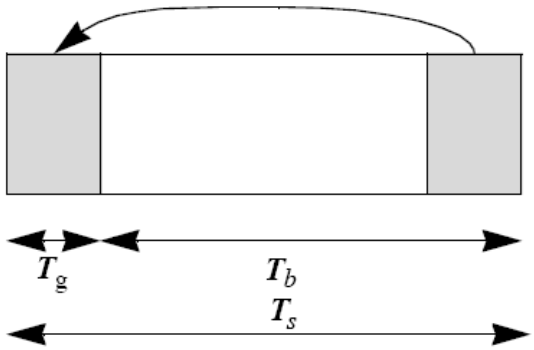

Figura 104: Prefijo cíclico IEEE 802.16d-2004
(Fuente: IEEE 802.16d-2004 [1])

Dichas muestras se denominan CP (Prefijo Cíclico) y su objetivo es evitar la ISI (Interferencia entre Símbolos), la cual es principalmente debida a los efectos del

multitrayecto (reflexiones que llegan con distintos retardos y provocan que puedan mezclarse rayos de distintos símbolos). El estándar IEEE 802.16d-2004 establece diferentes tamaños seleccionables para el prefijo cíclico, esto es, $G = T_g / T_b = 1/4, 1/8, 1/16$ o $1/32$. La elección de un determinado tamaño de prefijo cíclico dependerá de la respuesta al impulso del canal, siendo T_g mayor que ésta para que ningún rayo reflejado de un símbolo interfiera con el siguiente.

Por tanto, el receptor debe contar con una primera etapa que realice la extracción del prefijo cíclico de cada uno de los símbolos OFDM recibidos.

b) Diseño y justificación

El Extractor del prefijo cíclico diseñado consta de cuatro sistemas idénticos, realizando cada uno de ellos la extracción del prefijo cíclico para cada componente I y Q de cada antena. Por tanto, para cada una de estas componentes, el modelo diseñado es el siguiente (figura 105):

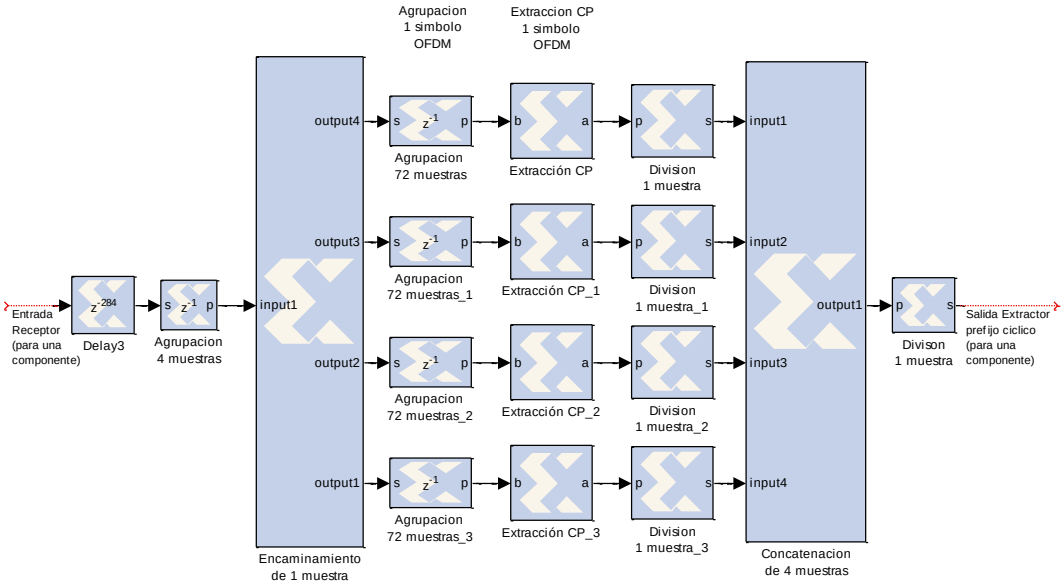

Figura 105: Extractor del prefijo cíclico diseñado (para cualquier componente de cualquier antena). Modelo

El Extractor del prefijo cíclico diseñado parte de la premisa de que el tamaño del prefijo cíclico es $G = T_g / T_b = 1/8$. Por tanto, una vez conocidas las muestras que representan dicho prefijo, debería ser suficiente con agrupar 1 símbolo OFDM mediante un bloque *Xilinx Serial to Parallel* y eliminar dichas muestras empleando un bloque *Xilinx BitBasher*.

Sin embargo, el bloque *Xilinx Serial to Parallel* no permite agrupaciones de bits tan grandes como se deseen, lo cual limita el número de bits de cada símbolo OFDM a los cual se puede aplicar la solución anterior directamente. En el caso de los símbolos OFDM-BPSK empleados en este proyecto (cada símbolo que llega a cada componente

de cada antena del receptor está formado por 288 muestras de 10 bits, es decir, 2880 bits por símbolo OFDM), este problema no afecta. De todas formas, se propone un modelo mediante el cual se podría ampliar el número de bits por símbolo, tanto por cuestiones de precisión como de variación de la modulación empleada.

El esquema diseñado realiza mediante 4 líneas en paralelo la operación de extracción del prefijo cíclico de un símbolo OFDM completo (para cada componente de cada antena). Inicialmente se distribuyen las muestras de cada símbolo por cada línea (bloque *Xilinx BitBasher* "Encaminamiento de 1 muestra"), esto es: muestra 1, 5, 9… para la primera línea; muestra 2, 6, 10… para la segunda línea; muestra 3, 7, 11… para la tercera línea; muestra 4, 8, 12… para la cuarta línea. A continuación, cada línea agrupa estas muestras hasta disponer de 1/4 de las muestras totales del símbolo OFDM (bloques *Xilinx Serial to Parallel* "Agrupacion 72 muestras"), teniendo en ese momento en total, entre las 4 líneas, la totalidad del símbolo OFDM. Finalmente, cada línea extrae las muestras correspondientes al prefijo cíclico de las que dispone (bloques *Xilinx BitBasher* "Extraccion CP"), siendo recombinadas las muestras restantes y serializadas para obtener el símbolo OFDM sin el prefijo cíclico.

Tal y como se comentó anteriormente, el símbolo OFDM-BPSK empleado en este proyecto (para cada componente de cada antena) consta de 288 muestras de 10 bits cada una (2880 bits por símbolo OFDM), de las cuales 32 se corresponden al prefijo cíclico (320 bits). Por tanto, cada una de las 4 líneas trabajará sobre 72 muestras (720 bits, como se muestra en la figura 106), extrayendo 8 muestras del prefijo cíclico (80 bits, como se muestra en la figura 107). El resultando a la salida es un símbolo OFDM-BPSK de 256 muestras (2560 bits).

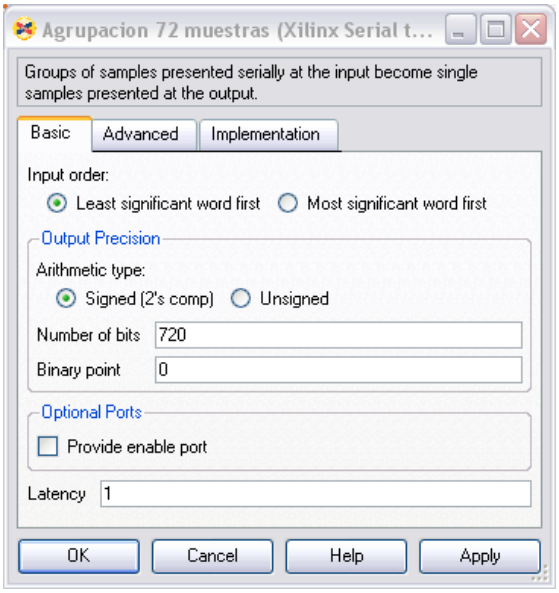

Figura 106: Extractor del prefijo cíclico diseñado. Configuración básica de los bloques Xilinx Serial to Parallel "Agrupacion 72 muestras"

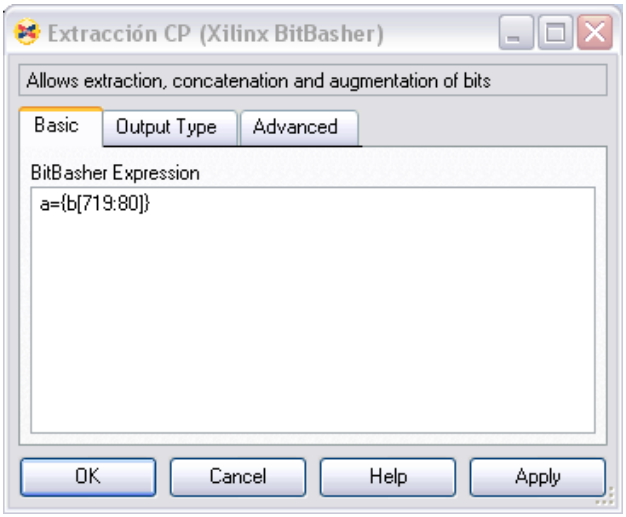

Figura 107: Extractor del prefijo cíclico diseñado. Configuración básica de los bloques Xilinx BitBasher "Extraccion CP"

4.2 Extractor del preámbulo

a) Requisitos IEEE 802.16d-2004

El estándar IEEE 802.16d-2004 establece la inclusión en el emisor, previamente a la operación de IFFT, de un preámbulo con el objetivo de permitir la sincronización entre el emisor y el receptor, así como la estimación del canal. Dicho preámbulo consiste en uno o dos símbolos OFDM de características conocidas que se inserta al principio de la trama.

Por tanto, en el receptor será necesario un elemento que localice el preámbulo y lo emplee para estimar el canal, compensando sus efectos sobre el resto de símbolos OFDM de la trama.

b) Diseño y justificación

El Extractor del preámbulo diseñado consta de cuatro sistemas idénticos, donde cada uno de ellos realiza la extracción del preámbulo para cada componente I y Q de cada antena. Por tanto, para cada una de estas componentes, el modelo diseñado es el siguiente (figuras 108 y 109):

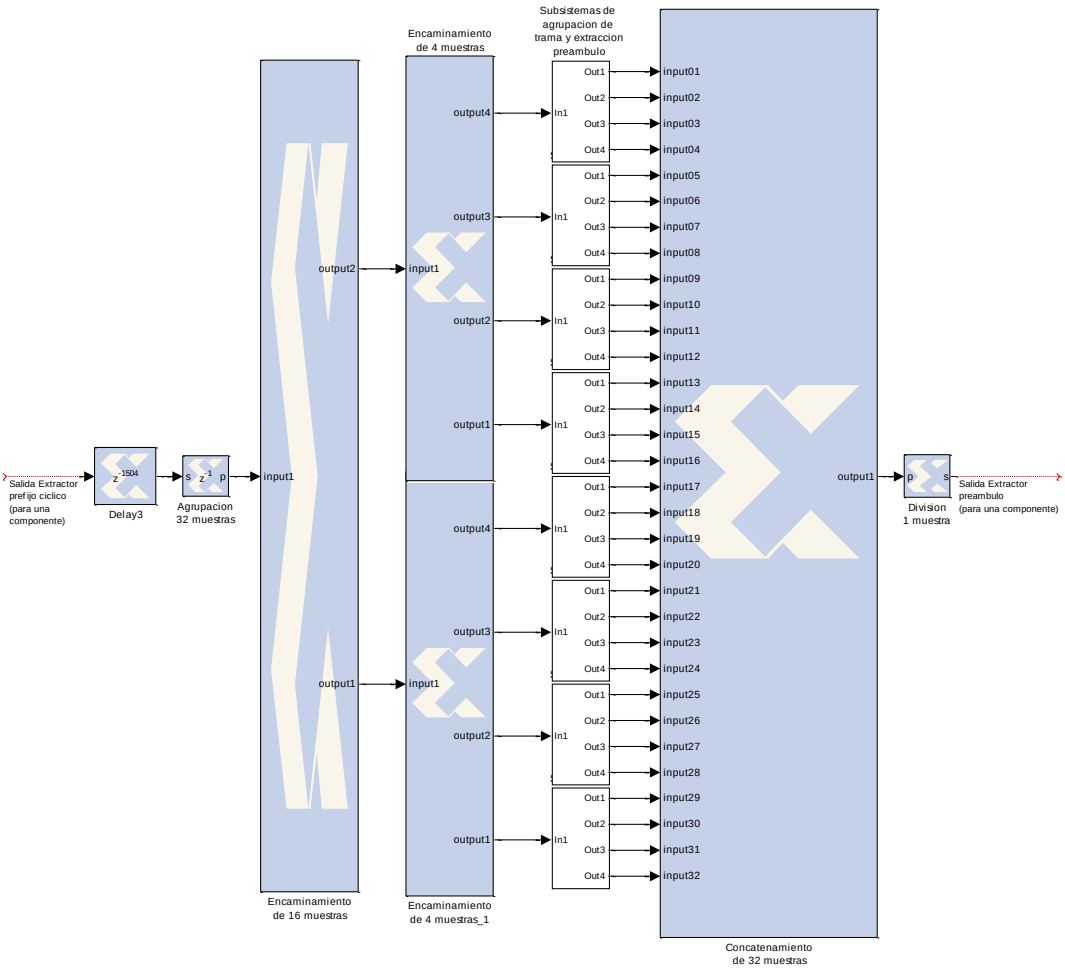

Figura 108: Extractor del preámbulo diseñado (para cualquier componente de cualquier antena). Modelo

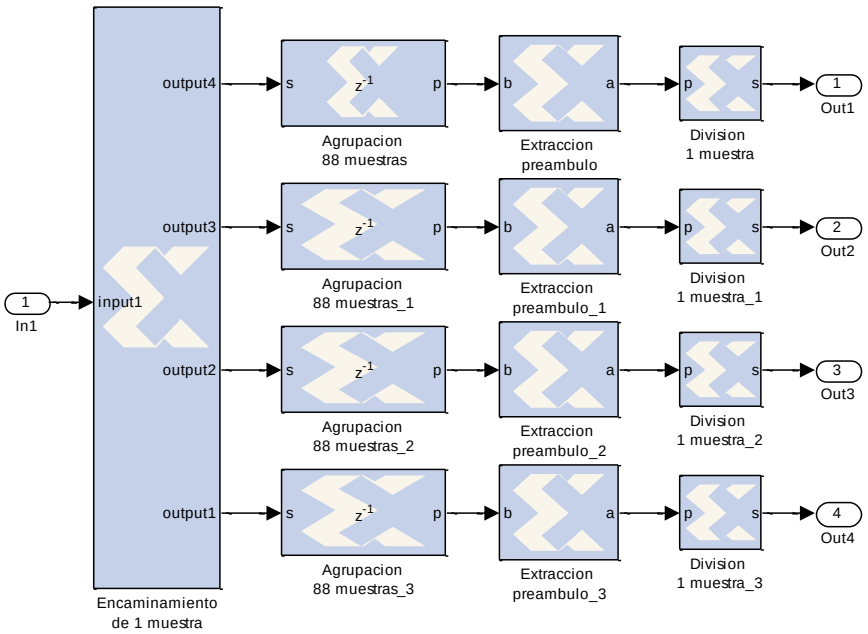

Figura 109: Subsistema de agrupación de trama y extracción del preámbulo diseñado. Modelo

Tal y como se comentó anteriormente, el Extractor del preámbulo debería emplearse a fin de permitir la sincronización entre el emisor y el receptor, así como la estimación del canal. Sin embargo, debido a que en este proyecto no se contempla la existencia de un canal, las operaciones asociadas a su existencia en el receptor no son necesarias.

Así, la extracción del preámbulo se realiza previamente a la operación de FFT a fin de reducir la complejidad computacional del sistema. Además, es posible realizar dicha operación debido a que se conoce la posición del preámbulo en todo momento, ya que no actúa canal ninguno, y que la operación de FFT afecta a símbolos OFDM completos.

La extracción del preámbulo implica trabajar sobre la trama al completo, identificando y eliminando el símbolo de preámbulo. Respecto al caso de este proyecto, y por cuestiones de sencillez computacional y de sincronismo, se considera la trama de tamaño fijo (formada por 10 símbolos OFDM-BPSK y el preámbulo).

El trabajar con dicha trama al completo requiere disponer de todas sus muestras en paralelo (bloque *Xilinx Serial to Parallel*) para poder eliminar el preámbulo mediante un bloque *Xilinx BitBasher*. Sin embargo y, tal y como ocurría en el Extractor del prefijo cíclico, el bloque *Xilinx Serial to Parallel* no permite agrupar cantidades de bits tan grandes como se desee. Por tanto, es necesario un diseño que realice la operación de extracción del preámbulo mediante diferentes líneas en paralelo.

El modelo diseñado se comporta de manera similar al empleado en el Extractor del prefijo cíclico, es decir, divide las muestras por diferentes líneas hasta disponer de toda la trama en paralelo y realizar la extracción del preámbulo. En este caso será necesario un árbol de bloques *Xilinx BitBasher* (“Encaminamiento de 16 muestras”, “Encaminamiento de 4 muestras” y Encaminamiento de 1 muestra”) hasta disponer de 32 líneas en paralelo con las muestras de la trama distribuidas de la siguiente manera: muestra 1, 33, 55…para la primera línea; muestra 2, 34, 56…para la segunda línea; muestra 3, 35, 57…para la tercera línea, etc. A continuación, cada una de las líneas agrupa estas muestras hasta disponer de 1/32 de las muestras totales de la trama (bloques *Xilinx Serial to Parallel* “Agrupacion 88 muestras”), teniendo en ese momento en total, entre las 32 líneas, la totalidad de la trama. Finalmente, cada línea extrae las muestras del preámbulo de las que dispone (bloques *Xilinx BitBasher* “Extracción preámbulo”), siendo recombinadas las muestras restantes y serializadas para obtener la trama sin el preámbulo.

Tal y como se ha comentado, la trama empleada en este proyecto consta de 10 símbolos OFDM-BPSK a la que se debe agregar el preámbulo. Cada uno de estos símbolos (para cada componente de cada antena), tras la extracción del prefijo cíclico, consta de 256 muestras de 10 bits cada una (2560 bits por símbolo OFDM), es decir, la trama de

entrada al Extractor del preámbulo cuenta con 28160 bits. Por tanto, cada una de las 32 líneas trabajará sobre 88 muestras (880 bits, como se muestra en la figura 110), extrayendo 8 muestras del preámbulo (80 bits, como se muestra en la figura 111). El resultando a la salida es una trama formada por 10 símbolos OFDM de 256 muestras con 10 bits cada una (25600 bits), para cada componente de cada antena.

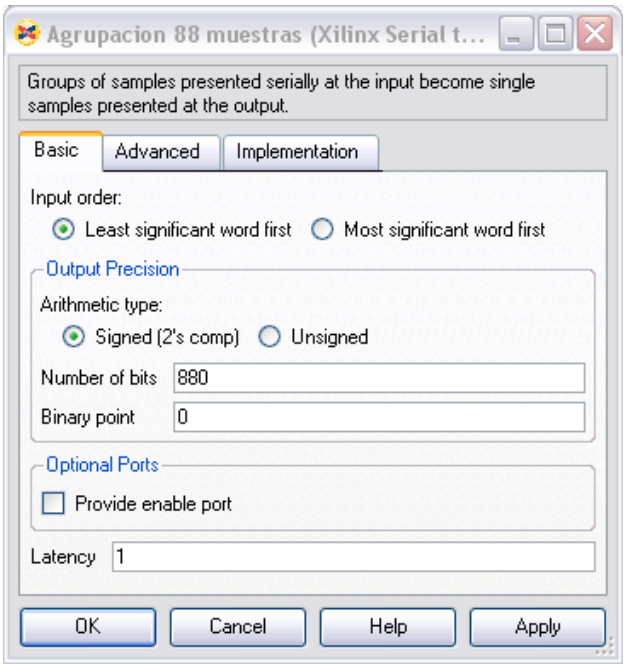

Figura 110: Extractor del preámbulo diseñado. Configuración básica de los bloques Xilinx Serial to Parallel "Agrupacion 88 muestras"

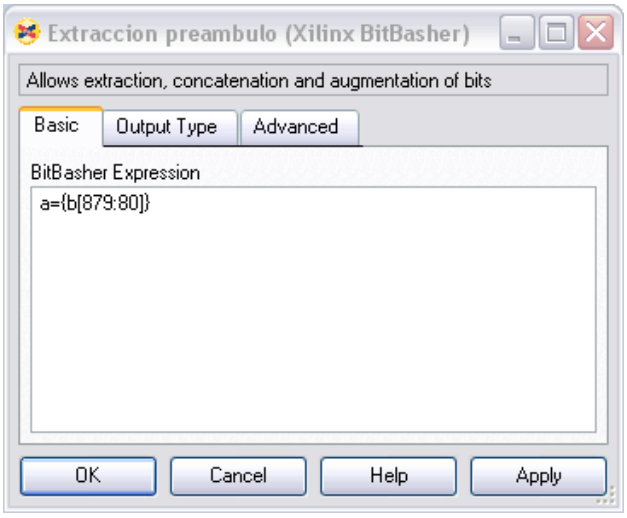

Figura 111: Extractor del preámbulo diseñado. Configuración básica de los bloques Xilinx BitBasher "Extraccion preambulo"

4.3 FFT

a) Requisitos IEEE 802.16d-2004

El estándar IEEE 802.16d-2004 establece la realización en el emisor de una operación de IFFT para la creación de la forma de onda temporal del símbolo OFDM. Esta operación se realiza una vez que dicho símbolo dispone en el dominio de la frecuencia de las subportadoras de datos, pilotos y nulas (bandas de guarda y subportadora de DC).

Así, el tamaño de la IFFT viene determinado por el número de subportadoras que conforman el símbolo OFDM. En el caso de este proyecto (modulación BPSK), el tamaño de dicha IFFT será de 256 subportadoras correspondientes a 192 subportadoras de datos, 8 de pilotos, 55 de bandas de guarda y una de DC.

Por tanto, el receptor deberá realizar una operación de FFT tras la extracción del prefijo cíclico y del preámbulo que permita trasladar el símbolo OFDM desde el dominio del tiempo al dominio de la frecuencia, con el objetivo de poder realizar las posteriores operaciones necesarias sobre las diferentes subportadoras que conforman dicho símbolo OFDM.

b) Diseño y justificación

La FFT diseñada consta de dos sistemas idénticos, donde cada uno de ellos realiza esta operación para las componentes real e imaginaria de cada antena. Por tanto, para cada una de estas antenas, el modelo diseñado es el siguiente (figuras 112 y 113):

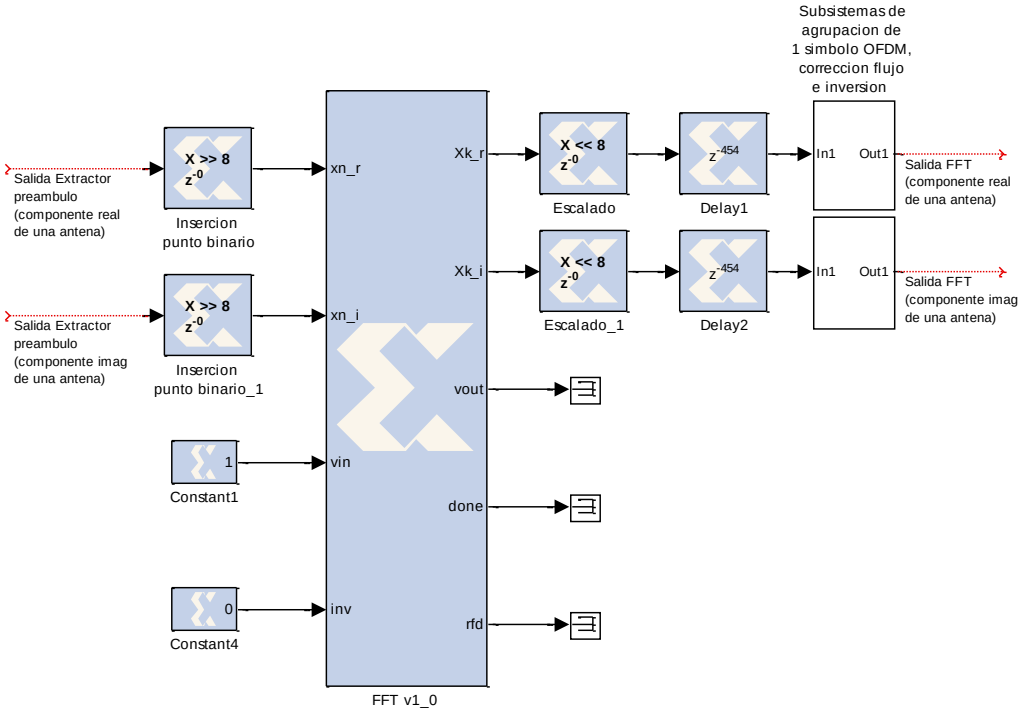

Figura 112: FFT diseñada (para la componente real e imaginaria de cualquier antena). Modelo

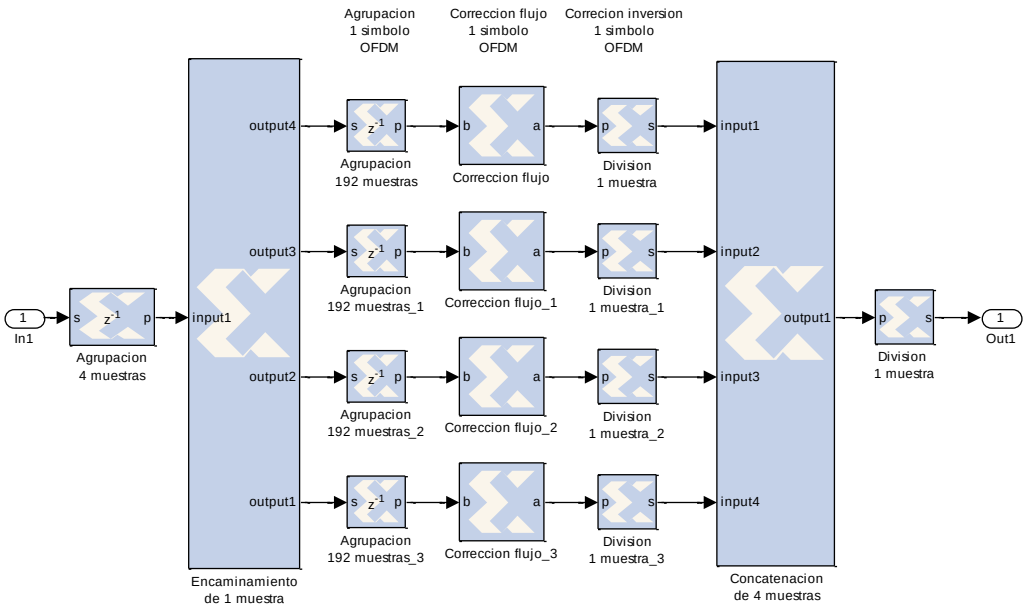

Figura 113: Subsistema de agrupación de 1 símbolo OFDM, corrección flujo e inversión diseñado. Modelo

Este sistema parte de la premisa de que el tamaño de la FFT y, por tanto, del símbolo OFDM, es de 256 subportadoras (192 subportadoras de datos, 8 de pilotos, 55 de bandas de guarda y una de DC). Por tanto, tras realizar la operación de FFT mediante el bloque *Xilinx FFT v1_0* (con ese tamaño) se esperaría recuperar el símbolo que el emisor introduce en la IFFT. Sin embargo, tal y como se comentaba en el apartado "3.3.12 Bloque Xilinx FFT v1_0", su operación de FFT produce tres efectos sobre la

salida: escalado por 1/N ó 1/(2N), ruptura del flujo de datos válidos e inversión de los mismos.

La corrección del efecto de escalado se realiza tanto a la entrada como a la salida del bloque *Xilinx FFT v1_0*. El bloque *Xilinx Shift* "Insercion punto binario" se encarga de incorporar el punto binario a los datos de entrada para mantener precisión tras el escalado, mientras que el bloque *Xilinx Shift* "Escalado" elimina dicho escalado multiplicando por N ó 2N los datos de salida.

La corrección de la ruptura del flujo de datos válidos a la salida (se triplican las muestras a la salida siendo únicamente válidas 1/3) se realiza tras solucionar el escalado. Para ello debería bastar con agrupar 1 símbolo OFDM, el cual contendrá el triple de muestras de las debidas, mediante un bloque *Xilinx Serial to Parallel* y eliminar las muestras de datos no válidos empleando un bloque *Xilinx BitBasher*.

Sin embargo, como ya se comentó en las etapas anteriores, el bloque *Xilinx Serial to Parallel* no permite agrupar cantidades de bits tan grandes como se desee. Por tanto, es necesario realizar esta corrección del flujo mediante diferentes líneas en paralelo, en este caso empleando 4. Inicialmente se distribuyen las muestras de cada símbolo por cada línea (bloque *Xilinx BitBasher* "Encaminamiento de 1 muestra"), esto es: muestra 1, 5, 9… para la primera línea; muestra 2, 6, 10… para la segunda línea; muestra 3, 7, 11… para la tercera línea; muestra 4, 8, 12… para la cuarta línea. A continuación, cada línea agrupa estas muestras hasta disponer de 1/4 de las muestras totales del símbolo OFDM (bloques *Xilinx Serial to Parallel* "Agrupacion 192 muestras"), teniendo en ese momento en total, entre las 4 líneas, la totalidad del símbolo OFDM. Finalmente, cada línea extrae las muestras de datos no válidas de las que dispone (bloques *Xilinx BitBasher* "Correccion flujo"), siendo recombinadas las muestras restantes y serializadas (aprovechando para corregir la inversión) para obtener el símbolo OFDM formado por los datos correctos.

Tal y como se comentó anteriormente, tras la extracción del prefijo cíclico y del preámbulo, el símbolo OFDM-BPSK empleado en este proyecto (para cada componente de cada antena) consta de 256 muestras de 10 bits cada una (2560 bits por símbolo OFDM). Tras la acción del bloque *Xilinx FFT v1_0* (figura 114), debido a la ruptura del flujo, se triplican el número de muestras, esto es, 768 por símbolo OFDM (7680 bits). A continuación, la corrección del escalado (figura 115) provoca que cada una de estas 768 muestras pase a encontrarse representada por 2 bits, teniendo 1536 bits por símbolo OFDM. Por tanto, cada una de las 4 líneas en paralelo trabajará sobre 192 muestras (384 bits, como se muestra en la figura 116) extrayendo 128 muestras de datos no válidos (256 bits, como se muestra en la figura 117).

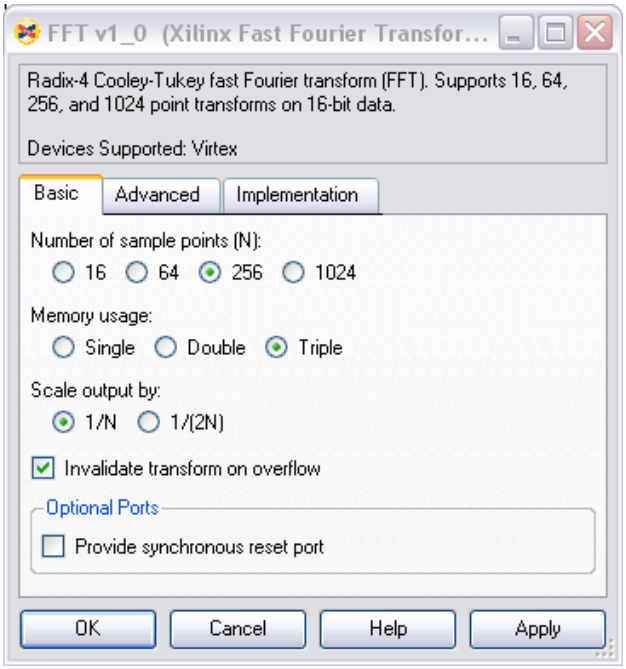

Figura 114: FFT diseñada. Configuración básica del bloque Xilinx FFT v1_0

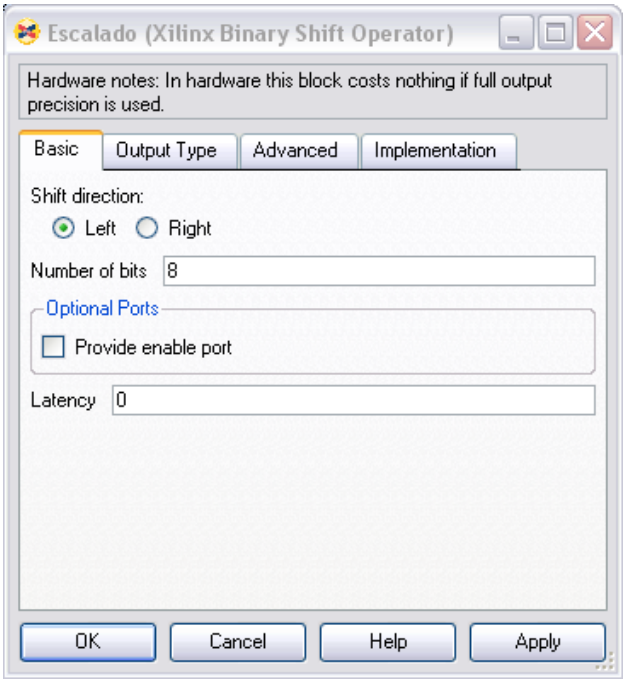

Figura 115: FFT diseñada. Configuración básica de los bloques Xilinx Shift "Escalado"

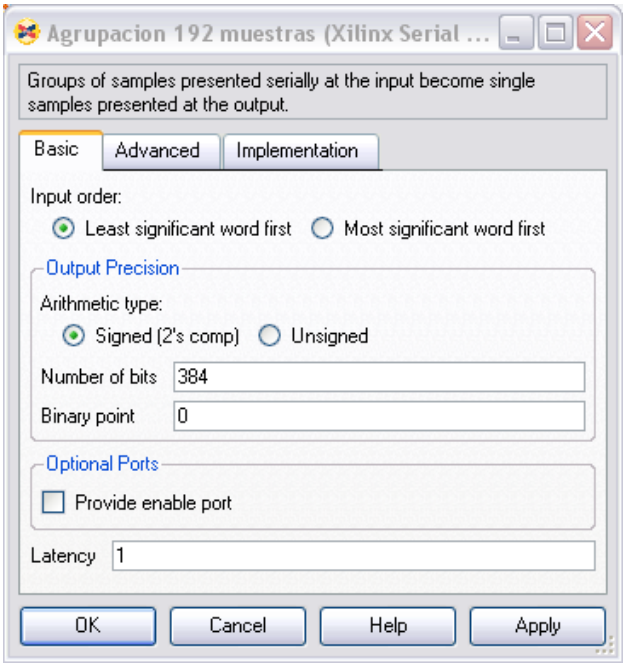

Figura 116: FFT diseñada. Configuración básica de los bloques Xilinx Serial to Parallel "Agrupacion 192 muestras"

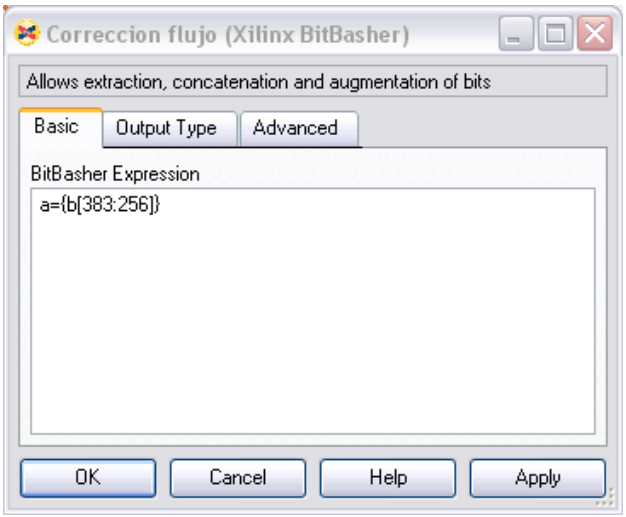

Figura 117: FFT diseñada. Configuración básica de los bloques Xilinx BitBasher "Correccion flujo"

4.4 Extractor de pilotos

a) Requisitos IEEE 802.16d-2004

El estándar IEEE 802.16d-2004 define en el emisor, previamente a las operaciones de inserción del preámbulo e IFFT, la inserción para cada símbolo OFDM de una serie de subportadoras a las de datos ya existentes (192 para el caso BPSK). Dichas subportadoras, tal y como se muestra en la figura 118, se tratan de las bandas de guarda, la subportadora de DC y las subportadoras de pilotos.

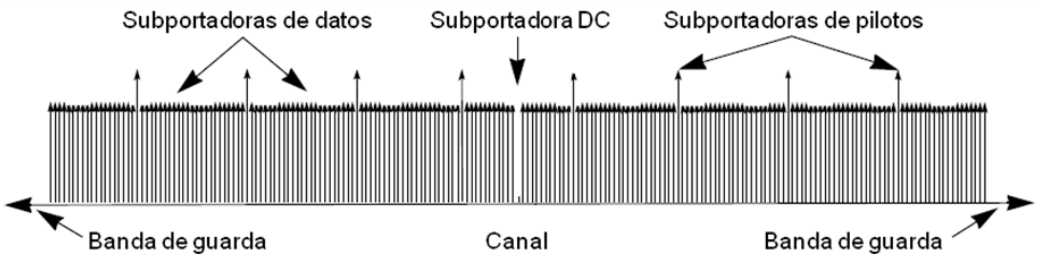

Figura 118: Descripción en frecuencia del símbolo OFDM IEEE 802.16d-2004
(Fuente: IEEE 802.16d-2004 [1])

Las bandas de guarda se incluyen lateralmente a fin de evitar la ACI (Interferencia de Canal Adyacente), siendo insertadas 28 subportadoras nulas en las frecuencias más bajas y 27 en las más altas del símbolo OFDM.

Las subportadoras de pilotos se incluyen con el objetivo de estimar el canal en el receptor y realizar la compensación adecuada. Se insertan 8 pilotos con un valor conocido en unas posiciones determinadas del símbolo OFDM.

La subportadora de DC se inserta en la frecuencia media del canal y su valor es nulo. Se considera que su posición es la 0, por tanto, las subportadoras del símbolo OFDM se consideran que se distribuyen desde la posición -128 (menor frecuencia) hasta la 127 (mayor frecuencia).

Por tanto, el receptor, tras la operación de FFT y extracción del preámbulo, debe eliminar las bandas de guarda y la subportadora DC. Además de emplear las subportadoras de pilotos para realizar las operaciones necesarias de estimación del canal con el objetivo de compensarlo.

Así, tras esta operación, el símbolo OFDM se encontrará formado únicamente por las subportadoras de datos.

b) Diseño y justificación

El Extractor de pilotos diseñado se encuentra formado por dos sistemas idénticos, donde cada uno de ellos realiza la misma operación para la componente real e imaginaria de cada antena. Por tanto, para cada una de estas antenas, el modelo diseñado es el siguiente (figura 119):

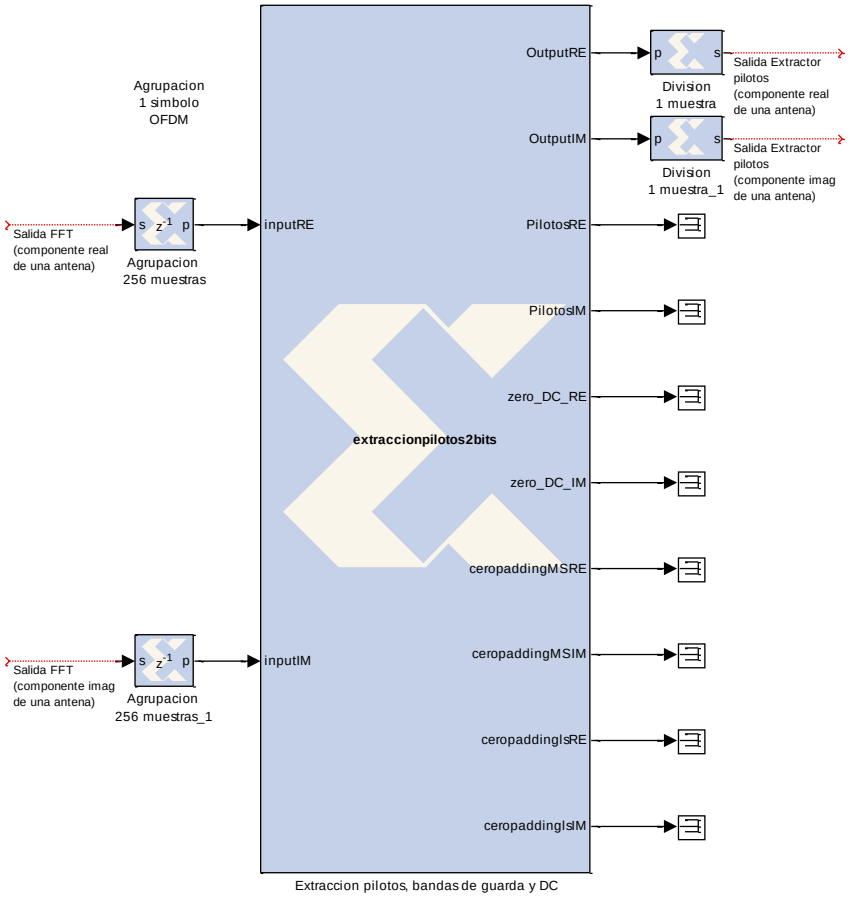

Figura 119: Extractor de pilotos diseñado (para la componente real e imaginaria de cualquier antena). Modelo

El modelo propuesto parte de la base de que no se contempla la existencia de canal. Por tanto, no serán necesarias las operaciones asociadas a su existencia (estimación y compensación).

Así, este diseño agrupa todos los bits de un símbolo OFDM y extrae las bandas de guarda (55), la subportadora de DC (1) y las subportadoras de pilotos (8). Para la primera acción se emplea un bloque *Xilinx Serial to Parallel* ("Agrupación 256 muestras"), mientras que para la segunda se utiliza un bloque *Xilinx MCode* ("Extracción pilotos, bandas de guarda y DC") que contiene una función en MATLAB® desarrollada para tal fin (`extraccionpilotos2bits.m`) e incluida en el apartado 9 ("Anexo A: Funciones MATLAB® implementadas").

Notar que en este caso no se han empleado líneas en paralelo para poder agrupar los bits que conforman el símbolo OFDM. Esto es debido a que tras la FFT el número de bits necesarios se reduce de tal manera que un bloque *Xilinx Serial to Parallel* tiene la capacidad de agruparlos.

Además, la decisión de emplear un código para extraer las subportadoras que no son de datos responde a sencillez que aporta y al deseo de mostrar la flexibilidad de la plataforma.

Respecto al número de bits implicados, el símbolo OFDM de entrada a esta etapa (para cada componente de cada antena) consta de 256 muestras de 2 bits cada una (512 bits). Dichas muestras se agrupan previamente al proceso de extracción, tal y como se muestra en la figura 120. Así, cada uno de estos símbolos, tras la extracción de las bandas de guarda (55 subportadoras), la subportadora DC (1) y las subportadoras de pilotos (8), está formado por 192 muestras de 2 bits cada una (384 bits). Por tanto, la trama resultante a la salida (formada por 10 símbolos OFDM) está compuesta por 3840 bits, para cada componente de cada antena.

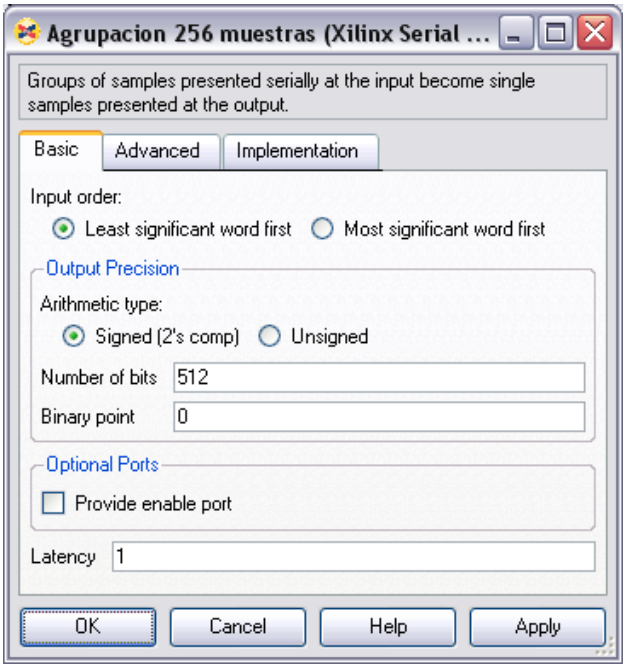

Figura 120 Extractor de pilotos diseñado. Configuración básica de los bloques Xilinx Serial to Parallel "Agrupacion 256 muestras"

4.5 Decodificador MIMO

a) Requisitos IEEE 802.16d-2004

El estándar IEEE 802.16d-2004 ofrece a posibilidad de emplear técnicas MIMO con el objetivo de mejorar la calidad del sistema de comunicación (incremento del radio de la celda y mayor capacidad). Así, en el emisor, tras la modulación y previamente a las operaciones de conformado del símbolo OFDM, se aplicará esta técnica de diversidad empleando dos antenas. Para ello se realiza la transmisión de la siguiente manera:

Primer uso del canal: *la antena 0 transmite s_0 y la antena 1 transmite s_1.*

Segundo uso del canal: *la antena 0 transmite $-s_1^*$ y la antena 1 transmite s_0^*.*

Donde:

s_0 es el primer símbolo que llega al codificador MIMO.

s_1 es el segundo símbolo que llega al codificador MIMO.

** es el conjugado.*

Notar que éste no se trata de un símbolo OFDM, sino del símbolo formado por la componente real e imaginaria.

Por tanto, el receptor debe ser capaz de deshacer esta operación teniendo en cuenta los efectos del canal y estimando los símbolos transmitidos, esto es:

$s_0' = h_0^* \cdot r_0 + h_1 \cdot r_1^*$

$s_1' = h_1^* \cdot r_0 - h_0 \cdot r_1^*$

Donde:

r_0 es el símbolo recibido en el primer uso del canal.

r_1 es el símbolo recibido en el segundo uso del canal.

h_0 es la respuesta del canal en el primer uso del canal.

h_1 es la respuesta del canal en el primer uso del canal.

** es el conjugado.*

b) Diseño y justificación

El Decodificador MIMO diseñado está formado dos sistemas análogos pero no idénticos, donde cada uno de ellos actúa sobre la componente real e imaginaria de cada una de las antenas de recepción. Por tanto, el modelo diseñado es el siguiente (figura 121):

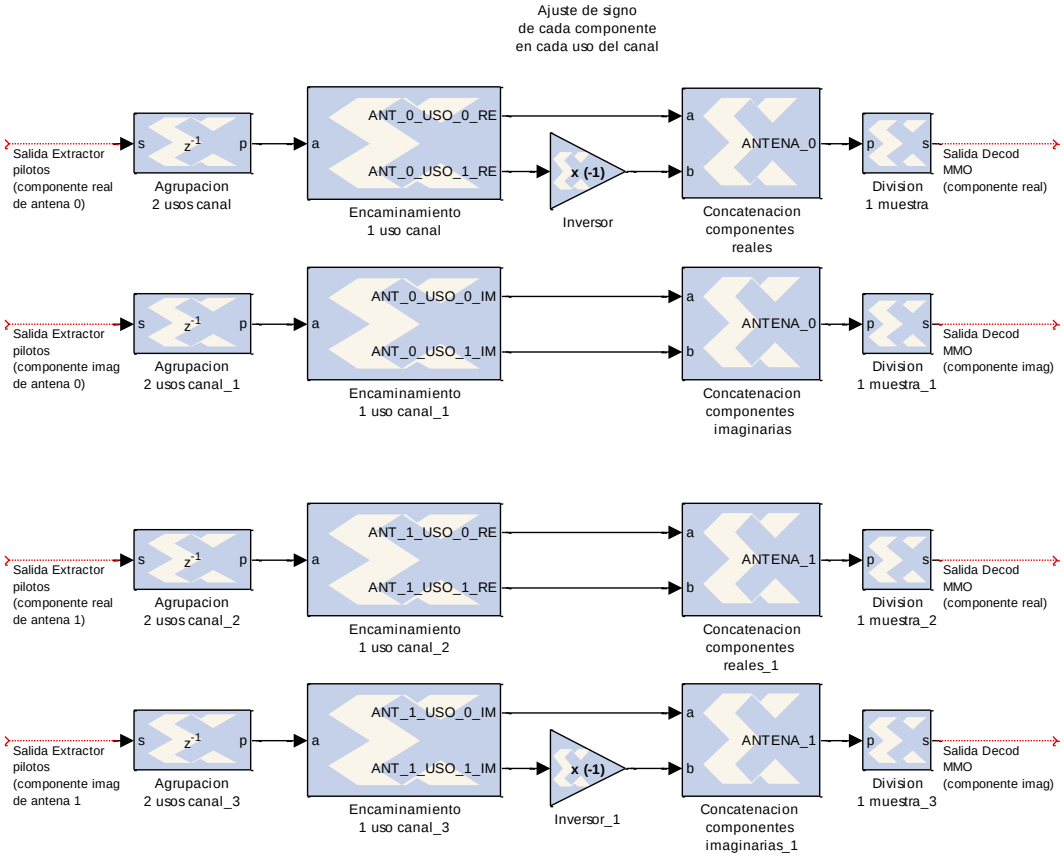

Figura 121: Decodificador MIMO diseñado. Modelo

Este modelo parte de la base de que no se dispone de canal. Por tanto, no es necesario estimar el símbolo recibido en cada uno de los usos del canal empleando la respuesta del canal durante dichos usos. En su lugar se realiza la operación inversa a la realizada en transmisión, trabajando de manera independiente con cada componente real e imaginaria de cada antena.

Inicialmente, este sistema agrupa los símbolos correspondientes a dos usos del canal (mediante los bloques *Xilinx Serial to Parallel* "Agrupacion 2 usos canal") durante los cuales la antena 0 recibe s_0 y $-s_1^*$, y la antena 1 recibe s_1 y s_0^*, respectivamente. A continuación se sitúan ambos usos en paralelo, de tal manera que se dispone de las componentes real e imaginaria recibida por cada antena en los dos usos del canal (bloques *Xilinx BitBasher* "Encaminamiento 1 uso canal"). El siguiente paso consiste en ajustar el signo de cada una de las componentes real e imaginaria para, finalmente, concatenar dichas componentes y recuperar los símbolos (s_0 y s_1), los cuales aparecen en ambas antenas a la salida formados por su parte real e imaginaria.

Respecto al número de bits implicados, esta etapa no modifica su número en la salida, es decir, la trama resultante (formada por 10 símbolos OFDM) está compuesta por 3840 bits para cada componente de cada antena (192 muestras de 2 bits cada una por

símbolo OFDM). Sin embargo, es interesante recordar que, al obtener s_0 y s_1 en ambas antenas, las salidas de una de ellas no serán empleadas en la entrada al demodulador.

Además, notar que cada componente (real e imaginaria) de cada símbolo de entrada consta de 2 bits. Así, cuando se agrupan los dos usos para cada componente de cada antena, se están agrupando 4 bits (como se muestra en la figura 122), los cuales se vuelven a dividir en 2 bits para su entrada en el demodulador.

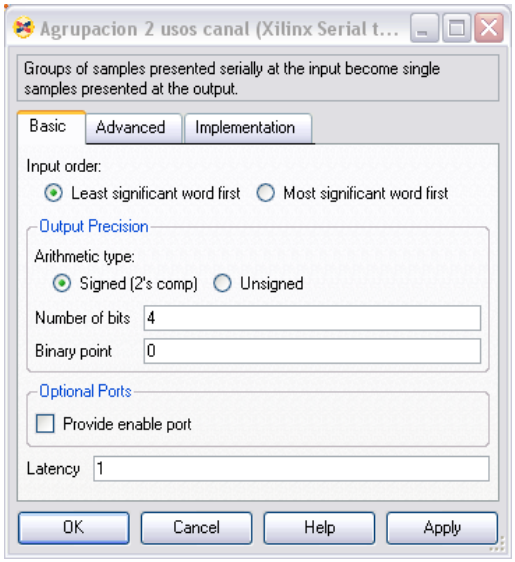

Figura 122 Decodificador MIMO diseñado. Configuración básica de los bloques Xilinx Serial to Parallel "Agrupacion 2 usos canal"

4.6 Demodulador

a) Requisitos IEEE 802.16d-2004

El estándar IEEE 802.16d-2004 define en el emisor, tras el proceso de entrelazado (*interleaving*) y previamente a las operaciones de conformado del símbolo OFDM, una etapa de modulación. Dicha etapa soporta constelaciones del tipo BPSK, QPSK, 16-QAM y 64-QAM. Además, cada punto de la constelación (donde b_0 indica el bit menos significativo) debe ser normalizado mediante un factor 'c' para conseguir la misma potencia media, como se muestra en la figura 123:

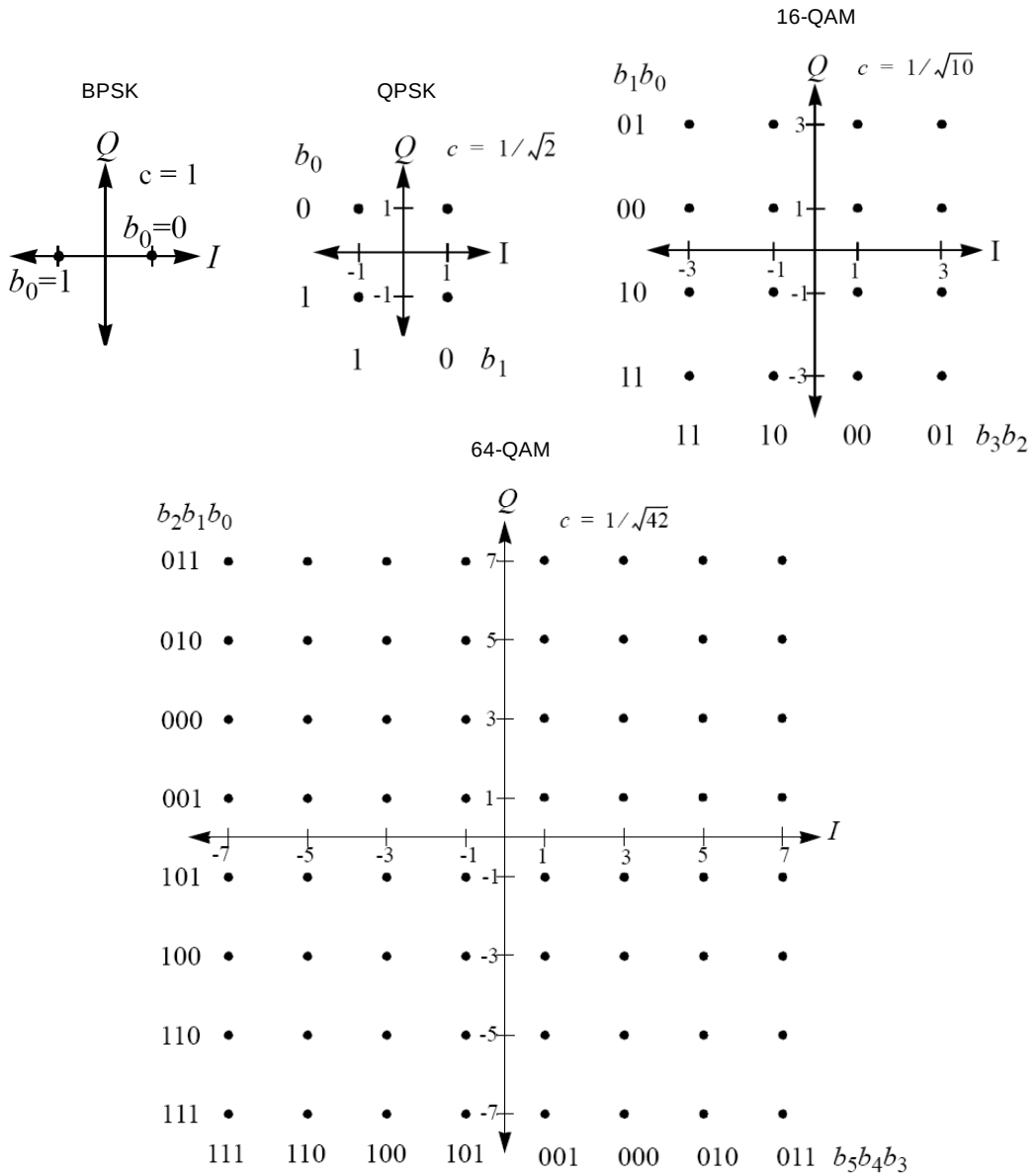

Figura 123: Constelaciones BPSK, QPSK, 16-QAM y 64-QAM IEEE 802.16d-2004
(Fuente: IEEE 802.16d-2004 [1])

Por tanto, el receptor, tras el proceso de demodulación del símbolo OFDM y la decodificación MIMO, debe contar con el demodulador correspondiente según el tipo de constelación empleada.

b) Diseño y justificación

El diseño del Demodulador se corresponde con el siguiente modelo (figura 124):

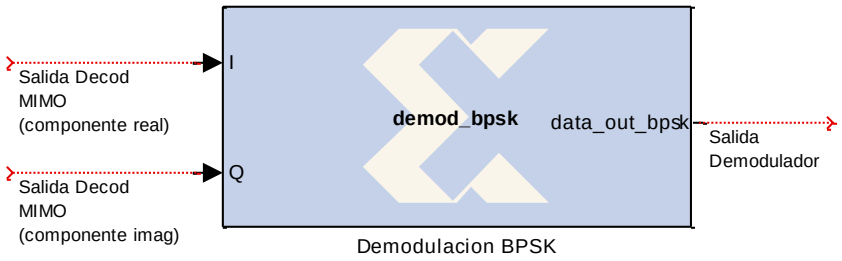

Figura 124: Demodulador diseñado. Modelo

El modelo propuesto realiza la demodulación de la constelación considerada en este proyecto, es decir, BPSK. Por tanto, se tiene en cuenta el hecho de que en el emisor se realiza la operación de modulación a través de la componente I, estableciendo un valor constante para Q (en este caso 0) para mantener la coherencia del sistema.

Así, el demodulador BPSK consiste en un bloque *Xilinx MCode* ("Demodulación BPSK") que contiene una función en MATLAB® desarrollada para tal fin (`demod_bpsk.m`) incluida en el apartado 9 ("Anexo A: Funciones MATLAB® implementadas"). La decisión de emplear un código para realizar la demodulación es debido a su sencillez de implementación y el deseo de mostrar la flexibilidad de la plataforma.

Por tanto, el demodulador BPSK, a través del código, implementa un sencillo decisor para la componente I con el umbral en 0. De tal forma que cualquier valor por debajo de ese umbral se considerará generado en el modulador por un bit de valor 1 y cualquier valor por encima del umbral se considerará generado en el modulador por un bit de valor 0. Tal y como se comentó anteriormente, la componente Q es eliminada debido a que en el emisor, el proceso de modulación de realiza a través de la componente I.

Respecto al número de bits implicados, cada componente (I y Q) de cada símbolo de entrada a este bloque consta de 2 bits, es decir, 384 bits por símbolo OFDM y componente (3840 bits por trama de 10 símbolos OFDM y componente). La operación de demodulación BPSK hace que se representen cada uno de esos símbolos de la componente I por 1 bit. Además, se elimina la componente Q. Por tanto, a la salida se obtienen 192 bits por símbolo OFDM (1920 bits por trama de 10 símbolos OFDM).

4.7 Desentrelazador

a) Requisitos IEEE 802.16d-2004

El estándar IEEE 802.16d-2004 define en el emisor, tras el proceso de perforación (*puncturing*) y previamente a la modulación, un proceso de entrelazado (*interleaving*) de todos los bits de datos. Este entrelazado tiene un doble objetivo: asegurar que los bits

codificados consecutivamente no vayan a subportadoras adyacentes y evitar que se formen colas de bits menos significativos al transformar los bits codificados en puntos de la constelación.

Las expresiones que definen este proceso de entrelazado, el cual se realiza en dos pasos, son las siguientes:

Primera permutación: $m_k = (N_{cbps}/12) \cdot k_{mod12} + floor(k/12)$

Segunda permutación: $j_k = s \cdot floor(m_k/s) + (m_k + N_{cbps} - floor(12 \cdot m_k/N_{cbps}))_{mod(s)}$

Donde:

k es la posición de un bit codificado antes de la primera permutación ($k = 0, 1, \ldots, N_{cbps}-1$).

m_k es la posición del bit k tras la primera permutación y antes de la segunda.

j_k es la posición del bit k tras la segunda permutación.

N_{cbps} es el número de bits codificados en un bloque.

N_{cpc} es el número de bits codificados por subportadora (1, 2, 4 o 6 para BPSK, QPSK, 16-QAM o 64-QAM, respectivamente).

$s = ceil(N_{cpc}/2)$.

Por tanto, el Desentrelazador debe realizar en el receptor la operación inversa a la que realiza el Entrelazador en el emisor. Ésta también consiste en dos permutaciones, de tal manera que la primera permutación del Desentrelazador es la inversa de la segunda permutación del Entrelazador y la segunda permutación del Desentrelazador es la inversa de la primera permutación del Desentrelazador.

Así, las expresiones que definen el proceso de desentrelazado son las siguientes:

Primera permutación: $m_j = s \cdot floor(j/s) + (j + floor(12 \cdot j/N_{cbps}))_{mod(s)}$

Segunda permutación: $k_j = 12 \cdot m_j - (N_{cbps} - 1) \cdot floor(12 \cdot m_j/N_{cbps})$

Donde:

j es la posición de un bit recibido antes de la primera permutación ($j = 0, 1, \ldots, N_{cbps}-1$).

m_j es la posición del bit j tras la primera permutación y antes de la segunda.

k_j es la posición del bit j tras la segunda permutación.

N_{cbps} es el número de bits del bloque recibido.

N_{cpc} es el número de bits codificados por subportadora (1, 2, 4 o 6 para BPSK, QPSK, 16-QAM o 64-QAM, respectivamente).

$s = ceil(N_{cpc}/2)$.

Respecto al tamaño de bloque empleado, el estándar IEEE 802.16d-2004 establece un tamaño de 192 bits en el caso de emplear modulación BPSK.

b) Diseño y justificación

El Desentrelazador diseñado se corresponde con el siguiente modelo (figura 125):

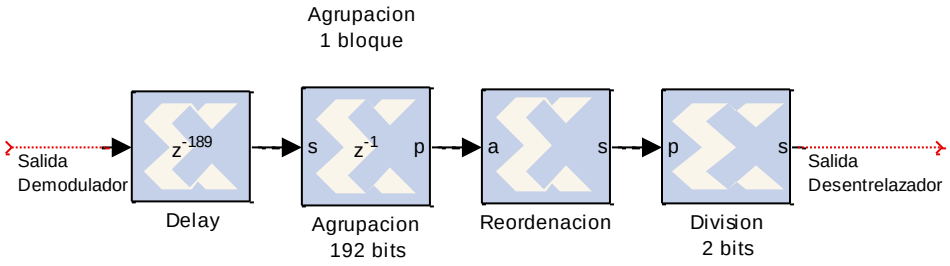

Figura 125: Desentrelazador diseñado. Modelo

Este modelo realiza el desentrelazado de los bits recibidos del Demodulador teniendo en cuenta que la modulación empleada es BPSK (bloques de 192 bits) y que la secuencia de reordenamiento es fija para cada bloque de bits recibidos.

El diseño, inicialmente y con el fin de disponer de todos los bits que conforman un bloque, agrupa dichos bits mediante un bloque *Xilinx Serial to Parallel* (“Agrupacion 192 muestras”). A continuación, mediante un bloque *Xilinx BitBasher* (“Reordenacion”) se realiza la operación de desentrelazado como tal, conociendo previamente la posición de salida asignada a cada uno de los bits de entrada.

Respecto al número de bits implicados, esta etapa no modifica su número a la salida. Cada símbolo OFDM de entrada consta de 192 bits (1920 bits por trama de 10 símbolos OFDM), lo cuales son agrupados para realizar correctamente el proceso de desentrelazado (como se muestra en las figuras 126 y 127), permaneciendo invariantes el número de bits a la salida (192 bits por símbolo OFDM).

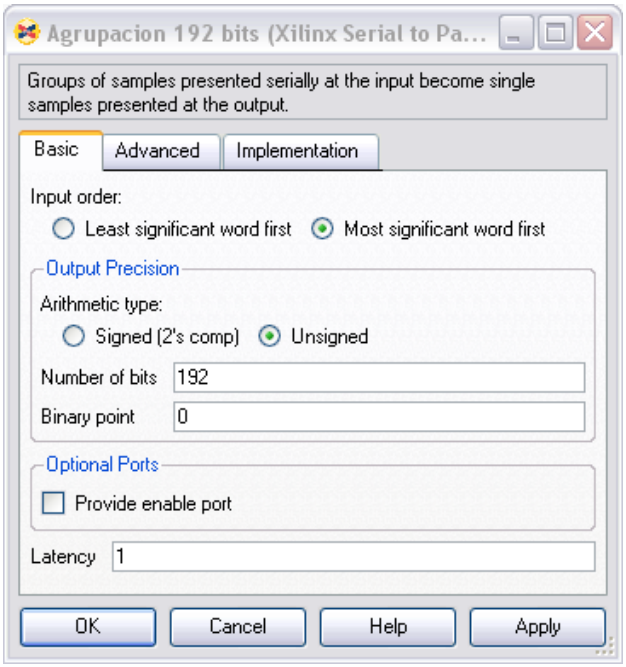

Figura 126: Desentrelazador diseñado. Configuración básica del bloque Xilinx Serial to Parallel "Agrupacion 192 bits"

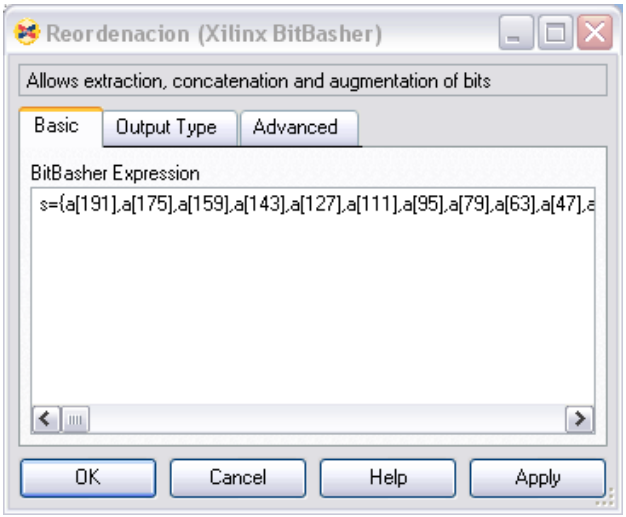

Figura 127: Desentrelazador diseñado. Configuración básica del bloque Xilinx BitBasher "Reordenacion"

4.8 Desperforador (Depuncturer)

a) Requisitos IEEE 802.16d-2004

El estándar IEEE 802.16d-2004 define en el emisor, tras el Codificador Convolucional, un proceso de perforación (*puncturing*) con el objetivo de ajustar la tasa de transmisión. Para el caso de BPSK, al emplear una tasa de codificación 1/2, el proceso de perforación consiste en concatenar las salidas del Codificador Convolucional ($X_1,Y_1 \rightarrow X_1Y_1$).

Por tanto, en el receptor se hace necesario un proceso de desperforación (*depuncturing*) previamente al Decodificador Viterbi para deshacer la perforación (*puncturing*) realizada en el emisor. Así, al emplear BPSK, el Desperforador deberá dividir en dos líneas los datos provenientes del Desentrelazador ($X_1Y_1 \rightarrow X_1,Y_1$) para dirigirlas hacia el Decodificador Viterbi.

b) Diseño y justificación

El diseño del Desperforador (*Depuncturer*) se corresponde con el siguiente modelo (figura 128):

Figura 128: Desperforador (Depuncturer) diseñado. Modelo

Este diseño parte del hecho de que la modulación de datos empleada es BPSK. Por ello, el proceso de desperforación (*depuncturing*) consistirá en dividir en dos líneas los datos provenientes del Desentrelazador ($X_1Y_1 \rightarrow X_1,Y_1$), siendo realizado este proceso mediante un bloque *Xilinx BitBasher* (“Desperforacion”).

El número de bits de entrada a esta etapa es de 192 por símbolo OFDM (1920 bits por trama de 10 símbolos OFDM). Este número no es modificado como consecuencia de la desperforación, sino que dichos bits son distribuidos (figura 129), contando cada línea de salida con 96 bits por símbolo OFDM (960 bits por trama de 10 símbolos OFDM).

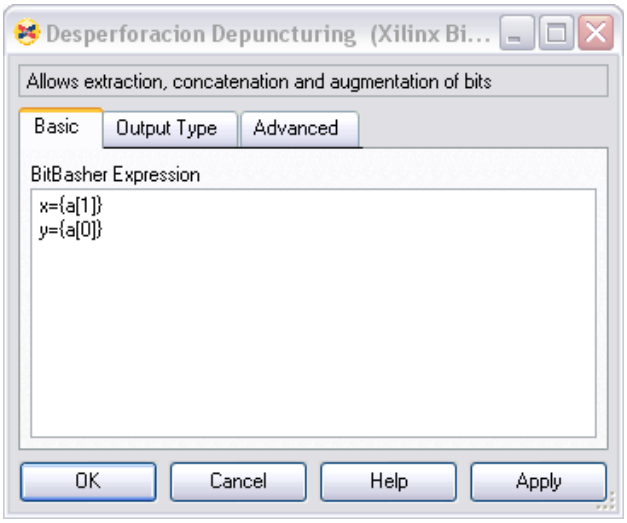

Figura 129: Desperforador (Depuncturer) diseñado. Configuración básica del bloque Xilinx BitBasher "Desperforacion"

4.9 Decodificador Viterbi

a) Requisitos IEEE 802.16d-2004

El estándar IEEE 802.16d-2004 establece en el emisor, tras la aleatorización de los bits de información, el empleo de un Codificador Convolucional con un código nativo de tasa 1/2, longitud restringida de 7 y dos polinomios generadores: $G_1=171_{OCT}$ (para la salida X) y $G_2=133_{OCT}$ (para la salida Y). En la figura 130 se muestra su representación:

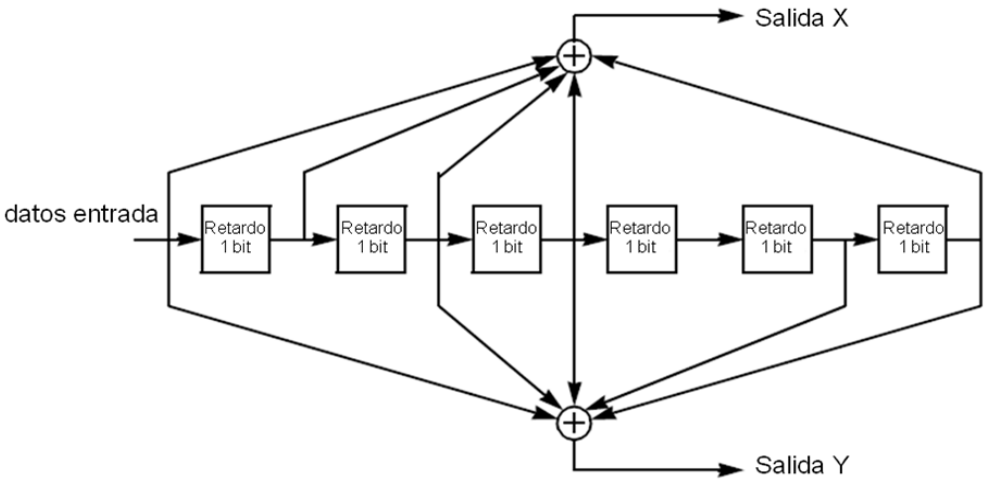

Figura 130: Codificador Convolucional IEEE 802.16d-2004
(Fuente: IEEE 802.16d-2004 [1])

Por tanto, el receptor deberá contar con un Decodificador que sea capaz de deshacer la acción en el emisor del Codificador Convolucional, obteniendo a su salida los bits que se

encaminarán hacia el Desaleatorizador con el fin de recomponer la secuencia originalmente transmitida.

b) Diseño y justificación

El Decodificador diseñado se corresponde con el siguiente modelo (figura 131):

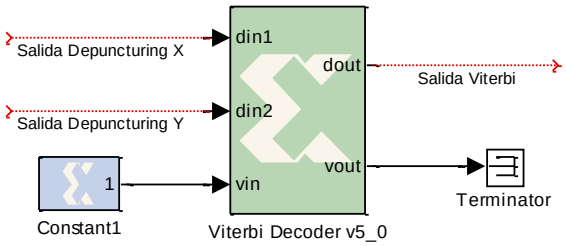

Figura 131: Decodificador Viterbi diseñado. Modelo

El diseño de esta etapa parte de las características del Codificador Convolucional situado en el emisor: longitud restringida de 7 y dos polinomios generadores: $G_1=171_{OCT}$ y $G_2=133_{OCT}$.

Para realizar la decodificación es suficiente con utilizar el bloque *Xilinx Viterbi Decoder v5_0* (descripción en el apartado "3.3.14 Bloque Xilinx Viterbi Decoder"). Este bloque permite realizar la decodificación de manera correcta mediante una adecuada configuración, tal y como se muestra en la figura 132:

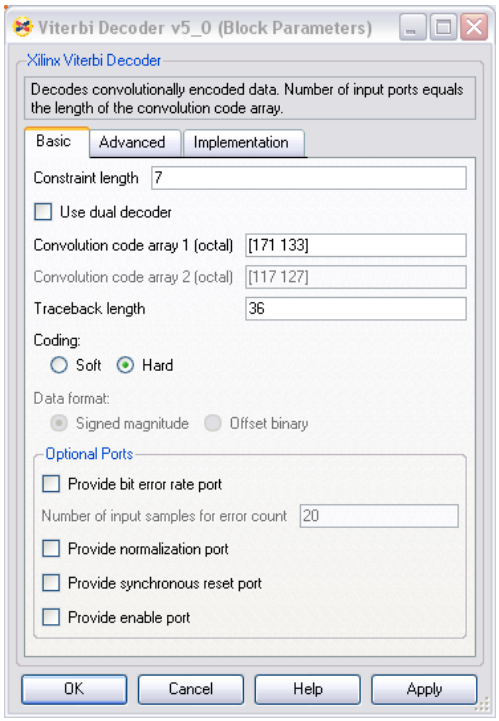

Figura 132: Decodificador Viterbi diseñado. Configuración básica del bloque Xilinx Viterbi Decoder v5_0

Respecto al número de bits, la entrada a esta etapa son dos líneas con 96 bits por símbolo OFDM cada una (960 bits por trama de 10 símbolos OFDM), es decir, en total las dos líneas de entrada transportan 192 bits por símbolo OFDM (1920 bits por trama de 10 símbolos OFDM). La decodificación conlleva que por cada bit de entrada de ambas líneas se produce un bit a la salida, es decir, el número de bits totales a la salida se reducen a la mitad. Por tanto, a la salida se tendrán 96 bits por símbolo OFDM (960 bits por trama de 10 símbolos OFDM).

4.10 Desaleatorizador

a) Requisitos IEEE 802.16d-2004

El estándar IEEE 802.16d-2004 define, como primera operación en el emisor, una aleatorización de los bits de información con el objetivo de evitar largas secuencias de 0 y 1. Dicho Aleatorizador consiste, básicamente, en un registro de desplazamiento de 15 bits con 2 funciones XOR y un vector de reinicio al inicio de cada trama ([1 0 0 1 0 1 0 1 0 0 0 0 0 0 0]), tal y como se muestra en la figura 133:

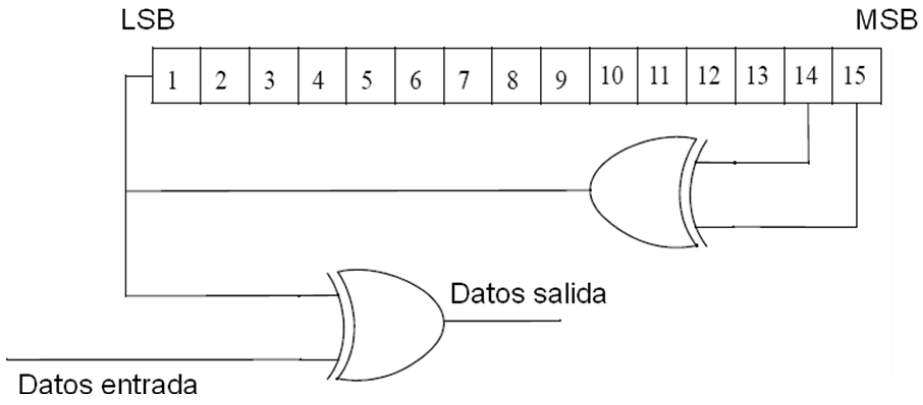

Figura 133: Aleatorizador IEEE 802.16d-2004
(Fuente: IEEE 802.16d-2004 [1])

Por tanto, el receptor deberá contar con una etapa que realice la operación inversa con el fin de recuperar los bits de información transmitidos originalmente. En este caso, el Desaleatorizador del receptor se define de la misma manera que el Aleatorizador del emisor.

b) Diseño y justificación

El diseño del Desaleatorizador se corresponde con el siguiente modelo (figura 134):

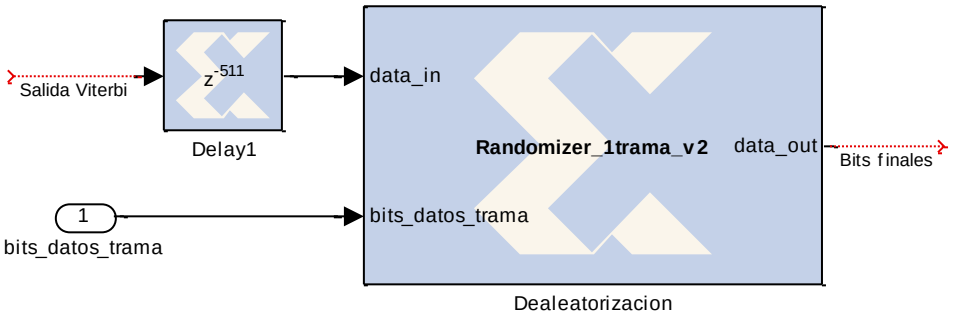

Figura 134: Desaleatorizador diseñado. Modelo

El principal componente del Desaleatorizador es un bloque *Xilinx MCode* ("Desaleatorizacion") que contiene una función MATLAB® creada para tal fin (`Randomizer_1trama_v2.m`) y se incluye en el apartado 9 ("Anexo A: Funciones MATLAB® implementadas"). La decisión de emplear código para realizar esta operación en lugar de bloques *Xilinx Register* se debe a que se requieren 15 de estos bloques, además de la lógica adecuada, lo cual es resuelto fácilmente mediante código. Además, permite mostrar la flexibilidad de la plataforma mediante el uso de un código MATLAB®.

Así, el código implementado realiza la desaleatorización de los datos de entrada en base a la función indicada en el estándar IEEE 802.16d-2004 (registro de desplazamiento con 2 funciones XOR y un vector de reinicio para cada trama). Al tratarse de un prototipo, con el objetivo de aumentar las prestaciones y la simplicidad, este código se encuentra adaptado al hecho de que las fuentes de datos empleadas en este proyecto (una fuente de 1 bit y otra de 8 bits) transmiten cíclicamente 1 trama de 10 símbolos OFDM (BPSK), siendo separada cada retransmisión por una cantidad de ceros con un tamaño equivalente a un símbolo OFDM en BPSK (96 bits). Además, se aprovecha el código implementado parar eliminar el *tailbyte* introducido en el emisor.

Respecto al número de bits implicados, a la entrada de esta última etapa del receptor se tienen 96 bits por símbolo OFDM (960 bits por trama de 10 símbolos OFDM). El proceso de desaleatorización no modifica este número, si bien la eliminación del *tailbyte* hace que a la salida se obtengan, definitivamente, 952 bits por trama de 10 símbolos OFDM. Posteriormente, estos bits deberán ser agrupados en un número adecuado (según el tipo de fuente) para recuperar los datos introducidos en el emisor. En este caso, puesto que una de las fuentes es de 1 bit, sólo será necesario agruparlos para la fuente de 8 bits mediante un bloque *Xilinx Serial to Parallel*.

5. Validación, simulación y resultados

La validación del receptor diseñado en este proyecto ha sido realizada mediante dos **fuentes de datos** introducidas en el emisor:

a) Fuente de 1 bit: contador libre ascendente de 1 bit (0, 1, 0, 1…).

b) Fuente de 8 bits: contador libre ascendente de 8 bits (0, 1, 2,…, 254, 255, 0, 1, 2…).

Ambas fuentes de datos son las que proporcionan la información a la **trama** empleada, la cual se **repite cíclicamente** y consta de **10 símbolos OFDM**. Así, la señal de entrada al receptor será el resultado de realizar sobre esta trama cíclica todas las operaciones correspondientes al emisor.

Para comprobar la validez del diseño realizado, se analiza y verifica en diferentes puntos del receptor, para ambas fuentes, la correspondencia de la señal con su equivalente en el emisor. Los puntos seleccionados se muestran en la figura 135:

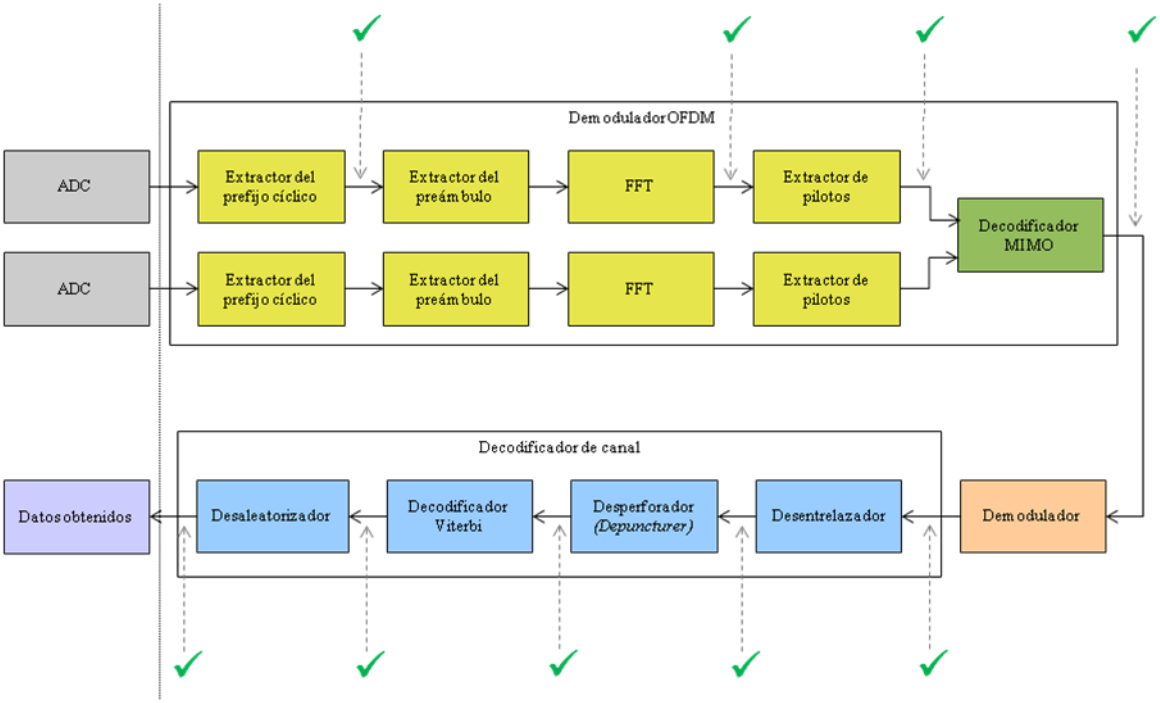

Figura 135: Puntos de validación del receptor diseñado para el estándar IEEE 802.16d-2004 utilizando MIMO 2x2, OFDM y BPSK

Así, en los siguientes apartados, se ofrece una muestra de dicha correspondencia para cada uno de los puntos seleccionados. Para ello se emplean capturas de imagen directamente tomadas de un bloque *Xilinx WaveScope* que contiene las señales adecuadas del emisor y receptor al principio de la trama y en formato hexadecimal para facilitar su comparación visual.

Previamente a la validación en cada uno de los puntos seleccionados, se adjunta en la figura 136 la propagación de la señal en cada uno de dichos puntos del receptor. Se puede observar cómo los datos finalmente obtenidos son válidos en el momento en el cual la señal "Errores Desaleatorizador" adquiere el valor 0, pues el Desaleatorizador funciona continuamente y la salida no será correcta hasta que llegue al mismo la señal propagada del resto de elementos.

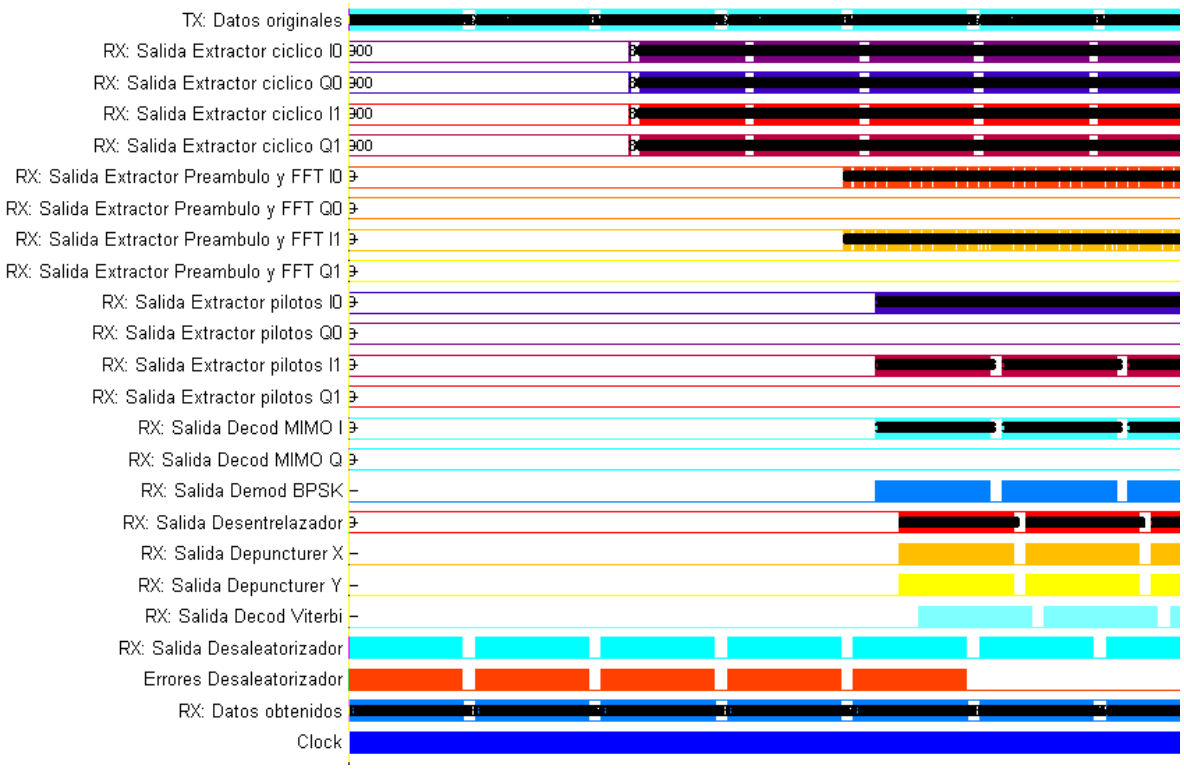

Figura 136: Validación del Receptor. Forma general de la señales

5.1 Extractor del prefijo cíclico

La señal, tras la acción del Extractor del prefijo cíclico, se encuentra formada por 4 líneas (componente I y Q para las dos antenas). Además, el valor de la componente Q no se encuentra a 0, aunque se podría esperar lo contrario debido a que se estableció a este valor fijo en el emisor (detalle en apartado "4.6 Demodulador"). Esto es causado por la acción de la IFFT en el emisor, la cual modifica el valor de Q, siendo diferente de 0.

Finalmente, comentar que en las capturas siguientes, las cuales se realizan al principio de la trama, se observan los mismos datos para ambas fuentes. Este hecho es totalmente lógico, pues el primer símbolo de la trama debe ser el preámbulo, el cual es el mismo en ambos casos.

A continuación se muestran las capturas realizadas al principio de la trama para ambas fuentes, en las cuales se observa la correspondencia entre las señales del emisor y receptor (figuras 137, 138, 139, 140, 141 y 142):

a) Fuente de 1 bit

Figura 137: Validación del Extractor del prefijo cíclico. Forma general de la señales de entrada y salida con fuente de 1 bit

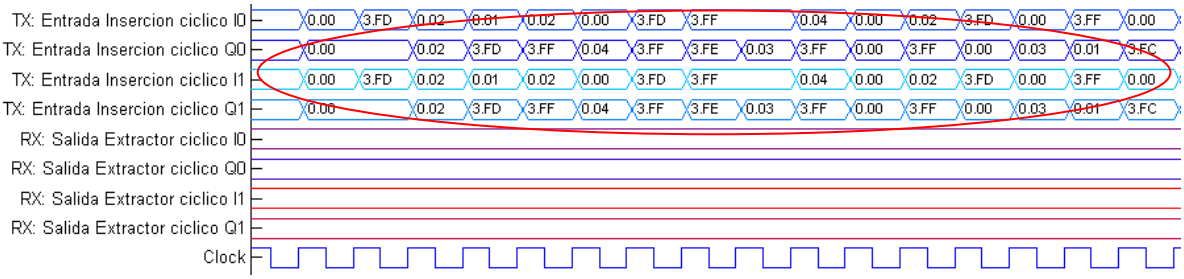

Figura 138: Validación del Extractor del prefijo cíclico. Muestra de los datos en el emisor con fuente de 1 bit

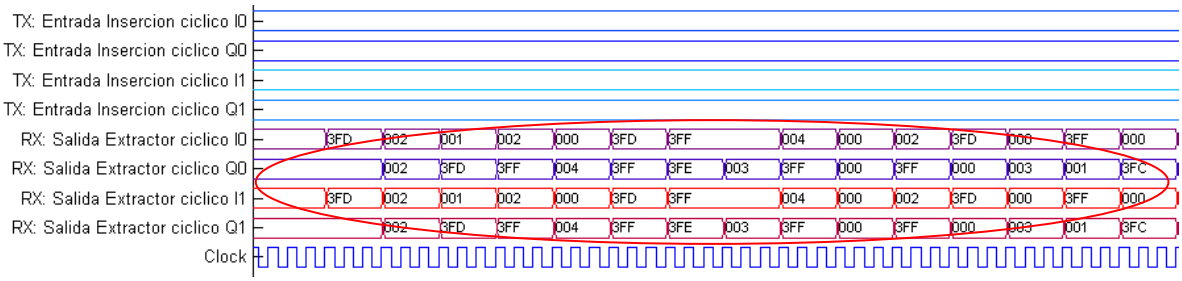

Figura 139: Validación del Extractor del prefijo cíclico. Muestra de los datos en el receptor con fuente de 1 bit

b) Fuente de 8 bits

Figura 140: Validación del Extractor del prefijo cíclico. Forma general de la señales de entrada y salida con fuente de 8 bits

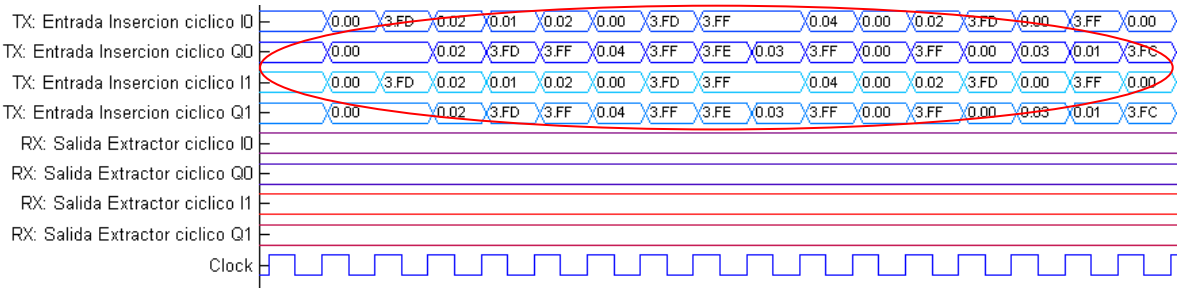

Figura 141: Validación del Extractor del prefijo cíclico. Muestra de los datos en el emisor con fuente de 8 bits

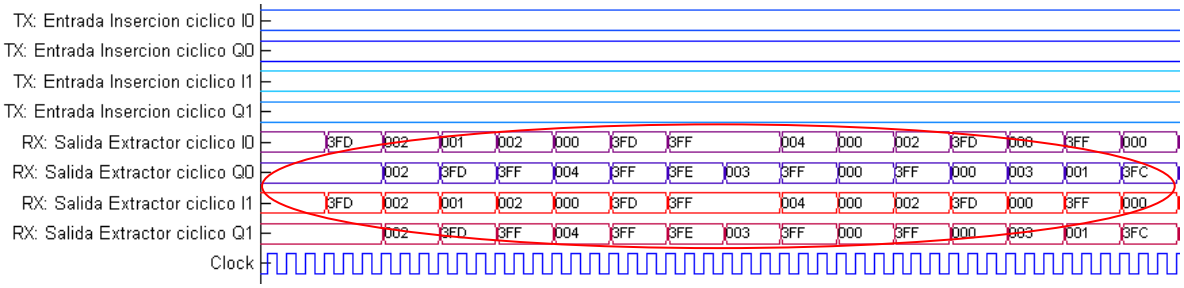

Figura 142: Validación del Extractor del prefijo cíclico. Muestra de los datos en el receptor con fuente de 8 bits

5.2 Extractor del preámbulo y FFT

La validación para el Extractor del preámbulo y la FFT se realiza de manera conjunta debido al diseño realizado en el emisor, en el cual las operaciones de inserción del preámbulo e IFFT se encuentran fuertemente relacionadas.

La señal, tras la acción de estas dos etapas, se encuentra formada por 4 líneas (componente I y Q para las dos antenas). Tal y como se esperaba, tras la operación de FFT, el valor de la componente Q se establece a 0. Además comentar el hecho de que al extraer el preámbulo (primer símbolo de la trama) no se observarán de aquí en adelante los mismos datos para ambas fuentes.

A continuación se muestran las capturas realizadas al principio de la trama para ambas fuentes, en las cuales se observa la correspondencia entre las señales del emisor y receptor (figuras 143, 144, 145, 146, 147 y 148):

a) Fuente de 1 bit

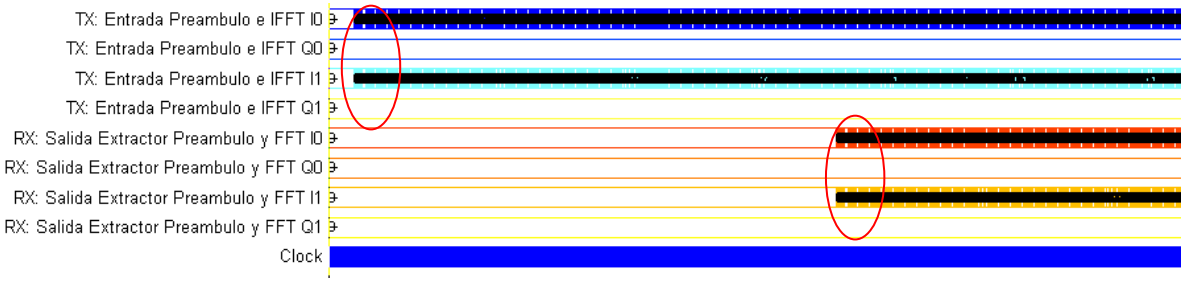

Figura 143: Validación del Extractor del preámbulo y FFT. Forma general de la señales de entrada y salida con fuente de 1 bit

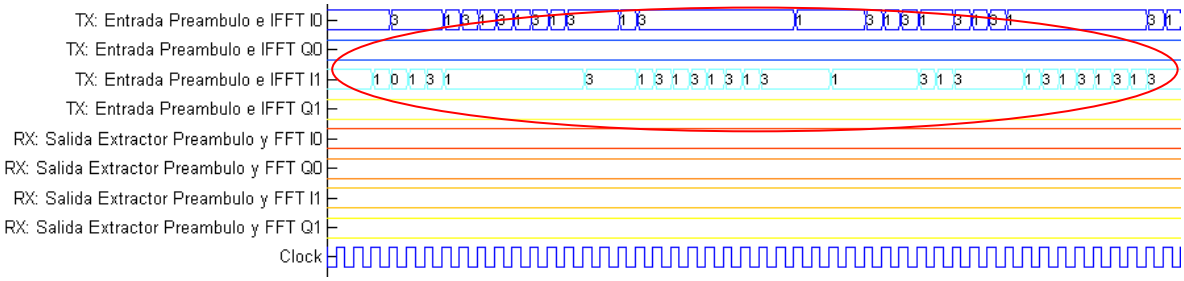

Figura 144: Validación del Extractor del preámbulo y FFT. Muestra de los datos en el emisor con fuente de 1 bit

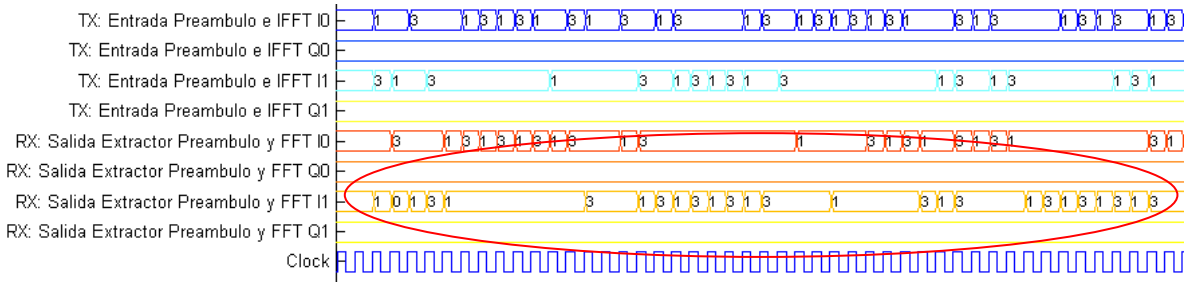

Figura 145: Validación del Extractor del preámbulo y FFT. Muestra de los datos en el receptor con fuente de 1 bit

b) Fuente de 8 bits

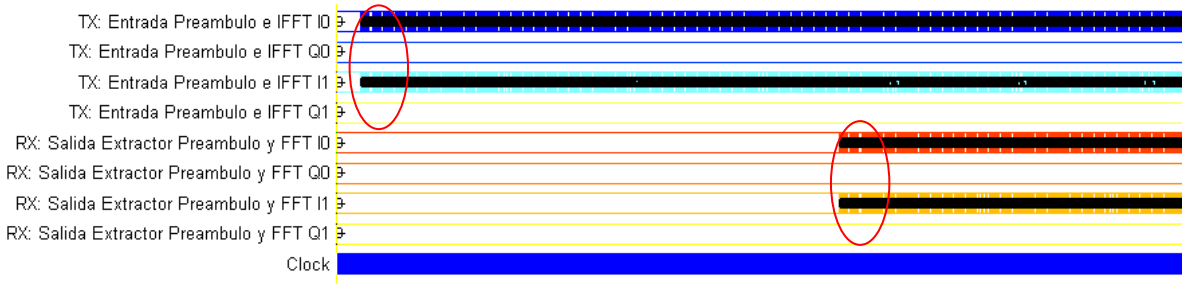

Figura 146: Validación del Extractor del preámbulo y FFT. Forma general de la señales de entrada y salida con fuente de 8 bits

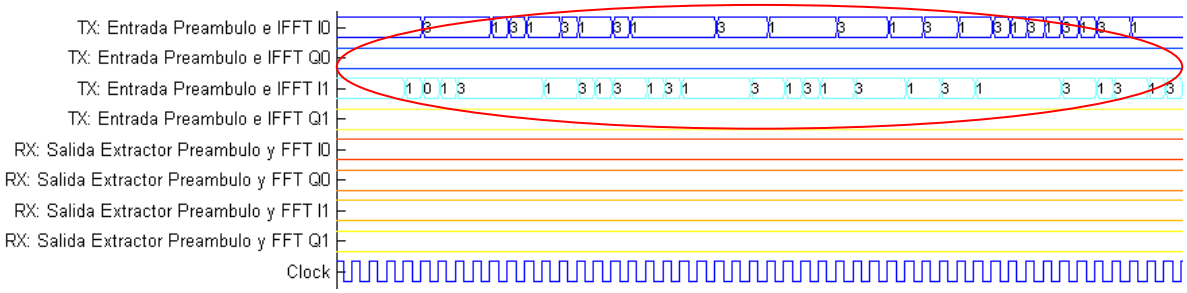

Figura 147: Validación del Extractor del preámbulo y FFT. Muestra de los datos en el emisor con fuente de 8 bits

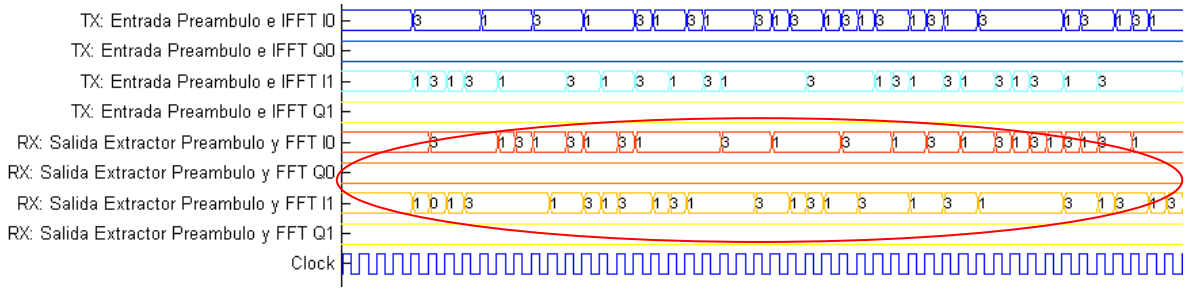

Figura 148: Validación del Extractor del preámbulo y FFT. Muestra de los datos en el receptor con fuente de 8 bits

5.3 Extractor de pilotos

La señal, tras la acción del Extractor de pilotos, se encuentra formada por 4 líneas (componente I y Q para las dos antenas). Además, lógicamente, el valor de la componente Q se mantiene a 0.

A continuación se muestran las capturas realizadas al principio de la trama para ambas fuentes, en las cuales se observa la correspondencia entre las señales del emisor y receptor (figuras 149, 150, 151, 152, 153 y 154):

a) Fuente de 1 bit

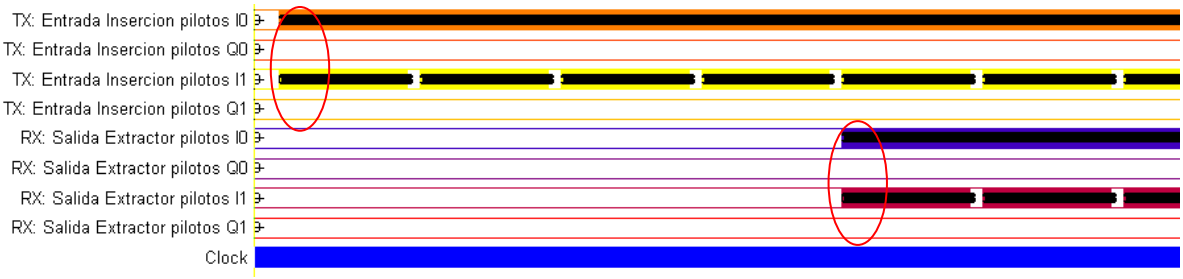

Figura 149: Validación del Extractor de pilotos. Forma general de la señales de entrada y salida con fuente de 1 bit

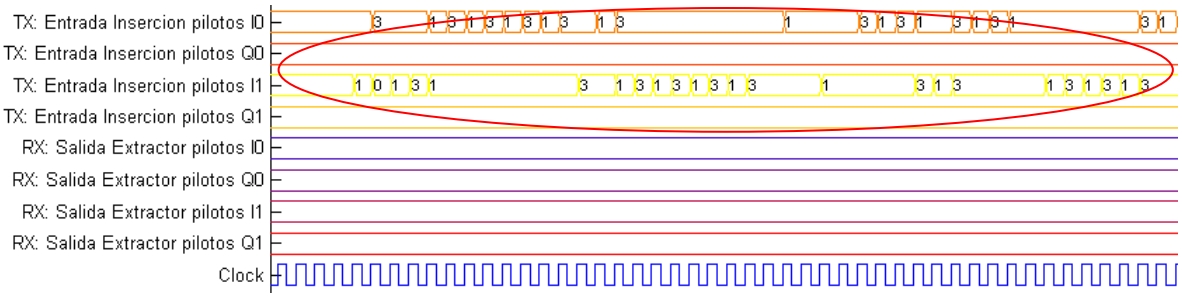

Figura 150: Validación del Extractor de pilotos. Muestra de los datos en el emisor con fuente de 1 bit

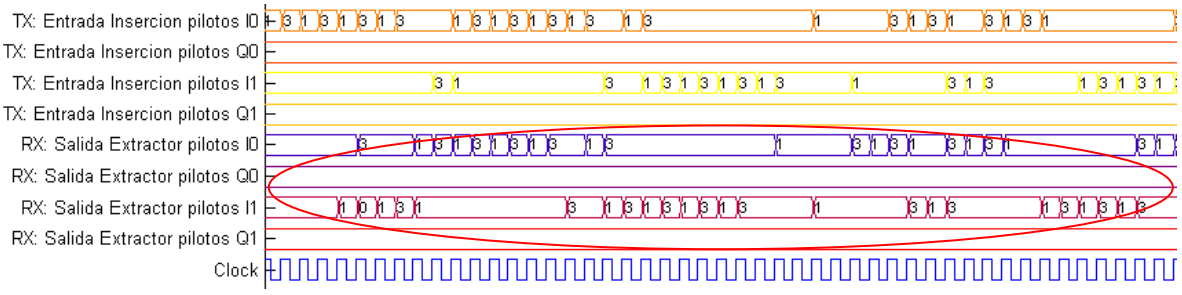

Figura 151: Validación del Extractor de pilotos. Muestra de los datos en el receptor con fuente de 1 bit

b) Fuente de 8 bits

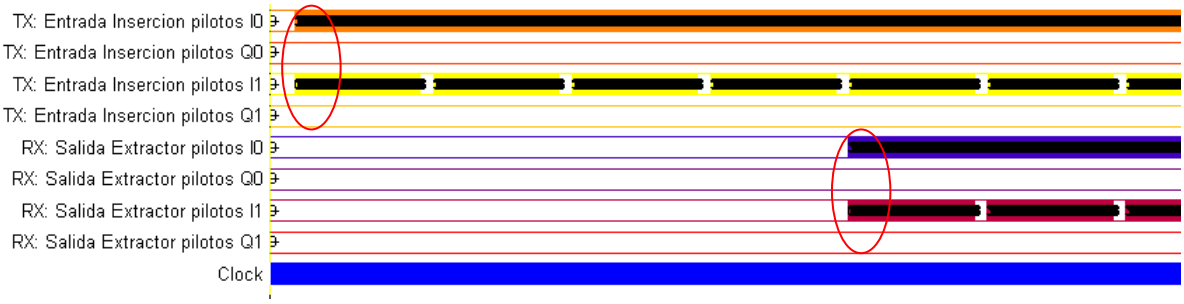

Figura 152: Validación del Extractor de pilotos. Forma general de la señales de entrada y salida con fuente de 8 bits

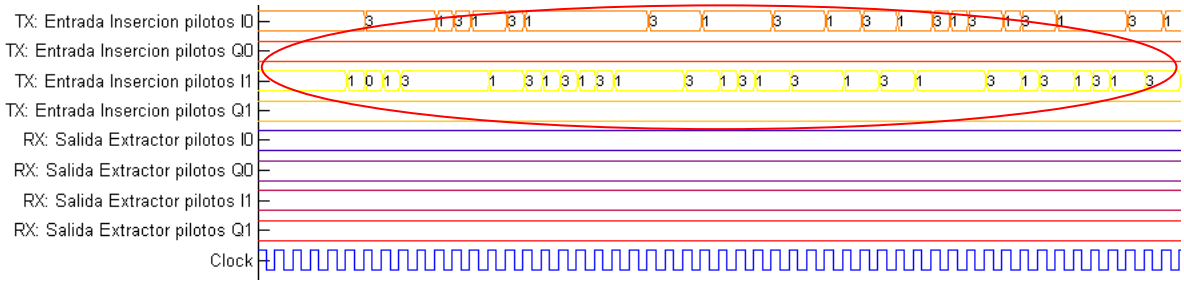

Figura 153: Validación del Extractor de pilotos. Muestra de los datos en el emisor con fuente de 8 bits

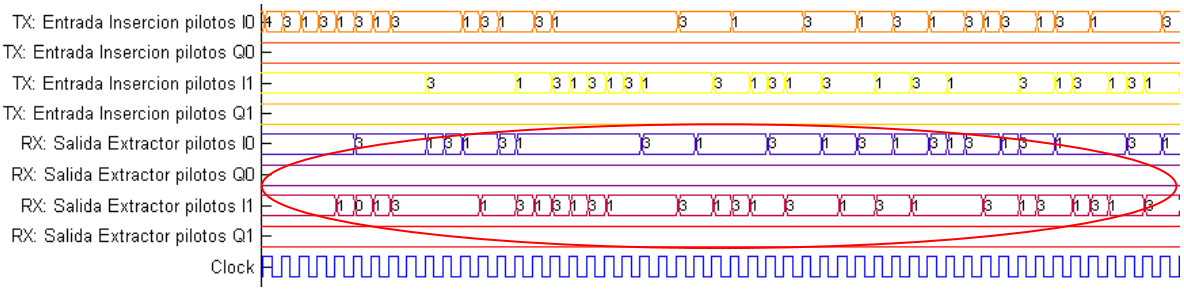

Figura 154: Validación del Extractor de pilotos. Muestra de los datos en el receptor con fuente de 8 bits

5.4 Decodificador MIMO

La señal, tras la acción del Decodificador MIMO, se encuentra formada por 2 líneas (componente I y Q), es decir, de aquí en adelante no se emplea la diversidad proporcionada por ambas antenas. Además, lógicamente, el valor de la componente Q se mantiene a 0.

A continuación se muestran las capturas realizadas al principio de la trama para ambas fuentes, en las cuales se observa la correspondencia entre las señales del emisor y receptor (figuras 155, 156, 157, 158, 159 y 160):

a) Fuente de 1 bit

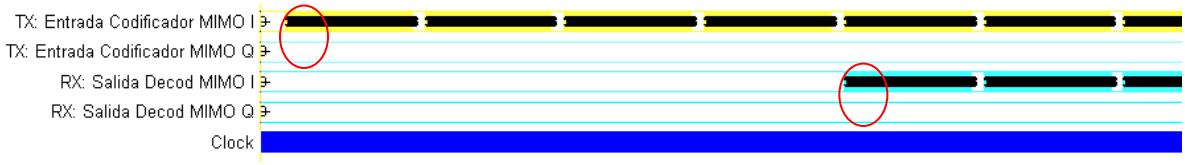

Figura 155: Validación del Decodificador MIMO. Forma general de la señales de entrada y salida con fuente de 1 bit

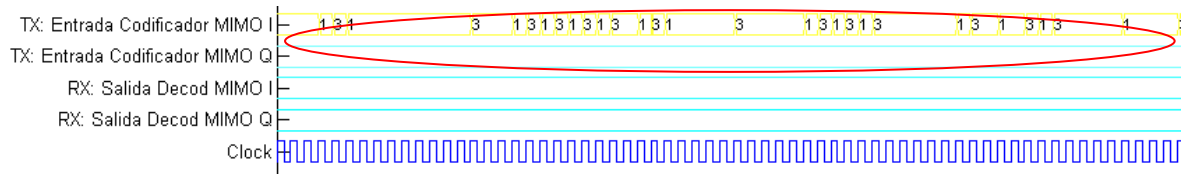

Figura 156: Validación del Decodificador MIMO. Muestra de los datos en el emisor con fuente de 1 bit

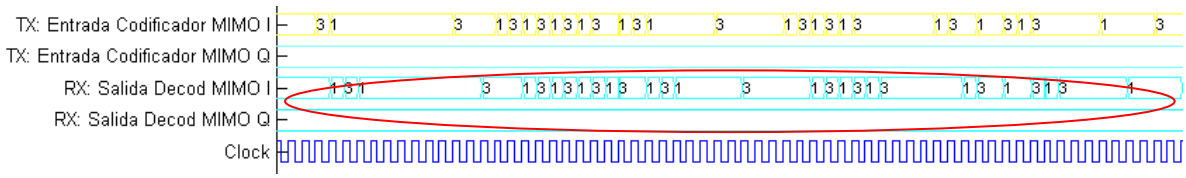

Figura 157: Validación del Decodificador MIMO. Muestra de los datos en el receptor con fuente de 1 bit

b) Fuente de 8 bits

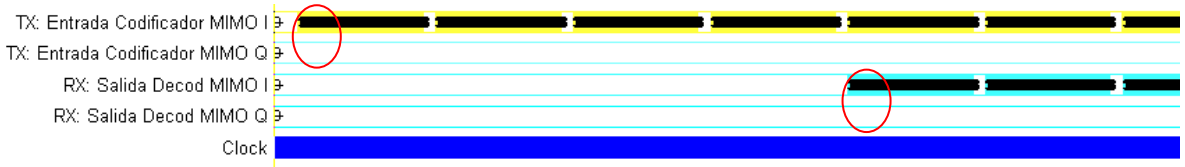

Figura 158: Validación del Decodificador MIMO. Forma general de la señales de entrada y salida con fuente de 8 bits

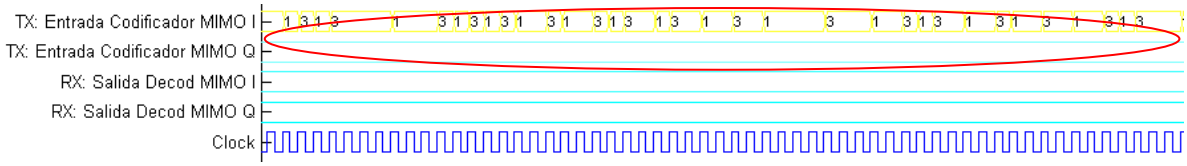

Figura 159: Validación del Decodificador MIMO. Muestra de los datos en el emisor con fuente de 8 bits

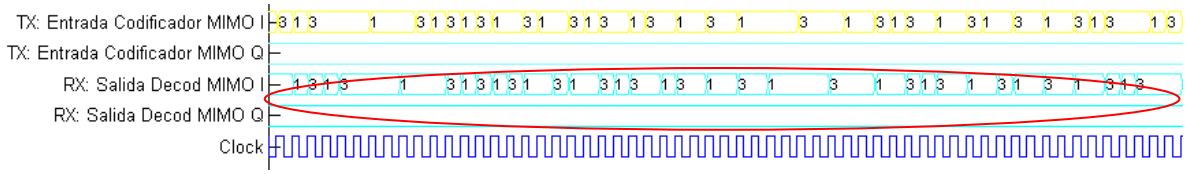

Figura 160: Validación del Decodificador MIMO. Muestra de los datos en el receptor con fuente de 8 bits

5.5 Demodulador

La señal, tras la acción del Demodulador (BPSK en este proyecto), se encuentra formada por 1 línea, es decir, de aquí en adelante no se emplean las componentes I y Q para representar la señal.

A continuación se muestran las capturas realizadas al principio de la trama para ambas fuentes, en las cuales se observa la correspondencia entre las señales del emisor y receptor (figuras 161, 162, 163, 164, 165 y 166):

a) Fuente de 1 bit

Figura 161: Validación del Demodulador. Forma general de la señales de entrada y salida con fuente de 1 bit

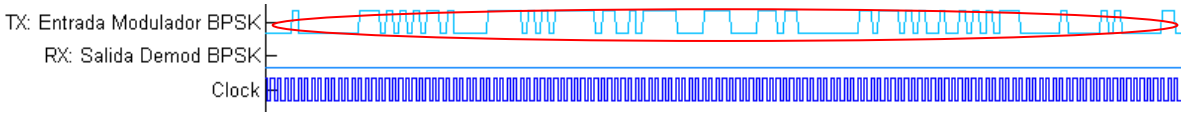

Figura 162: Validación del Demodulador. Muestra de los datos en el emisor con fuente de 1 bit

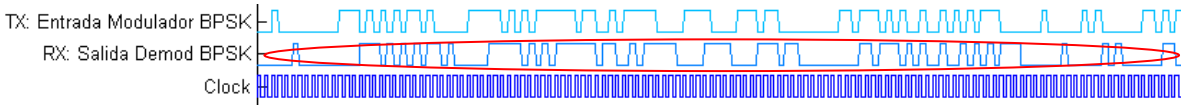

Figura 163: Validación del Demodulador. Muestra de los datos en el receptor con fuente de 1 bit

b) Fuente de 8 bits

Figura 164: Validación del Demodulador. Forma general de la señales de entrada y salida con fuente de 8 bits

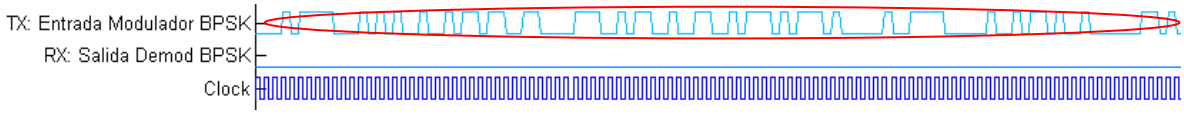

Figura 165: Validación del Demodulador. Muestra de los datos en el emisor con fuente de 8 bits

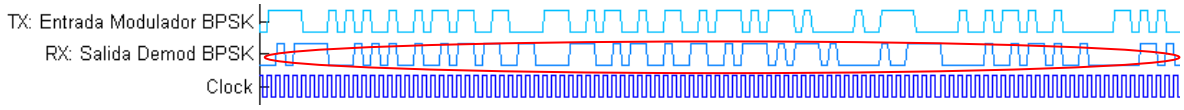

Figura 166: Validación del Demodulador. Muestra de los datos en el receptor con fuente de 8 bits

5.6 Desentrelazador

La señal, tras la acción del Desentrelazador, se encuentra formada por 1 línea puesto que únicamente realiza un mezclado de los bits.

A continuación se muestran las capturas realizadas al principio de la trama para ambas fuentes, en las cuales se observa la correspondencia entre las señales del emisor y receptor (figuras 167, 168, 169, 170, 171 y 172):

a) Fuente de 1 bit

Figura 167: Validación del Desentrelazador. Forma general de la señales de entrada y salida con fuente de 1 bit

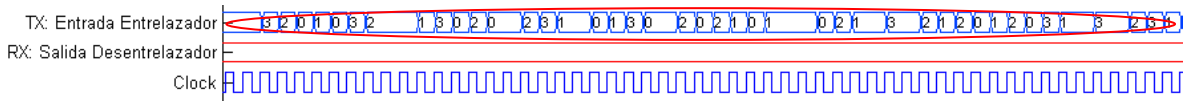

Figura 168: Validación del Desentrelazador. Muestra de los datos en el emisor con fuente de 1 bit

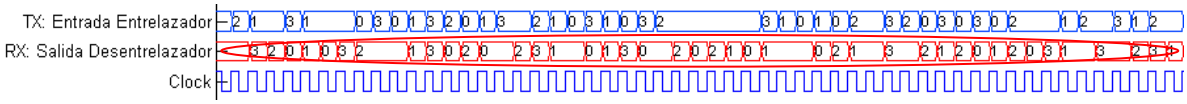

Figura 169: Validación del Desentrelazador. Muestra de los datos en el receptor con fuente de 1 bit

b) Fuente de 8 bits

Figura 170: Validación del Desentrelazador. Forma general de la señales de entrada y salida con fuente de 8 bits

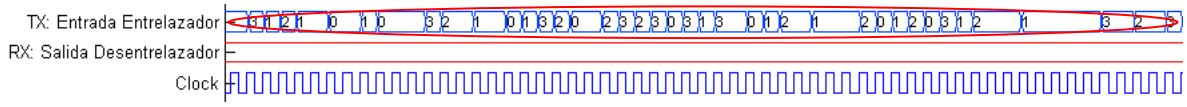

Figura 171: Validación del Desentrelazador. Muestra de los datos en el emisor con fuente de 8 bits

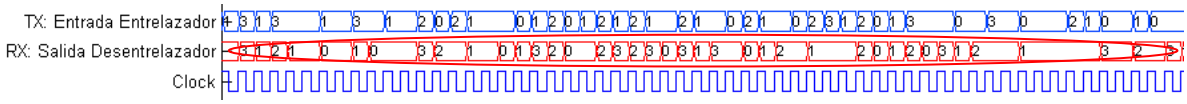

Figura 172: Validación del Desentrelazador. Muestra de los datos en el receptor con fuente de 8 bits

5.7 Desperforador (Depuncturer)

La señal, tras la acción del Desperforador (*Depuncturer*), se encuentra formada por 2 líneas (componente X e Y) con el objetivo de ser introducidas en el Decodificador Viterbi.

A continuación se muestran las capturas realizadas al principio de la trama para ambas fuentes, en las cuales se observa la correspondencia entre las señales del emisor y receptor (figuras 173, 174, 175, 176, 177 y 178):

a) Fuente de 1 bit

Figura 173: Validación del Desperforador (Depuncturer). Forma general de la señales de entrada y salida con fuente de 1 bit

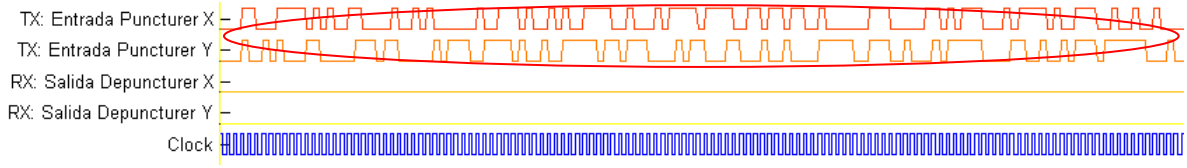

Figura 174: Validación del Desperforador (Depuncturer). Muestra de los datos en el emisor con fuente de 1 bit

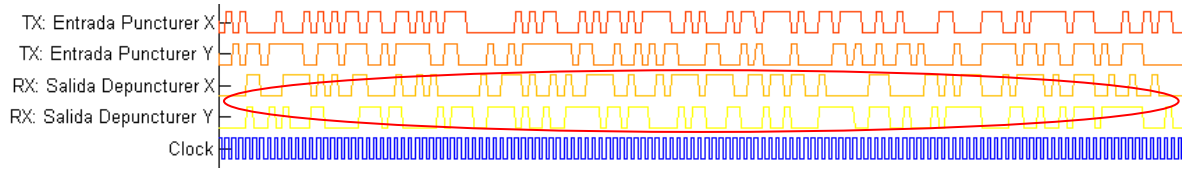

Figura 175: Validación del Desperforador (Depuncturer). Muestra de los datos en el receptor con fuente de 1 bit

b) Fuente de 8 bits

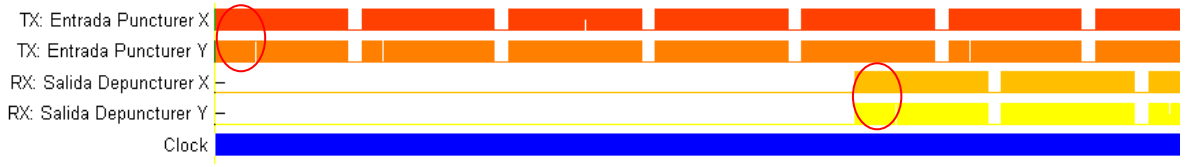

Figura 176: Validación del Desperforador (Depuncturer). Forma general de la señales de entrada y salida con fuente de 8 bits

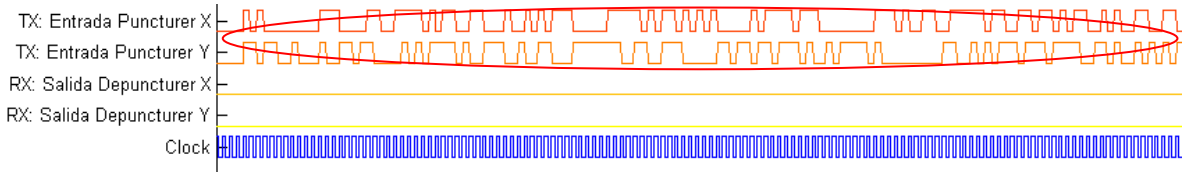

Figura 177: Validación del Desperforador (Depuncturer). Muestra de los datos en el emisor con fuente de 8 bits

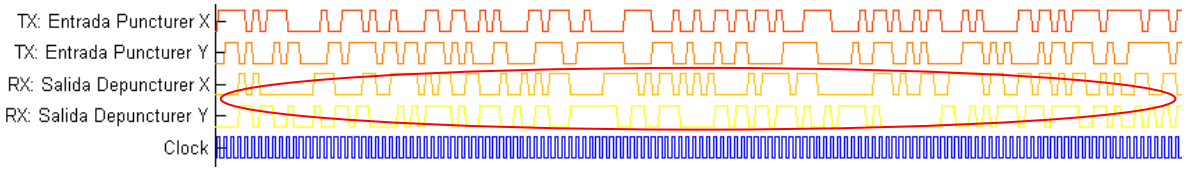

Figura 178: Validación del Desperforador (Depuncturer). Muestra de los datos en el receptor con fuente de 8 bits

5.8 Decodificador Viterbi

La señal, tras la acción del Decodificador Viterbi se encuentra formada por 1 línea con el objetivo de ser introducida en el último elemento del receptor: el Desaleatorizador.

A continuación se muestran las capturas realizadas al principio de la trama para ambas fuentes, en las cuales se observa la correspondencia entre las señales del emisor y receptor (figuras 179, 180, 181, 182, 183 y 184):

a) Fuente de 1 bit

Figura 179: Validación del Decodificador Viterbi. Forma general de la señales de entrada y salida con fuente de 1 bit

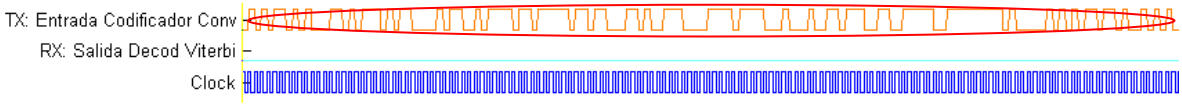

Figura 180: Validación del Decodificador Viterbi. Muestra de los datos en el emisor con fuente de 1 bit

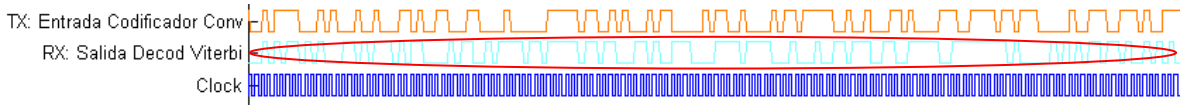

Figura 181: Validación del Decodificador Viterbi. Muestra de los datos en el receptor con fuente de 1 bit

b) Fuente de 8 bits

Figura 182: Validación del Decodificador Viterbi. Forma general de la señales de entrada y salida con fuente de 8 bits

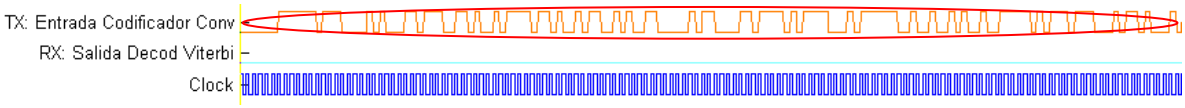

Figura 183: Validación del Decodificador Viterbi. Muestra de los datos en el emisor con fuente de 8 bits

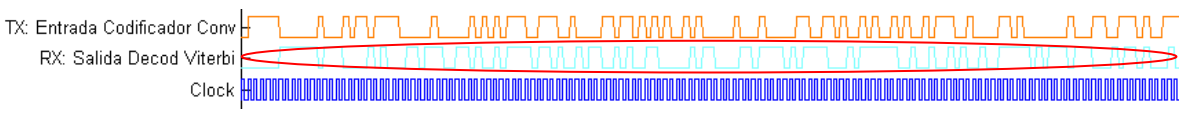

Figura 184: Validación del Decodificador Viterbi. Muestra de los datos en el receptor con fuente de 8 bits

5.9 Desaleatorizador

La señal, tras la acción del Desaleatorizador, se encuentra formada por 1 línea puesto que únicamente realiza operaciones a nivel de bit. Además, el bloque Desaleatorizador funciona continuamente, por lo que hasta que no se reciba la señal propagada del resto de elementos, no

proporcionará los datos correctos a la salida. Para indicar este momento se emplea la señal "Errores Desaleatorizador" mediante su establecimiento al valor 0.

Finalmente, comentar que dichos bits obtenidos a la salida deberán agruparse en un número adecuado según el tipo de fuente para recuperar los datos originales.

A continuación se muestran las capturas realizadas al principio de la trama para ambas fuentes, en las cuales se observa la correspondencia entre las señales del emisor y receptor (figuras 185, 186, 187, 188, 189 y 190):

a) Fuente de 1 bit

Figura 185: Validación del Desaleatorizador. Forma general de la señales de entrada y salida con fuente de 1 bit

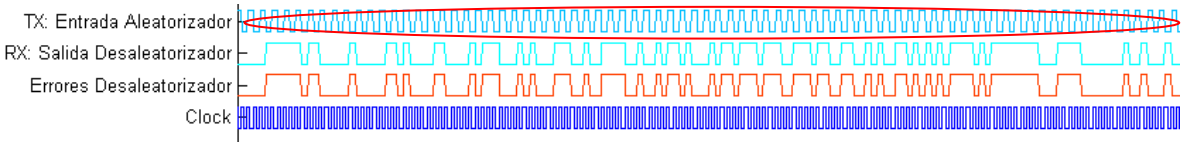

Figura 186: Validación del Desaleatorizador. Muestra de los datos en el emisor con fuente de 1 bit

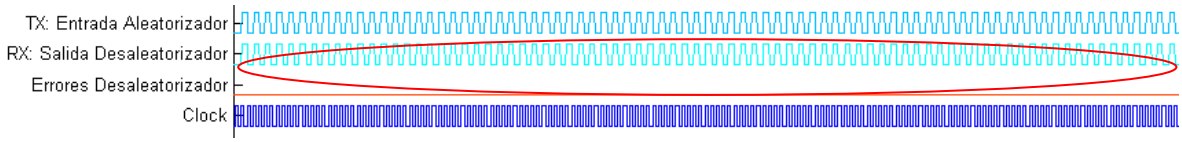

Figura 187: Validación del Desaleatorizador. Muestra de los datos en el receptor con fuente de 1 bit

b) Fuente de 8 bits

Figura 188: Validación del Desaleatorizador. Forma general de la señales de entrada y salida con fuente de 8 bits

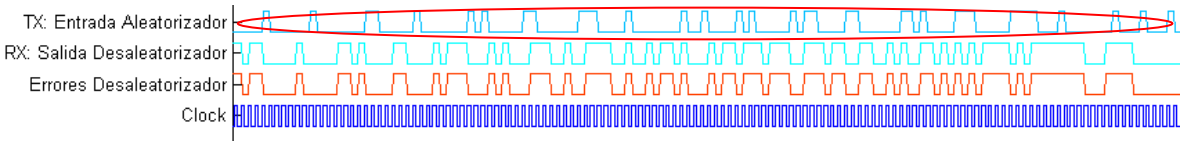

Figura 189: Validación del Desaleatorizador. Muestra de los datos en el emisor con fuente de 8 bits

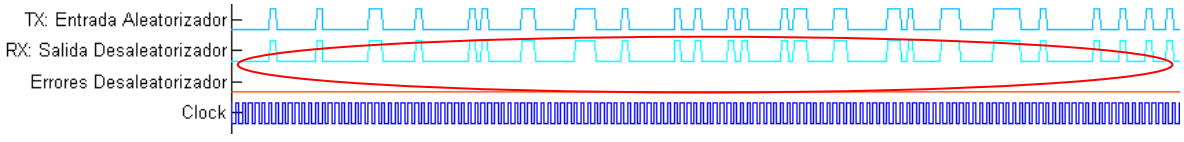

Figura 190: Validación del Desaleatorizador. Muestra de los datos en el receptor con fuente de 8 bits

5.10 Datos obtenidos

Los bits que conforman la señal tras la acción del Desaleatorizador deben ser agrupados en un número adecuado (según el tipo de fuente) para recuperar los datos introducidos en el emisor. En este caso, puesto que una de las fuentes es de 1 bit, sólo será necesario agruparlos para la fuente de 8 bits mediante un bloque *Xilinx Serial to Parallel*.

De la misma manera que en el apartado anterior, la señal “Errores Desaleatorizador” indicará el momento a partir del cual los datos obtenidos a la salida son correctos, estableciéndose al valor 0.

A continuación se muestran las capturas realizadas al principio de la trama para ambas fuentes, en las cuales se observa la correspondencia entre las señales del emisor y receptor (figuras 191, 192, 193, 194, 195 y 196):

a) Fuente de 1 bit

Figura 191: Validación de los datos obtenidos. Forma general de la señales de entrada y salida con fuente de 1 bit

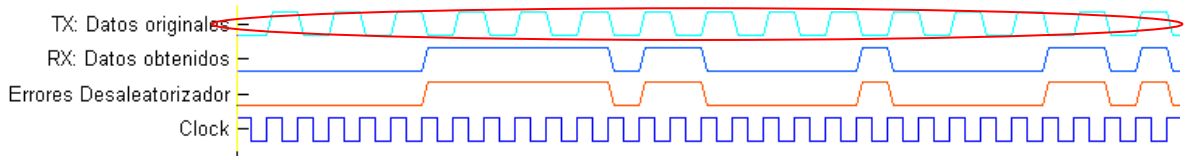

Figura 192: Validación de los datos obtenidos. Muestra de los datos en el emisor con fuente de 1 bit

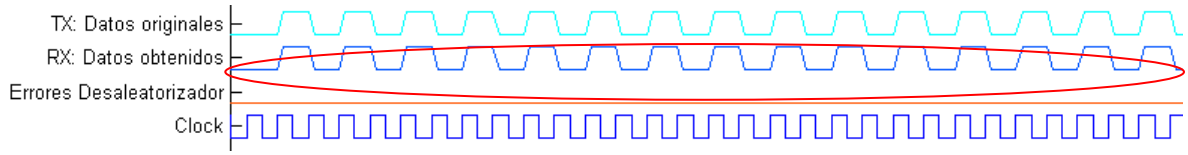

Figura 193: Validación de los datos obtenidos. Muestra de los datos en el receptor con fuente de 1 bit

b) Fuente de 8 bits

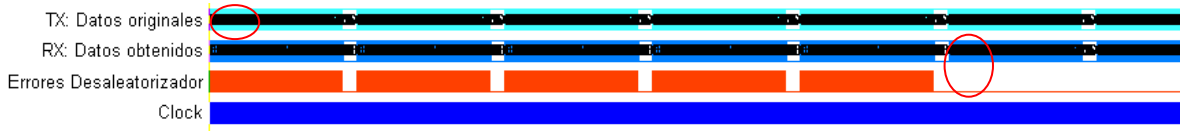

Figura 194: Validación de los datos obtenidos. Forma general de la señales de entrada y salida con fuente de 8 bits

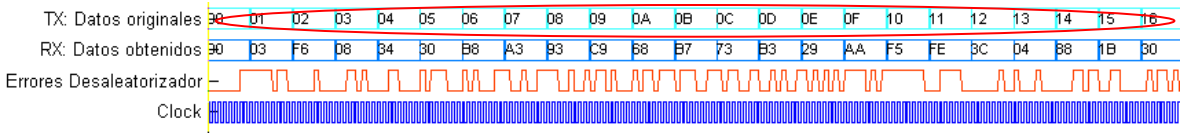

Figura 195: Validación de los datos obtenidos. Muestra de los datos en el emisor con fuente de 8 bits

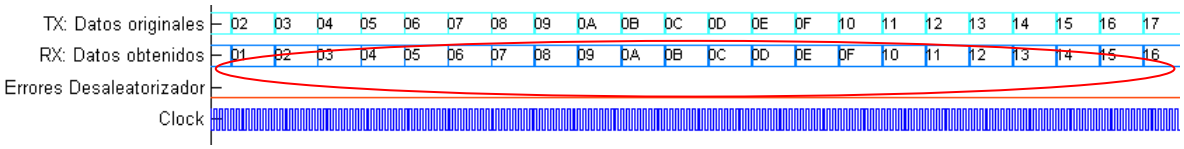

Figura 196: Validación de los datos obtenidos. Muestra de los datos en el receptor con fuente de 8 bits

6. Conclusiones y futuros trabajos

La realización de este proyecto y la consecución de sus objetivos aportan una serie de conclusiones, **tanto positivas como referidas a aspectos mejorables detectados**. Además, se han identificado posibles futuras líneas de trabajo asociadas a diferentes características de dicho proyecto.

Los siguientes apartados ilustran de manera detallada estas conclusiones y futuras líneas de trabajo.

6.1 Conclusiones

La consecución de los objetivos planteados al inicio de este proyecto implica una serie de conclusiones al respecto del diseño de este receptor para el estándar IEEE 802.16d-2004 utilizando MIMO 2x2, OFDM en la capa física y BPSK para la modulación de datos. Dichas conclusiones son tanto de carácter positivo como de aspectos mejorables, contando ambas con la misma importancia y sirviendo como base para la realización de diferentes trabajos en el futuro.

Así, las principales conclusiones positivas obtenidas son las siguientes:

- El diseño, simulación y validación de un **prototipo de receptor para el estándar IEEE 802.16d-2004 es posible mediante la plataforma de desarrollo para FPGA** basada en Simulink® *VHS-ADC Virtex-4* de Lyrtech junto a la herramienta *Xilinx System Generator for DSP.* Dicho modelo **se podría implementar realmente en la FPGA** sin necesidad de optimización hardware debido a los suficientes recursos de la *VHS-ADC Virtex-4*.

- Las **plataformas de desarrollo para FPGA basadas en Simulink®** cuentan con un **gran potencial y versatilidad para el desarrollo de sistemas de comunicaciones** debido a su alto nivel de abstracción, variedad de recursos, posibilidad de simulación previa y compilación automática del modelo en la FPGA (generando el código correspondiente).

- La combinación de las técnicas **MIMO** junto a la modulación **OFDM** constituyen el **futuro de los sistemas de comunicaciones inalámbricos de alta** capacidad debido a su robustez frente al multitrayecto (sin visión directa), lo cual produce un incremento en radio de la celda y la capacidad.

- La familia de estándares de acceso inalámbrico IEEE 802.16, especialmente el estándar **IEEE 802.16e-2005** (WiMAX Móvil) cuenta con importantes posibilidades de convertirse

en una **referencia en las comunicaciones inalámbricas de alta capacidad**. Este hecho se desprende tras el estudio y análisis, tanto de su definición, características y aplicaciones, como de su situación en el actual y futuro panorama de las comunicaciones frente a otros estándares inalámbricos.

Respecto a los aspectos mejorables detectados durante la realización, dichos son referidos, principalmente, a la plataforma de desarrollo para FPGA sobre Simulink®. Así, destacan los siguientes:

- El **proceso de aprendizaje inicial** de la plataforma de desarrollo es **costoso** debido a que, en muchos casos, la utilización y respuesta de los bloques no es todo lo intuitiva que se esperaría. Se hace necesario un estudio experimental de cada bloque previamente a su empleo en un sistema para asegurar una correcta interacción con el resto de bloques. Si bien, una vez se posee este conocimiento, el tiempo de desarrollo se reduce considerablemente respecto a otras plataformas no basadas en Simulink®.

- La **variedad de bloques y flexibilidad de la plataforma no siempre es suficiente** para implementar operaciones o lógicas de carácter avanzado de una manera sencilla. Para solucionar esta carencia, este tipo de plataformas de desarrollo incrementan y mejoran de manera constante su catálogo de bloques compatibles con Simulink®.

- **No todos los bloques que conforman este modelo, una vez sintetizado e introducido en la FPGA, se comportan de la misma manera que durante su simulación**. Así, se hace necesario un estudio de las señales sobre la propia FPGA en el caso de implementar de manera real el modelo.

6.2 Trabajos futuros

Los objetivos cumplidos y conclusiones obtenidas tras la realización de este proyecto constituyen un importante punto de partida para la realización de diferentes actividades futuras. Estos trabajos deberían seguir principalmente dos líneas:

- **Mejora** de las prestaciones del **receptor** diseñado para el estándar IEEE 802.16d-2004 (WiMAX Fijo) e implementación real sobre una FPGA.

- Aplicación del conocimiento adquirido sobre la plataforma de desarrollo para FPGA basada en Simulink® (*VHS-ADC Virtex-4* de Lyrtech [2] junto a la herramienta *Xilinx System Generator for DSP* [3]) para el **diseño de sistemas de comunicaciones basados en otros estándares**.

Una de las principales mejoras del receptor modelado consistiría en diseñar toda la lógica necesaria para permitir emplear, además de BPSK, cualquiera de las constelaciones contempladas por el estándar IEEE 802.16d-2004 para la modulación de datos: QPSK, 16-QAM y 64-QAM. Igualmente de importante sería trabajar en agregar la lógica asociada a la existencia de un canal de comunicaciones con el objetivo realizar la sincronización, estimarlo y compensar sus efectos. Además, el sistema de sincronismo de los bloques empleados podría ser mejorado mediante señales de control.

Pese a las interesantes líneas de actividad anteriores relacionadas con el receptor diseñado, quizás la más importante sería la síntesis, ejecución y validación de dicho receptor sobre una FPGA. Esto es debido a que se ha comprobado que no todos los bloques que conforman este modelo, una vez sintetizado e introducido en la FPGA, se comportan de la misma manera que durante su simulación. Así, se hace necesario ejecutar el modelo y capturar directamente datos de la FPGA, analizándolos para identificar la causa del comportamiento anómalo.

Además de las anteriores líneas de trabajo, resulta muy interesante aplicar el conocimiento adquirido sobre esta plataforma de desarrollo con el objetivo de diseñar sistemas de comunicaciones basados en otros estándares inalámbricos: principalmente IEEE 802.20 (Mobile-Fi) y las nuevas especificaciones de IEEE 802.11 (Wi-Fi).

7. Presupuesto

La ejecución de este proyecto, cuya duración ha sido de 10 meses, ha exigido una serie de recursos, tanto de carácter material como personal.

Así, el objetivo de los siguientes apartados es desglosar el coste imputable para cada uno de los elementos que conforman dichos recursos materiales y personales.

7.1 Coste material

La realización de este proyecto ha contado con los siguientes recursos materiales:

- Dos ordenadores portátiles personales valorados en 1.800 € y 800 € que incluyen el sistema operativo Windows XP, los cuales han sido y serán reutilizados. Por tanto, el coste imputable al proyecto, en base a su utilización y tiempo de vida estimado en 3 años, será de unos 350 € para el primero y 125 € para el segundo.

- Plataforma de desarrollo sobre FPGA basada en Simulink® denominada *VHS-ADC Virtex-4* de Lyrtech [2] valorada en 18.000 €. Dicha plataforma viene integrada en un ordenador de sobremesa con el sistema operativo Windows XP incluido. El coste imputable al proyecto será de unos 3.000 € debido a su reutilización en el departamento para la realización de otros proyectos.

- Herramienta software matemática MATLAB® (incluye Simulink®) [13] cuya licencia ronda los 3.000 €. Dicha licencia es compartida con el resto del departamento y seguirá siendo empleada. Por tanto, el coste imputable es de unos 100 € por equipo, es decir, 300 € en total.

- Herramienta software *Xilinx System Generator for DSP* [3] cuya licencia ronda los 6.000 €. Dicha licencia es compartida con el resto del departamento y seguirá siendo empleada. Por tanto, el coste imputable es de unos 200 € por equipo, es decir, 600 € en total.

- Conexión a Internet con un coste de unos 50 € / mes. Dichas conexión se ha empleado durante los 10 meses que ha durado la realización del proyecto, suponiendo un uso del 5% del total de su capacidad para tal fin. Por tanto, el coste imputable al proyecto es de unos 10 €.

- Impresión de documentos con un coste aproximado de 100 €.

Así, el **coste material** de la realización del proyecto es el mostrado en la tabla 3:

Concepto	Coste imputable (€)
Ordenadores portátiles con Windows XP	475
VHS-ADC Virtex-4 de Lyrtech con Windows XP	3.000
MATLAB® y Simulink®	300
Xilinx System Generator for DSP	600
Conexión Internet	10
Impresión documentos	100
TOTAL (Coste material)	**4.485**

Tabla 3: Coste material de la realización del proyecto

7.2 Coste personal

La realización de este proyecto ha contado con los siguientes recursos personales:

- Desarrollador del proyecto con un coste laboral de 70 € / hora (según el Colegio de Ingenieros de Telecomunicación [14]). Por tanto, los 10 meses de ejecución del proyecto con una dedicación media de 4 horas / día suponen un coste de 61.600 €.

- Director del proyecto (Víctor P. Gil Jiménez) con un coste laboral de 70 € / hora (según el Colegio de Ingenieros de Telecomunicación [14]) y una dedicación de unas 100 horas. Por tanto, el coste imputable es de 7.000 €.

Así, el **coste personal** de la realización del proyecto es el mostrado en la tabla 4:

Concepto	Coste imputable (€)
Desarrollador proyecto	61.600
Director proyecto	7.000
TOTAL (Coste personal)	**68.600**

Tabla 4: Coste personal de la realización del proyecto

7.3 Coste total

La realización de este proyecto tiene un **coste total** que debe incluir los costes materiales y personales, tal y como se muestra en la tabla 5:

Concepto	Coste imputable (€)
Coste material	4.485
Coste personal	68.600
TOTAL	**73.085**

Tabla 5: Coste total de la realización del proyecto

8. Bibliografía

[1] "Air Interface for Fixed Broadband Wireless Access Systems", IEEE STD 802.16-2004, octubre 2004.

[2] Sitio Web de Lyrtech, http://www.lyrtech.com/. Último acceso en mayo de 2009.

[3] Sitio Web de Xilinx, http://www.xilinx.com/. Último acceso en mayo de 2009.

[4] "Estudio práctico sobre el prototipado de un sistema MIMO 2x2 para WiMAX", David Díaz Martín y Roberto Prieto Alonso, Estudio Tecnológico de la Universidad Carlos III de Madrid, diciembre 2007.

[5] "Air Interface for Fixed and Mobile Broadband Wireless Access Systems", IEEE STD 802.16e-2005, febrero 2006.

[6] Sitio Web del WiMAX Forum, http://www.wimaxforum.org/. Último acceso en mayo de 2009.

[7] Sitio Web de Sociedad de la Información de la Fundación Telefónica, http://sociedadinformacion.fundacion.telefonica.com. Último acceso en mayo de 2009.

[8] Sitio Web del Centro de Investigación e Innovación en Telecomunicación de Méjico (CINIT), http://www.cinit.org.mx/. Último acceso en mayo de 2009.

[9] Sitio Web del Observatorio Tecnológico del Ministerio de Educación, http://observatorio.cnice.mec.es/. Último acceso en mayo de 2009.

[10] Sitio Web de la Universidad Galileo de Guatemala, http://www.galileo.edu/. Último acceso en mayo de 2009.

[11] Sitio Web del Departamento de Ingeniería Telemática de la Universidad Carlos III de Madrid, http://www.it.uc3m.es/. Último acceso en mayo de 2009.

[12] Sitio Web de Intel sobre WiMAX, http://www.intel.com/technology/wimax/. Último acceso en mayo de 2009. Último acceso en mayo de 2009.

[13] Sitio Web de The MathWorks, http://www.mathworks.com/. Último acceso en mayo de 2009.

[14] Sitio Web del Colegio Oficial de Ingenieros de Telecomunicación, http://www.coit.es. Último acceso en noviembre de 2009.

9. Anexo A: Funciones MATLAB® implementadas

El objetivo de este apartado es adjuntar el código en MATLAB® desarrollado para los bloques ***Xilinx MCode*** empleados en el **receptor**. Se debe recordar que dicho bloque no soporta el lenguaje MATLAB® completo, sino un subconjunto del mismo, contando con importantes restricciones. El apartado "3.3.8 Bloque Xilinx MCode" contiene una descripción detallada del funcionamiento de este bloque.

9.1 Randomizer_1trama_v2.m

El código de la función MATLAB® es la implementada en el **Desaleatorizador** es el siguiente:

```
function[data_out]=Randomizer_1trama_v2(data_in , bits_datos_trama)

%Vector de reinicio
persistent r,r=xl_state([1, 0, 0, 1, 0, 1, 0, 1 ,0 ,0 ,0 ,0 ,0,
0],{xlUnsigned,1,0}, 15);

%Número de bits transmitidos en la trama actual y tramas hasta el momento
persistent bits_trx, bits_trx = xl_state(0, {xlUnsigned,12,0});

%Contador
persistent contador, contador = xl_state(0, {xlUnsigned,12,0});

if contador<1055

        if bits_trx<bits_datos_trama
                %Dentro de una trama
                %Se construye la salida
                bit_rand=xfix({xlUnsigned,1,0},xl_xor(r(13),r(14)));
                data_out=xfix({xlUnsigned,1,0},xl_xor(bit_rand,data_in));
                r.push_front_pop_back(bit_rand);
                bits_trx=xfix({xlUnsigned,12,0},bits_trx+1);
                contador=xfix({xlUnsigned,12,0},contador+1);
        else
            %Se acabo la trama única que tiene en cuenta este código
            data_out=0;
            contador=xfix({xlUnsigned,12,0},contador+1);
        end

else %Se reinicia el sistema
    contador=0;
    bits_trx=0;
    data_out=0;
    r(0)=xfix({xlUnsigned,1,0},1);
    r(1)=xfix({xlUnsigned,1,0},0);
    r(2)=xfix({xlUnsigned,1,0},0);
    r(3)=xfix({xlUnsigned,1,0},1);
    r(4)=xfix({xlUnsigned,1,0},0);
    r(5)=xfix({xlUnsigned,1,0},1);
    r(6)=xfix({xlUnsigned,1,0},0);
    r(7)=xfix({xlUnsigned,1,0},1);
    r(8)=xfix({xlUnsigned,1,0},0);
    r(9)=xfix({xlUnsigned,1,0},0);
    r(10)=xfix({xlUnsigned,1,0},0);
    r(11)=xfix({xlUnsigned,1,0},0);
    r(12)=xfix({xlUnsigned,1,0},0);
    r(13)=xfix({xlUnsigned,1,0},0);
    r(14)=xfix({xlUnsigned,1,0},0);
end
```

9.2 Extraccionpilotos2bits.m

El código de la función MATLAB® es la implementada en el **Extractor de pilotos** (para una antena) es el siguiente:

```
function [OutputRE, OutputIM, PilotosRE, PilotosIM, zero_DC_RE, zero_DC_IM,
ceropaddingMSRE, ceropaddingMSIM, ceropaddinglsRE, ceropaddinglsIM]=
pilotos(inputRE, inputIM)

%Extracción de bandas de guarda, DC y pilotos (componente real)
ceropaddingMSRE=xl_slice(inputRE,511,456)
portadora10RE=xl_slice(inputRE,455,432);
piloto8RE=xl_slice(inputRE ,431, 430);
portadora9RE=xl_slice(inputRE,429,382);
piloto7RE=xl_slice(inputRE ,381, 380);
portadora8RE=xl_slice(inputRE,379,332 );
piloto6RE=xl_slice(inputRE,331 , 330);
portadora7RE=xl_slice(inputRE,329 ,282 );
piloto5RE=xl_slice(inputRE ,281, 280);
portadora6RE=xl_slice(inputRE,279,256);
zero_DC_RE=xl_slice(inputRE,255,254)
portadora5RE=xl_slice(inputRE,253,230);
piloto4RE=xl_slice(inputRE,229 ,228);
portadora4RE=xl_slice(inputRE,227,180);
piloto3RE=xl_slice(inputRE,179 , 178);
portadora3RE=xl_slice(inputRE,177, 130);
piloto2RE=xl_slice(inputRE ,129, 128);
portadora2RE=xl_slice(inputRE,127, 80);
piloto1RE=xl_slice(inputRE ,79,78);
portadora1RE=xl_slice(inputRE,77,54);
ceropaddinglsRE=xl_slice(inputRE,53,0);

%Extracción de bandas de guarda, DC y pilotos (componente imaginaria)
ceropaddingMSIM=xl_slice(inputIM,511,456);
portadora10IM=xl_slice(inputIM,455,432);
piloto8IM=xl_slice(inputIM ,431, 430);
portadora9IM=xl_slice(inputIM,429,382);
piloto7IM=xl_slice(inputIM ,381, 380);
portadora8IM=xl_slice(inputIM,379,332 );
piloto6IM=xl_slice(inputIM,331 , 330);
portadora7IM=xl_slice(inputIM,329 ,282 );
piloto5IM=xl_slice(inputIM ,281, 280);
portadora6IM=xl_slice(inputIM,279,256);
zero_DC_IM=xl_slice(inputIM,255,254)
portadora5IM=xl_slice(inputIM,253,230);
piloto4IM=xl_slice(inputIM,229 ,228);
portadora4IM=xl_slice(inputIM,227,180);
piloto3IM=xl_slice(inputIM,179 , 178);
portadora3IM=xl_slice(inputIM,177, 130);
piloto2IM=xl_slice(inputIM ,129, 128);
portadora2IM=xl_slice(inputIM,127, 80);
piloto1IM=xl_slice(inputIM, 79,78);
portadora1IM=xl_slice(inputIM,77,54);
ceropaddinglsIM=xl_slice(inputIM,53,0);

%Conformación de la salida (datos y pilotos)
OutputRE_aux =
xl_concat(portadora10RE,portadora9RE,portadora8RE,portadora7RE,portadora6RE,
portadora5RE,portadora4RE,portadora3RE,portadora2RE,portadora1RE)
OutputIM_aux =
xl_concat(portadora10IM,portadora9IM,portadora8IM,portadora7IM,portadora6IM,
portadora5IM,portadora4IM,portadora3IM,portadora2IM,portadora1IM)
PilotosRE_aux=
xl_concat(piloto8RE,piloto7RE,piloto6RE,piloto5RE,piloto4RE,piloto3RE,piloto
2RE,piloto1RE);
PilotosIM_aux=
xl_concat(piloto8IM,piloto7IM,piloto6IM,piloto5IM,piloto4IM,piloto3IM,piloto
2IM,piloto1IM);
```

```
OutputRE=xfix({xlSigned, 384, 0, xlRound, xlWrap}, OutputRE_aux);
OutputIM=xfix({xlSigned, 384, 0, xlRound, xlWrap}, OutputIM_aux);
PilotosRE=xfix({xlSigned, 16, 0, xlRound, xlWrap}, PilotosRE_aux);
PilotosIM=xfix({xlSigned, 16, 0, xlRound, xlWrap}, PilotosIM_aux);
```

9.3 Demod_bpsk.m

El código de la función MATLAB® es la implementada en el **Demodulador BPSK** es el siguiente:

```
function [data_out_bpsk]= demod_bpsk(I,Q)

%Salidas según el tipo de constelación una vez agrupados I y Q
persistent b0, b0 = xl_state(0,{xlUnsigned,1,0});

%BPSK (la Q no se emplea)
if I < 0
    b0=1;
else
    b0=0;
end

data_out_bpsk=b0;
```

10. Anexo B: Fundamentos de FPGA

Una FPGA se trata de un dispositivo semiconductor **programable** basado en una matriz de CLB (Bloques Lógicos Configurables) conectados a través de interconexiones programables. Este dispositivo se considera la evolución de los CPLD (Dispositivos Lógicos Programables Complejos).

Su invención, en 1984, se atribuye a Ross Freeman, quién ese mismo año fundó Xilinx [3] junto a Bernie Vonderchmitt y Jim Barnett, introduciendo la primera FPGA en el mercado, la XC2064, en 1985. Actualmente, Xilinx y Altera constituyen los dos principales fabricantes mundiales de FPGA de propósito general.

A diferencia de los ASIC (Circuitos Integrados para Aplicaciones Específicas), los cuales, tal y como indica su nombre, se construyen específicamente para una función, una FPGA puede programarse para diferentes aplicaciones. Además de poder programarlas para realizar diferentes funciones, existe la posibilidad de que las FPGA sean reprogramables o no. Hoy en día, el tipo dominante en el mercado son las basadas en SRAM (Memoria Estática de Acceso Aleatorio) que pueden ser reprogramadas, si bien siguen fabricándose FPGA de una única programación debido a su bajo coste respecto a los ASIC.

Los **componentes básicos** de los que consta una FPGA, tal y como se muestra en la figura 197, son:

- CLB: los bloques lógicos configurables son las unidades lógicas básicas de una FPGA. Aunque depende del dispositivo en concreto, un CLB consiste en una matriz conmutable configurable con 4 o 6 entradas, circuitería para realizar selecciones (como multiplexores) y *flip-flops* (biestables). Esta matriz es muy flexible y se puede configurar para manejar lógica combinatoria, registros de desplazamiento o RAM.

- Interconexiones: las FPGA constan de interconexiones flexibles que unen las señales entre los CLB y hacia y desde los I/O. El encaminamiento de estas conexiones es realizado por el software de diseño empleado.

- IOB: los bancos de entrada/salida se emplean para agrupar la gran cantidad de estándares de entrada/salida que soportan hoy en día las FPGA, los cuales se cuentan por docenas.

- Memoria: la mayoría de las FPGA constan de bloques de memoria RAM, lo cual es una gran ventaja al encontrarse integrada.

- DCM (Gestor del Reloj Digital): bloque empleado para gestionar el reloj digital.

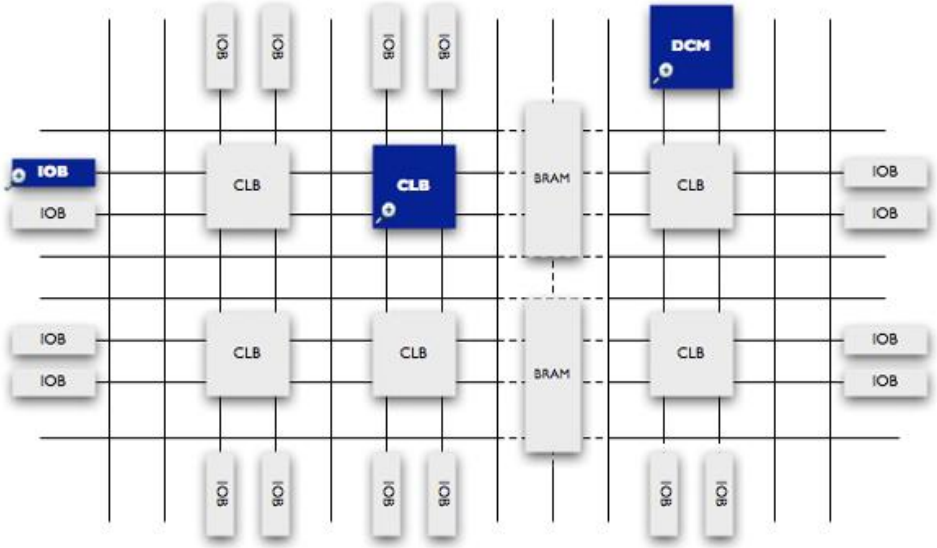

Figura 197: Estructura general de bloques de una FPGA

(Fuente: Xilinx [3])

La programación de una FPGA se basa en emplear un lenguaje de programación específico conocido como HDL, siendo los más conocidos: VHDL, Verilog y ABEL. Sin embargo, para reducir la complejidad y el tiempo necesario de desarrollo asociado al empleo de un lenguaje HDL, existen varias opciones de más alto nivel, entre las que destaca *National Instruments LabVIEW* que permite una programación gráfica de alto nivel.

Las FPGA comparten su mercado de aplicaciones con los ASIC, siendo recomendable uno u otro según la aplicación. Las mejoras en la velocidad de proceso y de capacidad de las FPGA, junto a su bajo coste, han provocado su incursión en campos de aplicación típicos de los ASIC. Sin embargo, todavía existen una serie de ventajas características propias de cada tipo de dispositivo:

- **Ventajas de ASIC** frente FPGA:
  - Menor coste unitario, para grandes cantidades de unidades.
  - Menor tamaño, debido a que el circuito se realiza a medida de la aplicación.
  - Mayor velocidad de reloj.
  - Menor consumo de potencia.
  - Total personalización, pues la fabricación del diseño se realiza para una aplicación específica.
- **Ventajas FPGA** frente ASIC:
  - Menor coste unitario en el caso de pocas unidades.

- Posibilidad de reprogramación.
- Menor tiempo de puesta en el mercado, pues no son necesarios los complejos y costosos típicos pasos del desarrollo con ASIC.
- Ciclo de diseño más simple debido al software que maneja el encaminamiento, emplazado y temporización.
- Ciclo de proyecto más predecible, pues se eliminan potenciales *re-spins* y capacidades del sustrato.
- Equipamiento sencillo.
- Aumento de la confidencialidad de las placas.

Por tanto, derivados de sus características, los mercados en los que la **aplicación de las FPGA** es directa son:

- Tecnología aeroespacial y defensa: procesamiento de imagen, generación de formas de onda y reconfiguración parcial para SDR (Software Defined Radio).
- Automoción: sistemas de asistencia, confort...
- Retransmisión: video y audio desde el estudio de producción y transmisión hasta el consumidor.
- Consumidor: pantallas digitales de información, redes de trabajo domesticas, auriculares inteligentes...
- Industria, medicina y ciencia: automatización de tareas, control de motores, imágenes médicas de alta calidad.
- Almacenamiento y servidores: soluciones para procesar datos para NAS (Almacenamiento Conectado a Red), SAN (Red de Área de Almacenamiento), servidores, dispositivos de almacenamiento...
- Comunicaciones inalámbricas: RF (Radio Frecuencia), banda base, conectividad, equipamiento inalámbrico, estándares como WCDMA, HSDPA, WiMAX...
- Comunicaciones por cable: soluciones para todos los dispositivos necesarios en el procesamiento de paquetes.

11. Anexo C: Fundamentos de MATLAB®

MATLAB®, *Matrix Laboratory*, se trata de una herramienta software matemática **orientada a matrices** que consiste en un lenguaje de computación técnico de alto nivel junto a un entorno interactivo [13], tal y como se muestra en la figura 198:

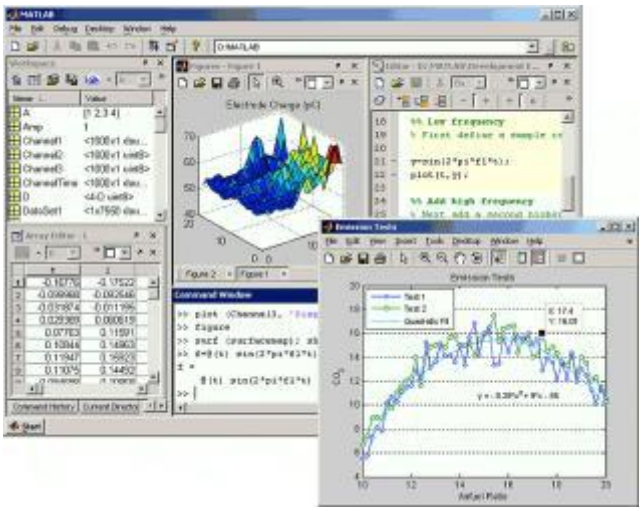

Figura 198: Interfaz gráfica de usuario de MATLAB®

(Fuente: The MathWorks [13])

Su creación por parte de The MathWorks se remonta a 1984, encontrándose disponible en la actualidad para múltiples plataformas y gozando de gran éxito en el campo de ciencia e ingeniería debido a **características** como:

- Mayor sencillez para la resolución de problemas técnicos que mediante el empleo de lenguajes de programación tradicionales como C, C++ y Fortran.
- Construcción de interfaces gráficas de usuario que acompañan a los algoritmos o aplicaciones desarrolladas.
- Adquisición de datos de múltiples fuentes externas como bases de datos o ficheros.
- Análisis de datos empleando mecanismos de computación numérica, proporcionándose diferentes funciones matemáticas (optimización, filtrado, estadísticas, análisis de Fourier…).
- Visualización de datos mediante gráficos editables en 2D y 3D, permitiendo exportarlos en diferentes formatos de ficheros gráficos.
- Producción de datos de salida en diferentes tipos de representación y formatos.

- Interoperabilidad con diversos lenguajes de programación y aplicaciones, como C, C++, Fortran, Java, Microsoft Excel…

- Arquitectura abierta, lo cual permite agregar nuevas funciones matemáticas específicas para diferentes campos mediante las llamadas *toolboxes*.

12. Anexo D: Fundamentos de Simulink®

Simulink®, Simulation and Model-Based Design, se trata de una herramienta software que consiste en un entorno gráfico interactivo (figura 199) y un conjunto de bibliotecas de bloques personalizables que permiten, mediante un alto nivel de abstracción, diseñar, simular, implementar, evaluar y optimizar **sistemas dinámicos** (variables en el tiempo) [13].

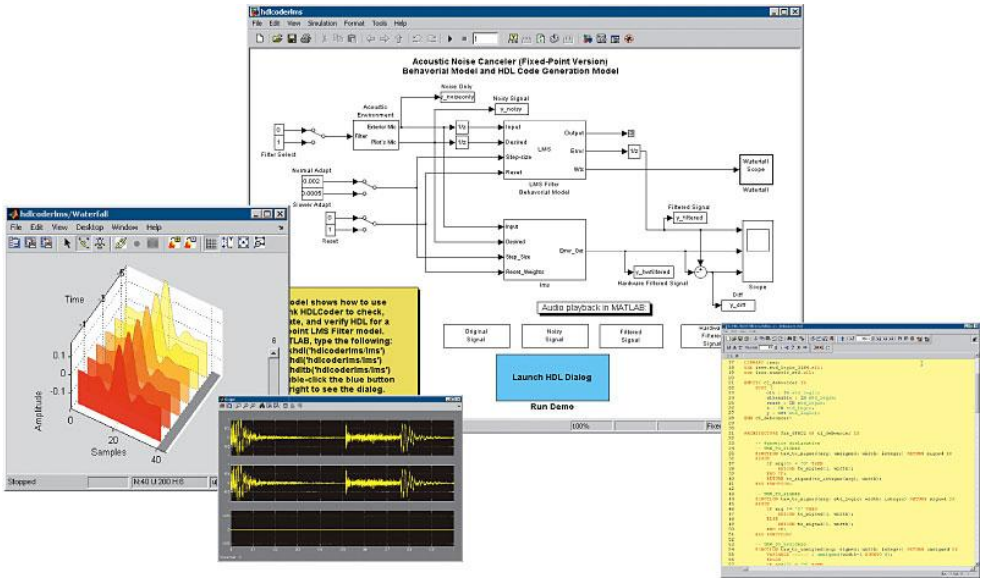

Figura 199: Interfaz gráfica de usuario de Simulink®

(Fuente: The MathWorks [13])

Esta herramienta, que se ejecuta sobre MATLAB®, fue creada por The MathWorks, encontrándose disponible para múltiples plataformas y disfrutando de una gran notoriedad debido a **características** como:

- Gran número de bibliotecas de bloques incluidas, como:
  - Bloques aritméticos: suma, producto…
  - Bloques dinámicos continuos y discretos: retardo, integración…
  - Bloques de estructuras: multiplexor, conmutador…
- Facilidad de creación de nuevos bloques para conformar nuevas bibliotecas en áreas específicas de conocimiento, como:
  - Procesamiento de señal.
  - Comunicaciones.

- Radiofrecuencia.
- Tecnología Aeroespacial.
- Procesamiento de video e imagen.
- Sistemas de control.

- Posibilidad de utilizar algoritmos creados en MATLAB® como bloques Simulink®. Además, se permite incorporar en un modelo código escrito en C, Fortran o ADA.
- Diseño de sistemas complejos con diferentes niveles de jerarquía mediante el empleo de subsistemas, los cuales aglutinan un conjunto de bloques y señales en un único bloque.
- Explorador de modelos con editor gráfico interactivo para navegar, crear, configurar, copiar, pegar, etc. Todas las señales, bloques, parámetros, propiedades y código generado asociado al modelo.
- Resolución de modelos en tiempo continuo, tiempo discreto e híbridos mediante diversos modos y posibilidades de simulación. Dichas simulaciones pueden ser dirigidas por eventos (*event driven*), dirigidas por datos (*data driven*) o dirigidas por tiempo (*time driven*).
- Depurador gráfico y analizador para examinar los resultados de las simulaciones, identificando posibles errores, inconsistencias y rendimiento del modelo.
- Herramientas de síntesis de los modelos, permitiendo generar código C, C++ y HDL a partir de los mismos.
- Integración total con MATLAB® a lo largo de todo el proceso de síntesis del modelo: desarrollo de algoritmos, definición de datos de evaluación, análisis y visualización de resultados...

13. Anexo E: Índice de tablas y figuras

TABLAS

FIGURAS