PROYECTO INTEGRADOR DE LA CARRERA DE INGENIERÍA EN TELECOMUNICACIONES

DISEÑO E IMPLEMENTACIÓN DE UN MODULADOR BASADO EN EL ESTÁNDAR DVB-S2 PARA FPGA EN VHDL

Federico Duca
Ingeniería

Mg. Guillermo Guichal
Director

Dra. Sol Pedre
Co-director

Miembros del Jurado
Ing. Roberto Costantini (IB, INVAP)
Ing. Leonardo Brocca (INVAP)

Diciembre de 2018

Instituto Balseiro
Universidad Nacional de Cuyo
Comisión Nacional de Energía Atómica
San Carlos de Bariloche, Río Negro, República Argentina

Resumen

Se realizó el estudio del estándar de segunda generación de transmisión de video satelital DVB-S2. Luego, se realizó un análisis de factibilidad de implementación del sistema de transmisión propuesto por dicho estándar y de su posible desempeño en cuanto a tasa de transmisión y ancho de banda. En base a esto, se simplificó dicho sistema a uno con un menor número de subsistemas y configuraciones por cuestiones de alcance del proyecto. Posteriormente, cada uno de los subsistemas del modulador seleccionado fue implementado en VHDL. Se utilizó como criterio de implementación, realizar diseños lo suficientemente flexibles para que la adaptación a otras configuraciones sea lo más simple posible. Finalmente, se realizaron simulaciones de comportamiento de cada uno de los subsistemas para validar el correcto funcionamiento de los módulos y, en algunos casos particulares, se contrastaron los resultados obtenidos con modelos computacionales.

Palabras clave: COMUNICACIONES SATELITALES, DVB-S2, FPGA, VHDL, BCH, LDPC

Abstract

The second generation standard of satellite video transmission DVB-S2 was studied. Afterward, a feasibility analysis of the implementation of the transmission system proposed by the standard and its possible performance in terms of transmission rate and bandwidth was carried out. Based on this, the transmission system was simplified to one constituted by a less number of subsystems and configurations because of the limited scope of this project. Subsequently, each of the subsystems of the selected modulator was implemented in VHDL. It was used as an implementation criterion, make designs flexible enough to achieve the adaptation to other configurations in the simplest way possible. Finally, behavior simulations of each of the subsystems were carried out to validate the correct functioning of the modules and, in some particular cases, the results were compared with computational models.

Keywords: SATELLITE COMMUNICATION, DVB-S2, FPGA, VHDL, BCH, LDPC

Índice de nomenclatura

Para los propósitos del presente documento, se aplica la siguiente nomenclatura en cuanto a abreviaturas y símbolos.

ACM	Adaptive Coding and Modulation (Código y Modulación Adaptativas)
APSK	Amplitude and Phase Shift Keying (Modulación por Desplazamiento de Amplitud y Fase)
AVC	Advance Video Coding (Codificación de Video Avanzada)
AWGN	Additive White Gaussian Noise (Ruido Aditivo Blanco y Gaussiano)
AXI	Advanced eXtensible Interface (Interfaz eXtensible Avanzada)
BCH	Bose-Chaudhuri-Hocquenghem «código»
BPSK	Binary Phase Shift Keying (Modulación por Desplazamiento de Fase Binario)
BS	Broadcast Service (Servicios de Radiodifusión)
BSS	Broadcast Satellite Service (Servicios de Radiodifusión Satelital)
CCM	Constant Coding and Modulation (Código y Modulación Constantes)
DFL	DATAFIELD Length (Largo del DATAFIELD)
DSP	Digital Signal Processor (Procesasdor Digital de Señales)
DTH	Direct To Home (Directo Al Hogar)
EOF	End Of Frame (Fin de Trama)

ETSI	European Telecommunications Standards Institute (Instituto Europeo de Normas de Telecomunicacione)
FEC	Forward Error Correction (Corrección de Errores “Hacia Adelante”)
FIFO	First In First Out (Cola «de elementos»)
FPGA	Field-Programmable Gate Array (Matriz de Compuertas Programables)
GF	Galois Field (Campo de Galois)
GS	Generic Stream (Trama Genérica)
HDL	Hardware Description Lenguage (Lenguaje de Descripción de Hardware)
HDTV	High Definition TeleVision (Televisión de Alta Definición)
HLS	High-Level Synthesis (Síntesis de Alto Nivel)
IP Core	Intelectual Property Core (Bloque de Propiedad Intelectual)
IRD	Integrated Receiver Decode (Receptores Decoficadores Integrados)
ITU	International Telecommunications Union (Unión Internacional de Telecommunicaciones)
LDPC	Low Density Parity Check «code» («código» Matriz de Chequeo de Baja Densidad)
LFSR	Linear Feedback Shift Register (Registro de Desplazamiento con Retroalimentación Lineal)
LSB	Least Significant Bit (Bit Menos Significativo)
MPEG	Moving Pictures Experts Group
MSB	Most Significant Bit (Bit Más Significativo)
PER	«MPEG TS» Packet Error Rate (Tasa de Error de Paquetes «MPEG TS»)
PID	Packet IDentifier (Identificador de Paquete)
PIRE	Potencia Isotrópica Radiada Equivalente

PL	Physical Layer (Capa Física)
PLS	Physical Layer Signalling (Señalización de Capa Física)
PRBS	Pseudo Random Binary Sequence (Secuencia Binaria Pseudo-Aleatoria)
PSK	Phase Shift Keying (Modulación por Desplazamiento de Fase)
QEF	Quasi-Error-Free (Casi Libre de Errores)
QPSK	Quaternary Phase Shift Keying (Modulación por Desplazamiento de Fase Cuaternaria)
RF	Radio Frequency (Radiofrecuencia)
SDTV	Standard Definition TeleVision (Televisión de Defnición Estándar)
SINR	Signal-to-Interference-plus-Noise Ratio (Relación Señal a Ruido más Interferencia)
SoC	System on Chip (Sistema en Chip)
SOF	Start Of Frame (Inicio de Trama)
TS	Transport Stream (Flujo de Transporte)
TV	Televisión
UP	User Packet (Paquete de Usuario)
UPL	User Packet Length (Largo del Paquete de Usuario)
VCM	Variable Coding and Modulation (Código y Modulación Variables)
VHDL	VhsicHDL
VHSIC	Very High Speed Integrated Circuit (Circuitos Integrados de Muy Alta Velocidad)

Índice de contenidos

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1. Motivación

La transmisión de datos entre satélites y estaciones terrenas requiere de protocolos robustos, adaptados a las particularidades de la aplicación y características del medio. En particular, la transmisión de imágenes presenta el desafío de requerir comunicaciones con gran ancho de banda. El estándar DVB-S2 propone un sistema de transmisión para estas aplicaciones que implementando un potente sistema de codificación, entre otras cosas, logra condiciones de recepción Cuasi Libres de Errores (QEF, por sus siglas en inglés) sobre canales AWGN y desempeños que distan del Límite de Shannon (límite teórico de máxima transferencia de información sin error [3]) entre 0.7 dB a 1.2 dB [2][1].

Las Matrices de Compuertas Programables (FPGA, por sus siglas en inglés) son ideales para el diseño y prueba concepto de prototipos para estas tecnologías y, en muchos casos, para la implementación del sistema de producción final. Además, las FPGAs aprovechan el paralelismo del hardware para superar la potencia de cómputo de los Procesadores Digitales de Señales (DSP, por sus siglas en inglés) disrumpiendo el paradigma de ejecución secuencial [4]. En un sistema de comunicaciones, las FPGAs permiten implementar un sistema flexible y reconfigurable de relativo bajo costo mientras que descargan al sistema central de forma parcial o total el procesamiento de señales requerido. En particular, existen sistemas de comunicaciones espaciales en la actualidad que emplean FPGAs [5].

Adicionalmente, en el ámbito de las FPGAs, existen módulos de lógica reutilizables denominados Bloques de Propiedad Intelectual (IP Cores) que son comercializados. Dentro de este mercado, se encuentran IP Cores que cumplen con funciones tanto completas como parciales de los moduladores y demoduladores propuestos en el estándar DVB-S2 [6][7][8].

1.2. Transmisión de video satelital

Hasta finales de 1990, la transmisión de la televisión digital al hogar no era práctica y resultaba costosa de implementar. Esto motivó en 1991 a un debate sobre cómo formar una plataforma paneuropea concertada para desarrollar la televisión digital terrestre. Hacia fines de ese año, cadenas de televisión, fabricantes de electrónica de consumo y organismos reguladores se reunieron para discutir la formación de un grupo que supervisaría el desarrollo de la televisión digital en Europa. Así fue como comenzó un proceso en el que rápidamente se añadieron compañías de todo el mundo, creando una alianza que mas tarde sería nombrada como Digital Video Broadcasting (DVB) Project.

Por el año 1993, DVB Project comenzó su trabajo siguiendo los siguientes puntos:

- La tarea inicial era desarrollar un conjunto completo de tecnologías de transmisión digital por satélite, cable y terrestre en un cuerpo de "pre-estandarización".
- Los sistemas propuestos servirían de contenedores para transportar cualquier combinación de imagen, audio o multimedia. De esta manera, estarían abiertos y preparados para cualquier tipo de servicio TV o nuevo medio que surgiera a lo largo del tiempo.
- El trabajo debía dar lugar a estándares del Instituto Europeo de Normas de Telecomunicaciones (ETSI) para las capas físicas, corrección de errores y el transporte para cada medio de entrega.
- Siempre que fuese posible, en las diferentes plataformas se debían tener elementos en común para reducir los costos para los fabricantes y, en consecuencia, a los usuarios. Sólo cuando no hubiera otra opción, habría diferencias entre los diferentes medios de entrega.
- El DVB Project no debía reinventar y usaría los estándares abiertos existentes siempre que estén disponibles.

El primer sistema propuesto fue el del estándar de transmisión satelital DVB-S [9] en el año 1994, junto con el estándar de transmisión por cable DVB-C [10]. Por cuestiones de marcos regulatorios mas complejos, el sistema de transmisión terrestre DVB-T [11] fue presentado en 1997. Este fue el inicio de los estándares desarrollados por DVB Project que se convirtieron en un referente mundial, habiendo conseguido en algunos casos la recomendación de la Unión Internacional de Telecomunicaciones (ITU, por sus siglas en inglés) [12]. Actualmente, se estima que casi mil millones de receptores DVB han sido desplegados [13].

El estándar DVB-S2 es la especificación DVB de segunda generación para aplicaciones de banda ancha satelital, desarrollada sobre el éxito de las especificaciones de primera generación, DVB-S para radiodifusión y DVB-DSNG [14] para servicios de recopilación y contribución de noticias satelitales, beneficiándose de los logros tecnológicos de la última década [2].

El estándar DVB-S especifica modulación QPSK y códigos convolucionales concatenados con codificación de canal Reed-Solomon, y es ahora usado mundialmente por la mayoría de los operadores satelitales para televisión y servicios de radiodifusión. DVB-DSNG especifica, en adición al formato DVB-S, el uso de modulación 8PSK y 16QAM para "satellite news gathering" y "contribution services".

Desde 1997, la tecnología de transmisión satélital digital ha evolucionado:

- Nuevos esquemas de codificación de canal, combinados con modulaciones de mayor orden, prometen alternativas más potentes a los esquemas de modulación y codificación de DVB-S/DVB-DSNG. El resultado es una ganancia de capacidad del orden el 30 % a un determinado ancho de banda

- Codificación y Modulación Variables (VCM) puede ser aplicada para proveer diferentes niveles de protección a errores para diferentes servicios (por ejemplo, SDTV y HDTV, audio, multimedia).

- En el caso de aplicaciones interactivas y punto a punto, la funcionalidad de la Codificación y Modulación Variables (VCM) se puede combinar con el uso de canales de retorno para lograr Codificación y Modulación Adaptativas (ACM). Esta técnica proporciona una protección de canal más eficaz y una adaptación de enlace dinámico a las condiciones de propagación. Los sistemas ACM prometen ganancias de capacidad satelital de 100 % hasta 200 %. Además, la disponibilidad del servicio se puede extender en comparación a un sistema de protección constante (CCM) como DVB-S o DVB-DSNG. Tales ganancias se logran informando a la estación del enlace ascendente del satélite de la condición del canal (por ejemplo, $SINR$) de cada terminal receptor a través de los canales de retorno terrestre o satelital.

- DVB-S y DVB-DSNG están estrictamente enfocados en un formato de datos único: MPEG Transport Stream (TS). Ahora es posible utilizar otros formatos de datos de entrada como múltiples TS o formatos de datos genéricos sin un aumento significativo de la complejidad.

Valéndose de todos estos avances se desarrolló la segunda generación para transmisiones satelitales. De esta manera, DVB-S2 logra ser un estándar único y muy flexible que cubre una gran variedad de aplicaciones satelitales. Esto se debe a que se caracteriza por tener:

- Un adaptador de flujos de entrada flexible adecuado para el funcionamiento con flujos de entrada simples y múltiples de diversos formatos (empaquetados o continuos)

- Un potente sistema de correción de errores (FEC), clave para lograr un rendimiento excelente en presencia de altos niveles de ruido e interferencia. El sistema FEC está basado en códigos LDPC (Low-Density Parity Check) [15] concatenados con códigos BCH (Bose-Chaudhuri-Hocquenghem) [16][17] que permiten un funcionamiento Casi Libre de Errores (QEF) operando a entre 0.7 dB a 1 dB del límite de Shannon dependiendo del modo de transmisión

- Una amplia variedad de tasas de codificación (desde 1/4 hasta 9/10)

- Un sistema de Codificación y Modulación Adaptativas (ACM) que permite que los parámetros de transmisión cambien de trama en trama en función de las condiciones particulares de entrega para cada usuario individual.

- Cuatro modos de modulación disponibles, con QPSK y 8PSK destinados a aplicaciones de transmisión en transpondedores satelitales no lineales conducidos cerca de la saturación. 16APSK y 32APSK, que requieren un nivel más alto de C/N, están dirigidos principalmente a aplicaciones profesionales. Variando en eficiencia espectral de $2\,\mathrm{bit\,s^{-1}\,Hz^{-1}}$ a $5\,\mathrm{bit\,s^{-1}\,Hz^{-1}}$, optimizadas para el funcionamiento con transpondes no lineales.

- Tres formas de pulso con factores de caída de 0.35, 0.25 y 0.20

- Un modo opcional de compatibilidad con versiones anteriores que usan modulación jerárquica para permitir que los receptores DVB-S continúen en funcionamiento y brindar capacidad y servicios adicionales a los receptores más nuevos.

1.3. Objetivos

El objetivo de este Proyecto Integrador consiste inicialmente en realizar un estudio sobre el estándar DVB-S2. Luego, realizar un análisis de factibilidad técnico para el uso del sistema DVB-S2 para comunicaciones de 1 Gbps sin superar los 400 MHz de ancho de banda. Finalmente, implementar un modulador basado en dicho estándar para FPGA en VHDL y realizar modelos del sistema planteado con herramientas de simulación para contrastar los resultados siempre que lo amerite.

1.4. Estructura del documento

El presente documento está dividido en 5 capítulos. En el Cap. 1, se presenta la motivicación y los objetivos del presente Proyecto Integrador y se introduce al estándar de transmisión de video satelital DVB-S2. En el Cap. 2, se detalla sobre el estándar DVB-S2, su sistema de transmisión, las áreas de aplicación y algunos parámetros sobre su desempeño. En el Cap. 3, se expone el análisis de factibilidad de la implementación del sistema de transmisión DVB-S2 con la tasa de transmisión de 1 Gbps y el modulador finalmente propuesto detallando las simplificaciones realizadas del diseño propuesto en el estándar. En el Cap. 4, se detalla la implementación realizada de cada uno de los subsistemas del modulador y los resultados obtenidos de las distintas pruebas de validación. Finalmente, en el Cap. 5, se exponen un análisis del trabajo realizado, realizando algunas conclusiones y propuestas para trabajos futuros.

Capítulo 2

Estándar DVB-S2

En el siguiente capítulo se exponen las bases del estándar de transmisión de video satelital DVB-S2. Se detalla la arquitectura del sistema de transmisión, sus posibles configuraciones para cada una de las áreas de aplicación y parámetros relevantes para denotar su desempeño.

2.1. Áreas de aplicación

El sistema DVB-S2 ha sido optimizado para las siguientes aplicaciones satelitales de banda ancha:

Servicios de Radiodifusión (BS) Televisión Digital multi-programa (TV) y Televisión de Alta Definición (HDTV)
Servicios de Radiodifusión para ser usados como distribución en las bandas primarias y secundarias en el Servicio por Satélite Fijo (FSS) y Servicios de Radiodifusión Satelital (BSS). DVB-S2 tiene la intención de proporcionar servicios Directo-Al-Hogar (DTH) para consumidores Receptores Decodificadores Integrados (IRD), así como los sistemas de antenas colectivas (Televisión Satelital de Antena Maestra - SMATV) y estaciones cabecera de televisión por cable. DVB-S2 puede considerarse como sucesor del estándar DVB-S [9], y puede introducirse para nuevos servicios y permitir una migración a largo plazo. Los BS se transportan en formato MPEG Transport Stream (TS). VCM puede aplicarse en múltiples flujos de transporte para lograr una protección de error diferenciada para diferentes servicios (TV, HDTV, audio, multimedia).

Servicios Interactivos (IS) Servicios de datos interactivos, incluido el acceso a Internet
DVB-S2 está destinado a proporcionar servicios interactivos a consumidores IRD y computadoras personales, donde la ruta de avance de DVB-S2 reemplaza la norma DVB-S [9] para sistemas interactivos. La ruta de retorno puede implementarse

utilizando varios sistemas interactivos DVB como DVB-RCS [18], DVB-RCP [19], DVB-RCG [20]) y DVB-RCC [21]. Los servicios de datos se transportan en formato de TS (único o múltiple) o en formato GS (único o múltiple). DVB-S2 puede proporcionar Codificación y Modulación Constantes (CCM), o Codificación y Modulación Adaptativas (ACM), donde cada estación receptora de satélite individual controla el modo de protección del tráfico que se le dirige.

Digital TV Contribution and Satellite News Gathering (DTVC/DSNG)

Las aplicaciones de DTVC por satélite consisten en transmisiones punto a punto o punto a multipunto que conectan las estaciones receptoras y de enlace ascendente fijas o transportables, no estando destinados a ser recibidos por el público en general. SNG se define como transmisión temporal y ocasional con poca antelación de televisión o sonido para fines de radiodifusión, utilizando estaciones terrenas de enlace ascendente altamente portátiles o transportables [22]. Los servicios se transportan en formato MPEG Transport Stream único (o múltiple). DVB-S2 puede proporcionar Codificación y Modulación Constantes (CCM), o Codificación y Modulación Adaptativas (ACM). En este último caso, una única estación receptora de satélite típicamente controla el modo de protección del múltiplex completo.

Distribución de contenido de datos / trunking y otras aplicaciones profesionales (PS)

Estos servicios son principalmente de punto a punto o de punto a multipunto, incluidos los servicios interactivos a las cabeceras profesionales, que redistribuyen los servicios por otros medios. Los servicios se pueden transportar en formato de flujo genérico (único o múltiple). El sistema puede proporcionar Codificación y Modulación Constantes (CCM), Codificación y Modulación Variables (VCM) o Codificación y Modulación Adaptativas (ACM). En este último caso, una única estación receptora de satélite controla típicamente el modo de protección del múltiplex TDM completo, o múltiples estaciones receptoras controlan el modo de protección del tráfico dirigido a cada uno. En cualquier caso, interactivo o no interactivo, el presente documento solo se refiere al canal de banda ancha directo.

2.2. Descripción del sistema de transmisión

2.2.1. Arquitectura del sistema

El sistema DVB-S2 está compuesto por una secuencia de bloques funcionales, tal como se expone en la Figura 2.1, que se describe a continuación:

MODE ADAPTATION (Adaptación de Modo, en español)
dependerá de la aplicación. Proporcionará interconexión y sincronización del flujo de entrada, eliminación de paquetes nulos, codificación CRC-8 para detección de errores a nivel paquete en el receptor, fusión de flujos de entrada y división en campos de data DATAFIELD. Para CCM y flujo de transporte de entrada única, el MODE ADAPTATION consistirá en una DVB-ASI (o DVB-parallel) "transparente" para la conversión de bit lógico y la codificación CRC-8. Se debe agregar un encabezado de banda base al frente del campo DATAFIELD para notificar al receptor el formato del flujo de entrada y la configuración del MODE ADAPTATION. Para aplicaciones que requieran políticas de fusión sofisticadas, de acuerdo con requisitos de servicio específicos como calidad de servicio, el MODE ADAPTATION puede realizarse opcionalmente en un dispositivo separado respetando todas las indicaciones de la especificación DVB-S2.

STREAM ADAPTER (Adaptador de Flujo, en español)
debe adaptar el largo de la trama de banda base y la aleatorización de banda base.

FORWARD ERROR CORRECTION ENCODING (Codificación de Corrección de Errores, en español)
se llevará a cabo mediante la concatenación de códigos BCH externos y códigos LDPC internos (con tasas 1/4, 1/3, 2/5, 1/2, 3/5, 2/3, 3/4, 4/5, 5/6, 8/9 y 9/10). Dependiendo del área de aplicación, el bloque codificado FEC tendrá una longitud 64 800 bit o 16 200 bit. Cuando se usan VCM y ACM, el FEC y el modo de modulación puede cambiar entre tramas pero permanecen constantes dentro de cada uno.

MAPPING (Mapeo, en español)
a las constelaciones QPSK, 8PSK, 16APSK y 32APSK debe ser aplicado dependiendo del área de aplicación.

PHYSICAL LAYER FRAMING (Entramado de Capa Física, en español)
debe realizarse de forma sincrónica con las tramas FEC, proporcionar la inserción de DUMMY PLFRAME (trama de capa física vacía) cuando no haya datos listos para transmitir, la inserción de señalización de capa física, inserción de símbolos piloto y aleatorización de capa física para la dispersión de energía. El sistema proporciona una estructura para el PHYSICAL LAYER FRAMING basada en SLOTs de 90 símbolos modulados lo que permite la sincronización confiable del receptor en la estructura del bloque FEC. Un SLOT está dedicado a la señalización de capa física, incluida la delimitación del inicio de trama y la definición del modo de transmisión. Este mecanismo es adecuado también para la configuración

del demodulador de VCM y ACM. La recuperación de portadora en el receptor puede verse facilitada por la introducción de un raster regular de símbolos piloto, sin embargo, un modo de transmisión sin piloto también está disponible ofreciendo un 2.4 % adicional de capacidad útil.

MODULATION (Modulación, en español)

debe ser aplicado para realizar el filtrado de banda base y así dar forma al espectro de la señal (raíz de coseno elevado con factores de caída 0.35, 0.25 y 0.20) y la modulación en cuadratura para generar la señal de RF.

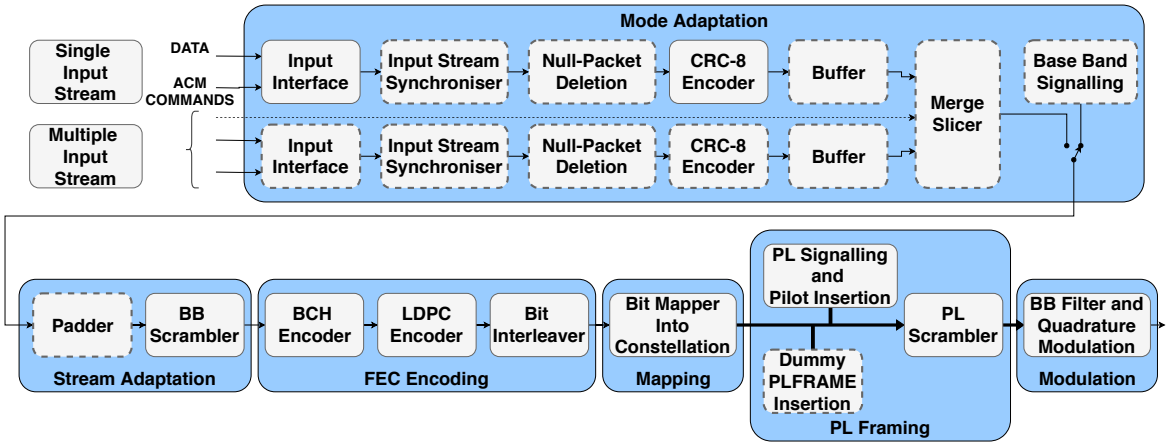


Figura 2.1: Diagrama de bloques funcional del sistema de transmisión DVB-S2 (adaptado de [1]).

2.2.2. Configuración del sistema

Las configuraciones del sistema para cada área de aplicación se expone en la Tabla 2.1. Aquellas funcionalidades y subsistemas que se definen como *Normativo* deben implementarse en los equipos transmisores y receptores para cumplir con el Estándar DVB-S2. Sin embargo, las pautas para la selección del modo no son indicadas por el Estándar (son dadas en otro documento [23]). Aquellas configuraciones que se definen como *Opcional* para un área de aplicación dada no necesitan implementarse en el equipo para cumplir con el Estándar. No obstante, en caso de implementar alguna de esas configuraciones, se debe cumplir con las especificaciones de dicho Estándar.

Tabla 2.1: Configuración del sistema y áreas de aplicación (adaptado de [1]).

Configuración del sistema	Servicios de Radiodifusión	Servicios Interactivos	DSNG	Servicios Profesionales
QPSK 1/4, 1/3 y 2/5	O	N	N	N
QPSK 1/2, 3/5, 2/3, 3/4, 4/5, 5/6, 8/9 y 9/10	N	N	N	N
8PSK 3/5, 2/3, 3/4, 5/6, 8/9 y 9/10	N	N	N	N
16APSK 2/3, 3/4, 5/6, 8/9 y 9/10	O	N	N	N
32APSK 3/4, 5/6, 8/9 y 9/10	O	N	N	N
Código y Modulación Constantes (CCM)	N	N[1]	N	N
Código y Modulación Variables (VCM)	O	O	O	O
Código y Modulación Adaptativas (ACM)	NA	N[2]	O	O
FECFRAME NORMAL (64 800 bit)	N	N	N	N
FECFRAME CORTO (16 200 bit)	NA	N	O	N
Flujo de Transporte (TS) Simple	N	N[1]	N	N
Flujo de Transporte (TS) Múltiple	O	O[2]	O	O
Flujo Genérico (GS) Simple	NA	O[2]	NA	O
Flujo Genérico (GS) Múltiple	NA	O[2]	NA	O
Factor de caída (0.35, 0.25 y 0.2)	N	N	N	N
INPUT STREAM SYNCHRONIZER	NA[3]	O[3]	O[3]	O[3]
NULL PACKET DELETION	NA[3]	O[3]	O[3]	O[3]
DUMMY FRAME INSERTION	NA[3]	N	N	N
Modo Banda-Ancha	O	O	O	O

N: *Normativo*, O: *Opcional*, NA: *No Aplicable*

[1] Receptores de Servicios Interactivos deben implementar CCM y Single Transport Stream.

[2] Receptores de Servicios Interactivos deben implementar ACM con al menos uno de las dos opciones: Multiple Transport Streams o Generic Stream (entradas simples o múltiples).

[3] Normativo para flujos de entrada TS simples o múltiples combinados con ACM/VCM o para flujos de entrada TS múltiples combinados con CCM.

2.3. Especificación de los subsistemas

La descripción de los subsistemas está organizada de acuerdo con el diagrama de bloques funcional de la Figura 2.1.

2.3.1. MODE ADAPTATION

El subsistema MODE ADAPTATION es el encargado de realizar la interfaz de entrada, la sincronización de los flujos de entrada, eliminación de los paquetes nulos, la codificación CRC-8, la fusión y recorte de los flujos de entrada y la inserción de la

señalización de banda base siempre y cuando la configuración adoptada lo requiera.

Las secuencias de entrada permitidas son:

- Transport Streams (TS, en español: Flujo de Transporte) simples o múltiples de transporte.
- Generic Streams (GS, en español: Flujo Genérico) simples o múltiples, empaquetados o continuos.

La salida consiste en una trama con los campos BBHEADER(80 bit) y DATAFIELD.

2.3.1.1. INPUT INTERFACE

Este subsistema debe mapear el formato de la entrada eléctrica al formato de bit lógico interno, siendo el primer bit indicado como el bit más significativo (MSB).

El subsistema admite los siguientes tipos de interfaces:

- MPEG TS
- GS
- ACM commands
- Mode Adaptation

Un TS se caracteriza por USER PACKETS (UP) de longitud constante UPL de un paquete MPEG (188 B) donde el primer byte es un SYNC-BYTE (47_{HEX}).

Un GS se caracteriza por un flujo continuo de bits o un flujo de UP de longitud constante. La longitud del UP (UPL) máxima es de 64 000 bit y $UPL = 0$ es interpretado como flujo continuo. Un flujo de paquetes de longitud variable o un paquete de longitud constante superior a la longitud máxima de UPL se tratará como un flujo continuo.

Para GS empaquetadas, si un byte de sincronización es el primer byte de UP, no se modificará, de lo contrario se insertará un byte de sincronización = 0 antes de cada paquete y el UPL se deberá aumentar en ocho.

La entrada de señalización Comando ACM debe permitir el ajuste por parte de una unidad de control de modo de transmisión externa de los parámetros de transmisión que adoptará el modulador DVB-S2 para una parte específica de los datos de entrada.

EL Mode Adaptation es una entrada opcional que consistirá en una secuencia de múltiples campos especificados en múltiples apéndices del Estándar.

2.3.1.2. INPUT STREAM SYNCHRONISER

NOTA: Este módulo no es relevante para TS simples - Servicios Radiodifusión.

El procesamiento de datos en el modulador DVB-S2 puede producir un retraso de transmisión variable en la información del usuario. Este subsistema garantiza una tasa de bits constante y el retardo de transmisión constante de extremo a extremo para flujos de entrada empaquetados.

2.3.1.3. NULL-PACKET DELETION

NOTA: Únicamente para ACM y TS.

Para los modos ACM y el formato de datos de entrada TS, se identificarán los paquetes MPEG nulos ($PID = 8191$) y se eliminarán. Esto permite reducir la tasa de información y aumentar la protección contra errores en el modulador. El proceso se lleva a cabo de forma tal que los paquetes nulos eliminados se puedan volver a insertar en el receptor en el lugar exacto en el que se encontraban originalmente.

2.3.1.4. CRC-8 ENCODER

NOTA: Solo para transmisiones empaquetadas

La parte útil del UP (excluyendo el SYNC-BYTE) debe ser procesada por un codificador CRC sistemático de 8 bits. El polinomio generador del codificador es:

$$g(x) = \left(X^5 + X^4 + X^3 + X^2 + 1\right)\left(X^2 + X + 1\right)(X + 1) = X^8 + X^7 + X^6 + X^4 + X^2 + 1$$

La salida del codificador CRC se calculará a partir de la secuencia de entrada $u(x)$ (UPL - 8 bits) de la siguiente forma:

$$CRC = \text{resto}\left[X^8 u(x) : g(x)\right]$$

El CRC-8 calculado reemplazará al SYNC-BYTE del siguiente UP y el SYNC-BYTE será insertado en el BBHEADER.

2.3.1.5. MERGER / SLICER

En este subsistema se almacenarán los flujos de entrada hasta que se pueda realizar el fraccionamiento de dichos flujos en un DATAFIELD. El largo del DATAFIELD (DFL) depende de la aplicación, sin embargo, debe respetarse la siguiente condición:

$$0 \leq DFL \leq K_{bch} - (10 \times 8)$$

Donde K_{bch} es longitud del mensaje a codificar por el código BCH, y 80 bits están dedicados al BBHEADER.

Este subsistema deberá concatenar, en una única salida, diferentes campos de datos leídos y cortados a partir de una de sus entradas. En presencia de una única transmisión, sólo se aplica la funcionalidad de fraccionador.

Después de la sustitución de SYNC-BYTE por el CRC-8, es necesario proporcionar al receptor un método para recuperar la sincronización de los UP. Por lo tanto, la distancia en bits entre el comienzo del DATAFIELD y el comienzo del primer UP completo (primer bit del CRC-8) será detectada y almacenada en el campo SYNCD del encabezado de banda base.

2.3.1.6. BASE BAND SIGNALLING

Se debe insertar un encabezado de banda base (BBHEADER) de longitud fija de 10 bytes al comienzo del DATAFIELD. El encabezado estará constituido por los siguientes campos, ordenados desde el más signficativo al menos significativo:

- MATYPE (2 B): Describe el formato del flujo de entrada, el tipo de MODE ADAPTATION y el factor de caída (Roll-Off)
- UPL (2 B): Longitud del UP en bits, rango de 0 a 65 535.
- DFL (2 B): Longitud del DATAFIELD en bits, rango de 0 a 58 112.
- SYNC (1 B): Copia del SYNC-BYTE del UP.
- SYNCD (2 B): Distancia en bits entre el comienzo del DATAFIELD y el primer bit del CRC-8.
- CRC-8 (1 B): Código aplicado a los primeros 9 bytes del BBHEADER

2.3.2. STREAM ADAPTATION

En este subsistema se realiza la adaptación de longitud de la trama de entrada a una longitud constante (K_{bch} bits) y posteriormente la aleatorización de la misma. Si bien K_{bch} es una longitud fija, no es un único valor dado que depende de ciertos parámetros de la configuración del susbistema FEC ENCODING tal como se expone en la Tabla 2.2. La trama debe ser completada cuando los datos de usuario disponibles para la transmisión no son suficientes para completar un BBFRAME o cuando se debe asignar un número entero de UP en un BBFRAME.

La trama de entrada está compuesta por el encabezado BBHEADER seguido de un DATAFIELD, mientras que la trama de salida es BBFRAME.

2.3.2.1. PADDER

Este subsistema es el encargado de completar la trama de entrada con $K_{bch} - DFL - 80$ bits (ceros) después del DATAFIELD para así obtener un BBFRAME de longitud fija de K_{bch} bits tal como se expone en la Figura 2.2. Ciertas aplicaciones del estándar implican que el DATAFIELD tengan el tamaño justo que no requiera añadir bits.

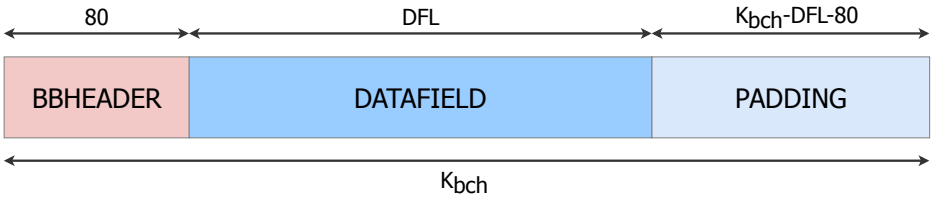


Figura 2.2: Composición de la trama BBFRAME antes de ser aleatorizado por el BASE BAND SCRAMBLER (adaptado de [1]).

2.3.2.2. BASE BAND SCRAMBLER

El BBFRAME debe ser aleatorizado para evitar una gran cantidad consecutiva de unos o de ceros. Para eso, se procesará bit a bit (XOR) la trama con una secuencia de aleatorización que debe ser sincrónica con el BBFRAME, comenzando desde el MSB y terminando después de K_{bch} bits. El polinomio generador de la secuencia binaria pseudoaleatoria (PRBS) se expone en (2.1).

$$g_{prbs}(x) = 1 + x^{14} + x^{15} \tag{2.1}$$

Cada vez que se inicia un BBFRAME, el registro de la PRBS debe ser inicializado con la secuencia binaria que se expone en (2.2).

$$100101010000000 \tag{2.2}$$

Una posible implementación del generador de la PRBS y el procesamiento con la trama de entrada se expone en la Figura 2.3.

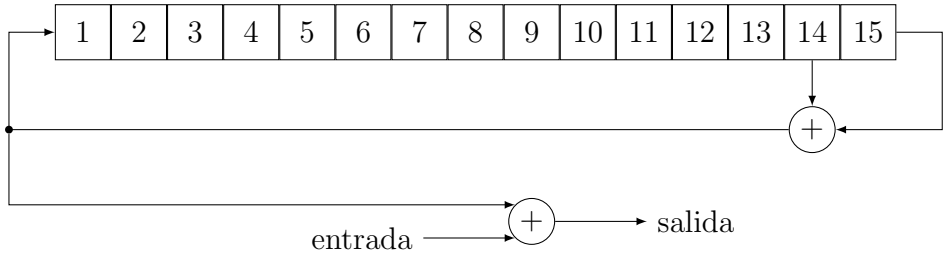


Figura 2.3: Posible implementación de la generación de la secuencia aleatoria y posterior aleatorización con la trama de entrada del subsistema SCRAMBLER DE BANDA BASE (adaptado de [1]).

2.3.3. FEC ENCODING

En este subsistema se realiza la codificación externa (BCH), la codificación interna (LDPC) y el entrelazado de bits. La trama de entrada serán BBFRAME de largo K_{bch} y la trama de salida serán FECFRAME de largo N_{ldpc}. Los bits de paridad del código externo sistemático BCH (BCHFEC) se agregan después del BBFRAME y los bits de paridad del codificador interno LDPC (LDPCFEC) se agregarán después del campo BCHFEC tal como se expone en la Figura 2.4. Los parámetros de codificación FEC para FECFRAME NORMAL y FECFRAME CORTO se exponen en la Tabla 2.2.

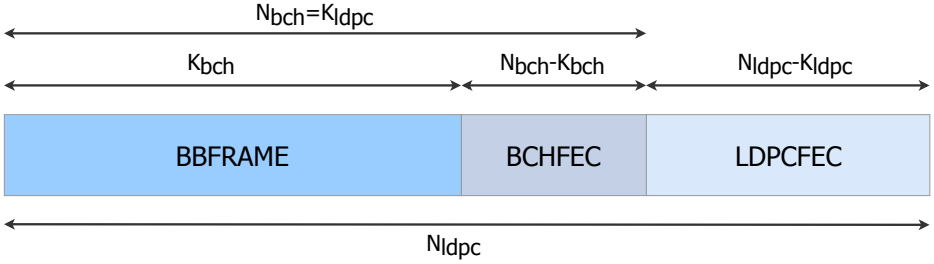


Figura 2.4: Composición del FECFRAME previo al BIT INTERLEAVER (adaptado de [1]).

2.3.3.1. BCH ENCODER

Se utilizará como codificación externa un código sistemático t-error corrector BCH (N_{bch},K_{bch}) para generar un paquete protegido contra errores. El polinomio generador del código que corrige t errores se obtiene multiplicando los primeros t polinomios fundamentales dados en la Tabla 2.3.

El Procedimiento 2.1 expone cómo codificar un mensaje $\bar{m} = (m_{K_{bch}-1}, \ldots, m_1, m_0)$ en una palabra de código $\bar{c} = (m_{K_{bch}-1}, \ldots, m_0, d_{N_{bch}-K_{bch}-1}, \ldots, d_0)$.

Tabla 2.2: Párametros de la codificación FEC (adaptado de [1]).

(a) FECFRAME NORMAL

Código LDPC	K_{bch}	$N_{bch} = K_{ldpc}$	N_{ldpc}	BCH t-error	Tasa LDPC
1/4	16008	16200	64800	12	1/4
1/3	21408	21600	64800	12	1/3
2/5	25728	25920	64800	12	2/5
1/2	32208	32400	64800	12	1/2
3/5	38688	38880	64800	12	3/5
2/3	43040	43200	64800	10	2/3
3/4	48408	48600	64800	12	3/4
4/5	51648	51840	64800	12	4/5
5/6	53840	54000	64800	10	5/6
8/9	57472	57600	64800	8	8/9
9/10	58192	58320	64800	8	9/10

(b) FECFRAME CORTO

Código LDPC	K_{bch}	$N_{bch} = K_{ldpc}$	N_{ldpc}	BCH t-error	Tasa LDPC
1/4	3072	3240	16200	12	1/5
1/3	5232	5400	16200	12	1/3
2/5	6312	6480	16200	12	2/5
1/2	7032	7200	16200	12	4/9
3/5	9552	9720	16200	12	3/5
2/3	10632	10800	16200	12	2/3
3/4	11712	11880	16200	12	11/15
4/5	12432	12600	16200	12	7/9
5/6	13152	13320	16200	12	37/45
8/9	14232	14400	16200	12	8/9
9/10	NA	NA	NA	NA	NA

Tabla 2.3: Polinomios fundamentales BCH (adaptado de [1]).

(a) FECFRAME normal

$g_1(x)$	$1 + x^2 + x^3 + x^5 + x^{16}$
$g_2(x)$	$1 + x + x^4 + x^5 + x^6 + x^8 + x^{16}$
$g_3(x)$	$1 + x^2 + x^3 + x^4 + x^5 + x^7 + x^8 + x^9 + x^{10} + x^{11} + x^{16}$
$g_4(x)$	$1 + x^2 + x^4 + x^6 + x^9 + x^{11} + x^{12} + x^{14} + x^{16}$
$g_5(x)$	$1 + x + x^2 + x^3 + x^5 + x^8 + x^9 + x^{10} + x^{11} + x^{12} + x^{16}$
$g_6(x)$	$1 + x + x^2 + x^4 + x^5 + x^7 + x^8 + x^9 + x^{10} + x^{12} + x^{13} + x^{14} + x^{15} + x^{16}$
$g_7(x)$	$1 + x + x^2 + x^5 + x^6 + x^8 + x^9 + x^{10} + x^{11} + x^{13} + x^{15} + x^{16}$
$g_8(x)$	$1 + x + x^2 + x^5 + x^6 + x^8 + x^9 + x^{12} + x^{13} + x^{14} + x^{16}$
$g_9(x)$	$1 + x^5 + x^7 + x^9 + x^{10} + x^{11} + x^{16}$
$g_{10}(x)$	$1 + x + x^2 + x^5 + x^7 + x^8 + x^{10} + x^{12} + x^{13} + x^{14} + x^{16}$
$g_{11}(x)$	$1 + x^2 + x^3 + x^5 + x^9 + x^{11} + x^{12} + x^{13} + x^{16}$
$g_{12}(x)$	$1 + x + x^5 + x^6 + x^7 + x^9 + x^{11} + x^{12} + x^{16}$

(b) FECFRAME corto

$g_1(x)$	$1 + x + x^3 + x^5 + x^{14}$
$g_2(x)$	$1 + x^6 + x^8 + x^{11} + x^{14}$
$g_3(x)$	$1 + x + x^2 + x^6 + x^9 + x^{10} + x^{14}$
$g_4(x)$	$1 + x^4 + x^7 + x^8 + x^{10} + x^{12} + x^{14}$
$g_5(x)$	$1 + x + x^2 + x^4 + x^6 + x^8 + x^9 + x^{11} + x^{13} + x^{14}$
$g_6(x)$	$1 + x + x^3 + x^7 + x^8 + x^9 + x^{13} + x^{14}$
$g_7(x)$	$1 + x + x^2 + x^5 + x^6 + x^7 + x^{10} + x^{11} + x^{13} + x^{14}$
$g_8(x)$	$1 + x^5 + x^8 + x^9 + x^{10} + x^{11} + x^{14}$
$g_9(x)$	$1 + x + x^2 + x^3 + x^9 + x^{10} + x^{14}$
$g_{10}(x)$	$1 + x^3 + x^6 + x^9 + x^{11} + x^{12} + x^{14}$
$g_{11}(x)$	$1 + x^4 + x^{11} + x^{12} + x^{14}$
$g_{12}(x)$	$1 + x + x^2 + x^3 + x^5 + x^6 + x^7 + x^8 + x^{10} + x^{13} + x^{14}$

Procedimiento 2.1. Codificación sistemática BCH:

1. Multiplicar el mensaje (polinomio) por $x^{N_{bch}-K_{bch}}$

$$m(x)x^{N_{bch}-K_{bch}} = m_{K_{bch}-1}x^{N_{bch}-1} + m_{K_{bch}-2}x^{N_{bch}-2} + \cdots + m_0 x^{N_{bch}-K_{bch}}$$

2. Dividir el resultado por el polinomio generador $g(x)$ y dejar el resto.

$$d(x) = \text{resto}\left[\frac{m(x)x^{N_{bch}-K_{bch}}}{g(x)}\right] \tag{2.3}$$

3. Establecer la palabra de código polinomial

$$c(x) = m(x)x^{N_{bch}-K_{bch}} + d(x)$$

2.3.3.2. LDPC ENCODER

Se utilizará como codificación interna un código sistemático LDPC (N_{ldpc},K_{ldpc}) para generar un paquete protegido contra errores. La salida de este codificador es un FEC-FRAME Normal o Corto (sin entrelazar), es decir, que tiene un tamaño $N_{ldpc} = 64800$ o $N_{ldpc} = 16200$ respectivamente. La transmisión de la palabra de código comienza en el orden dado desde el primer bit de información hasta el último bit de paridad.

El Prodimiento 2.2 expone cómo codificar un mensaje $\bar{i} = (i_0, i_1, \ldots, i_{K_{ldpc}-1})$ en una palabra de código $\bar{c} = (i_0, i_1, \ldots, i_{K_{ldpc}-1}, p_0, p_1, \ldots, p_{N_{ldpc}-K_{ldpc}-1})$.

Procedimiento 2.2. Codificación LDPC:

1. Inicializar los bits de paridad en cero

$$p_0 = p_1 = \cdots = p_{N_{ldpc}-K_{ldpc}-1} = 0$$

2. Acumular (XOR) de a 360 bits del mensaje iniciando del i_0 al i_{359}.

$$p_r = p_r \oplus i_m$$

Donde r es:

$$r = \left[x + [m]mod_{360} \times \frac{N_{ldpc} - K_{ldpc}}{360}\right] mod_{N_{ldpc}-K_{ldpc}}$$

Donde x son las direcciones de los bits de paridad indicadas en una **fila** de una dada tabla indicada en un Ápendice del Estándar [1].

3. Repetir el paso anterior utilizando las posiciones indicadas en la siguiente **fila** de la dicha tabla tomando los siguientes 360 bits del mensaje hasta haber procesado todos los bits del mensaje.

4. Finalmente realizar la siguiente operación secuencialmente iniciando en $i = 1$

$$p_i = p_i \oplus p_{i-1} \ (i = 1, \ldots, n - k - 1)$$

2.3.3.3. BIT INTERLEAVER

Exclusivamente para los formatos de modulación 8PSK, 16APSK y 32APSK, se implementará a la salida del LDPC ENCODER un subsistema encargado de entrelazar bits. Los datos se escriben en serie en el entrelazador en forma de columna, y se leen en serie por filas tal como se muestra en la Figura 2.5.

La configuración del BIT INTERLEAVER para cada modulación y formato de FECFRAME se especifica en la Tabla 2.4.

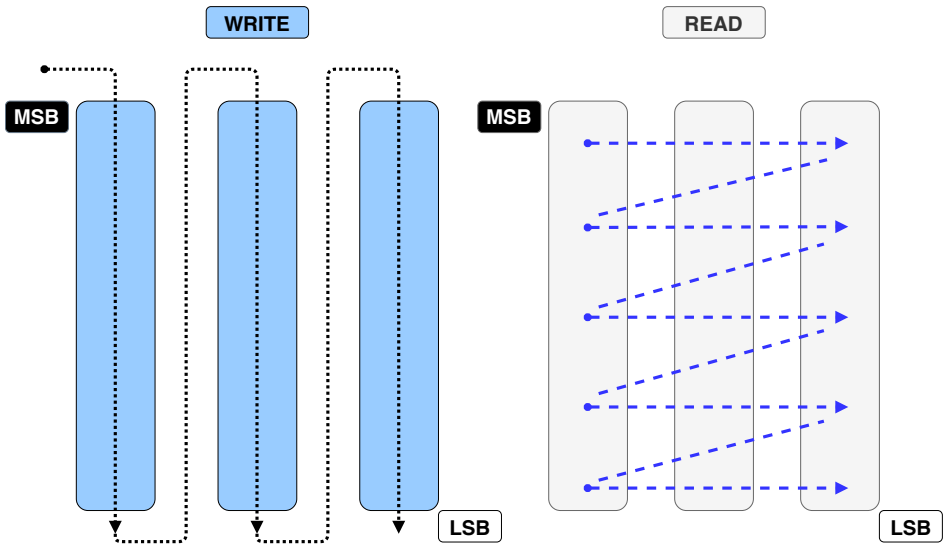
Tabla 2.4: Estructura y procedimiento de lectura y escritura del subsistema BIT INTERLEAVER (adaptado de [1]).

(a) FECFRAME NORMAL

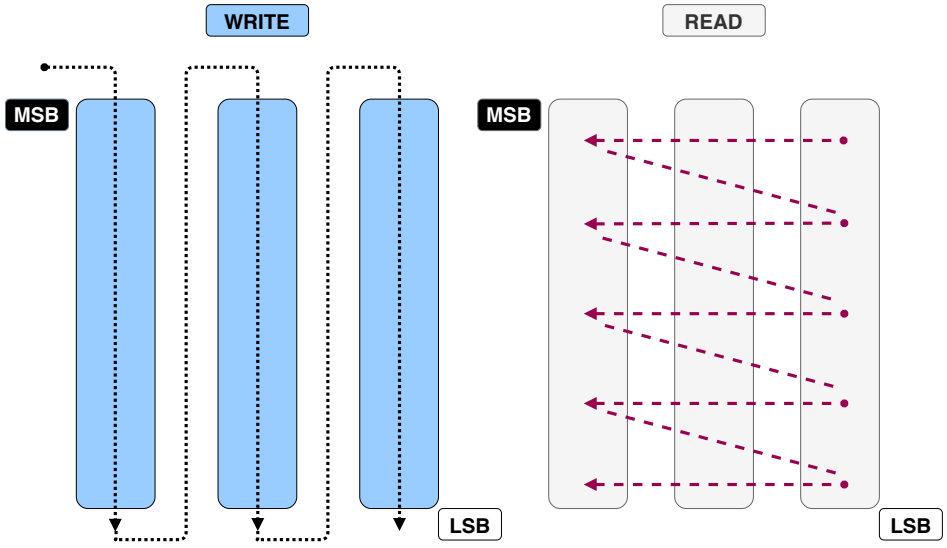
Modulación	Columnas	Filas
8PSK	21600	3
16APSK	16200	4
32APSK	12960	5

(b) FECFRAME CORTO

Modulación	Columnas	Filas
8PSK	5400	3
16APSK	4050	4
32APSK	3240	5



(a) Todas los configuraciones exceptuando modulación 8PSK, tasa 3/5 y longitud de FECFRAME NORMAL



(b) Modulación 8PSK, tasa 3/5 y longitud de FECFRAME NORMAL

Figura 2.5: Esquema de entrelazado de bits (adaptado de [1]).

2.3.4. MAPPING

2.3.4.1. Bit Mapper into Constellation

En este subsistema, cada FECFRAME será modulado en QPSK, 8PSK, 16APSK o 32APSK según la configuración optada por el sistema generando una salida de símbolos complejos. La trama de salida será una XFECFRAME (compleX FECFRAME) compuesto por $64800/\eta_{MOD}$ (XFECFRAME NORMAL) o $16200/\eta_{MOD}$ (XFECFRAME CORTO) símbolos de modulación donde η_{MOD} es el número de bits por símbolo de modulación. Cada símbolo debe ser un vector complejo en formato (I, Q) (I es la componente en fase y Q la componente en cuadratura) o en el formato equivalente $\rho\ exp(j\phi)$ (ρ es el módulo del vector y ϕ su fase).

Mapeo en QPSK y 8PSK

El sistema empleará las modulaciones QPSK y 8PSK con codificación Gray convencional y asignación absoluta (sin codificación diferencial). El mapeo de bits en la constelación QPSK seguirá la Figura 2.6a, mientras que el mapeo a 8PSK seguirá la Figura 2.6b. La energía promedio normalizada por símbolo será igual a $\rho^2 = 1$.

Mapeo en 16APSK

El sistema empleará modulación 16APSK cuya constelación está compuesta por dos anillos concéntricos de 4 y 12 puntos PSK uniformemente espaciados,respectivamente, donde el anillo interior posee radio R_1 y el anillo exterior radio R_2 tal como se expone en Figura 2.7a. La relación entre el radio del anillo externo y del anillo interno ($\gamma = R_2/R_1$) deberá cumplir con la Tabla 2.5. Solo dos valores son admitidos para las amplitudes de la constelación, que permiten la optimización del rendimiento de acuerdo con las características del canal (por ejemplo, portadora única o múltiples por transpondedor, uso de predistorsión no lineal):

- $E = 1$ (E es energía de símbolo promedio) correspondiente a $4[R_1]^2+12[R_2]^2 = 16$
- $R_2 = 1$

Tabla 2.5: Relación óptima de los radios de la constelación para 16APSK (canal lineal) (adaptado de [1]).

Tasa FEC	Modulación/código Eficiencia Espectral	$\gamma = R_2/R_1$
2/3	2.66	2.15
3/4	2.99	2.85
4/5	3.19	2.75
5/6	3.32	2.70
8/9	3.55	2.60
9/10	3.59	2.57

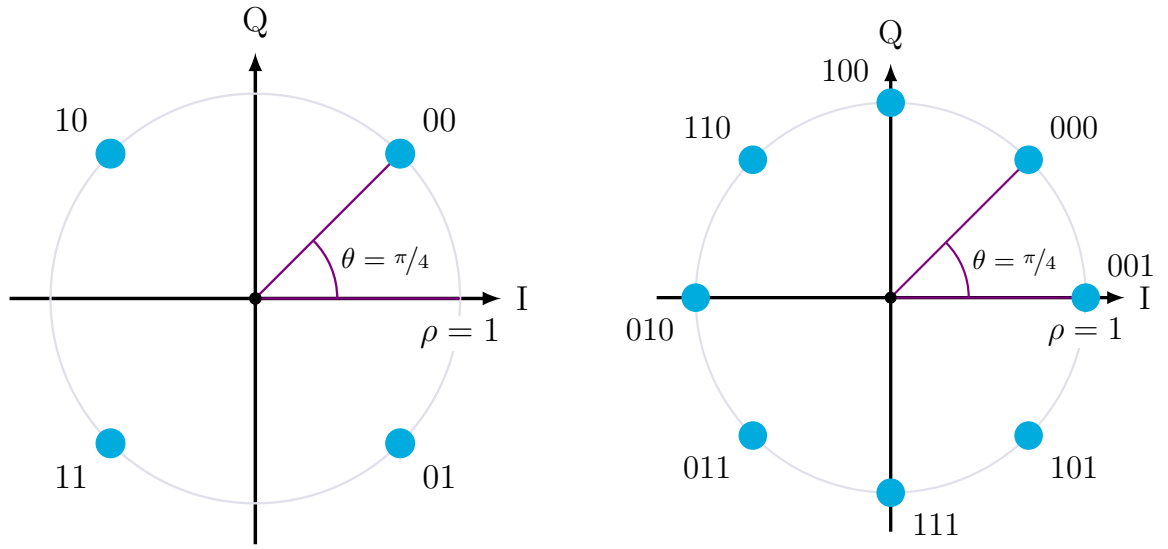
Mapeo en 32APSK

El sistema empleará modulación 32APSK cuya constelación está compuesta por tres anillos concéntricos de 4, 12 y 16 puntos PSK uniformemente espaciados, donde el anillo interior posee radio R_1, el anillo intermedio de radio R_2 y el anillo exterior radio R_3 tal como se expone en Figura 2.7b. Las relaciones entre el radio del anillo externo y del anillo interno ($\gamma_1 = R_2/R_1$) y el radio del anillo externo y del anillo intermedio ($\gamma_2 = R_3/R_1$) debe cumplir con la Tabla 2.7b. Solo dos valores son admitidos para las amplitudes de la constelación, que permiten la optimización del rendimiento de acuerdo con las características del canal (por ejemplo, portadora única o múltiples por transpondedor, uso de predistorsión no lineal):

- $E = 1$ (E es energía de símbolo promedio) correspondiente a $4[R_1]^2 + 12[R_2]^2 + 16[R_2]^2 = 32$
- $R_3 = 1$

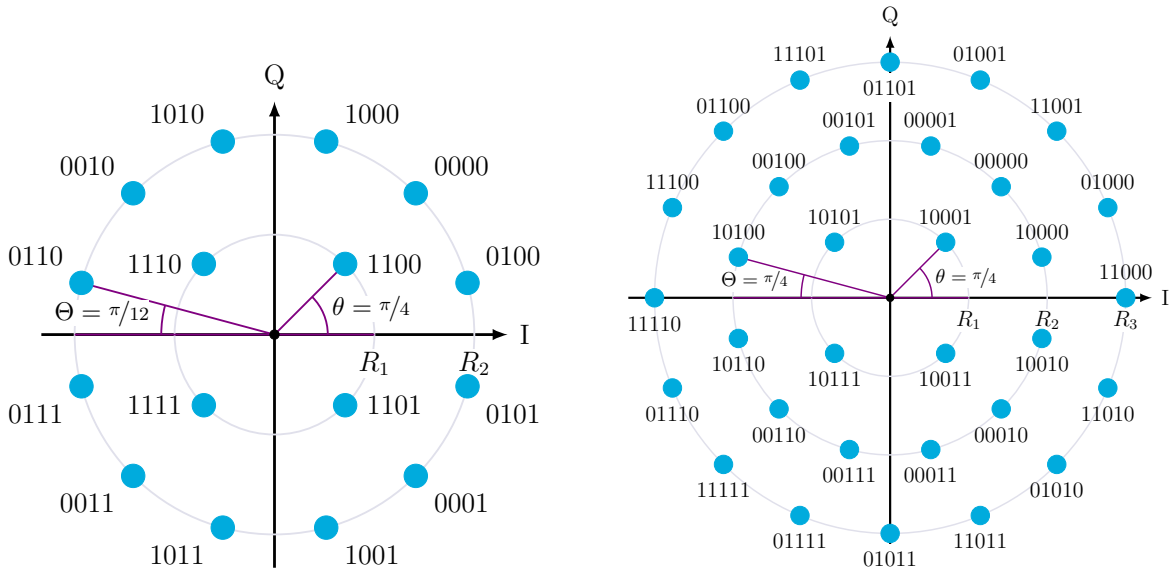
Tabla 2.6: Relaciones óptimas de los radios de la constelación γ_1 y γ_2 para 32APSK y canal lineal (adaptado de [1]).

Tasa FEC	Modulación/código Eficiencia Espectral	$\gamma_1 = R_2/R_1$	$\gamma_2 = R_3/R_1$
3/4	3.74	2.84	5.27
4/5	3.99	2.72	4.87
5/6	4.15	2.64	4.64
8/9	4.43	2.54	4.33
9/10	4.49	2.53	4.30



(a) QPSK **(b)** 8PSK

Figura 2.6: Mapeo a las constelaciones PSK (adaptado de [1]).



(a) 16APSK **(b)** 32APSK

Figura 2.7: Mapeo a las constelaciones APSK (adaptado de [1]).

2.3.5. PHYSICAL LAYER FRAMING

En este subsistema se genera la trama de capa física denominada PLFRAME mediante la realización de los siguientes procesos:

- Generación de DUMMY PLFRAME cuando ningún XFECFRAME está listo para procesar y transmitirse.
- Recorte de los XFECFRAME en un número entero S de SLOTs de longitud constante (90 símbolos); S está definido según la Tabla 2.7.

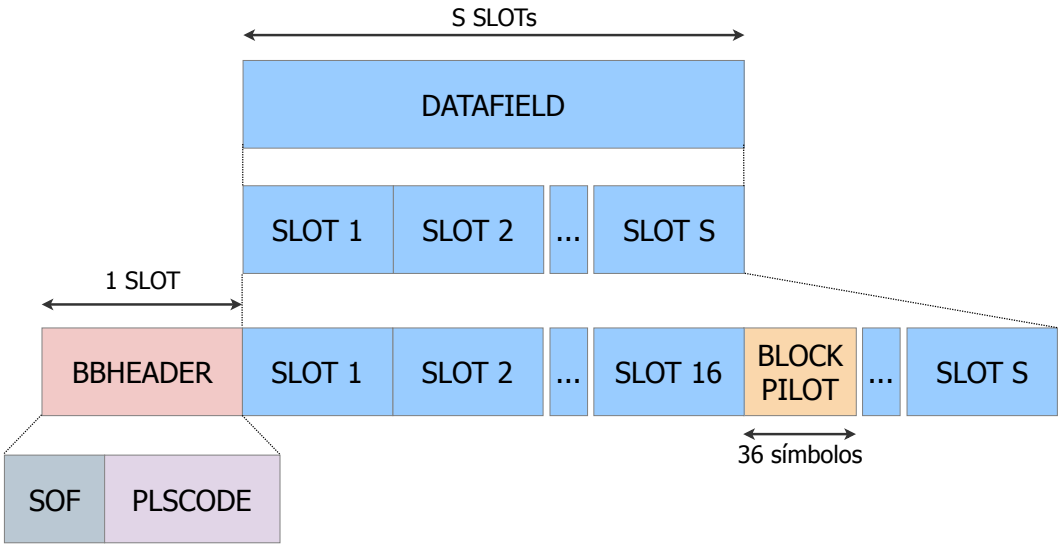


Figura 2.8: Composición de la trama PLFRAME (adaptado de [1]).

Tabla 2.7: Número de SLOTs por XFECFRAME según el tipo de FECFRAME y modulación utilizada (adaptado de [1]).

(a) XFECFRAME NORMAL

η_{MOD}	S	η (sin piloto)
2	360	99.72
3	240	99.59
4	180	99.45
5	144	99.31

(b) XFECFRAME CORTO

η_{MOD}	S	η (sin piloto)
2	90	98.90
3	60	98.36
4	45	97.83
5	36	97.30

- Generación e inserción de la señalización de capa física PLHEADER al inicio de cada XFECFRAME para configuración del receptor. El campo PLHEADER está compuesto de un SLOT.

- Inserción del BLOCK PILOT para aquellos modos que los requieran cada 16 SLOTs para ayudar a la sincronización del receptor. El BLOCK PILOT está compuesto por 36 símbolos piloto.

- Aleatorización de los símbolos modulados en fase y cuadratura.

2.3.5.1. DUMMY PLFRAME INSERTION

Un DUMMY PLFRAME estará constituido de un PLHEADER y de 36 SLOTs de portadoras sin modular ($I = Q = 1/\sqrt{2}$).

2.3.5.2. PHYSICAL LAYER SIGNALLING

En este subsistema se lleva a cabo la inserción del encabezado capa física denominado PLHEADER. El PLHEADER está diseñado para la sincronización del receptor y la señalización de la capa física. Una vez decodificado el PLHEADER, el receptor conoce la estructura y duración del PLFRAME, el esquema de modulación y codificación del sistema y la presencia o ausencia de los BLOCK PILOT.

El PLHEADER que tiene una duración de una SLOT (90 símbolos) está compuesto por los siguientes campos:

- SOF (26 símbolos): indica el inicio de la trama
- PLS (64 símbolos): el campo PLS (Physical Layer Signaling) es construido con un código binario no sistemático de longitud 64 y dimensión 7 con una distancia mínima $d_{min} = 32$, equivalente al código de Reed-Muller de primer orden bajo permutación. Los 7 bits de información del código constan de los campos MODCOD y TYPE definidos de la siguiente manera:
  - MODCOD (5 símbolos): indica la modulación y la tasa de FEC
  - TYPE (2 símbolos): indica la longitud del FECFRAME (normal o corto) y la presencia o ausencia de pilotos.

El PLHEADER será modulado utilizando $\pi/2$-BPSK de la siguiente forma:

$$\begin{aligned} I_{2i-1} = Q_{2i-1} &= \frac{1}{\sqrt{2}}(1 - 2y_{2i-1}) \\ I_{2i} = -Q_{2i} &= \frac{1}{\sqrt{2}}(1 - 2y_{2i}) \end{aligned} \tag{2.4}$$

Donde y_i con $i = 1, 2, \ldots, 90$ es la secuencia de bits del PLHEADER ya codificado.

Campo SOF

Este campo consiste en la secuencia de 26 símbolos que se expone a continuación donde el bit del lado izquierdo es el MSB del PLHEADER.

$$18D2E82_{HEX} = 01100011010010111010000010_2$$

Campo MODCOD

Este consiste en 5 bits que indican la codificación y modulación adoptada por el sistema según la Tabla 2.8.

Tabla 2.8: Valores del campo MODCOD según la tasa de codificación y modulación (adaptado de [1]).

MODCOD	Modo	
	Modulación	Tasa FEC
1	QPSK	1/4
2	QPSK	1/3
3	QPSK	2/5
4	QPSK	1/2
5	QPSK	3/5
6	QPSK	2/3
7	QPSK	3/4
8	QPSK	4/5
9	QPSK	5/6
10	QPSK	8/9
11	QPSK	9/10
12	8PSK	3/5
13	8PSK	2/3
14	8PSK	3/4
15	8PSK	5/6
16	8PSK	8/9
17	8PSK	9/10
18	16APSK	2/3
19	16APSK	3/4
20	16APSK	4/5
21	16APSK	5/6
22	16APSK	8/9
23	16APSK	9/10
24	32APSK	3/4
25	32APSK	4/5
26	32APSK	5/6
27	32APSK	8/9
28	32APSK	9/10
29	reservado	
30	reservado	
31	reservado	
0	Dummy PLFRAME	

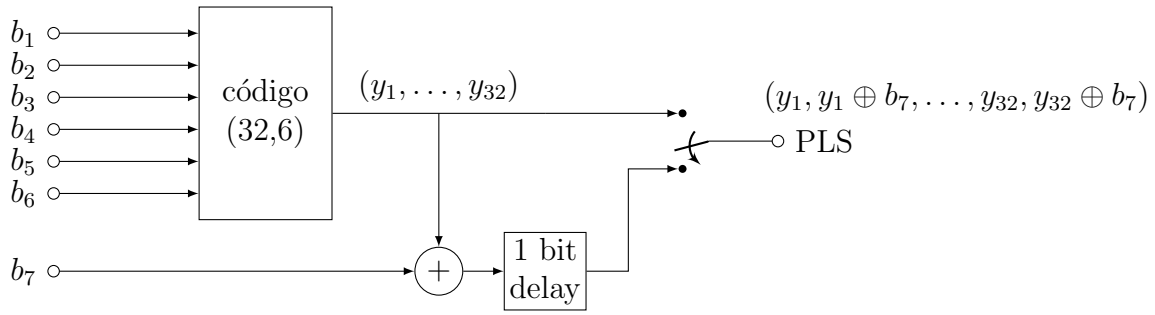


Figura 2.9: Esquema de codificación para generar el campo PLS a partir de los campos MODCOD y TYPE (adaptado de [1]).

Campo TYPE

Este campo consiste en 2 bits que indican la presencia o ausencia de señales piloto y el tamaño del FECFRAME. El LSB del campo TYPE indica la configuración de las señales pilotos (0 = sin pilotos, 1 = pilotos) y el MSB indica el tamaño del FECFRAME (0 = Normal: 64 800 bit, 1 = Corto: 16 200 bit).

Código PLS

El campo PLS consiste en los campos MODCOD y TYPE codificados de forma bi-ortogonal con un código (64,7) (y posteriormente procesado por una secuencia) que a su vez es construido a partir de otro código bi-ortogonal (32,6) tal como se expone en la Figura 2.9.

Esta construcción particular del campo PLS garantiza que cada bit impar en el código (64,7) sea siempre igual u opuesto al anterior dependiendo del valor del bit b_7 que corresponde al LSB del campo TYPE.

En cuanto al código (32,6) la matriz generadora del mismo se expone en la expresión Ecuación (2.5). De esta manera, el MSB del campo MODCOD se multiplica por la primera fila de la matriz, el siguiente bit con la segunda fila y así sucesivamente. Finalmente la palabra de código se obtiene de realizar la suma módulo 2 entre los 6 productos obtenidos.

$$G = \begin{bmatrix} 0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1 \\ 0\,0\,1\,1\,0\,0\,1\,1\,0\,0\,1\,1\,0\,0\,1\,1\,0\,0\,1\,1\,0\,0\,1\,1\,0\,0\,1\,1\,0\,0\,1\,1 \\ 0\,0\,0\,0\,1\,1\,1\,1\,0\,0\,0\,0\,1\,1\,1\,1\,0\,0\,0\,0\,1\,1\,1\,1\,0\,0\,0\,0\,1\,1\,1\,1 \\ 0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,1\,1\,1\,1\,1\,1\,0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,1\,1\,1\,1\,1\,1 \\ 0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1 \\ 1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1 \end{bmatrix} \tag{2.5}$$

Finalmente, la salida de 64 bits del código PLS debe ser procesada con la secuencia de la expresión Ecuación (2.6). El procesamiento se basa en aleatorizar los 64 bits de salida realizando la suma módulo 2 bit a bit con dicha secuencia.

$$01110001100111011000001111001001010100110100001000101101111111010 \quad (2.6)$$

2.3.5.3. PILOT INSERTION

Existen dos configuraciones posibles de la trama PLFRAME, con señales piloto o sin ellas. En caso que el servicio para el cual fue configurado el sistema exiga señales piloto, este subsistema se encargará de insertar dichas señales dentro del PLFRAME

Los BLOCK PILOT consisten en 36 símbolos piloto, símbolo no modulado identificado por $I = Q = 1/\sqrt{2}$. Los BLOCK PILOT deben insertarse cada 16 SLOTs después del PLHEADER. Sin embargo, si el BLOCK PILOT coincide con el comienzo del siguiente SOF entonces no debe insertarse.

La presencia o ausencia de las señales piloto en los modos VCM y ACM pueden cambiar trama a trama.

2.3.5.4. PHYSICAL LAYER SCRAMBLER

Antes de la modulación, cada PLFRAME (excluyendo el campo PLHEADER) debe ser procesado para lograr dispersión de energía de la trama. Para eso, se multiplican las muestras en fase y cuadratura $(I + jQ)$ por una secuencia compleja $(C_I + jC_Q)$ tal como se expone en la expresión Ecuación (2.7):

$$\begin{aligned} I_S &= IC_I - QC_Q \\ Q_S &= IC_Q + QC_I \end{aligned} \quad (2.7)$$

Donde I_S y Q_S son la rama fase y cuadratura de las muestras de salida.

Dado que la función de este subsistema consiste en realizar una dispersión de energía dentro de la trama, la tasa de la secuencia compleja previamente mencionada es la misma que la tasa de símbolo del PLFRAME. De esta manera, no hay un impacto en el ancho de banda de la señal. La secuencia de aleatorización tiene un período mayor que la duración máxima requerida de aproximadamente 70 000 símbolos).

La secuencia de aleatorización se reinicializará al final de cada PLHEADER. La duración del PLFRAME depende de la modulación seleccionada, por lo que la longitud de la secuencia de aleatorización se truncará a la longitud del correspondiente PLFRAME.

Las secuencias de código de aleatorización se construirán combinando dos secuencias de máxima longitud reales (x e y) generadas por medio de dos polinomios generadores de grado 18, en una secuencia compleja. Las secuencias resultantes constituyen segmentos de un conjunto de secuencias de Gold.

Las dos secuencias x e y se generan usando los polinomios primitivos (sobre $GF(2)$) expuestos en (2.8) y (2.9).

$$g_x(x) = 1 + x^7 + x^{18} \tag{2.8}$$

$$g_y(y) = 1 + y^5 + y^7 + y^{10} + y^{18} \tag{2.9}$$

La secuencia que depende del número de código de aleatorización elegido n se denota como z_n. Además, $x(i)$, $y(i)$ y $z_n(i)$ denotan el i-ésimo símbolo de la secuencia x, y, y z_n, respectivamente. Las m-secuencias x e y se deben inicializar con los siguientes valores:

- x se inicia con $x(0) = 1$, $x(1) = x(2) = \cdots = x(16) = x(17) = 0$.
- y se inicia con $y(0) = y(1) = \cdots = y(16) = y(17) = 1$.

Luego, las m-secuencias x e y se definen como:

- $x(i + 18) = [x(i + 7) + x(i)]$ mód $_2, i = 0, \ldots, 2^{18} - 20$.
- $y(i + 18) = [y(i + 10) + y(i + 7) + y(i + 5) + y(i)]$ mód $_2, i = 0, \ldots, 2^{18} - 20$.

La n-ésima secuencia de código Gold se define como:

$$z_n(i) = [x((i + n) \text{ mód } (2^{18} - 1)) + y(i)] \text{ mód }_2 \text{ para } i = 0, \ldots, 2^{18} - 2.$$

Estas secuencias binarias se convierten en secuencias de valores enteros R_n (R_n asumiendo los valores 0, 1, 2 y 3) mediante la siguiente transformación:

$$R_n(i) = 2z_n((i + 131072) \bmod (2^{18} - 1)) + z_n(i) \text{ para } i = 0, 1, \ldots, 66419.$$

Finalmente, la secuencia de códigos de aleatorización n-ésima compleja $C_I(i) + jC_Q(i)$ se define como:

$$C_I(i) + jC_Q(i) = exp(jR_n(i)\pi/2)$$

En el caso de los servicios de radiodifusión, $n = 0$ se utilizará como secuencia predeterminada para evitar la configuración manual del receptor o los retrasos de sincronización. n toma valores en el rango de 0 a 262 141 indicando el número de secuencia de dispersión. El uso de diferentes secuencias de aleatorización permite una reducción de la correlación de interferencia entre diferentes servicios. Para el mismo propósito, es posible reutilizar una versión desplazada de la misma secuencia en diferentes haces

de satélite. Además, n puede asociarse inequívocamente con cada operador de satélite, satélite o transpondedor permitiendo la identificación de una señal interferente mediante la detección de "firma"de codificación PL. No hay un método de señalización explícito para transmitir n al receptor.

2.3.6. MODULATION

2.3.6.1. BASE BAND (BB) FILTER

Las señales de entrada a este subsistema son procesadas para realizar la conformación de pulso. Para ello, se utiliza un filtro de raíz coseno elevado (SRRC) cuyo factor de caída (roll-off) puede ser 0.35, 0.25 y 0.20 dependiendo de los requisitos del servicio.

El filtro SRRC de banda base tendrá una función teórica definida por la siguiente expresión:

$$H(f) = \begin{cases} 1 & \text{para } |f| < f_N(1-\alpha) \\ \left[\frac{1}{2} + \frac{1}{2}\,\text{sen}\,\frac{\pi}{2f_N}\frac{f_N - |f|}{\alpha}\right]^{\frac{1}{2}} & \text{para } f_N(1-\alpha) \\ 0 & \text{para } \ |f| > f_n(1+\alpha) \end{cases}$$

Donde f_N es la frecuencia de Nyquist y α es el factor de caída.

2.3.6.2. QUADRATURE MODULATION

En este subsistema se realiza una etapa de conversión a frecuencia intermedia. El subsistema toma las señales en fase y en cuadratura, ya conformadas en pulsos, y las multiplica por el $\text{sen}(2\pi f_0 t)$ y $\cos(2\pi f_0 t)$ respectivamente (donde f_0 es la frecuencia portadora). Las dos señales resultantes son combinadas para obtener la señal de salida del modulador.

2.4. Parámetros de transmisión

2.4.1. Desempeño ante errores

Los requisitos de desempeño en condiciones QEF sobre un canal AWGN se exponen en la Tabla 2.9. La E_s/N_o [dB] ideal fue obtenida mediante simulación por computadora, con 50 iteraciones de decodificación de punto fijo LDPC [23], portadora perfecta y recuperación de sincronización, sin ruido de fase sobre canal AWGN para FECFRAME NORMAL. Para FECFRAME CORTO debe tenerse en cuenta una degradación adicional de 0.2 dB a 0.3 dB.

Tabla 2.9: Desempeño de la relación energía de símbolo promedio y densidad espectral de potencia de ruido (E_s/N_o) en transmisiones Casi-Libres de Errores ($PER = 10^{-7}$) para un canal AWGN (adaptado de [1]).

Modo		Eficiencia Espectral	E_s/N_o ideal [dB]
QPSK	1/4	0.490 243	−2.35
QPSK	1/3	0.656 448	−1.24
QPSK	2/5	0.789 412	−0.30
QPSK	1/2	0.988 858	1
QPSK	3/5	1.188 304	2.23
QPSK	2/3	1.322 253	3.10
QPSK	3/4	1.487 473	4.03
QPSK	4/5	1.587 196	4.68
QPSK	5/6	1.654 663	5.18
QPSK	8/9	1.766 451	6.20
QPSK	9/10	1.788 612	6.42
8PSK	3/5	1.779 991	5.50
8PSK	2/3	1.980 636	6.62
8PSK	3/4	2.228 124	7.91
8PSK	5/6	2.478 562	9.35
8PSK	8/9	2.646 012	10.69
8PSK	9/10	2.679 207	10.98
16APSK	2/3	2.637 201	8.97
16APSK	3/4	2.966 728	10.21
16APSK	4/5	3.165 623	11.03
16APSK	5/6	3.300 184	11.61
16APSK	8/9	3.523 143	12.89
16APSK	9/10	3.567 342	13.13
32APSK	3/4	3.703 295	12.73
32APSK	4/5	3.951 571	13.64
32APSK	5/6	4.119 540	14.28
32APSK	8/9	4.397 854	15.69
32APSK	9/10	4.453 027	16.05

Para calcular los presupuestos de potencia del enlace, se deben tener en cuenta las degradaciones específicas del canal de satélite.

La Tasa de Error de Paquete (PER) es la relación entre los paquetes de flujo de transporte útiles (188 B) recibidos correctamente y afectados por errores después de la corrección de errores.

La definición de QEF adoptada para DVB-S2 es menos de un evento de error no corregido por hora de transmisión en el nivel de un decodificador de servicio de TV individual de 5 Mbps, aproximadamente correspondiente a una relación de error de paquete de tren de transporte PER $< 10^{-7}$ antes del demultiplexor.

2.4.2. Tasas de transmisión y anchos de banda

El sistema DVB-S2 es apto para utilizar en diferentes anchos de banda de transpondedor del satélite y bandas de frecuencia. En las configuraciones de portadora única por transpondedor, la tasa del símbolo de transmisión se puede adaptar al ancho de banda del mismo (a $-3\,\mathrm{dB}$) para lograr la máxima capacidad de transmisión compatible con la degradación aceptable de la señal debido a las limitaciones del ancho de banda. En la configuración de multiples portadora, la tasa de símbolo puede coincidir con la ranura de frecuencia asignada al servicio por el plan de frecuencia, para optimizar la capacidad de transmisión manteniendo la interferencia mutua entre operadores adyacentes en un nivel aceptable [23][24].

Si bien, el estándar permite adaptar la tasa de símbolo al sistema empleado, no cualquier tasa de transmisión logrará cumplir requerimientos de transmisión de video útiles. En la Tabla 2.10 se expone una breve comparativa de la capacidad de transmisión de la primera y la segunda generación del estándar DVB para transmisiones satelitales [2]. Dado que la transmisión no solo contiene datos de usuario, se define la tasa efectiva de trasmisión que resulta menor. De esta manera, si se quisieran lograr determinados requerimientos de transmisión de video se debe definir las configuraciones del sistema DVB-S2 y calcular la capacidad de dicho sistema. Existen calculadoras que permiten a partir de una dada tasa de símbolo o ancho de banda calcular la capacidad del sistema para una dada configuración (modulación, tasa FEC, tipo de FECFRAME y factor de caída (Roll-off)) facilitando el cálculo [25][26].

Tabla 2.10: Comparativa de desempeño entre la primera y la segunda generación del estándar DVB para transmisiones satelitales (adaptado de [2]).

Sistema	**DVB-S**	**DVB-S2**	**DVB-S**	**DVB-S2**
Satélite PIRE [dBW]	51	51	53.7	53.7
C/N @27.5 MHz [dB]	5.1	5.1	7.8	7.8
Modulación	QPSK	QPSK	QPSK	8PSK
Tasa de Código	2/3	3/4	7/8	2/3
Factor de caída	0.35	0.2	0.35	0.25
Tasa de símbolo [Mbd]	27.5	30.9	27.5	29.7
Tasa efectiva [Mbps]	33.8	46[1]	44.4	58.8[2]
Programas SDTV	7 MPEG-2	10 MPEG-2	10 MPEG-2	13 MPEG-2
	15 AVC	21 AVC	20 AVC	26 AVC
Programas HDTV	1-2 MPEG-2	2 MPEG-2	2 MPEG-2	3 MPEG-2
	3-4 AVC	5 AVC	5 AVC	6 AVC

[1] Ganancia del 36 % [2] Ganancia del 32 %

Capítulo 3

Diseño del modulador basado en DVB-S2

En este capítulo se expone cuál fue el modulador propiamente implementado y los criterios de selección de los subsistemas del estándar DVB-S2 que se utilizaron para el diseño del mismo.

3.1. Análisis de factibilidad

Analizando el estándar (ver Cap. 2), se evidencia que la realización del sistema de transmisión para una determinada área de aplicación es un objetivo que excede el alcance de este proyecto. Esto se debe a que los subsistemas del modulador son numerosos, cada uno de ellos requiere más de una configuración y además algunos no resulta trivial implementarlos. Así fue como se optó acotar los subsistemas a implementar y sus configuraciones priorizando no perder la escencia del sistema DVB-S2. Sin embargo, siempre que fue posible, se implementaron los módulos de forma tal que facilite la futura adaptación a un mayor número de configuraciones.

Por otra parte, en la concepción este proyecto integrador, se ideó como objetivo obtener un modulador con el que se logre una tasa de transmisión de 1 Gbps sin exceder 700 MHz de ancho de banda. En el transcurso de la realización de este proyecto dos fuertes razones llevaron a limitar dicho objetivo. En primer lugar, investigando en distintos documentos oficiales referidos al estándar [24][23][2] se notó que las tasas típicamente expuestas rondaban aproximadamente en los 20 Mbaud equivalentes a 100 Mbps empleando la constelación de mayor orden (32APSK). En segundo lugar, el desempeño en términos generales de cualquier sistema está fuertemente ligado a la complejidad de su implementación. En el caso de implementar módulos en VHDL que trabajen con tasas tan elevadas, se requiere de un gran nivel de experiencia de diseño digital y tiempo para lograr un diseño óptimo en recursos. Este factor combinado con

el gran número de sistemas a implementar permiten concluir que el objetivo inicial de implementar el modulador en su totalidad con dicha tasa de transmisión presenta una factibilidad casi nula. De esta manera, se priorizó la implementación de un mayor número de sistemas por sobre obtener tasas de transmisión elevadas dado que aún con diseños poco óptimos podrían obtenerse tasas no muy lejanas a las típicas.

3.2. Arquitectura del modulador a implementar

A continuación, se detalla sistema por sistema siguiendo el diagrama en bloques de la Figura 2.1 los motivos por los cuales se realizó la selección de los susbsistemas y sus configuraciones:

MODE ADAPTATION: Este sistema es el más susceptible al área de aplicación siendo el sistema con mayor configurabilidad. Dado que la complejidad de la implementación resulta principalmente en lograr el funcionamiento de todas las configuraciones y no en la implementación de cada uno de los susbistemas, este sistema se descartó en su totalidad.

STREAM ADAPTATION:

- PADDER: realizar este subsistema resultaría incongruente si el sistema MODE ADAPTATION no es implementado, por lo que quedó descartado.
- BASE BAND SCRAMBLER: se optó por implementarlo dado que los sistemas de generación de secuencias pseudo-aleatorias y el procesamiento dichas secuencias con los datos a transmitir es ampliamente utilizado en los sistemas de comunicaciones.

FEC ENCODING:

- BCH ENCODER y LDPC ENCODER: Se optó implementar ambos codificadores dado que la codificación, en términos generales, cumple un rol fundamental en las comunicaciones para lograr mayores tasas de transmisión, robustez ante malas condiciones del canal, etc. Sin embargo, implementar códigos LDPC presenta una complejidad significativa por lo que se decidió acotar su funcionabilidad a la configuración que se encontrara mas simple para implementar.
- BIT INTERLEAVER: Si bien la utilización de entrelazadores es común para mitigar efectos como los errores en ráfaga dado que se optó únicamente por implementar la modulación QPSK este subsistema no debe ser implementado según el Estándar.

MAPPING: Este sistema resulta vital para realizar una transmisión por lo que se decidió implementar acotando su configuración a QPSK únicamente. Los otros sistemas que generan símbolos utilizan símbolos que quedan contenidos en la constelación QPSK ($\pi/2$-BPSK teniendo en cuenta la constelación para símbolos pares e impares y las señales piloto sin modular). De esta manera, se simplifica la integración de los distintos subsistemas.

PL FRAMING:

- DUMMY PLFRAME INSERTION: Este subsistema debe generar una trama cuando en la entrada del subsistema MERGER/SLICER del MODE ADAPTATION no hay datos de usuarios disponibles para transmitir. Dado que el sistema MODE ADAPTATION fue descartado, resulta incongruente implementar este subsistema.

- PL SIGNALLING: La inserción de encabezados es ampliamente utilizado en las comunicaciones para facilitar la transmisión de múltiples datos, optandose ser implementado.

- PILOT INSERTION: Las señales piloto permiten desde una mayor sincronización hasta una posible estimación del canal en el receptor. De esta manera, se optó por implementar.

- PL SCRAMBLER: Dado que ya se había optado por realizar el BASE BAND SCRAMBLER se encontró parcialmente redundante su implementación, por lo cual fue descartado para poder cumplir con el alcance de este proyecto.

MODULATION:

- BASE BAND FILTER: Se optó por implementar este subsistema únicamente para un único factor de caída.

- QUADRATURE MODULATION: Se optó por implementar este subsistema de manera tal que la salida del modulador no se encuentre en banda base.

Además de las configuraciones particulares de cada uno de los subsistemas mencionadas, existen configuraciones que tienen un impacto en todos los subsistemas como el largo del FECFRAME (normal y corto) y el tipo de codificación/modulación (CCM, VCM y ACM). En cuanto al tipo de FECFRAME, se implementó (excepto el

LDPC) con la capacidad de soportar ambas configuraciones. En cuanto a la codificación/modulación, se optó por implementar como constantes (CCM) dado que los otras configuraciones suponían una complejidad que supera el alcance de este proyecto.

Finalmente, la arquitectura del modulador implementado está compuesta por la secuencia de bloques funcionales que se expone en la Figura 3.1.

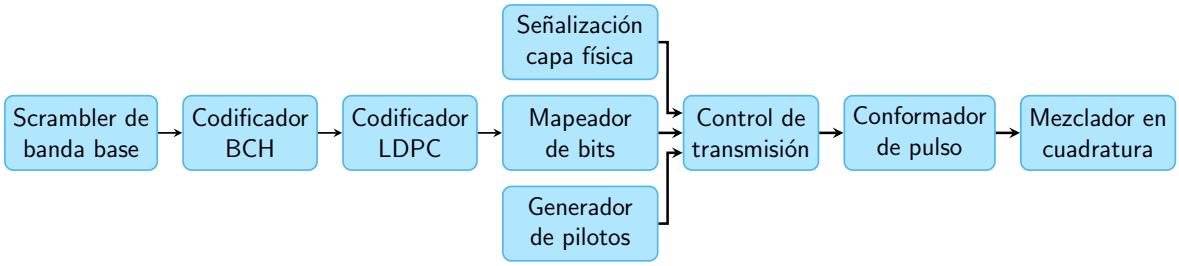


Figura 3.1: Diagrama en bloques del modulador diseñado

Capítulo 4

Implementación de los subsistemas del modulador

En el siguiente capítulo se detalla sobre la implementación de cada uno de los subsistemas del modulador diseñado. Cada uno de los subsistemas fue implementado en un módulo en VHDL bajo ciertos criterios generales que son detallados a continuación. Sobre cada uno de los módulos, se exponen sus interfaces, como realiza el control del flujo de datos y como realiza el procesamiento de los mismos. Finalmente, se exponen las pruebas a las cuales fueron sometidos para validar su correcto funcionamiento.

4.1. Criterios de implementación

Antes de comenzar a realizar la implementación de los subsistemas en módulos VHDL, se debió definir la FPGA sobre la cúal se trabajaría. Tres motivos llevaron a optar por utilizar FPGAs de Xilinx. En primer lugar, Xilinx proveé FPGAs de grado espacial que son utilizadas en la actualidad en satélites de comunicación [5]. En segundo lugar, se contaba con cierta experiencia previa al inicio este proyecto en el manejo de las herramientas de Xilinx (*Vivado*) utilizadas para realizar los módulos y las simulaciones de comportamiento. Finalmente, si bien una eventual migración a otro sistema presenta cierta complejidad, la complejidad recae principalmente en la implementación inicial de los subsistemas.

Basada en la decisión de utilizar FPGAs de Xilinx, se procedió a definir las interfaces que tendrían los módulos de forma tal que atiendan todas las necesidades de los distintos subsistemas utilizando una solución englobadora y que a la vez fuera estándar para poder reutilizar los módulos para otras funcionalidades. Realizando la investigación previa, se encontró que Xilinx a adoptado el protocolo *Advanced eXtensible Interface* (AXI) para los IP Cores. El protocolo AXI es parte de la especificación *Advanced Microcontroller Bus Architecture* (AMBA) ARM, una familia de buses para

microcontroladores. Cuando AMBA 4.0 fue introducido en 2010, se incluyó la segunda versión de AXI denominada *AXI4* con 3 tipos de interfaces: [27]

- *AXI4*: para requerimientos de alto rendimiento con mapeo de memoria.
- *AXI4-Lite*: para comunicaciones simples de bajo rendimiento con mapeo de memoria.
- *AXI4-Stream*: para transmisión de datos continuos de alta velocidad.

Dado que no se requería utilizar mapeo de memoria, se optó por utilizar *AXI4-Stream* para la comunicación de los módulos y plantear como *Genérico* aquellos parámetros de configuración que requieran los módulos.

La utilización de la señal TREADY no es obligatoria según el protocolo, sin embargo, permite que la tasa de procesamiento de los módulos sea independiente de la de los módulos adyacentes. Particularmente, se utiliza la señal TLAST para indicar el fin de trama ya que permite a los módulos recibir tramas de distintas longitudes adaptándose así las distintas configuraciones del modulador. En cuanto al bus de datos, si bien se implementó como *Genérico*, se optó por utilizar un ancho de 8 bits en la mayoría de los subsistemas por dos razones. En primer lugar, todas las posibles tramas de entrada y salida son perfectamente fraccionables por esta cantidad, lo que facilita la lógica del flujo de datos. Por otra parte, resulta un equilibrio entre la eficiencia del espacio y recursos necesarios para implementar el diseño en la FPGA y la eficiencia en cuanto a la tasa de procesamiento. Para los módulos de *Conformador de Pulso* y *Mezclador en Cuadratura* se requirió utilizar un ancho mayor dado que se encontró que mejoraba la resolución del filtro utilizado para la conformación del pulso.

Respecto a el control del flujo de datos, se encontró que existen ciertos comportamientos en común entre los distintos módulos. Con lo cuál, se decidió utilizar un esquema que posea la suficiente generalidad para la reutilización en los distintos módulos y la suficiente flexibilidad para ajustar el diseño a la particularidad de cada uno de ellos con el fin de facilitar las implementación. Se encontró que la utilización de máquinas de estado brinda una solución con las características mencionadas.

En cuanto al proceso de validación de las implementaciones, dos criterios fueron utilizados. El primer criterio, respecta a verificar que la lógica descripta en VHDL es efectivamente transferible a un circuito digital dentro de una FPGA, es decir, que el diseño sea sintentizables. El segundo, respecta a la validación de que el comportamiento del módulo implementado es el esperado. Para este proceso, se realizaron simulaciones de comportamiento y, en algunos casos, se plantearon modelos computacionales para contrastar los resultados obtenidos.

4.2. Scrambler de banda base

En este subsistema se realiza la aleatorización de la trama de entrada. Para eso se genera una secuencia pseudo-aleatoria (PRBS) con el polinomio generador expuesto en (2.1) que luego se opera bit a bit con dicha trama. El registro con el que se genera la PRBS es inicializado con la secuencia expuesta en (2.2) cada vez que se procesa una nueva trama.

Este subsistema fue implementado con la capacidad de soportar todas las configuraciones posibles del modulador sin la necesidad de ser configurado.

Interfaces

Las interfaces del módulo *Scrambler de banda base* se exponen en la Tabla 4.1.

La trama de entrada de este módulo consiste en un BBFRAME sin procesar, compuesto por el encabezado de banda base (BBHEADER) seguido por un campo de datos (DATAFIELD) y, eventualmente, un campo de relleno PADDING tal como se expone en la Figura 2.2. Por otra lado, la trama de salida es un BBFRAME.

Tabla 4.1: Interfaces del módulo *Scrambler de banda base*

Nombre	Tipo	Interfaz	Nota
BUS_WIDTH	`natural`	Genérico	Ancho de bus de datos Valor defecto := 8
clk	`std_logic`	Entrada	
rst	`std_logic`	Entrada	Activo en alto
DATAFIELD_TDATA	`std_logic_vector[7:0]`	Entrada	Ver BUS_WIDTH
DATAFIELD_TLAST	`std_logic`	Entrada	Fin de trama
DATAFIELD_TVALID	`std_logic`	Entrada	
DATAFIELD_TREADY	`std_logic`	Salida	
BBFRAME_TDATA	`std_logic_vector[7:0]`	Salida	Ver BUS_WIDTH
BBFRAME_TLAST	`std_logic`	Salida	Fin de trama
BBFRAME_TVALID	`std_logic`	Salida	
BBFRAME_TREADY	`std_logic`	Entrada	

Flujo de datos

Para controlar el flujo de datos se optó por utilizar una máquina de estados. Dicho flujo se resume en la lectura (recepción), procesamiento (aleatorización) y escritura (envío) de los datos. La máquina de estados implementada se expone en la Figura 4.1.

Las funciones de lectura y escritura de datos se implementaron en dos estados dado que resultó más simple.

La máquina de estados se inica en el estado *rd_init* donde se inicializa el proceso de lectura de datos indicando que el módulo se encuentra listo para procesar. La máquina se encuentra en este estado sólo durante un ciclo de reloj dado que sólo se realiza dicha inicialización.

En el estado *rd_idle* se espera hasta que se indique que en la entrada hayan datos válidos. Una vez que se encuentran datos válidos disponibles, una secuencia de acciones se llevan a cabo: dichos datos son almacenados para su posterior procesamiento, la señal que indica el fin de la trama es almacenada, se activa la habilitación del procesamiento y se activa el cambio de estado.

En el estado *scrambler*, se mantiene la habilitación del procesamiento de los datos hasta que la aleatorización haya concluido. Para detectar la finalización del procesamiento se implementó un contador que se decrementa por cada ciclo de reloj donde se haya mantenido la habilitación del procesamiento. Una vez finalizado el conteo, se deshabilita el procesamiento y la máquina pasa al siguiente estado.

En el estado *wr_init* se inicializa el proceso de escritura de los datos y, en caso de ser necesaria, la reconfiguración del sistema de procesamiento. En esta inicialización, se mapean los datos ya aleatorizados a la salida propiamente indicando que estos son válidos y, si corresponde, se indica el fin de la trama y se habilita la reconfiguración del sistema de procesamiento. La máquina se encuentra en este estado sólo durante un ciclo de reloj.

Finalmente, en el estado *wr_idle* se espera hasta que el siguiente módulo pueda leer los datos. Una vez enviados, la máquina pasa al estado *rd_init* para inicializar la siguiente lectura comenzando nuevamente con el ciclo descripto.

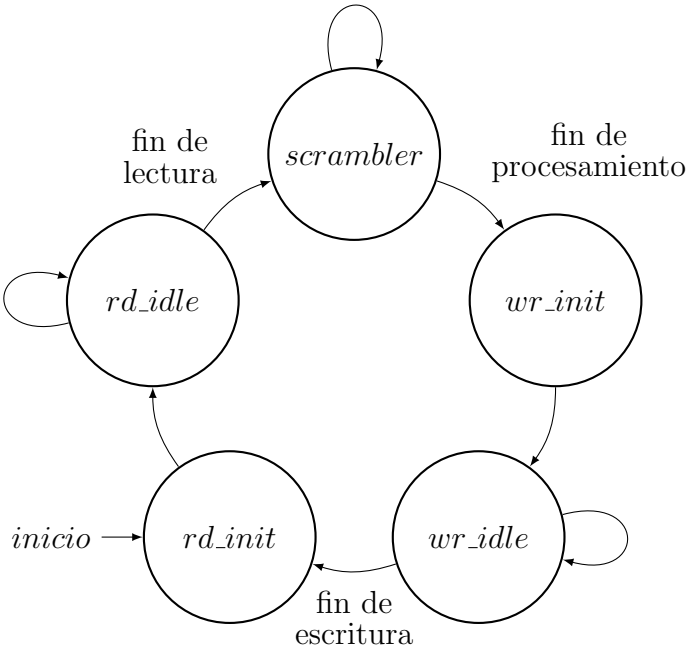


Figura 4.1: Máquina de estados del flujo de datos del *Scrambler de Banda Base*

Procesamiento

En cuanto al procesamiento de los datos, la aleatorización se realiza con un registro de desplazamiento siguiendo la implementación sugerida en el Estándar expuesta en la Figura 2.3. En cuanto al temporizado, se implementó de forma que por cada ciclo de reloj se produzca un deplazamiento del registro aleatorizando un sólo bit de la trama de entrada obteniendo en una tasa de aleatorización de 1 bit por ciclo de reloj. Sin embargo, si contemplamos los tiempos mínimos requeridos de escritura y lectura se obtiene que la tasa de procesamiento de este módulo es de 8 bits por 12 ciclos de reloj.

Pruebas

Para verificar el correcto comportamiento del módulo se realizaron las siguientes pruebas:

P1.1 Comprobar que la secuencia pseudo-aleatoria generada es consistente con la especificada en el Estándar.

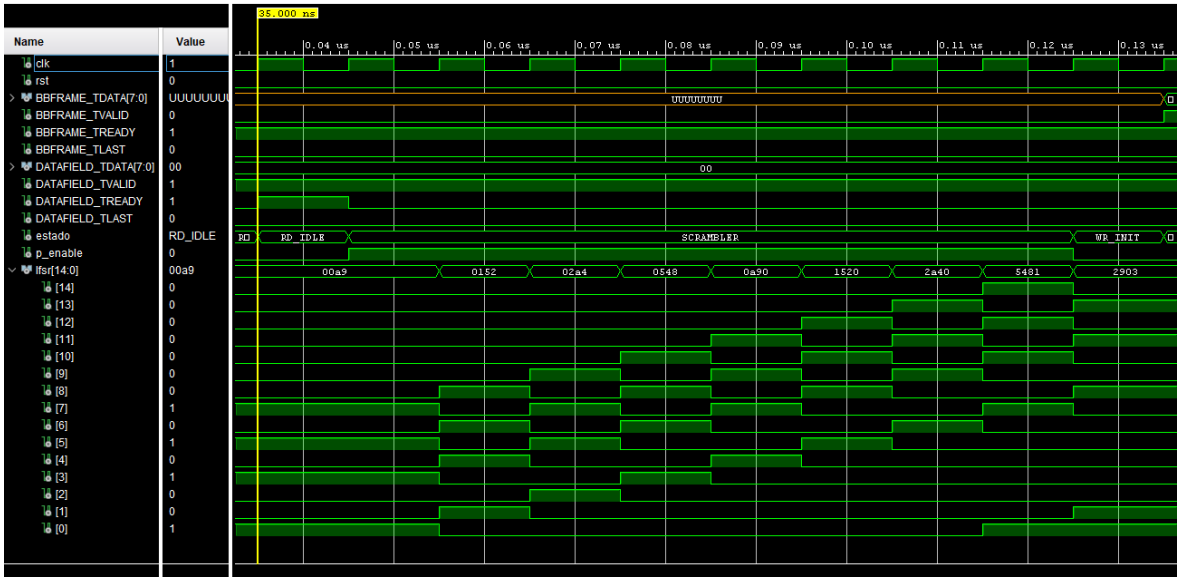
P1.2 Comprobar que se realiza el procesamiento (XOR) entre la trama de entrada y la secuencia pseudo-aleatoria.

P1.3 Comprobar que el registro de la secuencia pseudo-aleatoria se reinicializa con la secuencia de inicialización cuando se inicia una nueva trama.

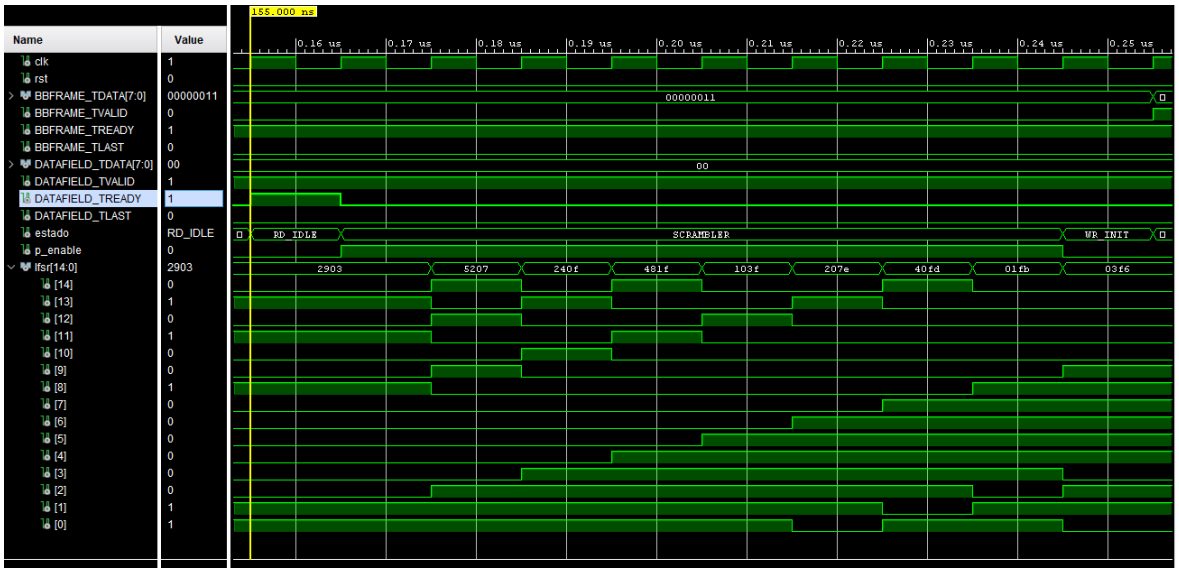
Respecto a la Prueba **P1.1**, se sometió al módulo a un flujo constante de datos nulos. De esta manera, del procesamiento (XOR) entre la trama de entrada nula y la secuencia pseudo-aleatoria se obtiene dicha secuencia. En la Figura 4.2 se expone la simulación bajo los condiciones mencionadas. Como se puede observar, el registro se inicia con la secuencia de inicialización deseada y se realiza el desplazamiento del registro por cada ciclo de reloj. En cuanto a la salida de la secuencia, se contrastaron los primeros 32 bits generados por la secuencia contra un modelo computacional obteniendo los mismos resultados.

Para realizar Prueba **P1.2**, se sometió al módulo a un flujo constante de datos cuyos bits pares eran cero y los impares unos. De esta manera, la trama de salida debiera resultar igual a obtenida en la Prueba **P1.1** con los bits pares invertidos. En la Figura 4.3 se expone la simulación bajo los condiciones mencionadas. Como se puede observar, la salida cumple con lo esperado si se contrasta con la Figura 4.2b. En cuanto a la salida de la secuencia, se contrastaron los primeros 32 bits procesados contra un modelo computacional obteniendo los mismos resultados.

Para realizar Prueba **P1.3**, se sometió al módulo a un flujo constante de datos cuyos bits pares eran cero y los impares, unos indicando siempre el fin de la trama. En la Figura 4.4 se expone la simulación bajo los condiciones mencionadas. Como se puede observar, la señalización del fin de trama y la reconfiguración del registro se llevan a cabo correctamente.



(a) De 35 ns a 135 ns.



(b) De 155 ns a 255 ns.

Figura 4.2: Simulación del *Scrambler de Banda Base* ante un flujo constante de datos nulos a su entrada. Para esta simulación se indicó que los datos a la entrada eran válidos y que el siguiente módulo estaba disponible para recibirlos en todo momento. Esta simulación fue llevada a cabo como parte de la Prueba **P1.1**.

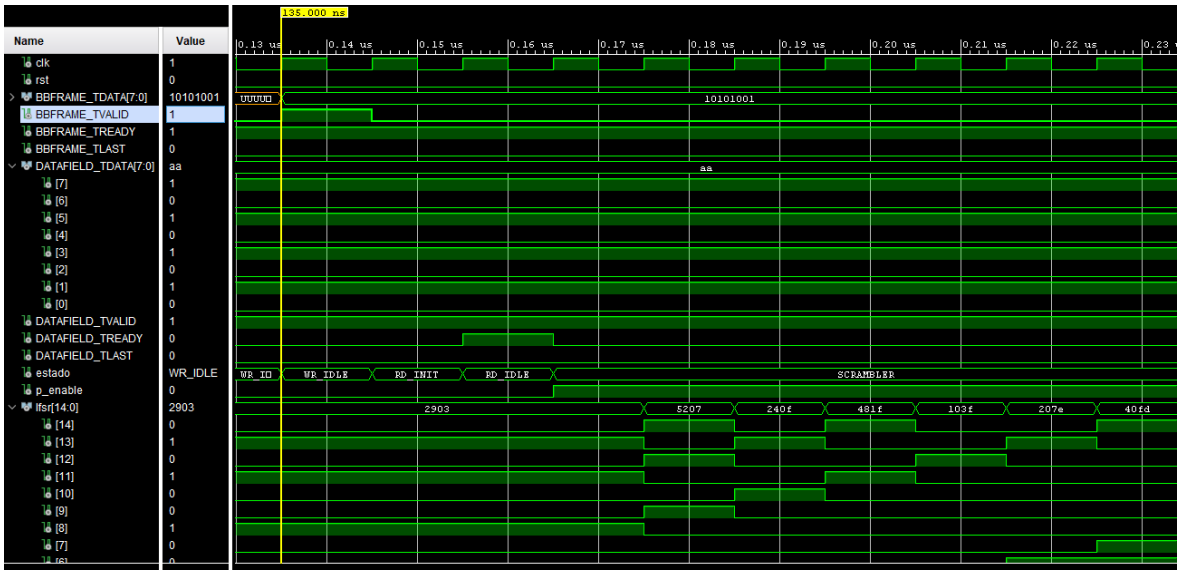


Figura 4.3: Simulación del *Scrambler de Banda Base* ante un flujo constante de datos de entrada cuyos bits pares eran cero y los impares unos. Para esta simulación se indicó que los datos a la entrada eran válidos y que el siguiente módulo estaba disponible para recibirlos en todo momento. Esta simulación fue llevada a cabo como parte de la Prueba **P1.2**.

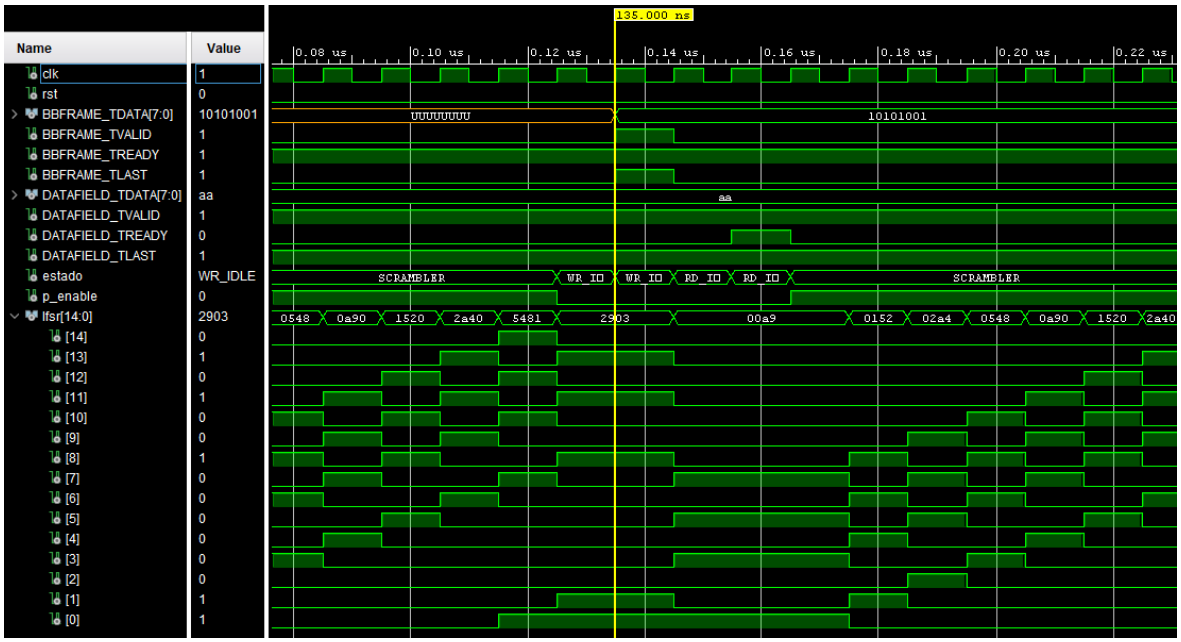


Figura 4.4: Simulación del *Scrambler de Banda Base* ante un flujo constante de datos de entrada cuyos bits pares eran cero y los bits impares eran unos. Para esta simulación se indicó que los datos a la entrada eran válidos, que la lectura realizada era el fin de trama y que el siguiente módulo estaba disponible para recibirlos en todo momento. Esta simulación fue llevada a cabo como parte de la Prueba **P1.3**.

4.3. Codificador BCH

En este subsistema se realiza la codificación BCH de forma sistemática de la trama de entrada siguiendo el Procedimiento 2.1.

Este subsistema fue implementado de forma tal que pueda ser configurado para soportar todas las configuraciones posibles del modulador. Los parámetros de configuración del modulador que impactan directamente en la configuración de este módulo son la tasa LDPC y el tipo de FECFRAME. Dichos parámetros deben ser indicados al momento de ser instanciado.

Interfaces

Las interfaces del módulo *Codificador BCH* se exponen en la Tabla 4.2.

La trama de entrada este módulo consiste es un BBFRAME, mientras que la trama de salida será dicho BBFRAME seguido por la paridad del BCH (BCHFEC) tal como se expone en la Figura 2.4.

Tabla 4.2: Interfaces del módulo *Codificador BCH*

Nombre	Tipo	Interfaz	Nota
BUS_WIDTH	`natural`	Genérico	Ancho de bus de datos Valor defecto := 8
SIZE	`natural`	Genérico	Tipo de FECFRAME Valor defecto := 0 [1]
RATE	`natural`	Genérico	Tasa FEC Valor defecto := 0 [2]
clk	`std_logic`	Entrada	
rst	`std_logic`	Entrada	Activo en alto
BBFRAME_TDATA	`std_logic_vector[7:0]`	Entrada	Ver BUS_WIDTH
BBFRAME_TLAST	`std_logic`	Entrada	Fin de trama
BBFRAME_TVALID	`std_logic`	Entrada	
BBFRAME_TREADY	`std_logic`	Salida	
BCHFRAME_TDATA	`std_logic_vector[7:0]`	Salida	Ver BUS_WIDTH
BCHFRAME_TLAST	`std_logic`	Salida	Fin de trama
BCHFRAME_TVALID	`std_logic`	Salida	
BCHFRAME_TREADY	`std_logic`	Entrada	

[1] FECFRAME NORMAL := 0, FECFRAME CORTO :=1.

[2] Tasa 1/4 := 0, Tasa 1/3 := 1, Tasa 2/5 := 2, Tasa 1/2 := 3, Tasa 3/5 := 4, Tasa 2/3 := 5, Tasa 3/4 := 6, Tasa 4/5 := 7, Tasa 5/6 := 8, Tasa 8/9 := 9 y Tasa 9/10 := 10.

Flujo de datos

Para controlar el flujo de datos se optó por utilizar una máquina de estados. Dicho flujo se resume en la lectura (recepción), procesamiento (codificación) y escritura (envío) de los datos. La máquina de estados implementada se expone en la Figura 4.5. Las funciones de lectura y escritura de datos se implementó en dos estados dado que resultó más simple.

La máquina de estados se inica en el estado *rd_init* donde se inicializa el proceso de lectura de datos indicando que el módulo se encuentra listo para procesar. La máquina se encuentra en este estado sólo durante un ciclo de reloj dado que sólo se realiza dicha inicialización.

En el estado *rd_idle* se espera hasta que se indique que en la entrada hayan datos válidos. Una vez que se encuentra datos válidos disponibles, una secuencia de acciones se llevan a cabo: dichos datos son almacenados para su posterior codificación, la señal que indica que los datos son los últimos de la trama es almacenada, se activa la habilitación del codificación y la máquina pasa al siguiente estado.

En el estado *coding*, se mantiene la habilitación del procesamiento de los datos hasta que la codificación de éstos haya concluido. Para detectar la finalización del procesamiento se implementó un contador que se decrementa por cada ciclo de reloj donde se haya mantenido la habilitación del procesamiento. Una vez finalizado el conteo, se deshabilita el procesamiento y la máquina pasa al siguiente estado.

En el estado *wr_m_init* se inicializa el proceso de escritura de los datos de entrada. Dado que el código es sistemático, los datos de entrada (el mensaje) son retransmitidos y luego es transmitida la paridad una vez finalizado su cálculo. En esta inicialización, se mapean los datos de entrada a la salida de datos indicando que estos son válidos. La máquina se encuentra en este estado sólo durante un ciclo de reloj dado que sólo se realiza dicha inicialización.

En el estado *wr_m_idle* se espera hasta que el siguiente módulo pueda leer los datos. Una vez enviados, se procede a cambiar al siguiente estado. Dicho estado depende si los datos son los últimos de la trama de entrada. En caso de no serlo, el cálculo de la paridad no se ha completado con lo cual el siguiente estado es *rd_init* para inicializar la siguiente lectura. Caso contrario, el siguiente estado es *wr_p_init* para inicializar la escritura de la paridad ya calculada.

En el estado *wr_p_init* se mapea a la salida de datos los bits de paridad correspondientes indicando que estos son válidos y, si corresponde, se indica el fin de la trama y se reconfigura el sistema de procesamiento. Para poder calcular si efectivamente son los últimos datos de la paridad, se empleó un contador. La máquina se encuentra en este estado sólo durante un ciclo de reloj dado que sólo se realiza dicha inicialización.

En el siguiente estado *wr_p_idle*, se espera hasta que el siguiente módulo pueda leer

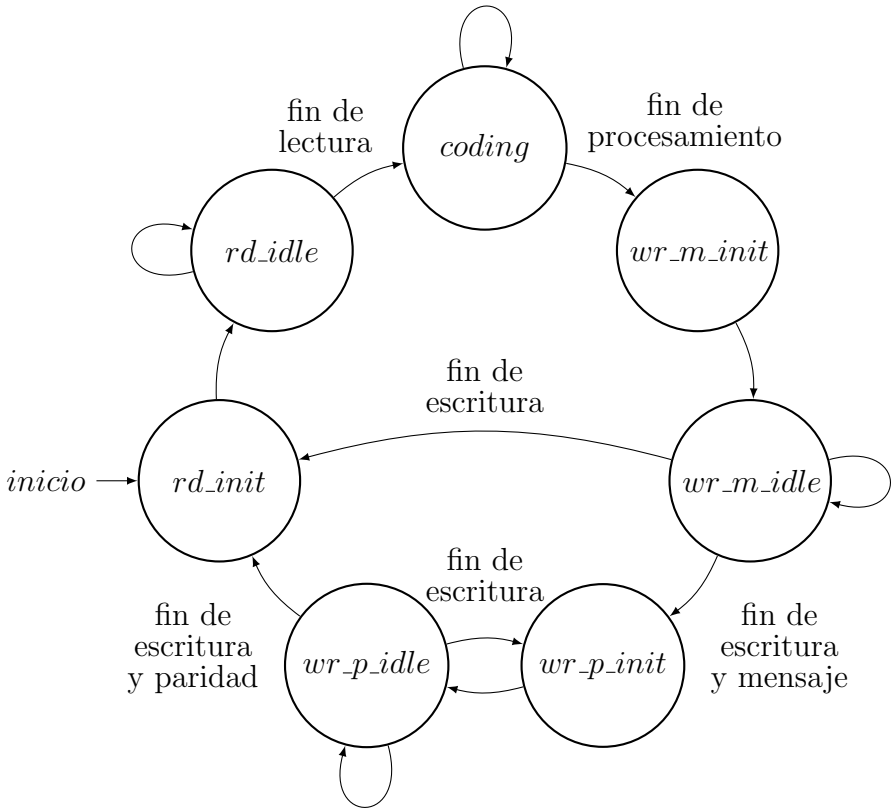


Figura 4.5: Máquina de estados del control de flujo de datos del *Codificador BCH*

los datos. Una vez enviados, se cambia al siguiente estado, que puede ser *wr_p_init* si no se concluyó con la escritura de la paridad o *rd_init*, en caso que se haya terminado, para inicializar la lectura de la siguiente trama de entrada comenzando nuevamente con el ciclo descripto.

Codificación

Para realizar la codificación se debe seguir el Procedimiento 2.1 donde se necesita emplear el polinomio generador del código, que no se expone explicítamente en el Estándar. Como se explicó en el Cap. 2, los polinomios generadores se obtienen de multiplicar los t primeros polinomios expuestos en la Tabla 2.3 donde t es la cantidad de errores que corrige el código. En las especificaciones de los parámetros del sistema FEC expuestos en la Tabla 2.2, se puede observar que el codificador BCH puede corregir 8, 10 y 12 errores dependiendo de tipo del FECFRAME y de la tasa de codificación. Analizando todas las posibles combinaciones se obtiene que existen solamente 4 polinomios generadores. Dichos polinomios generadores fueron calculados y verificados [28] se exponen en la Tabla 4.3.

Además, en dicho procedimiento, se expone que se debe calcular la paridad y luego concatenar el mismo al mensaje a codificar para formar la palabra de código. El

Tabla 4.3: Todos los posibles polinomios generadores BCH calculados según la especificación del Estándar. Los polinomios se expresan en hexadecimal para utilizar una notación mas compacta.

Frame	t	Polinomio generador
Normal	8	11C07255F712797BD19FC6D7504F9662B
Normal	10	160150CEDFC2A331F6A785703EFD12301B8BB6591
Normal	12	14E260E83845C511C50CF2CD8DC350889034785F7660255E7
Corto	12	14062DBEA9869B262CD23A39069528FE7D7D11905A5

cálculo de la paridad consiste en calcular el resto de la división entre el mensaje desplazado y el polinomio generador. Existen circuitos que con registros de desplazamiento realizan divisiones y permiten obtener el resto, pudiendo ser adaptados para codificar sistemáticamente. En la Figura 4.6 se expone una implementación de un codificador con un registro de desplazamiento de R etapas, donde R es el grado del polinomio generador [29][30][31].

En cuanto al procesamiento de los datos, se implementó un registro de desplazamiento siguiendo la implementación expuesta en la Figura 4.6. En cuanto al temporizado, se implementó de forma que por cada ciclo de reloj se produzca un deplazamiento del registro realizando la división de ese bit obteniendo una tasa de codificación 1 bit por ciclo de reloj. Por otra parte, si contemplamos los tiempos de escritura y lectura de la trama de entrada se adicionarían mínimo 4 ciclos de reloj para 8 bits procesados. Finalmente, si tomamos el tiempo mínimo de escritura de la paridad obtendríamos 2 ciclos de reloj por cada bit paridad.

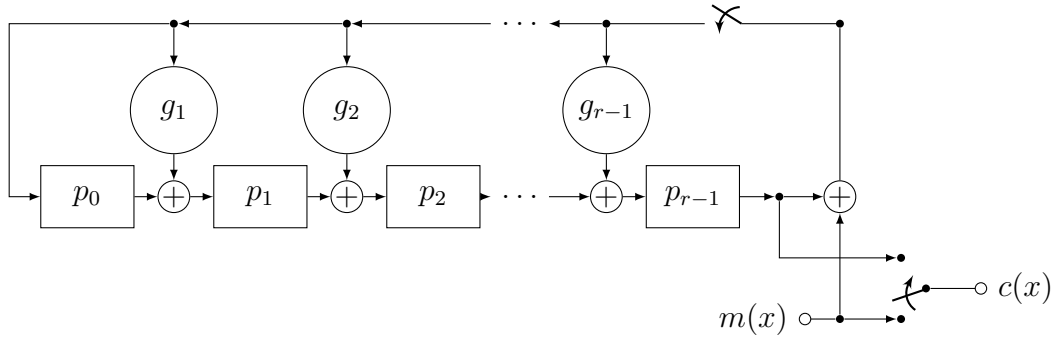


Figura 4.6: Codificador con un registro de desplazamiento de R etapas.

Pruebas

Para verificar el correcto comportamiento del módulo, se proponen las siguientes pruebas que no se realizaron por limitaciones del alcance:

P2.1 Comprobar que la lógica del sistema codificación funciona correctamente.

Para realizar Prueba **P2.1**, se encontró que comprobar el correcto funcionamiento de un registro de 192 etapas es difícil de depurar. Así se decidió iniciar con implementar un código que requiera mas registros, sin modificar la lógica del procesamiento. De esta manera, se optó por realizar inicialmente pruebas con un BCH (7,4), luego con un BCH (15,11) y finalmente con un BCH (16200,16008) correspondiente a FECFRAME NORMAL tasa 1/4 (12-errores de correción).

Para facilitar la depuración y poder enfocar el esfuerzo en la verificación de la lógica de procesamiento se adaptó la máquina de estados del control de flujo y el ancho de los buses de datos de entrada y salida. Se optó por modificar la máquina de forma que se ingrese con el mensaje en una sola lectura y que a la salida se escriba la paridad de una sola vez. Para eso, se eliminaron los estados *wr_m_init* y *wr_m_idle* y se modificó el estado *wr_p_init* de forma que mapee toda la paridad en simultáneo a la salida.

En la Figura 4.7 se expone el resultado de la simulación obtenida de someter al módulo modificado para efectuar el código BCH (7,4) ante el mensaje 1001. Para realizar dicha codificación primero se debió obtener el polinomio generador que resulta $x^3 + x + 1$ [29]. Como se observa, conociendo el polinomio generador y basándose en la implementación genérica expuesta en la Figura 4.6 se puede observar que el registro de desplazamiento funciona correctamente. Mas allá de esto, se sometió al módulo a otros mensajes obteniendo resultados que contrastados con un modelo computacional se concluyó que eran consistentes. Este mismo procedimiento fue repetido para la implementación de BCH (15,11) obteniendo nuevamente resultados consistentes con el modelo computacional. Finalmente, no se logró contrastar los resultados de las simulaciones para el código BCH(16200,16008).

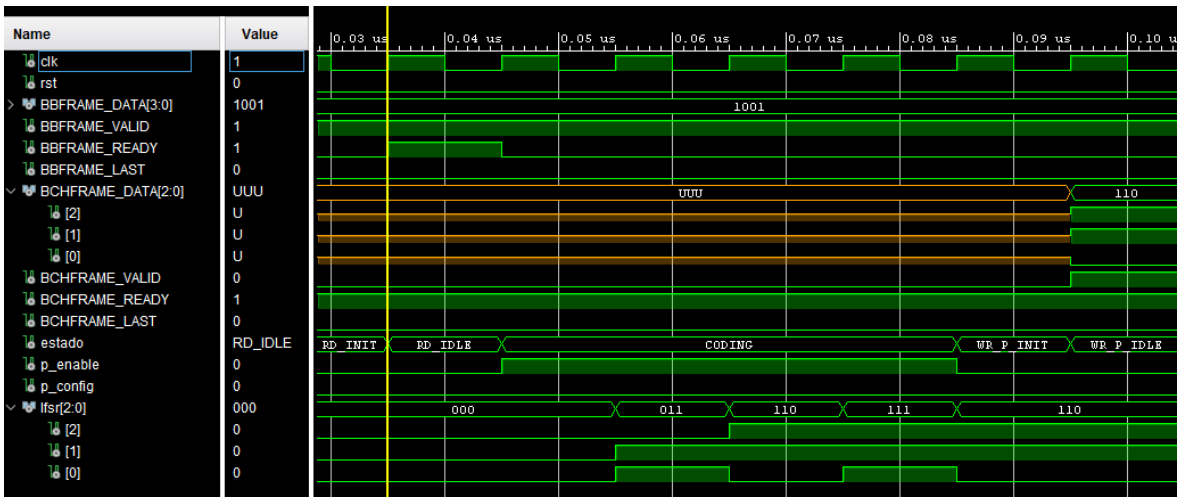


Figura 4.7: Simulación del módulo *Codificador BCH* modificado para que realizara la codificación BCH (7,4). Se sometió al módulo modificado en la entrada al mensaje 1001 indicando que este dato era válido en todo momento. Como se observa, la salida obtenida es 110 que consiste en la paridad que se debe agregar al mensaje para obtener la palabra de código (1 001 110). Este resultado fue contrastado con un modelo computacional siendo consistente. Esta simulación fue realizada en contexto del Prueba **P2.1**.

4.4. Codificador LDPC

En este subsistema se realiza la codificación LDPC de forma sistemática de la trama de entrada siguiendo el Procedimiento 2.2.

Este subsistema fue implementado únicamente para la tasa 4/5 y FECFRAME CORTO. Esta configuración es la única que presenta una tabla de posiciones paridad (ver Procedimiento 2.2) con todas las filas de igual tamaño facilitando la lógica de implementación. Mas allá, por cuestiones de limitación, este subsistema no fue implementado en su totalidad.

Interfaces

Las interfaces del módulo *Codificador LDPC* se exponen en la Tabla 4.4.

La trama de entrada este módulo consiste es un BBFRAME seguido por la paridad del BCH (BCHFEC). La trama de salida consiste en dichos campos seguidos por la paridad del LDPC (LDPCFEC) formando un FECFRAME sin entrelazar tal como se expone en la Figura 2.4.

Tabla 4.4: Interfaces del módulo *Codificador LDPC*

Nombre	Tipo	Interfaz	Nota
BUS_WIDTH	`natural`	Genérico	Ancho de bus de datos Valor defecto := 8
SIZE	`natural`	Genérico	Tipo de FECFRAME Valor defecto := 1 [1]
RATE	`natural`	Genérico	Tasa FEC Valor defecto := 7 [2]
clk	`std_logic`	Entrada	
rst	`std_logic`	Entrada	Activo en alto
BCHFRAME_TDATA	`std_logic_vector[7:0]`	Entrada	Ver BUS_WIDTH
BCHFRAME_TLAST	`std_logic`	Entrada	Fin de trama
BCHFRAME_TVALID	`std_logic`	Entrada	
BCHFRAME_TREADY	`std_logic`	Salida	
FECFRAME_TDATA	`std_logic_vector[7:0]`	Salida	Ver BUS_WIDTH
FECFRAME_TLAST	`std_logic`	Salida	Fin de trama
FECFRAME_TVALID	`std_logic`	Salida	
FECFRAME_TREADY	`std_logic`	Entrada	

[1] FECFRAME NORMAL := 0, FECFRAME CORTO :=1.

[2] Tasa 1/4 := 0, Tasa 1/3 := 1, Tasa 2/5 := 2, Tasa 1/2 := 3, Tasa 3/5 := 4, Tasa 2/3 := 5, Tasa 3/4 := 6, Tasa 4/5 := 7, Tasa 5/6 := 8, Tasa 8/9 := 9 y Tasa 9/10 := 10.

Flujo de datos

Para controlar el flujo de datos se optó por utilizar una máquina de estados. Dicho flujo se resume en la lectura (recepción), procesamiento (codificación) y escritura (envío) de los datos. La máquina de estados implementada se expone en la Figura 4.8. Las funciones de lectura y escritura de datos se encontró que era más simple de implementar en dos estados. Por otra parte, la codificación se realiza en dos estados de procesamiento separados. Un estado donde se realiza el almacenamiento de los bits del mensaje y el otro donde se realiza el procesamiento entre la paridad, es decir, el último paso del Procedimiento 2.2. Se debieron implementar en dos estados estas funciones dado que para el último estado se requiere haber procesado la trama de entrada por completo.

La máquina de estados se inica en el estado *rd_init* donde se inicializa el proceso de lectura de datos indicando que el módulo se encuentra listo para procesar. La máquina se encuentra en este estado sólo durante un ciclo de reloj dado que sólo se realiza dicha inicialización.

En el estado *rd_idle* se espera hasta que se indique que en la entrada hayan datos válidos. Una vez que se encuentra datos válidos disponibles, una secuencia de acciones se llevan a cabo: dichos datos son almacenados para su posterior codificación, la señal que indica que los datos son los últimos de la trama es almacenada, se activa la habilitación del codificación y la máquina pasa al siguiente estado.

En el estado *storage*, se mantiene la habilitación el almacenamiento de los datos en la paridad hasta que se haya concluido. Para detectar la finalización del almacenamiento se implementó un contador que se decrementa por cada ciclo de reloj donde se haya mantenido la habilitación del procesamiento. Una vez finalizado el conteo, se deshabilita el procesamiento y la máquina pasa al siguiente estado.

En el estado *wr_m_init* se inicializa el proceso de escritura de los datos de entrada. Dado que el código es sistemático, los datos de entrada (el mensaje) son retransmitidos y luego es transmitida la paridad una vez finalizado su cálculo. En esta inicialización, se mapean los datos de entrada a la salida de datos indicando que estos son válidos. La máquina se encuentra en este estado sólo durante un ciclo de reloj dado que sólo se realiza dicha inicialización.

Finalmente, en el estado *wr_m_idle* se esperará hasta que el siguiente módulo pueda leer los datos. Una vez envíado el dato, se procede a cambiar al siguiente estado. Dicho estado dependerá de si los datos enviados eran los últimos de la trama de entrada. En caso de no serlos, el cálculo de la paridad no se ha completado con lo cual se pasará al estado *rd_init* para inicializar la siguiente lectura, repitiendo el ciclo descripto anteriormente. Caso contrario, se pasará al estado *scrambler* para realizar el último paso de la codificación.

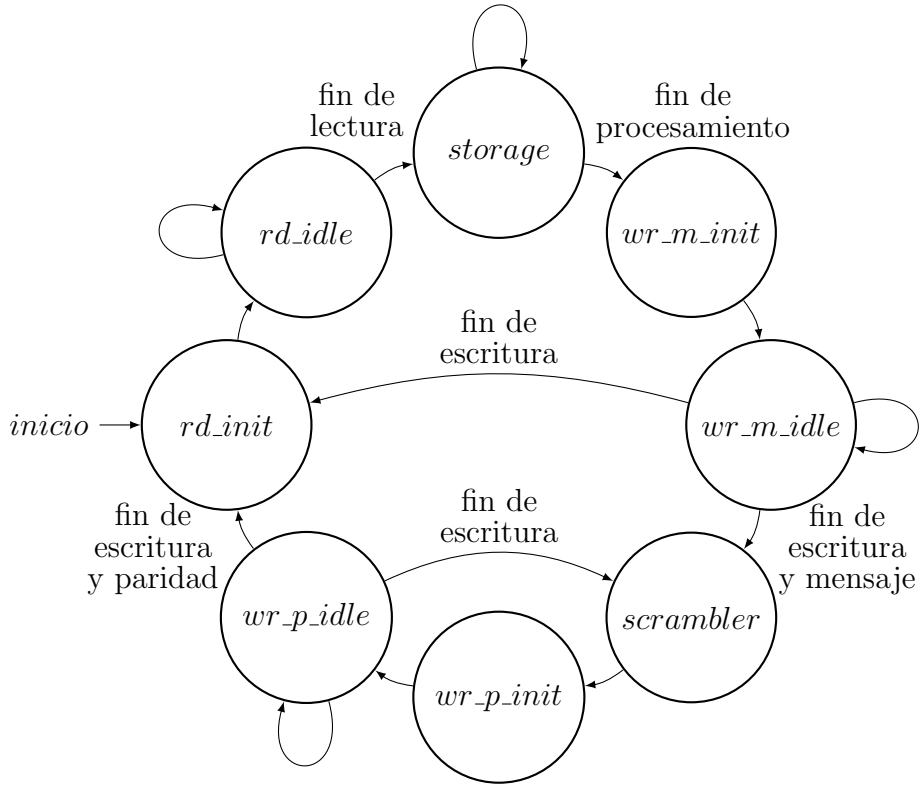


Figura 4.8: Máquina de estados del control de flujo de datos del *Codificador LDPC*

El estado *scrambler* se dejó reservado para futuros trabajos dado que no se logró concluir con este procesamiento. Así, inmediatamente se cambia al siguiente estado.

En el estado *wr_p_init* se mapea a la salida de datos los bits de paridad correspondiente, se indica que dichos datos son válidos y, si corresponde, se indica que el fin de la trama. Para poder calcular si efectivamente es el último dato, se empleo un contador que cuenta la cantidad de datos escritos. Dado que es sólo un estado de inicialización, luego de un ciclo de reloj el estado cambiará al siguiente.

En el siguiente estado *wr_p_idle*, se esperará hasta que el siguiente módulo pueda leer el dato. Una vez enviados los datos al siguiente módulo se cambiará al estado *wr_p_init* para continuar con la escritura de la paridad o al estado *rd_init*, en caso que se haya terminado la escritura de la paridad, para inicializar la siguiente lectura.

Codificación

Para realizar la codificación se debe seguir el Procedimiento 2.2. En dicho procedimiento, se expone que para formar la palabra de código se debe calcular la paridad almacenando los bits de mensaje en determinadas posiciones de la paridad y que luego, se realice sobre toda la paridad de forma secuencial la acumulación de los bit de paridad sobre la siguiente posición. Dichos procesos fueron separados en dos estados, dado que para el último paso se requiere haber acumulado todo el mensaje.

La primer parte del procesamiento indica que se deben almacenar determinados bits del mensaje en determinadas posiciones de la paridad indicadas en una tabla en el Estándar. Cada columna de dicha tabla es utilizada por cada 360 bits de mensaje, sin embargo, se debe realizar un desplazamiento de esas posiciones por cada bit procesado. Este desplazamiento fue implementadó sobre el registro de la paridad para optimizar la utilización de recursos.

Pruebas

Para verificar el correcto comportamiento del módulo, se proponen las siguientes pruebas que no se realizaron por limitaciones del alcance:

P3.1 Comprobar que se realiza el almacenamiento del mensaje en las posiciones correctas del registro de la paridad.

P3.2 Comprobar que se realiza el procesamiento sobre la paridad.

P3.3 Comprobar que se reconfigura el sistema de procesamiento ante una nueva trama.

4.5. Mapeador de bits

En este subsistema se realizar la conversión de bits a símbolos de la trama de entrada siguiendo la constelación determinada para cada modulación.

Este subsistema fue implementado únicamente para realizar el mapeo a la constelación QPSK. Más allá de esta limitación, este módulador soporta cualquier otra configuración del modulador sin la necesidad de ser configurado.

Interfaces

Las interfaces del módulo *Mapeador de bits* se exponen en la Tabla 4.5. Se utilizó la señal TLAST dado que si bien no tiene impacto en este módulo, si lo tiene en el siguiente. Además, se dividió la salida de datos en parte real e imaginaria dado que luego de la modulación la salida consiste en símbolos en vez de bits que requieren ser representados en fase y cuadratura.

En cuanto al ancho del bus de datos, se fijó en 8 bit porque es el tamaño mínimo posible de *AXI4-Stream* y todas las tramas posibles son perfectamente fraccionables por esta cantidad facilitando la implementación. Si bien el bus de datos de escritura es de 8 bit, sólo los 2 bit son necesarios para representar los datos de salida. Esto se debe a que la salida de este módulo entrega un símbolo por escritura.

La trama de entrada este módulo consiste es un FECFRAME, mientras que la trama de salida es un XFECFRAME.

Tabla 4.5: Interfaces del módulo *Mapeador de bits*.

Nombre	Tipo	Interfaz	Nota
BUS_WIDTH	`natural`	Genérico	Ancho de bus de datos Valor defecto := 8
MODU	`natural`	Genérico	Modulación Valor defecto := 2 [1]
clk	`std_logic`	Entrada	
rst	`std_logic`	Entrada	Activo en alto
FECFRAME_TDATA	`std_logic_vector[7:0]`	Entrada	Ver BUS_WIDTH
FECFRAME_TLAST	`std_logic`	Entrada	Fin de trama
FECFRAME_TVALID	`std_logic`	Entrada	
FECFRAME_TREADY	`std_logic`	Salida	
XFECFRAME_TDATA_IM	`std_logic_vector[7:0]`	Salida	Rama Cuadratura Ver BUS_WIDTH
XFECFRAME_TDATA_RE	`std_logic_vector[7:0]`	Salida	Rama Fase Ver BUS_WIDTH
XFECFRAME_TLAST	`std_logic`	Salida	Fin de trama
XFECFRAME_TVALID	`std_logic`	Salida	
XFECFRAME_TREADY	`std_logic`	Entrada	

[1] QPSK := 2, 8PSK := 3, 16APSK := 4 y 32APSK := 5.

Flujo de datos

Para controlar el flujo de datos se optó por utilizar una máquina de estados. Dicho flujo se resume en la lectura (recepción) y escritura (envío). La máquina de estados implementada se expone en la Figura 4.9. Las funciones de lectura y escritura de datos se implementaron en dos estados dado que resultó más simple. La conversión de bit a símbolo se realiza en el momento de mapear la salida de datos. De esta manera, cada vez que se realiza un proceso de escritura se está transmitiendo un símbolo.

La máquina de estados se inica en el estado *rd_init* donde se inicializa el proceso de lectura de datos indicando que el módulo se encuentra listo para realizar el mapeo. La máquina se encuentra en este estado sólo durante un ciclo de reloj dado que sólo se realiza dicha inicialización.

En el estado *rd_idle* se espera hasta que se indique que en la entrada hayan datos válidos. Una vez que se encuentra datos válidos disponibles, una secuencia de acciones se llevan a cabo: dichos datos son almacenados para su posterior mapeo, la señal que indica que los datos son los últimos de la trama es almacenada y la máquina pasa al siguiente estado.

En el estado *wr_init* se inicializa el proceso de escritura de los datos y de selección de bits a mapear de los datos leídos. En esta inicialización, se mapean de a un símbolo a la salida de datos indicando que estos son válidos y si corresponde se indica el fin de la trama. La máquina se encuentra en este estado sólo durante un ciclo de reloj.

Finalmente, en el estado *wr_idle* se espera hasta que el siguiente módulo pueda leer los datos. Una vez enviados, en caso que queden bits a mapear, la máquina pasa al estado *wr_init*, sino pasa a *rd_init* para inicializar la siguiente lectura comenzando nuevamente con el ciclo descripto.

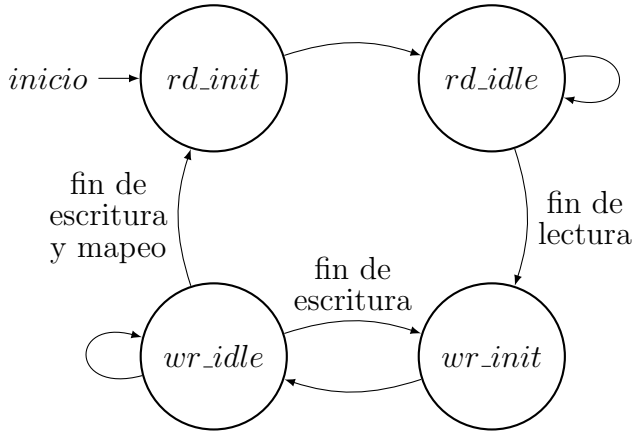


Figura 4.9: Máquina de estados del control de flujo de datos del *Mapeador de bits*.

Mapeo

En cuanto al procesamiento de los datos, el mapeo de los bits al símbolo corrspondiente se realiza con codificación Gray convencional y asignación absoluta (sin codificación diferencial) tal como se expone en la Figura 2.6a.

Pruebas

Para verificar el correcto comportamiento del módulo, se realizaron las siguientes pruebas:

P4.1 Comprobar que el mapeo a la constelación se realice correctamente.

P4.2 Comprobar que se indicara correctamente el fin de trama.

Para verificar las condiciones propuestas en la Prueba **P4.1** y la Prueba **P4.2**, se realizó un test integrador. Se sometió al módulo a un flujo constante de datos que exigiera utilizar todos los símbolos de la constelación indicando que son los últimos de la trama. En la Figura 4.10 se expone la simulación bajo los condiciones mencionadas. Como se puede observar, si bien la señal de fin de trama de la entrada está activa en todo momento, la señal de fin de trama a la salida se activa únicamente cuando

se transmite el último símbolo. Por otra parte, dado que la secuencia de entrada era 00101101, la salida debieran ser los símbolos de la constelación recorridos en sentido anti-horario $(+1,+1)$ $(-1,+1)$ $(-1,-1)$ $(+1,-1)$ representados en (I,Q). A diferencia de los módulos anteriores, la salida de datos en este caso requiere emplear una representación donde se puedan representar valores negativos; mas precisamente los valores posibles son 1, 0 y -1. Se optó por representar cada valor con dos bits de señales tipo signed tal como se exponen las últimas señales de la simulación. En consecuencia, los símbolos representados en *signed* son $(01,01)$ $(11,01)$ $(11,11)$ $(01,11)$ correspondiendose con el resultado de la simulación.

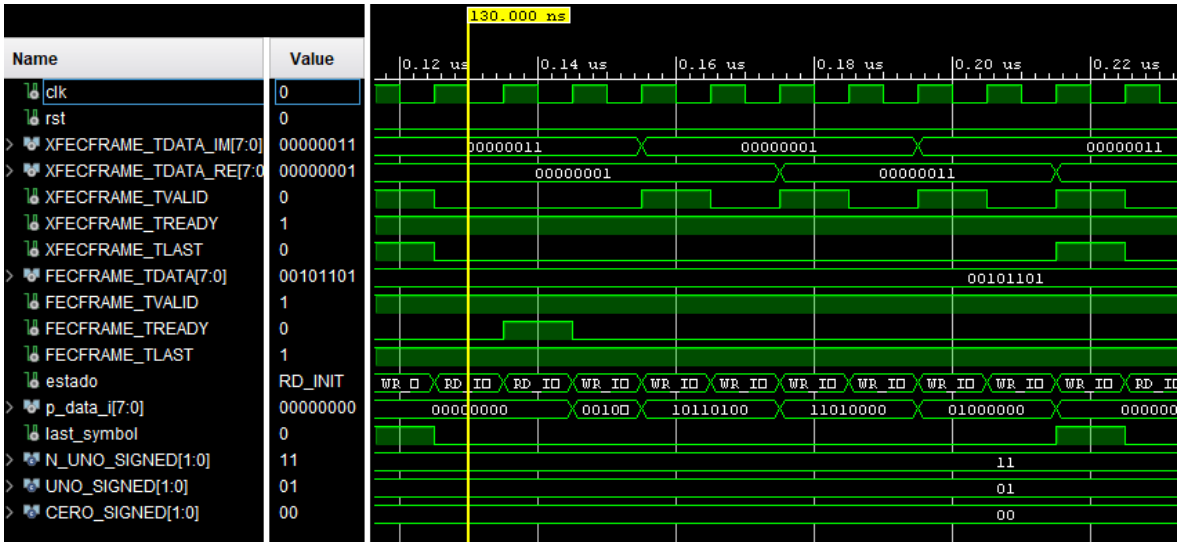


Figura 4.10: Simulación del *Mapeador de bits* ante un flujo constante con la secuencia de entrada 00100111 donde el bit de la izquierda es el MSB. Para esta simulación se indicaron que los datos a la entrada eran válidos, que era el fin de la trama de entrada y que el siguiente módulo estaba disponible para recibirlos en todo momento. Esta simulación fue llevada a cabo como parte de la Prueba **P4.1** y la Prueba **P4.2**.

4.6. Señalización de capa física

En este subsistema se realiza la generación del encabezado de capa física. La generación del encabezado requiere codificar y procesar campos de datos que indican sobre la configuración del modulador con respecto a la modulación, tasa, tipo de FECFRAME y utilización de pilotos. Además se debe realizar la modulación (π/2-BPSK) del encabezado.

Este subsistema fue implementado con la capacidad de soportar todas las configuraciones posibles del modulador debiendo ser configurado para un tal fin al momento de ser instanciado.

Interfaces

Las interfaces del módulo *Señalización de Capa Física* se exponen en la Tabla 4.6. Se optó por utilizar interfaces *AXI4-Stream* para la comunicación de este módulo por motivos de estandarización [27]. Se utilizó la señal TLAST dado que si bien no tiene impacto en este módulo si lo tiene en el siguiente. Además, se dividió la salida de datos en parte real e imaginaria dado que luego de la modulación (π/2-BPSK) la salida consiste en símbolos que requieren ser representados en fase y cuadratura.

En cuanto al ancho del bus de datos, se fijó en 8 bit porque es el tamaño mínimo posible de *AXI4-Stream* y es consistente con los módulos *Mapeador de bits* y *Generador de Pilotos*. Si bien el bus de datos de escritura es de 8 bit, sólo los 2 bit son necesarios para representar los datos de salida. Esto se debe a que la salida de este módulo entrega un símbolo por escritura.

Tabla 4.6: Interfaces del módulo *Señalización de Capa Física*.

Nombre	Tipo	Interfaz	Nota
BUS_WIDTH	`natural`	Genérico	Ancho de bus de datos Valor defecto := 8
SIZE	`natural`	Genérico	Tipo de FECFRAME Valor defecto := 1 [1]
MODCOD	`natural`	Genérico	Campo MODCOD Valor defecto := 8 [2]
PILOTS	`natural`	Genérico	Tipo de FECFRAME Valor defecto := 1 [3]
clk	`std_logic`	Entrada	
rst	`std_logic`	Entrada	Activo en alto
PLHEADER_TDATA_IM	`std_logic_vector[7:0]`	Salida	Rama Cuadratura Ver BUS_WIDTH
PLHEADER_TDATA_RE	`std_logic_vector[7:0]`	Salida	Rama Fase Ver BUS_WIDTH
PLHEADER_TLAST	`std_logic`	Salida	Fin de trama
PLHEADER_TVALID	`std_logic`	Salida	
PLHEADER_TREADY	`std_logic`	Entrada	

[1] FECFRAME NORMAL := 0, FECFRAME CORTO :=1.

[2] Ver Tabla 2.8.

[3] Sin piloto := 0, Con piloto := 1.

Flujo y generación de datos

Para controlar el flujo de datos se optó por utilizar una máquina de estados. Dicho flujo se resume en la generación del encabezado y la escritura (envío) del mismo. La máquina de estados implementada se expone en la Figura 4.11. La función escritura de datos se implementó en dos estados dado que resultó más simple. En cuanto a la generación del encabezado, se implementó en tres estados para acortar los caminos de la lógica combinacional.

En el estado PLS (32,6) se realiza la codificación (32,6) de la generación del campo PLS. Para eso se realiza el producto punto entre el vector binario formado por la concatenación de los campos MODCOD y el MSB del TYPE (ya conocidos) con la matriz generadora (binaria) especificada en (2.5). este cálculo se realiza en un ciclo de reloj por lo que una vez realizada dicha codificación se pasa al siguiente estado.

En el estado PLS (64,7) se toma la salida del codificador (32,6) y se genera el campo PLS. Para esto a los bits impares del campo PLS se le asigna secuencialmente la salida del codificador (32,6) mientras que a los bits pares se realiza una xor entre el bit anterior y el LSB del campo TYPE. Este cálculo se realiza en un ciclo de reloj por lo que una vez realizada la codificación se pasa al siguiente estado.

En el estado *scrambler*, se realiza la xor bit a bit entre la secuencia (2.6) y el campo PLS. Además se concatena el campo SOF y el campo PLS procesado para así formar los bits del encabezado de capa física.

En el estado *wr_init* se inicializa el proceso de escritura de los datos. En esta inicalización, se modula bit a bit el encabezado indicando que el símbolo a la salida es válido y, si corresponde, se indica el fin del encabezado. Para la selección del bit del encabezado a modular se implementó un contador que cuenta la cantidad de símbolos escritos. La máquina se encuentra en este estado sólo durante un ciclo de reloj.

Finalmente, en el estado *wr_idle* se espera hasta que el siguiente módulo pueda leer el símbolo del encabezado. Una vez envíado, la máquina pasa al estado *wr_init* para la siguiente escritura del encabezado de capa física.

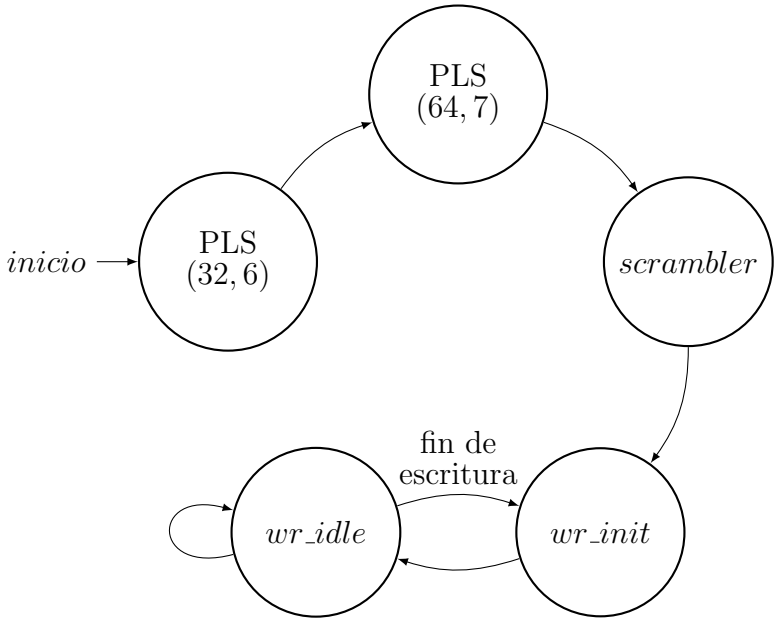


Figura 4.11: Máquina de estados del control de flujo de datos del *Señalización de capa física*.

Pruebas

Para verificar el correcto comportamiento del módulo, se realizaron las siguientes pruebas:

P5.1 Comprobar que la codificación (32,6) se realiza correctamente

P5.2 Comprobar que la codificación (64,7) se realiza correctamente

P5.3 Comprobar que el procesamiento (XOR) de campo PLS con la secuencia dada se realiza correctamente.

P5.4 Comprobar que la modulación $\pi/2$-BPSK se realiza correctamente.

Para verificar las condiciones propuestas en la Prueba **P5.1**, **P5.2** y **P5.3** se realizó un único test integrador. Se sometió al módulo a la generación del encabezado para la tasa 4/5 y FECFRAME CORTO indicando que el siguiente módulo estaba disponible para lectura en todo momento. Se decidió generar el encabezado para dichas configuraciones dado que son las seleccionadas en la implementación del módulo *Codificador LDPC*. En la Figura 4.12 se expone la simulación bajo los condiciones mencionadas. Como se puede observar, la salida del codificador (64,7) (señal *pls_code*) es A 5A5 A5A 5A5 A5A 5A5 y el PLHEADER generado es 18D 2E8 2D4 382 66C FF6 E78 85F. Este resultado fue contrastado contra el resultado obtenido realizando la codificación de forma analítica y con un módulo computacional resultando consistentes.

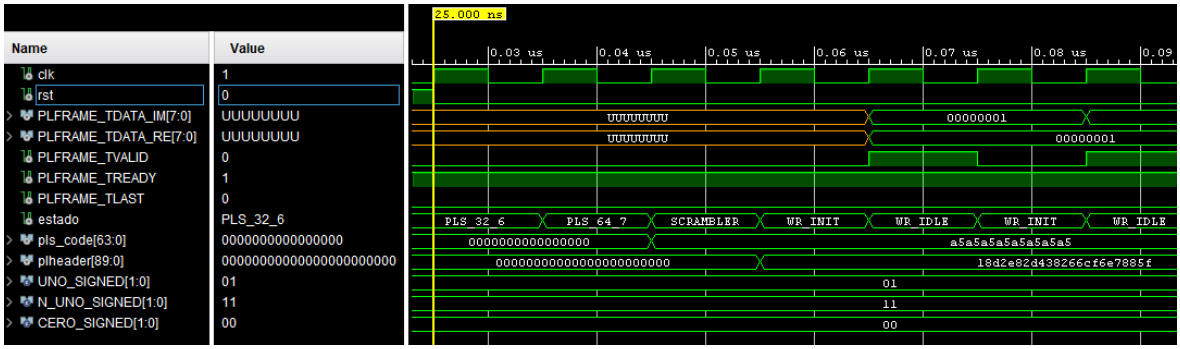


Figura 4.12: Simulación del *Señalización de Capa Física* de la generación del encabezado para la tasa 4/5 y FECFRAME CORTO indicando que el siguiente módulo estaba disponible para lectura en todo momento. Esta simulación fue llevada a cabo como parte de la Prueba **P5.1**, **P5.2** y **P5.3**.

Para verificar las condiciones propuestas en la Prueba **P5.4** se sometió al módulo a la generación del encabezado indicando que el siguiente módulo estaba disponible para lectura en todo momento. El encabezado comienza con el campo SOF que consiste en una secuencia fija $0110001101 \ldots 0010$, por lo que con observar la modulación de los primeros 4 bits se comprenden todos los símbolos posibles de la constelación. Los 4 símbolos obtenidos para los primeros 4 bits serían $(+1,+1)$ $(+1,-1)$ $(-1,-1)$ $(-1,+1)$ representados en (I,Q). En la Figura 4.13 se expone la simulación bajo los condiciones mencionadas. Se observa que la salida para los primeros 4 símbolos son $(01,01)$ $(01,10)$ $(10,10)$ $(10,01)$. Esto se debe a que dado que se requeria utilizar una representación capaz de representar valores negativos; más precisamente los valores 1, 0 y -1. De esta manera, se decidió representar cada valor con dos bits de señales del tipo *signed* tal como se exponen las últimas señales de la simulación. En consecuencia, se concluye que la modulación es correcta.

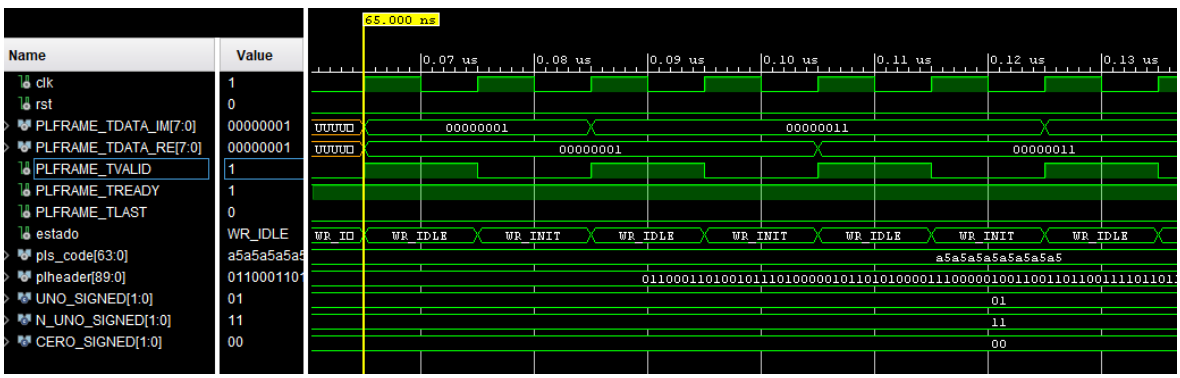


Figura 4.13: Simulación del *Señalización de Capa Física* para verificar la correcta modulación del encabezado de capa física. Esta simulación fue llevada a cabo como parte de la Prueba **P5.4**.

4.7. Generador de pilotos

En este subsistema se genera el BLOCK PILOT que consiste en 36 de símbolos sin modular.

Este subsistema es insensible a las distintas configuraciones del modulador.

Interfaces

Las interfaces del módulo *Generador de Pilotos* se exponen en la Tabla 4.7.

Tabla 4.7: Interfaces del módulo *Generador de Piloto*

Nombre	Tipo	Interfaz	Nota
BUS_WIDTH	`natural`	Genérico	Ancho de bus de datos Valor defecto := 8
clk	`std_logic`	Entrada	
rst	`std_logic`	Entrada	Activo en alto
PILOT_TDATA_IM	`std_logic_vector[7:0]`	Salida	Rama Cuadratura Ver BUS_WIDTH
PILOT_TDATA_RE	`std_logic_vector[7:0]`	Salida	Rama Fase Ver BUS_WIDTH
PILOT_TLAST	`std_logic`	Salida	Fin de trama
PILOT_TVALID	`std_logic`	Salida	
PILOT_TREADY	`std_logic`	Entrada	

Se utilizó la señal TLAST dado que si bien no tiene impacto en este módulo si lo tiene en el siguiente. Además se dividió la salida de datos en parte real e imaginaria dado que la salida consiste en símbolos que requieren ser representados en fase y cuadratura.

En cuanto al ancho del bus de datos, se fijó en 8 bit porque es el tamaño mínimo posible de *AXI4-Stream* y es consistente con la salida del módulo *Mapeador de bits* y *Señalización de Capa Física*. Si bien el bus de datos es de 8 bit, sólo los 2 bit menos significativos son datos válidos. Esto se debe a que la salida de este módulo entrega de a un símbolo por escritura.

Flujo de datos

El flujo de datos consiste en controlar la señal de fin de trama indicando que se transmitieron los 36 símbolos sin modular. Para eso se empleó un contador que cuenta la cantidad de símbolos que faltan escribir.

Pruebas

Se realizó una única prueba para comprobar la correcta señalización de fin de trama. Para ello, se sometió al módulo a una simulación donde el siguiente módulo estuviese listo para realizar la lectura del símbolo piloto. En la Figura 4.14 se exponen los resultados de dicha de simulación cumpliendo con esperado.

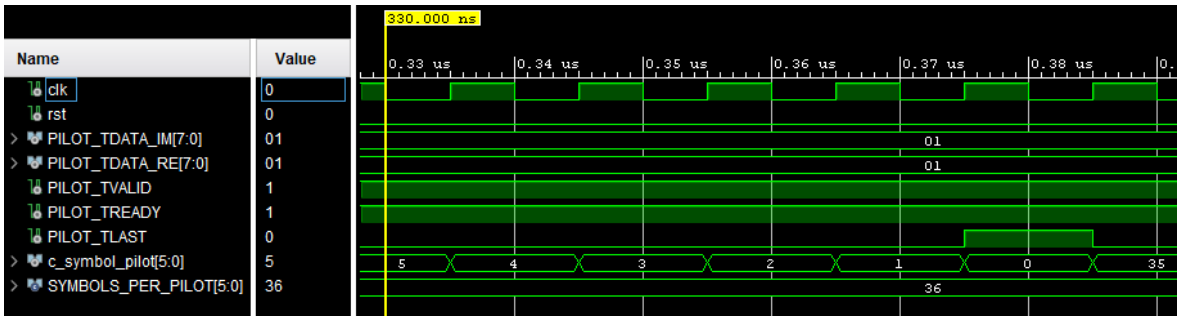


Figura 4.14: Simulación del *Generador de pilotos* para comprobar la correcta señalización del fin de trama luego de escribir 36 símbolos.

4.8. Control de transmisión

En este subsistema se realiza la integración del encabezado de capa física, el XFECFRAME y las señales piloto para formar el PLFRAME (ver Figura 2.8). En esta etapa, además, se realiza la conversión de símbolos a muestras.

Este subsistema fue implementado con la capacidad de soportar todas las configuraciones posibles del modulador. Sin embargo, este debe ser configurado en función de la utilización (o no) de señales piloto.

Interfaces

Las interfaces del módulo *Control de transmisión* se exponen en la Tabla 4.8. Se utilizó la señal TLAST sólo en la entrada dado que facilita la implementación de este módulo.

En cuanto al ancho del bus de datos, se fijó en 8 bit porque es el tamaño mínimo posible de *AXI4-Stream* y es consistente con la salida de los módulos de entrada *Mapeador de bits*, *Señalización de Capa Física*, *Generador de Pilotos*. Si bien el bus de datos es de 8 bit, sólo los 2 bit menos significativos son datos válidos tal como se expone en las respectivas secciones de los módulos de entrada.

Tabla 4.8: Interfaces del módulo *Control de transmisión.*

Nombre	Tipo	Interfaz	Nota
BUS_WIDTH	`natural`	Genérico	Ancho de bus de datos Valor defecto := 8
PILOTS	`natural`	Genérico	Tipo de FECFRAME Valor defecto := 1 [1]
clk	`std_logic`	Entrada	
rst	`std_logic`	Entrada	Activo en alto
XFECFRAME_TDATA_IM	`std_logic_vector[7:0]`	Entrada	Rama Cuadratura Ver BUS_WIDTH
XFECFRAME_TDATA_RE	`std_logic_vector[7:0]`	Entrada	Rama Fase Ver BUS_WIDTH
XFECFRAME_TLAST	`std_logic`	Entrada	Fin de trama
XFECFRAME_TVALID	`std_logic`	Entrada	
XFECFRAME_TREADY	`std_logic`	Salida	
PLHEADER_TDATA_IM	`std_logic_vector[7:0]`	Entrada	Rama Cuadratura Ver BUS_WIDTH
PLHEADER_TDATA_RE	`std_logic_vector[7:0]`	Entrada	Rama Fase Ver BUS_WIDTH
PLHEADER_TLAST	`std_logic`	Entrada	Fin de trama
PLHEADER_TVALID	`std_logic`	Entrada	
PLHEADER_TREADY	`std_logic`	Salida	
PILOT_TDATA_IM	`std_logic_vector[7:0]`	Entrada	Rama Cuadratura Ver BUS_WIDTH
PILOT_TDATA_RE	`std_logic_vector[7:0]`	Entrada	Rama Fase Ver BUS_WIDTH
PILOT_TLAST	`std_logic`	Entrada	Fin de trama
PILOT_TVALID	`std_logic`	Entrada	
PILOT_TREADY	`std_logic`	Salida	
PLFRAME_TDATA_IM	`std_logic_vector[7:0]`	Salida	Rama Cuadratura Ver BUS_WIDTH
PLFRAME_TDATA_RE	`std_logic_vector[7:0]`	Salida	Rama Fase Ver BUS_WIDTH
PLFRAME_TVALID	`std_logic`	Salida	
PLFRAME_TREADY	`std_logic`	Entrada	

[3] Sin piloto := 0, Con piloto := 1.

Flujo de datos

Para controlar el flujo de datos se optó por utilizar una máquina de estados. En cuanto al control del flujo de datos de entrada, trata sobre la selección de las interfaces *AXI4-Stream* de los módulos de entrada para adquirir los datos y conformar el PLFRAME. La máquina de estados implementada se expone en la Figura 4.15. Esta máquina de estados podría interpretarse como un multiplexor donde el estado es el control de selección y las entradas/salidas son las señales *AXI4-Stream*.

La máquina de estados comienza en el estado IDLE donde se encuentra hasta que todos los módulos de entrada tengan datos válidos. Este estado se implementó para controlar la transmisión de los datos principalmente.

En el estado PLHEADER se seleccionan las señales del módulo *Señalización de Capa Física*. En este estado se permanece hasta que se indica el fin de trama.

El estado XFECFRAME se seleccionan las señales del módulo *Mapeador de bits*. En cuanto a las condiciones de cambio de estado depende de si en la configuración adoptada para el modulador se utilizan señales piloto. En caso de no utilizarse, este estado queda a la espera al fin de trama para pasar al estado PLHEADER. En caso de utilizar señales piloto, se comprobaran dos condiciones: la primera, el fin de trama donde el siguiente estado corresponde al PLHEADER, y la segunda, la escritura de 90 SLOTs de campo XFECFRAME donde el siguiente estado corresponde al PILOTS. En caso de darse ambas condiciones en simultáneo, se prioriza la primera condición.

En el estado PILOTS se seleccionan las señales del módulo *Generador de Pilotos*. En este estado se permanece hasta que se indica el fin de trama pasando nuevamente al estado XFECFRAME.

En cuanto al control de datos de salida, la habilitación de transmisión es el estado de la máquina descripta anteriormente. Una vez que la máquina sale del estado inicial IDLE, se mantiene la habilitación de transmisión de forma indefinida. La salida de este módulo consiste en muestras del símbolo leído del respectivo módulo de entrada. Se optó por trabajar con una tasa de transmisión de 1 símbolo por cada 8 ciclos de reloj y con una representación de 4 muestras por símbolo. Así, las muestras transmitidas consisten en el símbolo leído seguido por un *padding* de 3 ceros cuya separación temporal es 2 ciclos de reloj. La validación de lectura se realiza en conjunto con la validación de escritura.

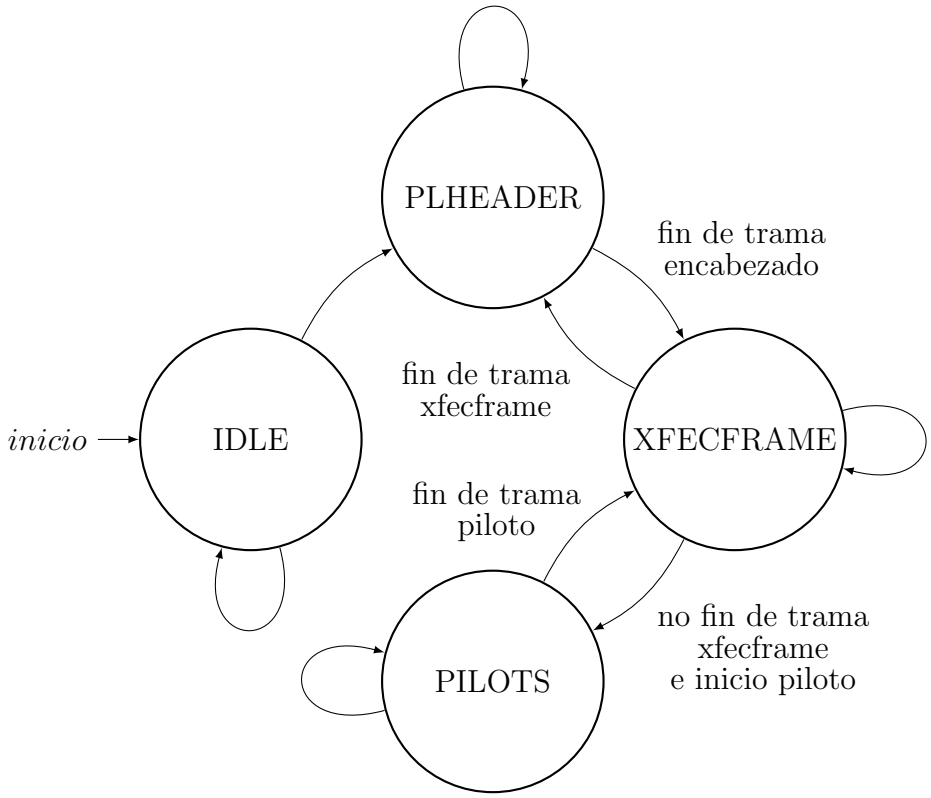


Figura 4.15: Máquina de estados del control de flujo de datos de entrada del *Control de transmisión.*

4.9. Conformador de pulso

En este subsistema se realiza la conformación de forma del pulso. El filtro empleado es un Raíz Coseno Elevado (SRRC).

Este subsistema fue implementado para un factor de caída (0.25), sin embargo, la solución implementada permite la adaptación a otra configuración.

Esto se debe a que se utilizó un IPCore de Xilinx denominado FIR Compiler. Este módulo debe ser instanciado una vez para la rama en fase y otra para la rama en cuadratura. Dicho módulo implementa el filtro deseado especificando el vector de coeficientes del filtro. Los coeficientes del SRRC con factor de caída 0.25 fueron obtenidos con una herramienta de diseño de filtros. No se realizaron pruebas sobre este módulo dado que el IPCore no puede ser modificado mas allá de su dicha configuración.

4.10. Mezclador en cuadratura

En este módulo se realiza la conversión los pulsos conformados en fase y cuadratura en banda base a frecuencia intermedia. Para esto, se debe realizar la multiplicación de ambas ramas por el coseno y -seno de la frecuencia intermedia y sumar ambas señales

resultantes.

Este subsistema fue implementado con la capacidad de soportar todas las configuraciones posibles del modulador sin la necesidad de ser configurado.

Interfaces

Las interfaces del módulo *Mezclador en cuadratura* se exponen en la Tabla 4.9. En cuanto al ancho del bus de datos, se empleo de forma tal que se pueda adaptar al ancho del bus de datos de la salida del *Conformador de pulso*.

Tabla 4.9: Interfaces del módulo *Mezclador en cuadratura*.

Nombre	Tipo	Interfaz	Nota
BUS_WIDTH	`natural`	Genérico	Ancho de bus de datos Valor defecto := 16
clk	`std_logic`	Entrada	
rst	`std_logic`	Entrada	Activo en alto
I_TDATA	`std_logic_vector[15:0]`	Entrada	Ver BUS_WIDTH
I_TVALID	`std_logic`	Entrada	
Q_TDATA	`std_logic_vector[15:0]`	Entrada	Ver BUS_WIDTH
Q_TVALID	`std_logic`	Entrada	
FI_TDATA	`std_logic_vector[15:0]`	Salida	Ver BUS_WIDTH
FI_TVALID	`std_logic`	Salida	

Conversión a frecuencia intermedia

La forma convencional de realizar la conversión a frecuencia intermedia es produciendo sinusiodes y utilizar dos multiplicadores y un sumador. Sin embargo, dado que no hay especificaciones respecto a la frecuencia intermedia, se optó una que permita una implementación mas simple. En caso que dicha frecuencia sea exactamente cuatro veces menor que la frecuencia de muestreo, los valores de las sinusiodes (iniciando con una fase nula) toman simplemente los valores 1, -1 y 0 siguiendo las siguientes secuencias:

$$\cos \Rightarrow 1, 0, -1, 0, 1, 0, -1, 0, 1, 0, -1, \ldots$$

$$-\operatorname{sen} \Rightarrow 0, -1, 0, 1, 0, -1, 0, 1, 0, -1, 0, \ldots$$

De esta manera, podemos ver que salida en frecuencia intermedia sería:

$$i(0), -q(1), -i(2), q(3), i(4), -q(5), -i(6), -q(7), i(8), ...$$

Donde $i(n)$ y $q(n)$ son las salidas de cada uno de los filtros conformadores de la rama fase y cuadratura respectivamente.

Como se puede observar, esto significa que en este caso toda la modulación puede reemplazarse por un bloque que alternadamente deje pasar una muestra de la rama fase o cuadratura y que a su vez alterne su polaridad de dos en dos.

Capítulo 5

Conclusiones

> *"La experiencia es un peine que te lo dan cuando te quedas pelado."*
>
> — Oscar Natalio "Ringo" Bonavena

Se realizó el estudio del estándar de segunda generación de transmisión de video satelital DVB-S2. Complementando dicho estudio, se realizó un análisis de factibilidad técnico de la implementación del sistema de transmisión propuesto en dicho estándar y las implicancias sobre su diseño para logar comunicaciones de 1 Gbps sin superar los 400 MHz de ancho de banda. Se pudo concluir que la realización de dicho sistema de transmisión en su completitud resulta un trabajo que excede el alcance de este proyecto dado al gran número de sistemas y configuraciones posibles de los mismos. Además, se pudo concluir que la tasa de transmisión deseada inicialmente presenta una complejidad que no solo escapa al alcance de este proyecto sino que también es superior a las tasas típicas según documentos oficiales [23][24].

El diseño del modulador implementado consistió únicamente en acotar el sistema de transmisión del estándar DVB-S2 a uno con un número de menor de subsistemas y configuraciones respetando siempre las especificaciones de dicho estándar. Dado que se buscaba seguirlo lo más posible, no se realizó un estudio para evaluar el rendimiento y posibles mejoras del sistema de transmisión DVB-S2. Sin embargo, realizando el estudio del estándar se encontraron documentos que proponen modificaciones sobre el sistema como reemplazar el codificador BCH por un Reed-Salomon obteniendo mejores desempeños a costa de una mayor complejidad de implementación [32][33].

Se logró implementar los módulos en VHDL que cumplieran con las funciones de los subsistemas del modulador diseñado de forma exitosa en términos generales. En el caso del filtro conformador de pulso, se prefirió utilizar un IP Core dado que estaba dipsonible en la herramienta utilizada para realizar la síntesis y simulación de los módulos VHDL (Vivado Design Suite). Además, mientras que realizar un modelo básico del filtro resulta trivial, utilizar un módulo completamente verificado y optimizado

permitió concentrar el esfuerzo en la implementación del resto de los subsistemas. En el caso del codificador LDPC, la implementación realizada no resuelve completamente el procedimiento de la codificación dado que se tuvieron complicaciones en obtener un diseño que fuese sintetizable. Las mayores limitaciones en la implementación de este codificador consisten en disponer de un registro lo suficientemente largo como para poder almacenar la paridad, simplificar la lógica de almacenamiento de los bits de paridad para que el requerimiento de recursos no sea excesivo y lograr mayores tasas de procesamiento de las que se pudieran alcanzar con la lógica propuesta dado que es el de menor tasa de procesamiento de los que trabajan con bits. Se propone para futuros trabajos dos posibles alternativas de implementación para atender estas cuestiones. La primera, describir la codificación en Sintesis de Alto Nivel (HLS, por sus siglas en inglés) dado que permite una creación más rápida de IP [34]. En segundo lugar, se propone emplear SoC (Sistema en Chip) FPGA para realizar la codificación LDPC en el procesador, facilitando dicha implementación. Sin embargo, como desventaja de esta última alternativa se encuentra que requiere el esfuerzo adicional de realizar la comunicación entre el procesador y la FPGA propiamente.

Se realizaron simulaciones de comportamiento de los módulos VHDL implementados obteniendo resultados satisfactorios. En algunos casos, adicionalmente se contrastaron los resultados de dichas simulaciones con modelos computacionales obteniendo consistencia entre ambos modelos de forma satisfactoria. Por otra parte, si se propusiera la comercialización o utilización profesional de estos módulos, las pruebas realizadas resultarían escasas. Sin embargo, realizar pruebas del sistema para cada uno de los casos posible representa un trabajo sumamente extenso y que requeriría de pruebas automatizadas presentando una dificultad adicional.

En cuanto al desempeño del modulador, se encuentra que el módulo que limita la tasa de transmisión es el subsistema *Control de Transmisión*, cuya tasa es de 4 muestras por símbolo cada 8 ciclos de reloj. Tomando la frecuencia de reloj utilizada para las simulaciones de comportamiento (100 MHz) se obtendría que la tasa de símbolo del modulador diseñado es de 12.5 Mbd equivalente a una tasa neta de bits de 25 Mbps. Realizando otra lectura, se podría considerar que la limitación está dada por el esquema de modulación dado que con un esquema de mayor orden se podría obtener una mayor tasa de bits. Sin embargo, si se utilizara otra constelación que no sea QPSK, requeriría de la implementación de un descentralazador de bits aumentando el trabajo de implementación. También, se podría obtener un mejor desempeño utilizando una mayor frecuencia de reloj empleando placas con FPGAs de alta gama [35], siempre y cuando los diseños realizados permitan dicho temporizado. Más allá de esto, se puede concluir que el potencial de la tasa de símbolo del modulador implementado es buena en contraste con las típicas teniendo en cuenta las limitaciones de este proyecto.

Bibliografía

[1] ETSI EN 302 307-1: Digital Video Broadcasting (DVB); Second generation framing structure, channel coding and modulation systems for Broadcasting, Interactive Services, News Gathering and other broadband satellite applications; Part 1: DVB-S2. Estándar europeo, European Telecommunications Standards Institute (ETSI), 2014-11. V1.4.1.

[2] 2nd Generation Satellite. Ficha técnica, Digital Video Broadcasting (DVB) Project Office, 2012-08.

[3] Sklar, B. Digital communications, tomo 2. Prentice Hall Upper Saddle River, 2001.

[4] Enabling technologies for SDR: Comparing FPGA and DSP performance, 2006. URL `https://www.bdti.com/MyBDTI/pubs/20061115_sdr06_fpgas.pdf`, accedido: 2018-11-30.

[5] Iridium NEXT uses Xilinx space-grade FPGAs. URL `\https://www.eetindia.co.in/news/article/iridium-next-uses-xilinx-space-grade-fpgas`, accedido: 2018-02-16.

[6] CREONIC. DVB-S2 Demodulator IP Core. URL `https://www.creonic.com/de/ip-core/dvb-s2-demodulator-de/`, accedido: 2018-02-22.

[7] IPRIUM. DVB-S2 Modulator IP Core. URL `https://www.iprium.com/ipcores/id/dvbs2-modulator/`, accedido: 2018-02-22.

[8] DVB-S2 LDPC Decoder. URL `www.wasiela.com/dvb-s2-ldpc-decode`, accedido: 2018-11-28.

[9] ETSI EN 300 421: Digital Video Broadcasting (DVB); Framing structure, channel coding and modulation for 11/12 GHz satellite services. Estándar europeo, European Telecommunications Standards Institute (ETSI). V.1.1.2.

[10] ETSI EN 300 429: Digital Video Broadcasting (DVB); Framing structure, channel coding and modulation for cable systems. Estándar europeo, European Telecommunications Standards Institute (ETSI), 1994-12. Edition 1.

[11] ETSI EN 300 744: Digital Video Broadcasting (DVB); Framing structure, channel coding and modulation for digital terrestrial television. Estándar europeo, European Telecommunications Standards Institute (ETSI), 1997-03. Edition 1.

[12] Press Release: ITU recommends DVB-S. URL `https://www.dvb.org/resources/public/pressreleases/old/pr007_itu_recommends_dvb-s_941129.pdf`, accedido: 2018-02-16.

[13] History of DVB. URL `https://www.dvb.org/about/history`, accedido: 2018-02-22.

[14] ETSI EN 301 210: Digital Video Broadcasting (DVB); Framing structure, channel coding and modulation for Digital Satellite News Gathering (DSNG) and other contribution applications by satellite. Estándar europeo, European Telecommunications Standards Institute (ETSI).

[15] Gallager, R. Low-density parity check codes. Tesis Doctoral, Massachusetts Institute of Technology (MIT), 1963.

[16] Hocquenghem, A. Codes correcteurs d'erreurs. *Chiffres*, **2** (2), 147–56, 1959.

[17] Bose, R. C., Ray-Chaudhuri, D. K. On a class of error correcting binary group codes. *Information and control*, **3** (1), 68–79, 1960.

[18] ETSI EN 301 790: Digital Video Broadcasting (DVB); Interaction channel for satellite distribution systems. Estándar europeo, European Telecommunications Standards Institute (ETSI).

[19] ETSI ETS 300 801: Digital Video Broadcasting (DVB); Interaction channel through Public Switched Telecommunications Network (PSTN)/ Integrated Services Digital Networks (ISDN). Estándar europeo, European Telecommunications Standards Institute (ETSI).

[20] ETSI EN 301 195: Digital Video Broadcasting (DVB); Interaction channel through the Global System for Mobile communications (GSM). Estándar europeo, European Telecommunications Standards Institute (ETSI).

[21] ETSI ES 200 800: Digital Video Broadcasting (DVB); DVB interaction channel for Cable TV distribution systems (CATV). Estándar europeo, European Telecommunications Standards Institute (ETSI).

[22] Recommendation ITU-R SNG.770-1: Uniform operational procedures for satellite news gathering (SNG). Inf. téc., International Telecommunication Union (ITU).

[23] ETSI TR 102 376: Digital Video Broadcasting (DVB) User guidelines for the second generation system for Broadcasting, Interactive Services, News Gathering and other broadband satellite applications; Part I (DVB-S2). Reporte técnico, European Telecommunications Standards Institute (ETSI).

[24] DVB BlueBook A171-1: Digital Video Broadcasting (DVB) Implementation guidelines for the second generation system for Broadcasting, Interactive Services, News Gathering and other broadband satellite applications; Part I (DVB-S2). Reporte técnico, European Telecommunications Standards Institute (ETSI).

[25] SATbroadcasts. DVB-S/S2 Bitrate Calculator. URL `http://www.satbroadcasts.com/DVB-S_Bitrate_and_Bandwidth_Calculator.html`.

[26] ROKS. Bitrate calculator DVB-S2. URL `https://roks-tv.com/pages/dvb_s2`.

[27] AXI UG761. Reference guide, Xilinx Inc., 2011. (v 13.1).

[28] COM-1209: a soft high-speed DVB-S2 BCH code decoder and encoder VHDL source code overview. Inf. téc., ComBlock, 2009.

[29] Proakis, J. G., Salehi, M., Zhou, N., Li, X. Communication systems engineering, tomo 2. Prentice Hall New Jersey, 1994.

[30] El Gouri, R., Ahmed, W. A., Lichioui, A., Hlou, L. Conception and implementation of a bch code on a fpga board. *International Journal of Engineering & Technology*, **2** (4), 293, 2013.

[31] Haykin, S. Communication systems. John Wiley & Sons, 2008.

[32] Iva, B., Krešimir, M., Emil, D. Dvb-s2 model based on reed-solomon outer encoder.

[33] Jokela, T. Performance analysis of substituting dvb-s2 ldpc code for dvb-t error control coding system. En: Broadband Multimedia Systems and Broadcasting, 2008 IEEE International Symposium on, págs. 1–5. IEEE, 2008.

[34] Vivado High-Level Synthesis. URL `https://www.xilinx.com/products/design-tools/vivado/integration/esl-design.html`, accedido: 2018-11-29.

[35] ZCU104 Evaluation Board. User guide, Xilinx Inc., 2018-10. UG1267 (v1.1).

[36] de Lima, E. R. Architecture and algorithms for the implementation of digital wireless receivers in FPGA and ASIC: ISDB-T and DVB-S2 cases. Tesis Doctoral, 2016.

Agradecimientos

"Yo soy yo y mi circunstancia, y si no la salvo a ella no me salvo yo."

— José Ortega y Gasset (Meditaciones del Quijote, 1914)

Agradezco al Instituto Balseiro por haber otorgado la posibilidad de realizar mis estudios en su institución. A aquellas personas que lo componen, que permiten que el IB trascienda con su respetado reconocimiento. Al ingresar al IB esperaba entrar en una aula y sorprenderme, y he tenido la suerte de haber tenido docentes que superaron ampliamente la ya alta expectativa. A todos ellos que con su esfuerzo, su pasión y dedicación hacen que verdaderamente la excelencia académica sea un emblema del IB, mil y un gracias.

El Instituto Balseiro subsiste por el apoyo de distintos entes que a su vez dependen del estado argentino. Es por eso que extiendo mi gratitud a cada uno de las personas que habitan este tan particular rincón del mundo que a través de su aportes han permitido la realización de mi carrera.

Agradezco a quienes permitieron que re-descubriera un poco de pasión en las telecomunicaciones a través de la realización de este proyecto. A mis directores por la predisposición de tutelarme, en particular a Willy por haberme guiado y compartido un lugar en sus oficinas. A Leo que también tuvo la predisposición de colaborar en todo momento. A Adrián y el resto del staff de Satellogic que dieron la chispa inicial para idear el proyecto.

En esta larga y ardua experiencia, han habido muchos altos y muchos bajos. Algunos momentos que han sacado más de una lágrima de felicidad y otros que han tocado fondo, que han dolido y que lograron flaquearme. Sin embargo, la buena compañía que tuve la gran e increíble suerte de tener a mi lado fue la que permitió en mis peores momentos volver a levantar a la cabeza y seguir. A cada uno de los Teleco IB15, gracias por todos los buenos momentos juntos y por tanta ayuda. Y si bien, son los que más han estado y los que más alegrías me han dado, también he tenido la suerte de tener amigos de

otras carreras como El Rodri y Augusto o de otros países como los *cubichi* Raimel y Chang. Inclusive no sólo tener amistades sino el hecho de estar rodeado de personas que en el día a día en un simple acto de convivencia, compartiendo un rato almorzando en los pabellones, reunidos en algún bar, cenando en alguna casa lograron contagiarme alegría en mi vida. Ni hablar de esas coincidencias cósmicas, como haber encontrado a Dana detrás de la barra de la cafetería del Beck, mi novia y mayor soporte y compañía en todo este tiempo.

Agradezco a todos y cada uno de los de mi familia, desde mi Abu por "prestarme un poco de su inteligencia" hasta mi tía y padrino por siempre hacer lo imposible por estar presentes en cada una de las vueltas a Buenos Aires. Agradezco especialmente a Mamá y Papá por su contención, su apoyo, por haberme inculcado sus valores y una infinidad de cosas más que no pueden ser enumeradas. Además, agradezco a esa familia adoptiva, a la familia que pude elegir: mis amigos. He tenido muchísima suerte en mi vida de haberme cruzado con tantas increíbles personas ya desde la primaria y el barrio (Diego, Gonza, Nachito, Orio, Priori), en la secundaria (Ari, Dami, Drach, Gerce, Pato, Timpi) o coincidencias de la vida misma (Chapo, Tomi, Juli y tantos más). Y como todos estamos en este viaje llamado vida, vamos y venimos, estamos y no estamos, así también lo hacen las amistades. No siempre están abiertas las puertas pero cuando lo están, siempre será con la misma alegría.