Universidad Autónoma de Baja California

Facultad de Ingeniería, Arquitectura y Diseño

Maestría y Doctorado en Ciencias e Ingeniería

Implementación de codificador y decodificador de canal utilizando FPGAs para aplicaciones en sistemas de comunicación óptica inalámbrica

TESIS

que para cubrir parcialmente los requisitos necesarios para obtener el

grado de

MAESTRO EN INGENIERIA

Presenta

LUIS ALBERTO LUNA ESPINOSA

Ensenada, Baja California, Diciembre del 2013.

Universidad Autónoma de Baja California

Facultad de Ingeniería, Arquitectura y Diseño

Implementación de codificador y decodificador de canal utilizando FPGAs para aplicaciones en sistemas de comunicación óptica inalámbrica

TESIS

que para cubrir parcialmente los requisitos necesarios para obtener el grado de

MAESTRO EN INGENIERÍA

Presenta

Luis Alberto Luna Espinosa

Aprobada por:

Dr. Juan de Dios Sánchez López
Director de Tesis

Dr. Miguel Enrique Martínez Rosas
Miembro del Comité

Dr. Juan Ivan Nieto Hipólito
Miembro del Comité

Dr. Manuel Moisés Miranda Velasco
Miembro del Comité

Dra. Mabel Vázquez Briseño
Miembro del Comité

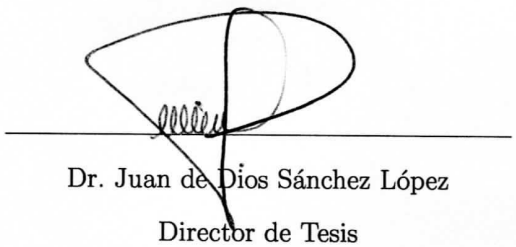
Ensenada, Baja California, Diciembre del 2013.

Resumen de la tesis de **Luis Alberto Luna Espinosa**, presentada como requisito parcial para la obtención del grado de MAESTRO EN INGENIERÍA del programa de Maestría y Doctorado en Ciencias e Ingeniería (MYDCI) de la UABC. Ensenada Baja California, México, Noviembre del 2013.

Implementación de codificador y decodificador de canal utilizando FPGAs para aplicaciones en sistemas de comunicación óptica inalámbrica

En este trabajo de tesis se ha realizado un análisis de los requerimientos que presenta el canal óptico inalámbrico, específicamente la turbulencia atmosférica. Este trabajo plantea la utilización de codificaciones de canal para poder mejorar el desempeño este tipo de enlaces o algún otro con características similares. Diversas técnicas de codificación de canal han sido analizadas, de las cuales se ha seleccionando la codificación convolucional debido a su balance entre complejidad y desempeño. También se describe una relación que permite el fácil el desarrollo de la implementación de codificadores y decodificadores. Adicionalmente se hacen algunas observaciones útiles para futuras implementaciones obtenidas mediante una serie de pruebas con $N_0 = 0$.

Resumen Aprobado por:



Dr. Juan de Dios Sánchez López

Director de Tesis

Palabras Clave: *Codificación de canal, Codigos convolucionales, FPGA, Comunicaciones ópticas inalámbricas (COI)*

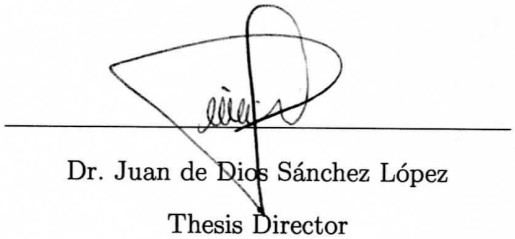
Ensenada, Baja California, Diciembre del 2013.

Abstract of the tesis by **Luis Alberto Luna Espinosa**, submitted as partial requirement for obtaining the degree MASTER in ENGINEERING from the Masters and Doctorate program in Science and Engineering (MYDCI) at UABC. Ensenada Baja California, Mexico, November 2013.

Implementation channel coder and decoder using FPGAs for applications in wireless optical communication systems

In this thesis the requirements of the wireless optic channel has been analized, specificly the atmospheric turbulence. Channel coding has been proposed to improve the performance of this kind of link or another with similar behavior. Several coding techniques has been analized, from which convolutional coding has been selected because of its tradeoff between complexity and performance. Also we describe a relationship that allows an easy development of convolutional coders and decoders. Additionally we make some useful suggestions for future implementations gathered from experiments with $N_0 = 0$.

Abstract Aproved by:



Dr. Juan de Dios Sánchez López

Thesis Director

Key Words: *Channel coding, Convolutional Codes, FPGA, Optical Wireless*

Ensenada, Baja California, December del 2013.

Dedicatoria

A mi esposa, Miriam Maritza Calderon Jauregui, quien me ha apoyado en todo momento siendo paciente y comprensible con mis actividades y decisiones, y poniendo mis pies en la tierra cuando locuras he tratado de hacer.

Gracias por aguantarme.

Agradecimiento Especial

Agradezco especialmente a mi director de tesis, el Dr. Juan de Dios Sánchez López por brindarme su amistad, apoyo, pláticas amenas, consejos, conocimiento y paciencia, por la confianza depositada en mí para formar parte de su grupo de trabajo y permitirme aportar a tan importante proyecto.

A mis sinodales el Dr. Miguel Enrique Martínez Rosas, Dr. Juan Ivan Nieto Hipolito, Dr. Manuel Moises Miranda Velasco y la Dra. Mabel Vázquez Briseño, por su apoyo, consejos, amistad y sus oportunos comentarios.

Agradezco también al Dr. Salvador Villarreal, quien siempre me pudo dedicar unos minutos de su tiempo para realizar oportunas observaciones y esclarecer algunas dudas.

Agradecimientos

Quiero expresar mi agradecimiento:

A los compañeros de maestría de mi generación por su apoyo, moral principalmente, y a los compañeros de las generaciones pasadas que nos mostraron una parte del camino a seguir por medio de sus experiencias, consejos y vivencias. Un agradecimiento especial a Francisco Villalpando y Diego Bustillos, quienes siempre se permitieron dedicarnos algo de su tiempo a pesar de su ajustada agenda para atendernos de manera paciente.

Al Consejo Nacional de Ciencia y Tecnología (CONACyT) por el apoyo económico brindado y a la Facultad de Ingeniería, Arquitectura y Diseño (FIAD) de la Universidad Autónoma de Baja California (UABC), nuestra segunda casa, por las facilidades otorgadas para la realización de éste trabajo.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

Los sistemas de comunicaciones hoy en día son una necesidad casi vital para algunos sectores de la población y la industria. Estos sistemas, en función de su aplicación, deben cumplir con ciertos regímenes de calidad en diversos aspectos. Si la aplicación implica información privilegiada que solo algunos usuarios deben tener, entonces este sistema de comunicaciones requiere un cierto grado de seguridad, o bien, si el sistema requiere un constante y amplio flujo de información, entonces se dice que este requiere una alta tasa de transmisión, tal vez, la información que se envía resulta ser indispensable que llegue sin modificación alguna, entonces se dirá este sistema requiere una baja probabilidad de error o una alta integridad de los datos. En función del escenario es que estos parámetros pueden ser priorizados para satisfacer la necesidad, pero en general si la integridad de los datos no alcanza un determinado nivel, los demás parámetros carecen de sentido, es por ello que todo sistema de comunicaciones implica ciertos métodos, y/o técnicas para asegurar un cierto nivel de integridad. Un sistema de comunicaciones también implica un medio de transmisión imperfecto el cual afecta el desempeño del sistema de comunicaciones al introducir ruido o algunos tipos de perturbaciones y restricciones. Es aquí donde entran un conjunto de métodos y técnicas

que pueden mejorar el desempeño del sistema en alguno o algunos de los parámetros previamente comentados. Este conjunto de métodos y técnicas puede estar integrado por algún tipo de modulación, codificación de canal, estimación del canal, código de línea, demodulación, etc. Cada una de estas técnicas o métodos tienen sus ventajas y desventajas en función del comportamiento del canal y de la aplicación en cuestión, de modo que no existe un conjunto estándar para toda aplicación que sea siempre más efectivo y es por eso que el estudio de estas técnicas sigue teniendo amplio interés. Con lo anterior en mente es que en el presente este trabajo se muestra la implementación en una plataforma FPGA de un código convolucional, que es un tipo de codificación de canal que permite la corrección de errores para mejorar la integridad de los datos. Esta codificación se implementó a partir de componentes modulares (Entidades) con la finalidad de tener aplicación en sistemas de comunicaciones ópticas, ópticas inalámbricos o en alguna otra parte del espectro electromagnético. Además, con el fin de facilitar la implementación de los mismos, se creó un programa (script) en Matlab, el cual a partir de estos componentes modulares crea un codificador y decodificador en función de unos parámetros de control permitiendo crear una amplia variedad de códigos convolucionales. Adicionalmente se realizó un estudio teórico de algunos códigos de línea que permitieran tener una mayor funcionalidad al sistema y se llevó a cabo la implementación de un código Unipolar Retorno a Cero (URZ), el cual facilita la recuperación de la señal de reloj indispensable para la sincronía de los datos.

1.1. Antecedentes

Las telecomunicaciones son una de las bases sobre las cuales descansa el desarrollo de un país. Los sistemas de comunicaciones vía fibra óptica son actualmente la columna vertebral de las redes de comunicaciones de voz y de datos a nivel global. Dichas

comunicaciones son limitadas debido a que los usuarios finales no cuentan con el mismo tipo de canal de transmisión, esto debido a diversos motivos, tales como dificultad para excavar en ciertas zonas o bien el costo de las instalaciones asociadas a la fibra óptica [47, 23], generando con esto el problema de conexión de la ultima milla [47, 27, 2], que se puede ver como un cuello de botella en cuanto a la transferencia de información. Es en este nicho de aplicación donde los sistemas de Comunicaciones Ópticas Inalámbricas (COI) pueden ser utilizadas con ventaja. Otras aplicaciones para los sistemas de COI pueden ser: enlaces de respaldo, enlaces temporales y para casos de contingencia [47]. Es por esto que al comienzo de este siglo, este tipo de sistemas atrajeron el interés debido a las ventajas citadas anteriormente. Sin embargo, la interacción de las ondas electromagnéticas con la atmósfera a frecuencias ópticas es más fuerte que a las frecuencias correspondientes a las microondas [44, 40]. Un haz láser que se propaga a través de la atmósfera puede ver reducida su intensidad debido a fenómenos tales como el esparcimiento y absorción molecular, entre otras causas [19]. Las variaciones en el índice de refracción debido a la turbulencia degenera la calidad del haz láser, distorsionando el frente de fase y modulando aleatoriamente la potencia de la señal [11]. La presencia de nubes pueden impedir completamente el paso del haz, dejando el enlace de comunicaciones inoperante [49]. A la fecha, se han realizado diversas investigaciones sobre la propagación de la luz en la atmósfera y de diferentes técnicas y sistemas para demostrar la factibilidad de enlaces de comunicaciones ópticas inalámbricos sobre trayectorias verticales y horizontales. Un punto importante, es el hecho de que el láser enfoca la energía de la señal en un haz angosto de alrededor de un miliradián [24] lo cual hace que estos sistemas de COI sean más seguros que los enlaces de radiofrecuencia (RF). Por la anterior razón, requiere de menos potencia y no produce interferencia. En resumen las principales ventajas de los sistemas COI son a) no requieren licencias; b) no hay emisión de RF c) no se requieren excavaciones; d) gran ancho de banda (comparable con los sistemas de fibra óptica). En la figura 1.1 se muestra el modelo conceptual

de un sistema de comunicaciones ópticas inalámbrico, donde puede observarse que el haz láser es afectado a lo largo de su trayectoria por variaciones del índice de refracción ocasionadas a su vez por la turbulencia atmosférica [36].

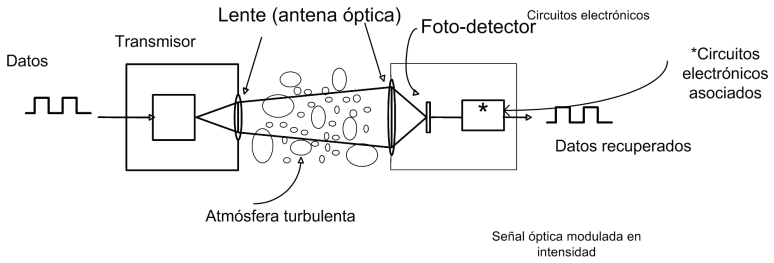


Figura 1.1: Modelo de un enlace de comunicaciones ópticas en el espacio libre atmosférico.

1.2. Tecnologías de acceso.

En las décadas pasadas, el ancho de banda de un solo enlace en las dorsales de las redes ha sido incrementado en casi 1000 veces, gracias al uso de la multicanalización por división de longitud onda (WDM) [10]. Los sistemas de fibra óptica actuales pueden proporcionar capacidades de varios gigabits por segundo al usuario final. Sin embargo, solo el 10 % de los negocios u oficinas, poseen acceso directo a la fibra óptica, de modo que el resto se conecta a ésta por tecnologías de transmisión que emplean cables de cobre ó señales de radio, lo que es conocido como cuello de botella de la última milla [47]. Si bien existen sistemas de comunicaciones de banda ancha basados en las tecnologías DSL o en cable módem, el ancho de banda de este tipo de tecnologías es aun limitado [2]. Los sistemas de RF que utilizan frecuencias de portadoras menores que la de las ondas milimétricas no pueden entregar datos a velocidades especificadas por IEEE 802.3z Gbit Ethernet. Velocidades de 1 Gbps y mayores sólo pueden ser entregadas por haz láser u ondas milimétricas. Sin embargo la tecnología de ondas milimétricas

es mucho menos madura que la tecnología de los láseres [47]. El acceso a las redes de banda ancha basados en comunicaciones ópticas puede ser realizado por medio de redes ópticas pasivas (PONS por sus siglas en inglés) las cuales se basan en el uso de la fibra óptica ó por medio de sistemas de comunicaciones ópticas inalámbricas. La industria de las comunicaciones ópticas inalámbricas ha experimentado un sano crecimiento en la década pasada a pesar de los altibajos de la economía. Lo anterior se debe a las tres principales ventajas sobre otras tecnologías competidoras. En primer lugar, como tecnología de comunicaciones inalámbricas, ocupa un nicho de aplicación. El costo de las comunicaciones ópticas inalámbricas es en promedio de alrededor del 10 % del costo de un sistema de fibra óptica [47]. Se requiere además de sólo unas cuantas horas para su instalación, tiempo similar para establecer un enlace de radioeléctricos (RF), cuando para instalar la fibra óptica puede llevar varios meses. En segundo lugar, los sistemas COI tienen mayor alcance que los sistemas basados en ondas milimétricas. Los sistemas COI pueden cubrir distancias mayores a un Km., cuando sistemas de ondas milimétricas requieren de repetidores para la misma distancia. Además los sistemas de ondas milimétricas son afectados por la lluvia, cuando los sistemas COI son afectados por la niebla [28]. Sin embargo experimentos realizados en Londres, ciudad famosa por su niebla, muestran que la ocurrencia de niebla con la suficiente densidad para hacer caer un enlace con norma de seguridad clase I a 155 Mb/s sobre un km es menor de 0.1 % [28]. Otros estudios realizados en los Estados Unidos han demostrado que en la mayoría de las ciudades de este país, es posible lograr enlaces de 1.25 Gb/s con una disponibilidad del 99 % en distancias promedio de una milla, con láseres de 1W [28]. Esto último es uno de los problemas a resolver, ya que este tipo de láseres es caro y no es seguro para la vista.

1.2.1. Comparación de tecnologías de acceso

Existen diversas alternativas de solución para el problema de la última milla, las que se basan en tres tipos de tecnologías de transmisión: cable de cobre, enlaces de radiofrecuencia y enlaces ópticos inalámbricos. En la tabla 1.1 se hace una comparación de las diferentes tecnologías de acceso, donde se puede observar que las tecnologías de transmisión que presentan más afinidad a los sistemas de fibra óptica, en relación a su velocidad de transmisión de bit, son los sistemas de ondas milimétricas y los sistemas de COI. Los primeros, si bien son una alternativa atractiva, son muy sensibles a la lluvia y la tecnología de ondas milimétricas aún no es lo suficientemente madura [47]. Por otro lado, los sistemas de COI pueden hacer uso de los desarrollos tecnológicos de los sistemas de comunicaciones ópticos basados en fibra óptica. Esta alternativa ha surgido en fechas recientes, como una solución para la conexión de dorsales de fibra óptica en redes metropolitanas. Como se mencionó anteriormente, tales sistemas ofrecen muchas ventajas tales como tener un gran ancho de banda, operación en bandas del espectro electromagnético no reguladas, instalación relativamente rápida, alta directividad, con potenciales aplicaciones en lugares donde no exista la infraestructura de comunicaciones de fibra óptica, en enlaces de sistemas de satélites y como vía alterna de comunicaciones en caso de desastres [2].

1.3. Revisión del estado de arte de los sistemas de comunicaciones ópticos inalámbricos

Debido al prometedor desempeño que pueden jugar los sistemas COI en la solución del problema de la ultima milla se han realizados múltiples estudios orientados a facilitar la implementación de este tipo de sistemas. Algunos estudios están enfocados

Tecnología de acceso	Tasa de transmisión	Distancia límite (típicas)	Comentarios
Fibra óptica	10 Gbps por longitud de onda	Hasta 100 Km sin regeneración	Distancias interoceánicas.
Sistemas COI	100 Mbps a 2.5 Gbps	2 Km (100 Mbps) 1 Km (2.5 Gbps)	Punto a punto y en red. Afectado por la neblina.
Ondas Milimétricas	1.55 Mbps a 1.25 Gbps	5 Km (155 Mbps) 10 Km (1.25 Gbps)	Punto a punto y en red. Afectado por la lluvia.
DSL	1 a 100 Mbps (asimétrico)	6 Km	Compromiso entre velocidad y distancia.
Microondas	75 Mbps	50 Km	Punto a punto y en red.
Wi-Fi	54 Mbps	100 m	Compromiso entre velocidad y distancia.
Cable-Módem	1 a 10 Mbps	10 Km con amplificadores	Compromiso entre velocidad y distancia.

Tabla 1.1: Comparación de tecnologías acceso

a la caracterización de los efectos que provocan los estados meteorológicos en el sistema, como bien se ha reportado en varios trabajos [20, 32, 26]. La neblina es uno de los factores de atenuación más grande que presentan los sistemas COI con respecto a otros fenómenos naturales como la lluvia. También se han hecho variantes sobre dichos estudios como el caso del trabajo de Muhammad [32] en el cual se consideran diferentes tipos de neblina (continental y marítima), presentando la neblina continental una atenuación del tipo gaussiana, mientras que la neblina marítima presenta una atenuación gaussiana sesgada, esto debido al tipo de movimientos que esta presenta. Otro tipo de estudios, referentes a la neblina también, son los que presentan Khan y Awan [21, 3], quienes han caracterizado el nivel de atenuación relacionándolo con la cantidad de agua en la niebla y la temperatura ambiente, con lo que proponen que a partir de un censado de estos parámetros el transmisor pueda ajustar el nivel de potencia para mantener el enlace operante. Algunos otros estudios [7, 18, 43] sugieren la utilización de longitudes

de onda superiores que interactúen menos con las partículas de agua suspendidas en la neblina de tal manera que la atenuación sea menor. También se han realizado estudios en lugares donde la neblina no es frecuente, como es el caso de Zabidi [48], quien ha presentado un trabajo en regiones tropicales donde la neblina no es común como para ser un efecto apreciable, de modo que la lluvia es el siguiente factor a considerar en cuanto a atenuación. Este factor resulta ser dependiente de la intensidad de la lluvia, e independiente de la longitud de onda utilizada por el haz. Estos factores resultan en una disminución de la potencia del haz recibido en una parte, mientras que existen factores como la turbulencia atmosférica que modulan la intensidad provocando el desvanecimiento de la señal por decenas o centenas de milisegundos [8]. Un par de modelos ampliamente aceptados para modelar este tipo de perturbación son la distribución log-normal, para regímenes de turbulencia débil, mientras que la distribución Gama, K, K-I, son utilizadas para turbulencias de régimen fuerte [22, 12, 30]. Estos regímenes están dados por la matriz de índice de refracción C_n^2 que describe al medio, que para valores de 10^{-16} se considera turbulencia débil, 10^{-14} se considera régimen medio, y para valores de 10^{-13} se considera turbulencia fuerte[48].

Al igual que existen trabajos publicados donde se caracteriza el sistema COI bajo diferentes condiciones, también los existen para mejorar el desempeño en general, tal y como el estudio de Zhu [49], en el cual se analiza el desempeño de sistemas para receptores basados en el empleo de la máxima verosimilitud símbolo por símbolo, con diversidad temporal ó espacial de sistemas COI para un canal con desvanecimiento del tipo log-normal en régimen de turbulencia débil, mismos que comparados contra un receptor de umbral igual por símbolo, tienen una mejora significativa, para sistemas incoherentes utilizando modulación de intensidad y detección directa (MI/DD).

Otro estudio interesante al respecto ha sido la utilización de promedios de apertura óptica para reducir la varianza de la intensidad recibida, obteniendo un sistema con

una matriz de índice de refracción menor, y por tanto un sistema con una turbulencia aparentemente menor [33, 41]. Esto permite que otras técnicas de mitigación se enfoquen a resolver el problema pudiendo asumir un régimen de turbulencia menor que el presente en el sistema en cuestión. Las codificaciones de canal son otra herramienta ampliamente utilizada para mejorar el desempeño de los sistemas de comunicaciones, ya que permiten mejoras al sistema al incrementar la calidad del enlace debido a que estos pueden detectar y/o corregir errores. Estas mejoras, bajo ciertos escenarios, pueden ser la disminución de la probabilidad de error de bits, incremento del ancho de banda o disminución de la potencia, manteniendo una misma calidad de enlace [42]. Este tipo de técnica suele ser muy recurrida ya que no suele implicar costos tan elevados de hardware, y puede ser portable a diversos tipos de sistemas con modificaciones menores, lo cual facilita la implementación de los mismos. Aunque no eliminen en su totalidad los efectos que introduce el canal óptico turbulento, estas pueden ser complementadas con otras técnicas mencionadas previamente en los párrafos anteriores (diversidad espacial, temporal, promediación de apertura, etc). Amplios estudios han sido reportados al respecto, como tal es el caso de Zhu [50], en el cual realiza un estudio de los límites del desempeño de diversos tipos de códigos utilizando modulación en intensidad con detección directa (MI/DD) en un escenario de turbulencia débil. Zhu, hace un comparativo entre códigos de bloques, códigos convolucionales y turbo-códigos. De dicho estudio es importante resaltar que para un tiempo de bit (T_b) mayor que el tiempo de desvanecimiento del canal (τ_0), la probabilidad de error del sistema incrementa, por lo cual Zhu propone un entrelazado de profundidad K, tal que $KT_b > \tau_0$, debe mejorar el sistema considerablemente, mientras que $KT_b >> \tau_0$ no supone mejora alguna, por lo que no es recomendable, ya que esto solo incrementará la latencia de los datos. De los resultados de Zhu se observa que el mejor desempeño es obtenido por los turbo códigos, aunque como ya se comentó pudiera deberse al uso de el entrelazado[8], el cual permite distribuir los errores a lo largo de una secuencia de un tamaño determinado.

Otro parámetro de interés en los sistemas de comunicaciones que puede prohibir el uso de ciertas técnicas es la latencia de los datos. En general la latencia se refiere al tiempo transcurrido entre la generación de los datos en la fuente del transmisor hasta la llegada de los mismos al destinatario en el receptor, pero en cuanto a las codificaciones de canal solo suele referirse al tiempo que toman los datos en ser decodificados, siendo esto el objeto de estudio de Maiya [29] al comparar los códigos convolucionales contra los LDPC (Low-Density Parity Check), mismos que son ampliamente referenciados por sus capacidades correctivas y alta tasa de transmisión. No obstante los códigos LDPC son un tipo de códigos de bloques, lo cual implica que el proceso de decodificación no puede iniciar si no hasta que todo el bloque ha llegado al receptor en su totalidad, introduciendo un retraso significativo, además este tipo de códigos en general obtienen su mejor desempeño cuando el tamaño de los bloques es realmente grande, siendo en algunos casos impráctico. En cambio los códigos convolucionales pueden iniciar el proceso de decodificación a penas una pequeña cantidad de datos es recibida y solo requieren una cierta cantidad de tiempo para llenar una memoria con los estados posiblemente transcurridos para poder entregar un continuo de datos, que si bien en teoría no tiene límite, en la práctica se suele segmentar con tramas de sincronización para asegurar un determinado estado.

1.4. Hipótesis

En base a lo anterior se plantean las siguientes hipótesis:

- Es posible habilitar un enlace de comunicaciones ópticas inalámbricas mediante la implementación de un conjunto de técnicas tales como diversidad espacial, estimación de canal, codificación de canal y entrelazado de datos.

- Existe una codificación de canal que sea balanceada en cuanto a la complejidad de la implementación y el desempeño ante un canal óptico afectado por turbulencia atmosférica.

1.5. Objetivos.

En base a las hipótesis previamente planteadas y dadas las limitantes de tiempo es que en este trabajo de tesis se plantea la comprobación de la segunda hipótesis únicamente. Para esto es que se plantea el siguiente objetivo general, mismo que se logrará realizar mediante los siguientes objetivos específicos.

1.5.1. Objetivo general.

Diseñar e implementar los bloques de codificación y decodificación de canal que permitan mejorar el desempeño en sistemas de comunicaciones ópticos inalámbricos afectados por turbulencia óptica atmosférica, utilizando dispositivos FPGA.

1.5.2. Objetivos específicos.

- Seleccionar un tipo de codificación versátil y flexible para canales con desvanecimiento y similares.
- Realizar la implementación de un codificador y decodificador sobre una plataforma FPGA.
- Realizar pruebas bajo condiciones de ruido cero para asegurar la funcionalidad del codificador y decodificador.

1.6. Importancia del estudio.

Actualmente se cuentan con amplios estudios teóricos del desempeño de diversas codificaciones de canal considerando diversos factores y escenarios, no obstante pocas son reportadas, lo cual es de suma importancia ya que es con las implementaciones que salen a la luz diversas consideraciones que deben ser tomadas por los futuros trabajos, además de que estos permiten validar los factores que se suponen actualmente, siendo esto cierto para el caso de las comunicaciones ópticas inalámbricas como para muchos otros sistemas de comunicaciones en los cuales esta codificación de canal puede ser portada.

1.7. Estructura de capítulos.

A continuación se describirá de forma breve en qué consiste cada capítulo de esta tesis, para darle al lector una idea de la información que encontrará.

En el **capítulo 2** se mencionaran diversos tipos de codificaciones de canal más comunes y sus características, así como las mejoras que estos pueden proveer a un sistema determinado.

En el **capítulo 3** se comentará sobre el canal atmosférico y su interacción con las ondas ópticas. Se mencionaran sus características, problemas y algunos de los modelos que son los más apropiados para algunos fenómenos referentes a la propagación de ondas ópticas. Se concluye con la selección de una codificación de canal apta para este y otros canales que presenten problemas similares.

En el **capítulos 4** se abordará la implementación del codificador y decodificador convolucional, al igual que el esquema general propuesto y probado bajo condiciones

de ruido cero. Adicionalmente se presenta un análisis del método de implementación de múltiples codificadores y decodificadores.

En los **capítulos 5** y **6** se presentan los resultados correspondientes a la implementación, además de la conclusión a la que se ha llegado a lo largo de este trabajo de tesis, al igual que se comentan algunas consideraciones para futuras implementaciones.

Capítulo 2

Códigos de canal.

En este capítulo se mencionarán los diversos tipos de codificaciones, su funcionamiento, propiedades y características, así como las capacidades correctivas teóricas [5, 6, 17, 34, 37, 38, 42].

2.1. Generalidades de los códigos de canal

2.1.1. Tipos de codificaciones.

Las codificaciones según la forma de codificar se pueden dividir en dos tipos: codificación de forma de onda y codificación por secuencias estructuradas. Las codificaciones de forma de onda son aquellas que intentan mejorar la probabilidad de una correcta interpretación de los datos recibidos en el receptor, básicamente se intenta hacer señales muy distintas entre sí para que le sea más sencillo al receptor diferenciarlas (Señales antipodales $-Seno$, $Seno$ y ortogonales $Seno$, $Coseno$, como por ejemplo la figura 2.1).

Las codificaciones de secuencias estructuras son aquellas codificaciones que implican

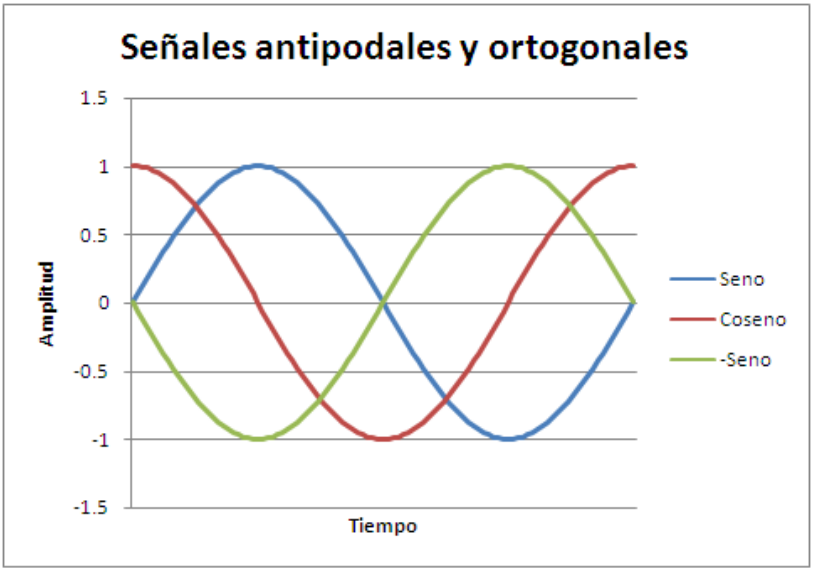


Figura 2.1: Codificación por forma de onda: a) y b) son antipodales, mientras que c) y d) son antipodales.

el uso de información redundante proveniente del mensaje a transmitir con la finalidad de poder detectar y/o corregir errores durante la transmisión. Para aquellas transmisiones con información redundante se puede definir un parámetro llamado tasa de transmisión el cual define la cantidad de bits de información en relación con la cantidad de bits del mensaje codificado (k bits del mensaje, n bits del código), siendo la tasa de transmisión k/n, que por obvias razones este valor no puede superar la unidad.

2.1.2. Tipos de control de errores.

Ahora bien, un sistema de comunicaciones capaz de detectar errores debe ser capaz de poder corregirlos también y en función de cómo lo haga se pueden considerar de las siguientes formas:

- Corrección por retransmisión, que como su nombre lo indica, consiste básicamente

en retransmitir los datos que se detecten errados, consecuentemente implica una comunicación dúplex para poder notificar los datos errados.

- Corrección de errores (en inglés, Forward Error Correction o FEC) , este método permite por medio de las codificaciones poder detectar y/o corregir los datos errados en el receptor, evitando retransmisiones.

2.1.3. Beneficios de la codificación.

La codificación en general supone una mejora ante la probabilidad de error de transmisión de datos, lo cual en función de las condiciones dadas y las necesidades se puede traducir en alguno de los tres tipos de beneficios o una combinación de ellos, como se ilustra en la figura 2.2.

- Mejora la calidad de transmisión al haber menor probabilidad de error (figura 2.2 trayectoria A).
- Reducir la potencia de transmisión. Debido a que usualmente la potencia ayuda a facilitar la recepción de los datos y por tanto mejorar la calidad de la transmisión, esta puede ser sustituida por una codificación obteniendo un resultado similar para ciertos casos (figura 2.2 trayectoria B).
- Aumentar la tasa de transmisión. Al aumentar la tasa de transmisión tenemos más bits transmitidos con una misma potencia, provocando con esto que la probabilidad de error aumente, por ello una codificación permitiría restaurar la probabilidad de error que el incremento en la tasa de transmisión provocaría.

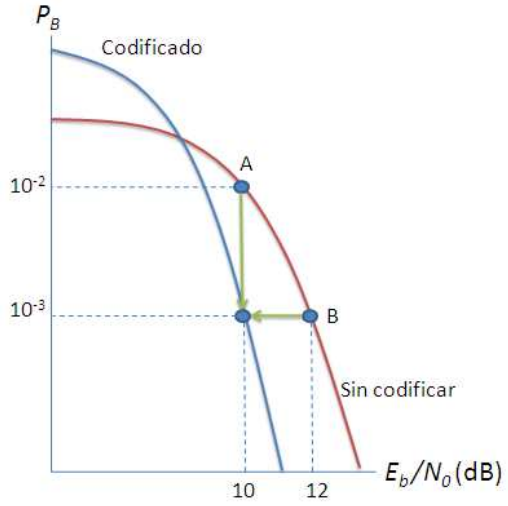


Figura 2.2: Comparación entre una transmisión codificada y una sin codificar.

2.1.4. Tipos de canales.

Los tipos de canales son modelos que describen el comportamiento del medio a través del cual se propagan los mensajes y son ciertamente fundamentales para entender cómo es que se ven afectadas las comunicaciones.

- **Canal discreto sin memoria (DMC)**: Es un canal con un alfabeto discreto en el cual la señal recibida depende únicamente del dato enviado.
- **Canal binario simétrico (BSC)**: Es un caso particular del DMC en el que el alfabeto discreto es binario 0 y 1, de modo que las probabilidades son simétricas, tanto de fracaso $P(1|0) = P(0|1) = p$, como la probabilidad de éxito $P(0|0) = P(1|1) = 1 - p$.
- **Canal Gaussiano**: Supone un alfabeto de entrada discreto y un alfabeto de salida continuo, ya que el ruido característico de este medio se agrega a la señal siendo dicho ruido una señal con media cero y una varianza dada σ^2.

2.2. Códigos de bloques lineales.

Los códigos de bloques lineales surgen de querer codificar k bits en una secuencia de n bits denotados códigos de bloques (n, k), estos códigos pueden tratar cualquier tipo de símbolo. Para poder entender cómo es que la codificación se lleva a cabo es necesario entender el ámbito en el cual se desarrollan, por ello es necesario saber algo sobre los espacios de vectoriales que componen a los elementos de la codificación [6]. Supongamos $k = 3$, y $n = 6$, de modo que existen $2^3 = 8$ secuencias distintas para el mensaje (llamados k-tuples) y $2^6 = 64$ secuencias distintas para los posibles códigos (llamados n-tuples), a estas secuencias (8 para k, y 64 para n) se les conocen como espacios vectoriales, y a un conjunto o parte de estas dentro de estos espacios se les llaman subespacios. Debido a que el espacio de k es menor que el de n, y para llevar a cabo la codificación es necesaria una asignación de uno a uno para las palabras codificadas, resulta ser que solo es requerido un subespacio de n es decir solo 2^k (8 secuencias en este caso) para codificar todos los posibles mensajes. Ahora bien, los subespacios no pueden simplemente elegirse al azar dentro de las posibles secuencias, por ello se proponen los códigos de bloques lineales, mismos que se obtienen cumpliendo con dos sencillas reglas:

- La secuencia de ceros debe estar incluida.
- La suma de cualquiera de los elementos del subespacio debe ser un elemento del subespacio también.

Cabe mencionar que de tener un espacio para n más grande, es decir, incrementar la redundancia facilita el poder detectar y/o corregir errores en la transmisión, mientras que disminuye la tasa de transmisión, de modo que en función de las necesidades este parámetro entrara en juego.

2.2.1. Matriz generadora.

Una forma fácil y obvia de asignar los códigos a los mensajes es por medio de una tabla que los relaciones, pero para códigos grandes una tabla resulta ineficiente, tanto para análisis como para la implementación de la codificación, por ello es que se utilizan matrices generadoras [6]. Para entender cómo funciona la matriz generadora se deben definir los elementos que participan, tales como el mensaje $m = m_1, m_2, m_3, ..., m_k$, un vector con k elementos, la matriz generadora $G = v_1, v_2, v_3, ..., v_k$ donde v_i es un vector de n elementos que son bases para formar el subespacio de n que será utilizado para la codificación, y por último las palabras codificadas U que resultan de multiplicar la matriz generadora por el mensaje a enviar. Una forma general de una matriz generadora se presenta en la ecuación (2.1).

$$G = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \tag{2.1}$$

Como se observa, la matriz generadora está compuesta de k vectores o bases para generar el subespacio de las palabras codificadas que serán utilizadas, de modo que para codificar un mensaje m solo es necesario multiplicarlo por la matriz generadora G.

La matriz generadora debe cumplir con el hecho de que los vectores base deben ser linealmente independientes, ya que de lo contrario habría palabras codificadas repetidas ante diversos mensajes, lo cual no satisface la relación de uno a uno para mensaje-palabra codificada. Ahora bien, existen condiciones que no son estríctamente necesarias para el funcionamiento del mismo, pero sí facilitan el trabajo con los mismos, tal es el caso de los códigos de bloques sistemáticos, los cuales utilizan una matriz generadora de la forma $G = [P|I_k]$, donde P es una matriz que define a los bits de paridad y I_k es

una matriz identidad que básicamente duplica el mensaje, obteniendo como resultado un mensaje codificado de la forma $U = [x_1, x_2, x_3, ..., y_1, y_2, y_3, ...]$, siendo los elementos x los bits de paridad y los elementos y los bits del mensaje.

Este tipo de matrices aun tienen utilidad en el área de comunicaciones para canales bien conocidos y aplicaciones muy especificas, aunque recientemente se le han estado encontrando aplicaciones en otras áreas, como es el caso de estudios realizados con la decodificación de los genomas de diversos seres vivos, en los cuales, el código en el cual esta codificada la información no es conocida [9].

2.2.2. Matriz de revisión de paridad

Al enviar el mensaje a través del canal, resulta evidente que este pueda sufrir alteraciones, de modo que debe existir un método para validar los códigos recibidos, dicho método es implementado mediante la matriz de revisión de paridad. Si recordamos el hecho de que los 2^k códigos posibles son un subespacio de las 2^n secuencias, esto nos indica que cualquiera que no sea uno de estos n-tuples, será considerado un error de transmisión, y es precisamente a estas secuencias a las que detecta la matriz de revisión de paridad. La matriz de paridad está definida como $H = [I_{k-n}|P^T]$; mientras que la matriz de paridad transpuesta $H^T = [I_{k-n}; P]$, de modo que $UH^T = 0$ para los 2^k códigos posibles del subespacio. De recibir un código alterado no perteneciente al subespacio la multiplicación obtendríamos $UH^T \neq 0$. Nótese que la matriz de paridad detecta aquellos códigos alterados no pertenecientes al subespacio, de modo que si un código es recibido erróneamente como un código perteneciente al subespacio pero diferente del que se ha enviado será un error indetectable.

Síndrome. Detección y corrección de errores.

Ahora bien, una vez detectado un error este debe poder ser corregido, y esto se consigue al relacionar un determinado tipo de error con un síndrome en particular, de modo que el cálculo del síndrome conlleve a detectar el error y por ende poder corregirlo. Supongamos que la secuencia recibida r es compuesta por un código válido y un(os) error(es) en alguno(s) de los bits de la secuencia, de modo que $r = U + e$, donde U es un código valido y e el vector que describe los errores, el síndrome es calculado como $S = rH^T$, siendo esto reducido a $S = eH^T$ ya que $UH^T = 0$ para los códigos validos. Si las secuencias están compuestas por n bits, esto significa que existen 2^n posibles secuencias a recibir, de las cuales 2^k son validas, resultando 2^{n-k} secuencias erróneas detectables, y por tanto 2^{n-k} síndromes distintos, cada uno asociado a una secuencia errónea. El método de corrección de errores se lleva a cabo de la siguiente manera:

- Elaborar una tabla de relación entre el síndrome y el error asociado al mismo.
- Ubicar el síndrome y su error asociado $S = rH^T$.
- Restar el error, que en el caso de sistemas binarios es igual restarlo que sumarlo.

2.2.3. Capacidades de corrección y detección de errores.

La tarea del decodificador consiste en definir que código ha llegado al receptor y para ello lo que realiza es una comparación entre el vector recibido r y los códigos validos U, de modo que aquel vector se parezca más será considerado como el que se ha transmitido. Para ilustrar esto supónganse el esquema ilustrado en la figura 2.3.

(a) Se ha enviado U pero se ha recibido r_1, de modo que el decodificador realiza una comparación entre los códigos U y V contra el vector de datos recibido, resultando

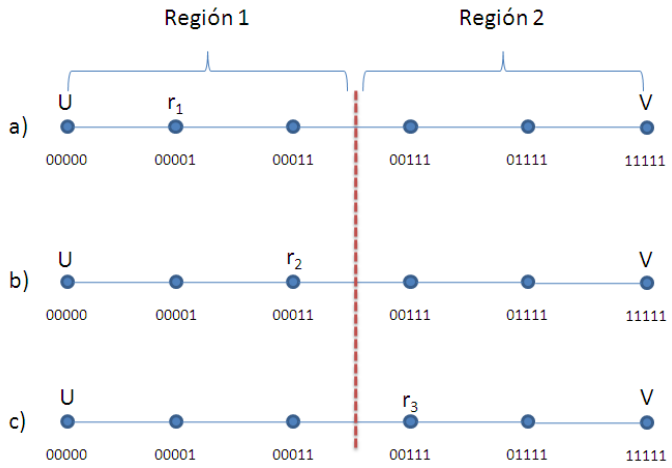


Figura 2.3: Capacidades de corrección de errores en función del mensaje recibido r.

en una diferencia de 1 bit entre r_1 y U, mientras que la diferencia entre V y r_1 es de 4 bits, por lo cual el decodificador decide que el mensaje transmitido debió ser U.

(b) Se ha enviado U pero se ha recibido r_2, de modo que el decodificador realiza una comparación entre los códigos U y V contra el vector de datos recibido, resultando en una diferencia de 2 bit entre r_2 y U, mientras que la diferencia entre V y r_1 es de 3 bits, por lo cual el decodificador decide que el mensaje transmitido debió ser U.

(c) Se ha enviado U pero se ha recibido r_3, de modo que el decodificador realiza una comparación entre los códigos U y V contra el vector de datos recibido, resultando en una diferencia de 3 bit entre r_3 y U, mientras que la diferencia entre V y r_1 es de 2 bits, por lo cual el decodificador decide que el mensaje transmitido debió ser V, siendo este último caso un error, ya que el mensaje recibido se parece más a V.

Ahora bien, de este ejemplo podemos observar que las capacidades de detección de errores (A, B, C, porque en C también se detecta el error, pero se corrige mal) y de

corrección de errores (A y B) están relacionadas con la *Distancia de Hamming* $d(U, V)$ entre los códigos validos, misma que está definida como la cantidad de elementos no cero que resulta de la diferencia entre los elementos de dos códigos validos, U y V en este caso. Resulta evidente que entre más pequeña sea dicha distancia (menor sea la diferencia entre estos) más fácilmente se pueden cometer errores, por lo cual la distancia mínima d_{min}, definida como la *Distancia de Hamming mínima* entre los códigos validos resulta ser un parámetro importante dentro del diseño de códigos, y es un indicador de que tan fuerte es el código, claro, este parámetro no es fácil o incluso posible de calcular para todo tipo de códigos.

Considerando esta métrica lo que prosigue es calcular la *Distancia de Hamming* entre los códigos validos y buscar el mínimo, pero esto resulta ser innecesariamente trabajoso, ya que tendríamos que calcular la diferencia de todas las combinaciones posibles en busca de la menor. Existe una característica de los códigos llamada el *Peso de Hamming* $w(U)$, el cual está definido como el número de elementos no cero de la secuencia U, el cual puede resultar muy útil para la obtención de la *Distancia de Hamming mínima*. Considere como ejemplo $U = [10101]$ y $V = [01110]$, donde $w(U) = 3$ y $w(V) = 3$. Ahora bien, $U + V = [11011]$ y $w(U + V) = 4$ mismo que es igual a $d(U, V)$, esto debido a que la suma realizada mediante el modulo-2 nos arroja los bits en los que difieren ambas secuencias, es decir, $U + V = [11011]$, de lo mismo que del contar los bits (es decir obtener $w(U + V)$) obtenemos la distancia entre ambas.

Esto resulta muy importante al recordar que para seleccionar un subespacio para la codificación uno de los requisitos fue que la suma de los vectores del subespacio fuera otro vector dentro del mismo subespacio, es decir, si U y V son vectores del subespacio que utilizamos para codificar, entonces $U + V = X$, donde X también debe pertenecer al subespacio, de modo que todos los vectores del subespacio resultan ser la suma de algún otro par de vectores, de tal forma que para obtener la Distancia de Hamming

entre dos vectores $d(U+V)$, es lo mismo que obtener el *Peso de Hamming* $w(X)$. Esto es ideal para el cálculo de d_{min}, ya que solo es necesario obtener el *Peso de Hamming* de cada uno de los códigos validos y buscar el menor, obteniendo d_{min} más fácilmente. Cabe mencionar que para el cálculo del la *Distancia Mínima* por medio de los *Pesos de Hamming* no se considera el vector de ceros. Una vez obtenida la distancia de Hamming podemos formalizar la capacidad detección de errores como (2.2).

$$e = d_{min} - 1 \tag{2.2}$$

La cantidad de corrección de errores queda definida como (2.3)

$$t = \lfloor \frac{d_{min} - 1}{2} \rfloor \tag{2.3}$$

donde $\lfloor x \rfloor$ es el entero más grande sin exceder x.

En tanto que se puede diseñar un decodificador capaz de únicamente detectar o corregir errores, se puede diseñar uno que haga ambos, es decir, que pueda detectar α errores y corregir β errores simultáneamente tomando en cuenta que $\beta \leq \alpha$, de modo que la capacidad correctiva de un código dado es $d_{min} \geq \beta + \alpha + 1$

2.2.4. Corrección por medio de borraduras

Un receptor puede estar diseñado de tal manera que pueda designar cuando un elemento presente una *borradura* (erasure), esto significa que el receptor no está seguro de que fue lo que recibió, si un *1* o un *0*. Si recordamos los niveles lógicos **TTL**, un cero lógico está comprendido entre *0 - 0.8V*, mientras que un uno lógico está entre *2.4 - 5V*, dejando una zona indefinida entre los *0.8 - 2.4V*, de modo que una borradura se asemeja a esta zona indefinida, la cual es declarada por el receptor como una borradura. Ahora bien, esto nos sirve ya que el marcar un elemento como inseguro nos da la posición del posible error, permitiendo de esta forma incrementar el poder de corrección

del decodificador. En general se dice que el decodificador tiene como límite $d_{min} \geq \rho + 1$, siendo ρ es la cantidad de borraduras que se pueden corregir. De igual forma que al detectar y corregir errores, se pueden declarar borraduras y corregir errores simultáneamente, y al igual que en el caso anterior este también tiene restricciones, pero de la forma $d_{min} \geq 2\alpha + \gamma + 1$, donde γ son las borraduras y α son las correcciones. Cabe mencionar que la borradura al no ser un *0* o un *1*, para el caso del alfabeto binario, el utilizarlo requiere de un elemento más en el alfabeto, mismo que indicará cuando este sea una borradura, de modo que para cualquier codificador que utilice borraduras, su alfabeto se verá incrementado en un elemento.

2.2.5. Probabilidad de una decodificación errónea

Dentro del cálculo de probabilidades existen diversos tipos de distribuciones de probabilidad dependiendo de la naturaleza del evento a estudiar. En el caso en el que se desea estudiar la ocurrencia o no de un evento se le conoce como distribución binomial ilustrada en la ecuación (2.4)

$$P(x = j) = \frac{n! p^j (1-p)^{n-j}}{j!(n-j)!} \tag{2.4}$$

donde j es el número eventos favorables que se desea estudiar, n el número de ensayos o veces que se realiza el experimento, p es la probabilidad de éxito del evento estudiado.

NOTA: por eventos favorables se entienden los eventos que se estudian, pudiendo ser estos errores, ausencias, objetos dañados, etc, una vez etiquetados j son los eventos de estudio y no deben confundirse.

Ahora bien, supongamos un código $(6, 3)$ con $d_{min} = 3$ el cual es transmitido por

un canal donde la probabilidad de error de bit es p. Para saber la probabilidad de que el decodificador realice una decodificación errónea debemos recordar la capacidad de corrección de bits que estaba dada por la ecuación (2.3), de tal manera que este decodificador puede corregir 1 bit, esto quiere decir que para llevar a cabo una decodificación errónea son necesarios 2, 3, 4, 5 o 6 errores debemos calcular la probabilidad de que se cometan dichos errores y sumarlas, ya que todas estas producen la codificación errónea.

2.3. Códigos cíclicos

Son un tipo de códigos de bloques lineales basados en el corrimiento de los datos a través de una serie de registros. Son ampliamente utilizados ya que estos permiten que se les describa mediante polinomios de orden n, los cuales resultan ser muy manejables. Los coeficientes del polinomio son los elementos de los códigos. Un código (n,k), será cíclico si puede ser descrito mediante la siguiente propiedad: Sí el n-tuple $U = (u_0, u_1, u_2, \ldots u_{n-1})$ es un código valido dentro del espacio S, entonces $U(1) = (u_{n-1}, u_0, u_1, u_2, \ldots u_{n-1}, u_{n-2})$ obtenido al recorrer los datos dentro de los registros, debe también ser un código valido del espacio S. Una forma general de estos polinomios se ilustra en la ecuación (2.5).

$$U(X) = u_0, u_1X, u_2X^2, \ldots u_{n-1}X^{n-1} \tag{2.5}$$

2.3.1. Estructura algebraica de los códigos cíclicos

Si $U(X)$ es un polinomio de orden $n-1$ que representa a un código valido, entonces $U^{(i)}(X)$ es el residuo de dividir $X_iU(X)$ entre X^n+1, esto conforme a (2.5), y se denota como (2.6) y (2.7).

$$\frac{X^i U(X)}{X^n + 1} = q(X) + \frac{U^i(X)}{X^n + 1} \tag{2.6}$$

$$X^i U(X) = q(X)(X^n + 1) + U^i(X) \tag{2.7}$$

siendo $U^{(i)}(X)$ el residuo.

2.3.2. Propiedades de los códigos cíclicos

Un código valido estará formado por $U(X) = m(X)g(X)$, un polinomio de orden $n-1$ que representa a un código valido, donde $m(X)$ es un polinomio de orden $k-1$ cuyos coeficientes son los bits del mensaje, mientras que $g(X)$ es un polinomio generador de orden $n-k$. Estos polinomios tienen sus estructuras de la forma (2.8) y (2.9) respectivamente.

$$m(X) = m_0, m_1 X, m_2 X^1, \ldots, m_{k-1} X^{k-1} \tag{2.8}$$

$$g(X) = g_0, g_1 X, g_2 X^1, \ldots, g_{n-k} X^{n-k} \tag{2.9}$$

donde g_0 y g_{n-k} tienen que ser igual a 1.

Al observar cómo se forma $U(X)$, resulta evidente que todo código valido debe ser divisible entre $g(X)$ sin quedar residuo alguno.

2.3.3. Codificación sistemática

Al igual que en los códigos de bloques lineales, también se pueden codificar sistemáticamente los códigos cíclicos. Esto se logra al multiplicar el polinomio del mensaje $m(X)$ por X^{n-k} con lo que estaríamos recorriendo los bits del mensaje a la derecha, faltando agregar los bits de paridad a la izquierda. Los bits de paridad son el residuo de dividir $X^{n-k}m(X)$ entre el polinomio generador $g(X)$, esto significa que $X^{n-k}m(X)$ está formado por (2.10)

$$X^{n-k}m(X) = q(X)g(X) + p(X) \tag{2.10}$$

de manera que al agregar los bits de paridad al lado izquierdo de la igualdad tenemos un código valido $U(X)$, esto se obtiene sumando a ambos lados de la ecuación $p(X)$ (Recordemos que al sumar $p(X)$ del lado derecho, estos se cancelan por la aritmética binaria), de esto obtenemos (2.11)

$$X^{n-k}m(X) + p(X) = q(X)g(X) = U(X) \tag{2.11}$$

2.3.4. Detección de errores

Para que un código sea válido, este tiene que ser divisible entre $g(X)$ y con residuo cero, de acuerdo con (2.11). Ahora bien, supongamos la transmisión de un código valido el cual ha sido perturbado de modo que el receptor percibe $Z(X) = U(X)+e(X)$, donde $e(X)$ es un polinomio de orden $n-1$ cuyos coeficientes representan los bits que fueron alterados. La división de $Z(X)$ entre $g(X)$ nos dará un residuo, mismo que es conocido como el síndrome $S(X)$, siendo un polinomio de orden $n-k-1$, que al igual que en el caso de los bloques lineales está asociado al error. Dicha relación está dada por las

ecuaciones (2.12), (2.13)

$$U(X) = m(X)g(X) \tag{2.12}$$

$$Z(X) = U(X) + e(X) \tag{2.13}$$

Sustituyendo (2.12) en (2.13), tenemos (2.14)

$$Z(X) = m(X)g(X) + e(X) \tag{2.14}$$

Al ser $S(X)$ el residuo de $Z(X)$ entre $g(X)$, $Z(X)$ lo podemos expresar como (2.15)

$$Z(X) = g(X)q(X) + S(X) \tag{2.15}$$

de modo que podemos igualar ambas expresiones de $Z(X)$, (2.14) y (2.15) para obtener $e(X)$ en (2.18):

$$m(X)g(X) + e(X) = g(X)q(X) + S(X) \tag{2.16}$$

$$e(X) = g(X)q(X) + S(X) + m(X)g(X) \tag{2.17}$$

$$e(X) = g(X)[q(X) + m(X)] + S(X) \tag{2.18}$$

De este análisis podemos observar que $S(X)$ es tanto el residuo de $Z(X)/g(X)$ como el residuo de $e(X)/g(X)$. De esta forma solo es necesario calcular los síndromes que

producen determinados errores. Posteriormente cuando un código sea recibido erróneamente (es decir, la división entre el código recibido y $g(X)$ tenga residuo diferente de cero), podremos mediante el residuo saber que error fue el que lo ocasionó.

2.4. Códigos convolucionales

Un código convolucional es generado cuando una secuencia de datos atraviesa una serie de registros de estados finitos, los cuales forman una secuencia nueva al hacer una combinación lineal de los datos en dichos registros. Existen mK registros y n generadores de funciones lineales. El codificador es alimentado con m bits a la vez, mientras que la salida está dada por los n generadores, de modo que la tasa del codificador puede definirse aproximadamente como m/n. El parámetro K es llamado *constante de restricción* y define la cantidad de registros a través de los cuales pasa un bit de información influyendo la salida del codificador [42]. En la figura 2.4 se ilustra un codificador convolucional de la forma general.

En un ciclo de reloj son ingresados k bits al codificador y generados n datos de salida del decodificador, mientras que le tomarán K ciclos de reloj recorrer los K estados a un bit que ingresa al codificador. Los codificadores convolucionales a pesar de no tener una restricción en cuanto al tamaño de las secuencias de entrada, se suelen truncar con una cantidad de $K-1$ ceros con la finalidad de limpiar los registros.

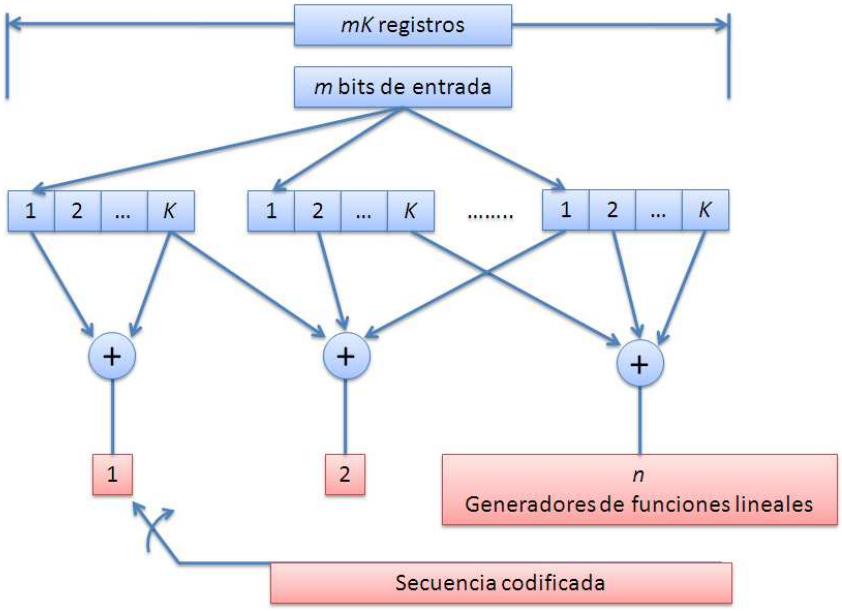


Figura 2.4: Diagrama general de un codificador convolucional.

2.4.1. Representaciones del codificador

Representación polinomial

Al igual que en los códigos cíclicos es posible representar al codificador y al mensaje como polinomios para su análisis con métodos algebraicos. Los polinomios generadores en este caso describen las interconexiones entre los registros de corrimiento y el punto de suma (siendo X^0 la referencia al registro de entrada de los datos). Esta representación resulta muy conveniente a la hora de calcular U, ya que esta se obtiene por medio de una multiplicación entre $m(X)$ y $g_i(X)$, siendo $m(X)$ el mensaje y $g_i(X)$ los diversos puntos de suma que existen en el codificador. De las multiplicaciones obtenemos i resultados parciales, los cuales solo es necesario intercalar para obtener $U(X)$.

Representación de estados y diagrama de estados

Los códigos convolucionales son conocidos también como máquinas de estados finitos. Las máquinas de estados en general son maquinas que tienen memoria de los sucesos anteriores, de modo que su funcionamiento depende en gran medida de los sucesos anteriores. A dichos estados se les puede definir como paquetes del menor tamaño posible de información que condensan los sucesos anteriores de los cuales depende la salida de la máquina. En un código convolucional la cantidad de estados dependen de $m(K-1)$, donde m es la cantidad de bits de entrada en un momento dado y de K la constante de restricción o numero de registros de memoria del codificador. Los códigos convolucionales están relacionados con las cadenas de Markov, en el sentido de que la probabilidad de que el sistema caiga dentro de un estado está directamente relacionada con el estado anterior, es decir, $P(X_{i+1}|X_i)$ donde X_i es el estado actual y X_{i+1} es el estado futuro. Una forma de representar codificadores sencillos es mediante un diagrama de estados, el cual indica en sus estados los últimos $K-1$ bits que han entrado al codificador, mismos que son necesarios para conocer la salida del decodificador. Los caminos entre los estados denotan el valor de la entrada e indican hacia qué estado está transitando el codificador. Nótese que el número de caminos que puedan tener los estados depende directamente del tamaño del alfabeto que se esté utilizando, de modo que para alfabetos binarios se tienen únicamente dos caminos. En la figura 2.5 se ilustra el codificador que utiliza $g_1 = [111]$ y $g_2 = [101]$, de modo que los estados (los posibles valores de los $K-1=2$ registros) son: *a=00, b=10 c=01, d=11.*

Las líneas indican el cambio de un estado a otro, siendo las líneas punteadas el camino tomado debido a un 1 en la entrada, mientras que las líneas sólidas dibujan el trayecto de un estado a otro debido a un 0 en la entrada. El número al lado de cada línea (punteada o sólida) indica la salida del codificador y este tendrá tantas cifras como puntos de suma tenga el codificador, en este caso dos. Debe observarse que no es

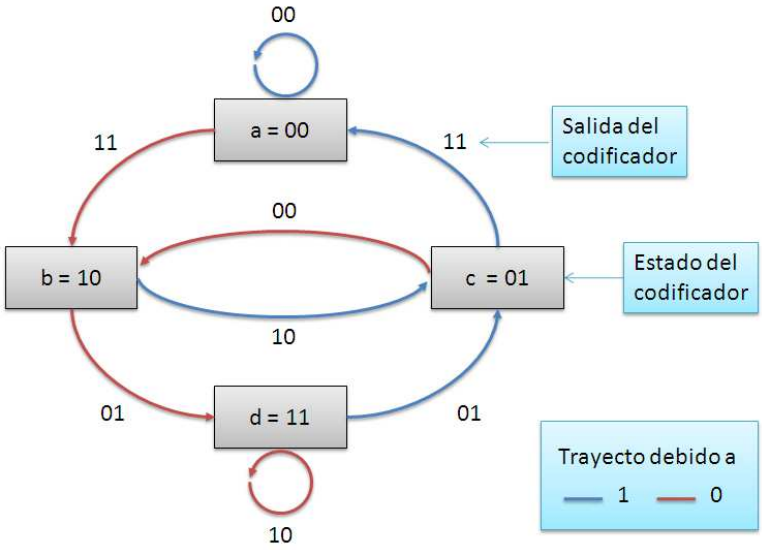


Figura 2.5: Diagrama de estados de un codificador con $K = 3$, $g_1 = [111]$ y $g_2 = [101]$.

posible saltar de un estado a otro arbitrariamente, esto es debido a que solo se tienen dos elementos en el alfabeto. Para generar un código utilizando este diagrama, solo es necesario suponer un estado inicial y seguir las líneas conforme se conozcan los bits de entrada.

2.4.2. Diagrama de árbol

Este diagrama es poco eficiente pero muy didáctico, debido a que permite observar la transición de los estados de una forma más clara. Consiste en ramificar a partir de un estado dado a los posibles estados a los que se puede llegar dependiendo del valor de entrada. Si el bit de entrada es un 0, en el diagrama de árbol siempre tomaremos el camino superior, mientras que si es un 1 tomaremos el camino de abajo. En función del valor de entrada podemos ver en el diagrama la sucesión de estados que ocurre en el codificador. Una característica de este diagrama es que a diferencia del diagrama de estados, este si puede tener una representación del tiempo, de modo que a partir

de la profundidad en la que uno esté dentro del árbol se puede saber cuántos bits del mensaje han entrado al codificador. Cabe mencionar que el diagrama de árbol se debe extender tanto como tantos bits tenga el mensaje, razón por la cual es ineficiente su uso para mensajes largos, alfabetos mayores a dos símbolos. En la figura 2.6 se ilustra un diagrama de árbol con $K = 3$.

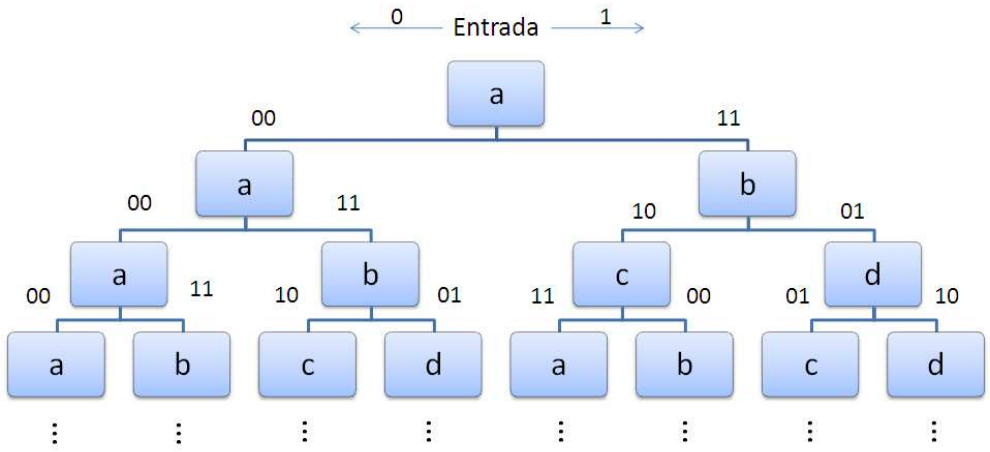


Figura 2.6: Diagrama de árbol con $K = 3$.

2.4.3. Diagrama de trellis

El diagrama de trellis resulta de la observación de que el sistema es recursivo, es decir, dos estados cual quiera presentan los mismos estados disponibles después de K avances dentro de sus posibles trayectos. El diagrama de trellis, puede ser visto como una compresión del diagrama de árbol, ya que este no se extiende más que en el eje del tiempo. En la figura 2.7 se ilustra un diagrama de trellis para un codificador convolucional con $K = 3$, $g_1 = [111]$ y $g_2 = [101]$:

El diagrama de trellis condensa toda la información necesaria para entender el decodificador, incluye la transición de los estados (líneas sólidas y punteadas), los datos de

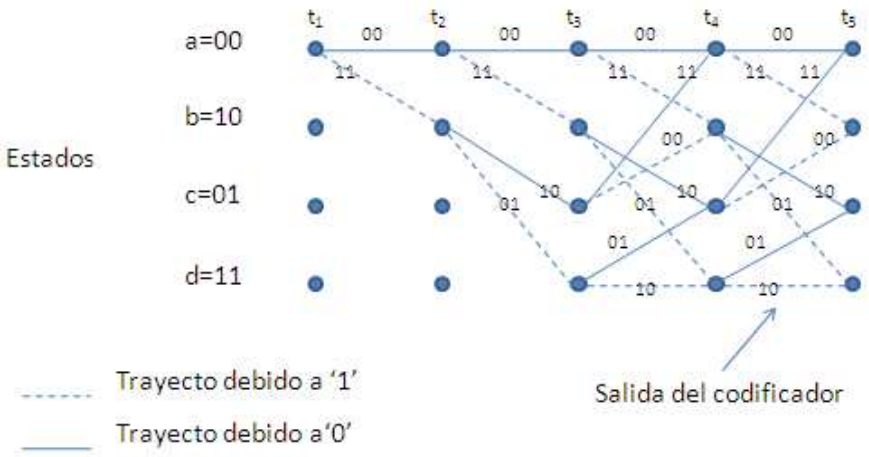


Figura 2.7: Diagrama de trellis con $K = 3$, $g_1 = [111]$ y $g_2 = [101]$.

entrada (línea sólida para un 0 y línea punteada para un 1), los datos de salida (número junto a la línea de transición de estado), considera el tiempo (las transiciones hechas de izquierda a derecha). Obsérvese que cada estado (representado como un nodo) tiene dos posibles caminos como salida, y después del tercer avance se puede accesar a cada estado por medio de dos caminos, esto está relacionado con el tamaño del alfabeto, mientras que la cantidad de avances necesarios para la recursividad está relacionado con la cantidad de registros K que posea el codificador.

2.4.4. Consideraciones para la decodificación

Un decodificador por máxima verosimilitud es aquél que de una secuencia recibida Z busque de entre todas las secuencias validas $U^{(m)}$ aquella que sea la más parecida $U^{(m')}$, es decir (2.19).

$$P(Z|U^{(m')}) = Max(P(Z|U^{(m)})) \tag{2.19}$$

donde $U^{(m')}$ denota a la secuencia elegida y $U^{(m)}$ denota a las posibles secuencias

validas.

La función de máxima verosimilitud depende del canal del que se trate. Suponiendo un canal con ruido blanco aditivo de tipo gaussiano (**AWG**), es decir, un ruido con media cero y sin memoria, es decir que todos los bits de la secuencia son afectados de igual forma y de manera independiente, se puede calcular la verosimilitud, para cuando la tasa de transmisión es de $1/n$, representado en la ecuación (2.20).

$$P(Z|U^{(m)}) = \prod_{i=1}^{\infty} P(Z_i|U_i^{(m)}) = \prod_{i=1}^{\infty}\prod_{j=1}^{n} P(z_{ij}|u_{ij}^{(m)}) \tag{2.20}$$

donde Z_i es el i-ésimo segmento del código recibido de la secuencia Z, $U_i^{(m)}$ es el i-ésimo segmento valido de la secuencia valida $U^{(m)}$, z_{ji} es el j-ésimo bit del segmento Z_i y u_{ji} es el j-ésimo bit del segmento $U_i^{(m)}$, siendo cada segmento compuesto por n símbolos (número de puntos de suma).

El trabajo del decodificador es elegir aquella secuencia o segmento que maximice la probabilidad de que una determinada secuencia o segmento haya sido transmitido, es decir (2.21).

$$Max(\prod_{i=1}^{\infty}\prod_{j=1}^{n} P(z_{ij}|u_{ij}^{(m)})) \tag{2.21}$$

2.4.5. Tipos de canales: Decisión rígida y suave

Es bien sabido que para la transmisión de los códigos a través de un medio, es necesario el transformar las secuencias en señales capaces de viajar por el medio, a este proceso se le conoce como modulación, mismo que tiene su contra parte que convierte una señal a las secuencias codificadas, esto conocido como demodulación. Ahora bien, en el canal en el cual se transmite, siempre hay perturbaciones conocidas como ruido, como

es el caso del ruido blanco aditivo del tipo gaussiano (**AWG**), previamente mencionado, el cual va a afectar a nuestra señal mientras se propague por el medio, de modo que el receptor recibe una señal de la forma (2.22).

$$r(t) = s_i(t) + n(t) \tag{2.22}$$

donde $r(t)$ es la señal recibida compuesta por $si(t)$ es la señal que representa a las símbolos que componen la secuencia codificada $n(t)$ es una componente de ruido agregado por el medio.

Suponiendo un alfabeto binario $s_i(t)$ puede ser tanto $s_1(t)$ o $s_2(t)$, señales que representarían por ejemplo un 0 y un 1 respectivamente. Ahora bien, existen dos formas en las que el demodulador puede tratar esta información:

- El demodulador define que símbolo fue transmitido y se lo hace llegar al decodificador. Es decir, calcula $P(z|s_1)$ contra $P(z|s_2)$ y elige el mayor. A este decodificador se le conoce como decodificador de decisión rígida.
- El demodulador define que tanto parece la señal recibida a un determinado símbolo, es decir, no entrega un 1 o un 0 si no un valor fraccionario, 3/8 por ejemplo, de modo que el decodificador haga uso de esta información para determinar que símbolo se transmitió. A este tipo de decodificador se le conoce como decodificador de decisión suave.

Cabe mencionar que mientras que el demodulador puede hacer una estimación de que tan probable es que el símbolo recibido sea tal o cual, el decodificador no puede hacer eso, de modo que el decodificador sí tiene que definir el mensaje de una forma rígida.

En un canal con ruido AWG un decodificador de decisión suave con 8 niveles de cuantización logra conseguir una mejora de 2dB con respecto a un decodificador de decisión rígida, mientras que una señal analógica (la cual implica niveles de cuantización prácticamente infinitos) tiene 2.2dB, por lo que a partir de 8 niveles de cuantización resulta en una mejora insignificante. Está claro que mientras una decodificación tipo suave tiene un mejor desempeño que una rígida, también es cierto que el demodulador entrega 3 bits en lugar de 1, de modo que la complejidad del sistema incrementa.

2.4.6. Decodificador Viterbi

Se basa en un decodificador por máxima verosimilitud y en la recursividad que presenta el diagrama de Trellis. Este algoritmo no depende de la cantidad de símbolos que tenga la secuencia recibida, pudiendo ser infinita en teoría, aunque normalmente se restringe a una cantidad de símbolos. El algoritmo de esta decodificación consiste en hacer una comparación conforme la secuencia es recibida en los instantes t_i (con $i = 1, 2, 3...$) con todos los caminos de Trellis entrando a los diversos estados en esos momentos, dicha comparación deja unas marcas sobre los caminos de la *Distancia de Hamming* existente entre la secuencia recibida y cada uno de los caminos entrando a los diversos estados δ_{xy} donde x es el estado de origen y y es el estado al que se llega, posteriormente se calcula la *Distancia de Hamming Acumulada* para ese instante t_i que es la suma de los δ_{xy} desde el tiempo t_1 hasta t_i entrando a cada estado. Después se comparan, en base al estado final, las *Distancias de Hamming Acumuladas*, es decir se compara δ_{xy} con δ_{zy} para cada uno de los estados, eliminando aquellas *Distancias de Hamming Acumuladas* más grandes, de modo que solo quede un único camino entrando a cada estado, convirtiéndose la *Distancia de Hamming Acumulada* en la *Métrica del Estado* Γ_x, lo cual facilita el cálculo de las posteriores *Distancias de Hamming Acumuladas*. El proceso es repetido para los tiempos t_{i+1} posteriores y después de un número

de comparaciones y avances dentro del diagrama de Trellis (en práctica 4 o 5 veces la constante de restricción), los caminos van teniendo un mismo camino de origen del cual es posible extraer los datos codificados.

Implementación del decodificador

Célula Suma-Comparación-Selección (ACS) Supóngase que deseamos calcular la Métrica del Estado del estado a (Γ'_a) para un momento $t_(i+1)$, según el procedimiento, debemos calcular la *Distancia de Hamming Aculada* de los trayectos que entran a dicho estado y eliminar el mayor. Ahora bien, conocemos los estados desde los cuales se puede acceder al estado a, es decir a y c (esto lo sabemos porque conocemos el codificador), en lugar de realizar el cálculo de la *Distancia de Hamming Acumulada* para el tiempo t_{i+1} que requerimos, podemos hacer uso de la *Métrica del Estado* (Γ) de los estados por los cuales se accede al estado de interés, es decir Γ_a y Γ_c y sumarlo a la *Distancia de Hamming* de los trayectos $\delta_{aa'}$ y $\delta_{ac'}$ respectivamente, realizar la comparación entre los dos y posteriormente eliminar el mayor, convirtiéndose la *Métrica del Estado* (Γ'_a) de t_{i+1} en la *Métrica del Estado* actual.

Este mismo proceso es repetido para los demás estados y posteriormente se avanza en a la siguiente unidad de tiempo. Un diagrama de bloques que ilustra las operaciones hechas por la unidad de ACS se muestra en la figura 2.8

Como resultado de la célula ACS obtenemos la Métrica de Estado Γ, la cual es necesaria para el siguiente ciclo del decodificador, y además obtenemos la trayectoria asociada a dicha Métrica de estado. La trayectoria en sí no aporta información útil para la decodificación, no así, los bits que producen dichas trayectorias, que son los que el decodificador debiera almacenar.

Debido a que existen 2^{K-1} estados, la cantidad de bits que se deben almacenar son

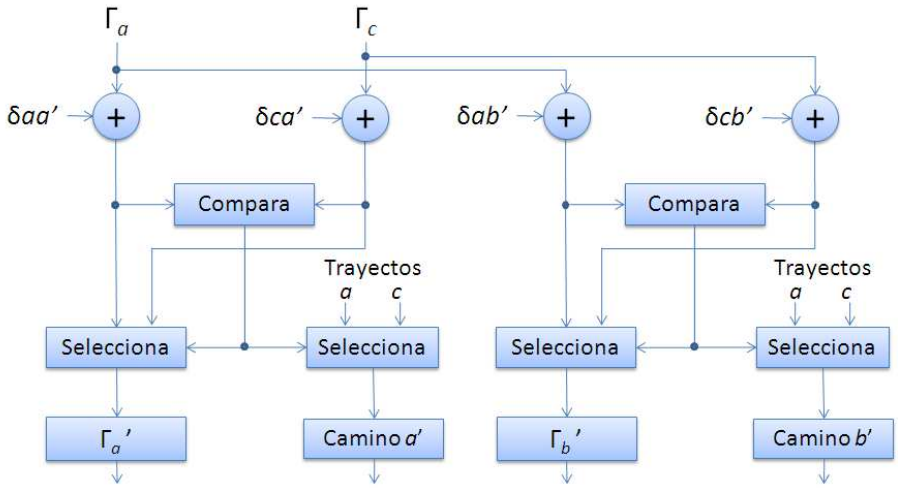


Figura 2.8: Diagrama a bloques de una célula ACS.

(2.23)

$$u = h2^{K-1} \tag{2.23}$$

Esto para tasas de transmisión de $1/n$, donde h es el número de unidades de tiempo que se desea almacenar (en la práctica 4 o 5 veces la constante de restricción K). Esto conlleva a que la complejidad del decodificador incremente exponencialmente con respecto a K.

2.4.7. Propiedades de los códigos convolucionales

Como en el caso de los códigos de bloques, para conocer la fuerza o resistencia de nuestro código requerimos conocer la distancia mínima existente entre las posibles secuencias, y debido a que los códigos convolucionales son lineales también, es igual de valido medir la Distancia Mínima entre las secuencias posibles y la secuencia de 0's, como ya se había demostrado para los códigos de bloques. La distancia mínima es

obtenida cuando el camino asociado a los datos transmitidos es eliminado de los caminos almacenados en la memoria, de modo que una forma de medir la distancia mínima para el camino asociado a la secuencia de 0's, solo basta con buscar el camino que diverge de la secuencia de 0's y vuele con la distancia menor. Como ejemplo tenemos la ilustración de la figura 2.9

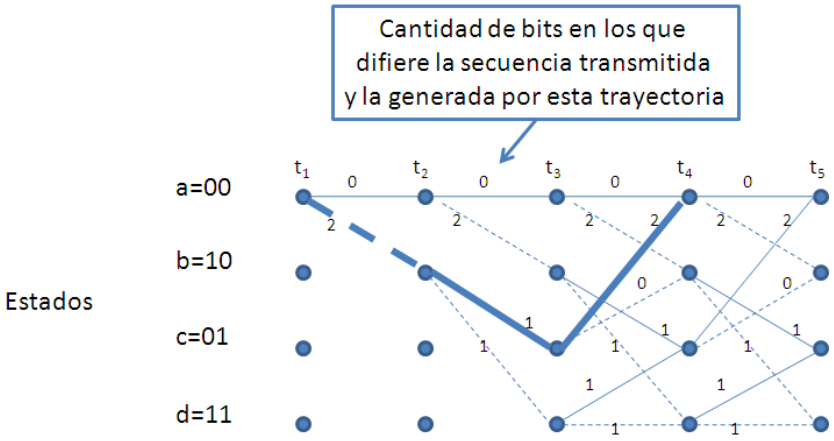


Figura 2.9: Distancia mínima de los códigos convolucionales obtenida a partir de la secuencia de 0's.

Para conocer las propiedades de distancias entre los códigos existe un método más analítico que simplemente contar la cantidad de bits en los que difieren las secuencias posibles, y consiste en el cálculo de una función de transferencia, en la cual los bits en los que difieren las secuencias son contabilizados en el exponente de una variable, es decir un trayecto que difiera en un bit será denotado como D^1, mientras que un trayecto con una diferencia de dos bits será denotado como D^2, y un trayecto que no difiera será denotado como $D^0 = 1$. Ahora bien, recordemos que para la medición de la distancia entre los códigos podemos utilizar como referencia la secuencia de 0's, de modo que nos interesan todas aquellas secuencias que inician en el estado a y vuelven a él, por lo que resulta conveniente establecer un estado $a = 00$ que indicaría el inicio de nuestra secuencia y un estado $e = 00$ que denotaría el regreso a la secuencia de 0s,

tal y como se ilustra en la figura 2.10

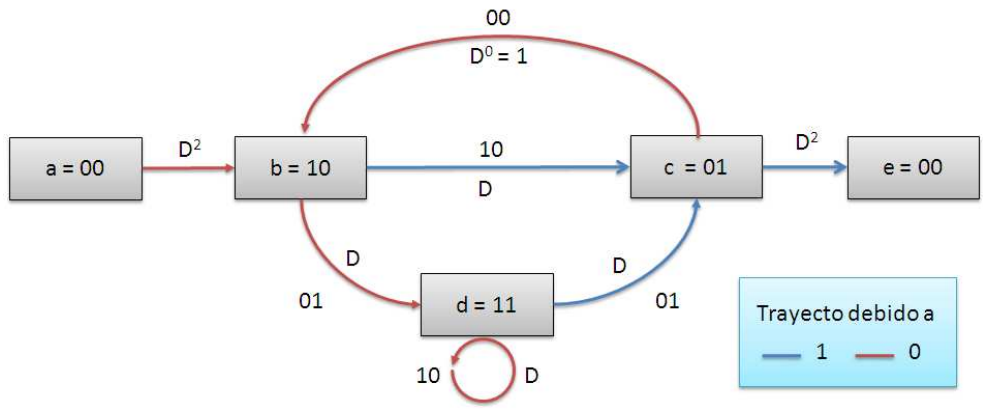


Figura 2.10: Diagrama de estados para obtener la función de transferencia.

De observar los valores que provocan las trayectorias entre los estados y compararlos con la trayectoria que genera la secuencia de 0s obtenemos los bits en los que estas difieren y por tanto los coeficientes de D, de modo que podemos escribir unas ecuaciones de estado con la forma (2.24)

$$X_b = D^2 X_a + X_c \tag{2.24}$$

donde X_b representa al estado b y sus entradas con sus respectivos bits en que difiere de la secuencia de 0's. Los demás estados quedan de la forma (2.25)

$$X_c = DX_b + DX_dX_d = DX_b + DX_dX_e = D^2X_c \tag{2.25}$$

Una vez teniendo las ecuaciones de los estados podemos definir una función de transferencia, relacionando la salida/entrada (X_e/X_a) (2.26)

$$T(D) = \frac{D^5}{1-2D} \tag{2.26}$$

Esta función de transferencia solo es útil si la expandimos de la forma (2.27)

$$\sum_{n=0}^{\infty} az^n = \frac{a}{1-z} \tag{2.27}$$

donde $a = D^5$ y $z = 2D$, de modo que:

$$T(D) = \frac{D^5}{1-2D} = \sum_{n=0}^{\infty} D^5(2D)^n = D^5 + 2D^6 + 4D^7 \ldots \tag{2.28}$$

Cada termino de la sumatoria indica la cantidad de trayectos (la constante que acompaña a la variable D) que tienen un determinado número de bits en los que difieren (el exponente de la variable).

La función de transferencia puede ser utilizada también para denotar el numero de trayectos que se recorrieron mediante una variable denotada L, o bien la cantidad de 1's que generaron el trayecto mediante una variable generalmente denotada N.

2.4.8. Capacidad de corrección de errores

Al igual que en los códigos de bloques, la distancia mínima es un indicador de la fuerza de los códigos, de modo que las capacidades correctivas están asociadas a esta de forma (2.3)

No obstante, esto no implica que el decodificador solo pueda corregir t errores a lo largo de una secuencia de longitud cualquiera, sino que esto depende de la forma en la que estos errores estén distribuidos, por lo que se deben observar los limites de probabilidad de error P_B mismos que están dados en (2.29)

$$P_B \leq \frac{dT(D,N)}{dN}\Big|_{N=1, D=2\sqrt[2]{p(1-p)}} \tag{2.29}$$

donde p es la probabilidad de error del símbolo de canal.

2.4.9. Códigos convolucionales bien conocidos

Para la elección de códigos convolucionales, se deben de tomar en cuenta un par de especificaciones. La primera de ellas es evitar aquellos que tengan la propiedad de *error catastrófico*, en segundo lugar, se desea una distancia de Hamming lo mayor posible, y en tercer instancia cualquier otro factor necesario para el sistema en particular, como la distribución de 1s y 0s, por ejemplo. El *error catastrófico* está definido como un evento en el cual una cantidad finita de símbolos pueden llevar a una cantidad infinita de errores en la decodificación. Esta propiedad puede ser observada cuando los polinomios que describen los puntos de suma tienen factores comunes. En la tabla 2.1 se observan una serie de códigos convolucionales bastante conocidos y utilizados, descritos por primera vez por Odenwalder en 1970 [35]. Estos códigos cumplen con las condiciones antes descritas, de evitar los errores catastróficos y capacidades correctivas altas para una determinada constante de restricción.

Tasa de transmisión	Constante de restricción	Distancia mínima	Vector del código (Octal)
1/2	3	5	(5, 7)
1/2	4	6	(15, 17)
1/2	5	7	(23, 35)
1/2	6	8	(53 ,75)
1/2	7	10	(133,171)
1/2	8	10	(247, 371)
1/2	9	12	(561, 753)

Tabla 2.1: Códigos convolucionales conocidos descritos por Odenwalder [35].

2.5. Otros tipos de codificaciones

2.5.1. Códigos de Hamming

Son un tipo de códigos de bloques con la cualidad de poder corregir un único error en cada código transmitido. Están conformados por n bits de paridad y k bits de mensaje $(n, k) = (2m-1, 2m-1-m)$, donde $m = 2, 3, \ldots$ Todos los códigos de Hamming tienen una distancia mínima de 3, lo cual corrobora el hecho de que tengan la capacidad de corregir un único error, o bien, detectar 2 errores.

2.5.2. Códigos BCH

Son una generalización de los códigos de Hamming para la corrección de múltiples errores, siendo estos códigos cíclicos. Estos códigos son proveídos en tablas que relacionan a un polinomio generador $g(X)$ con diversas propiedades como la longitud del código, tamaño del alfabeto, capacidad de corrección de errores. Este tipo de códigos normalmente son usados con alfabetos no binarios conocidos como Reed-Solomon. Cabe mencionar que ante un canal de ruido blanco aditivo del tipo Gaussiano (AWG), este tipo de códigos suelen tener un mejor desempeño a tasas de 1/3 o 3/4.

2.5.3. Códigos Reed-Solomon

Son un tipo de códigos cíclicos basados en los códigos BCH. Estos códigos utilizan un conjunto de m bits como símbolos, de modo que la secuencia codificada se trata de k símbolos constituidos de mk bits, mismos que son codificados en n símbolos, donde $n = k + r$ siendo r los símbolos de paridad. Al ser un símbolo igual a un conjunto de m bits, realmente la longitud de una palabra codificada consta de $(k + r)m$ bits.

Este tipo de codificación ha sido ampliamente utilizada en la codificación de audio en los CD debido a su capacidad de corrección de errores ante las explosiones de ruido (*burst noise*), ya que este tipo de códigos tienen la capacidad de corregir símbolos completos, pudiendo ser estos un único bit o bien todos los bits del símbolo erróneos. Nótese que un único error provoca que el símbolo sea erróneo, de modo que un canal que presente numerosos errores distribuidos suele degrada considerablemente el desempeño del sistema.

2.5.4. Códigos concatenados

La concatenación de dos códigos ha sido una técnica ampliamente utilizada actualmente debido a sus capacidades y/o propiedades, tales como la utilización de dos codificaciones simples para sustituir una extremadamente complicada, obteniendo resultados similares. Básicamente el éxito de esta técnica se basa en un primer código denominado código externo (*outer code*) y posteriormente el código interno *inner code*). La finalidad de estos es que el código interno realice una disminución de la cantidad de errores o altere la forma en la que estos aparecen de modo que el código externo se enfrente a otro "tipo de canal"más conveniente para este. Una combinación ampliamente utilizada es un código Reed-Solomon como código externo y un código convolucional con decisión suave como código interno. Otras variantes implican el uso de un entrelazado, el cual se encarga de mezclar los bits para obtener una distribución de errores a lo largo de una secuencia y no un grupo de errores ante canales que presenten explosiones de ruido (burst noise) [34]. Los turbo códigos son un tipo de códigos caracterizados por la utilización de códigos sistemáticos conectados entre sí por entrelazadores, y por una decodificación iterativa. Un esquema genérico de un turbo codificador sistemático se ilustra en la figura 2.11.

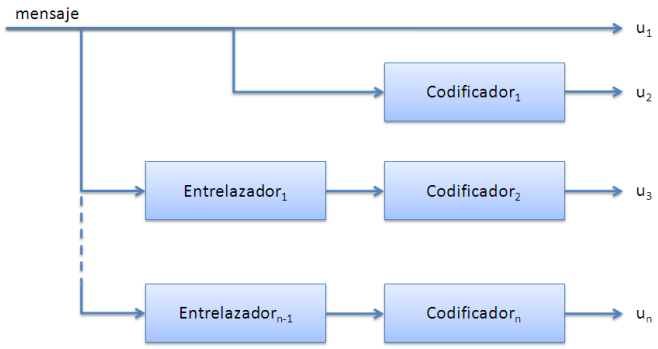


Figura 2.11: Esquema general de un turbo código.

2.6. Conclusiones del capítulo.

En este capítulo se han discutido los beneficios de utilizar codificaciones de canal. Se han descrito algunas de las codificaciones de canal existentes, su funcionamiento y principales características como tasa de transmisión, capacidades correctivas, y algunas de sus variantes. Se ha visto que su utilidad está limitada en gran medida a la aplicación en cuestión. Los códigos de bloques en general son sencillos, los cuales permiten una rápida implementación con mejoras significativas al sistema, no obstante, esta codificación exige que toda la trama haya llegado antes de poder iniciar la decodificación, por lo que grandes codificaciones con tramas grandes pueden introducir un retraso significativo, además de que se requiere una sincronización de trama precisa. Los códigos convolucionales pueden ofrecer mejores capacidades correctivas y un procesamiento continuo conforme arriban los datos, aunque su complejidad incrementa exponencialmente en función de la constante de restricción k. Las codificaciones concatenadas permiten modificar el canal aparente sobre el cual trabaja la codificación externa, una vez que la codificación interna interactuó con el canal real, esto permite que dos codificaciones relativamente sencillas puedan obtener un desempeño similar a una codificación especifica y compleja, no obstante la selección de el codificador tanto interno como externo debe ser adecuada para el canal.

Capítulo 3

Comunicaciones ópticas en la atmósfera.

3.1. Introducción

El inicio de las comunicaciones ópticas surgió a partir de la invención del láser (Light Amplification by Stimulated Emission of Radiation) en 1960. En sus principios se le intentó buscar aplicaciones en todo tipo áreas, como las comunicaciones, militares, medicas, óptica adaptiva, censado remoto, entre otras. Todas estas aplicaciones han enfrentado en mayor o menor medida las adversidades que el medio presenta, además de las propias de las ondas ópticas. El medio de propagación en la mayoría de los casos, es afectada por la turbulencia atmosférica que es un efecto producto de los cambios de índice de refracción del medio relacionada con los cambios de temperatura. Este fenómeno en algunos casos dificulta extremadamente las mediciones deseadas, esta junto con otras razones fue la que orilló a los astrónomos a enviar telescopios al espacio, como es el caso del Hubble, que fue lanzado en 1990 [46], ya que la turbulencia en trayectorias

verticales presenta aún más desafíos que la trayectoria horizontal.

3.1.1. Antecedentes históricos

A mediados del decimoséptimo siglo, la comunidad científica tenía la creencia de que la luz se propagaba en línea recta, atravesaba objetos transparentes y se reflejaba en superficies opacas. La ley de Snell ya era conocida, la difracción ya había sido descubierta por Grimaldi, y la doble refracción por Bartholinus. En 1670 Huygens indicó que las leyes de refracción y reflexión podían ser descritas mediante teoría de ondas. A inicios de 1800 fue cuando Young pudo medir la longitud de onda de diversas fuentes luminosas, y Fresnel observó que los efectos de difracción podían deberse a comportamientos de ondas. En 1850 la velocidad de la luz fue medida y aceptada como $3\text{x}10^8 m/s$. Posteriormente en 1873 Maxwell publico sus estudios sobre las ondas electromagnéticas. Hertz en 1887 verificó la teoría de Maxwell al descubrir el efecto fotoeléctrico, pero fue Einstein, mediante la teoría cuántica de Plank quien pudo explicar el fenómeno y denominar fotones a los paquetes discretos de energía de las ondas de luz.

Aunque la teoría cuántica ofrece una explicación más precisa de los fenómenos ópticos, la teoría clásica sigue siendo ampliamente utilizada para ciertas aplicaciones. Debido a que el espectro electromagnético es muy grande, longitudes de onda desde kilómetros hasta unidades X (Unidad $X = 10^{-13}$ metros). En la figura 3.1 se observa el espectro electromagnético conocido.

3.1.2. Modelos de las ondas ópticas

Existen diversos modelos que describen el comportamiento de la propagación de las ondas en el espectro visible e infrarrojo, estos se consideran que son transmitidos a lo largo del eje Z y algunos de ellos son:

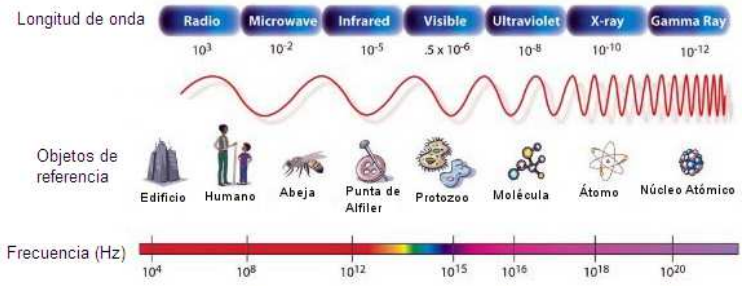


Figura 3.1: Espectro electromagnético conocido.

- Onda Plana: Supone una onda plana con amplitud (A) y fase (ϕ) constante, es utilizada para modelar la luz proveniente de las estrellas o astros fuera de la atmósfera, considera $z = 0$ (3.1).

$$U_0(x, y, 0) = A_0 e^{i\phi_0} \tag{3.1}$$

- Onda esférica: Supone una onda proveniente de una fuente con una pequeña apertura y un determinado ángulo de divergencia, considera $z = 0$ y está dada por (3.2)

$$U_0(x, y, 0) = \lim_{R \to 0} \frac{e^{ikR}}{4\pi R} \tag{3.2}$$

donde $R = |\mathbf{R}| = \sqrt[2]{x^2 + y^2 + z^2}$

En el análisis ingenieril, usualmente se utilizan estos dos modelos junto con otra serie de fenómenos que son de sumo interés, tales como la difracción, la atenuación atmosférica y la turbulencia. Estos fenómenos son considerados como lineales para la mayoría de los escenarios donde es necesario su estudio [25].

3.1.3. Difracción

La difracción es un fenómeno característico de las ondas que se basa en la desviación de estas al encontrar un obstáculo o al atravesar una rendija. La difracción ocurre en todo tipo de ondas, desde ondas sonoras, ondas en la superficie de un fluido y ondas electromagnéticas como la luz visible y las ondas de radio. Este fenómeno también provoca que la energía del haz enfocado disminuya, ya que parte del haz es redirigido. Un haz láser se ve severamente afectado por este fenómeno debido a la turbulencia atmosférica. El esparcimiento del haz debido a la difracción está en función de la longitud de onda (λ) del haz, de la forma del frente de fase (esférica, uniforme) y del tamaño de la apertura del emisor.

3.1.4. Efectos atmosféricos

Algunas condiciones meteorológicas pueden producir efectos adversos a la visibilidad humana, al igual que a la propagación de las ondas electromagnéticas, ya que son lo mismo, son la neblina, brisa, lluvia, partículas suspendidas en el medio, nieve, entre otras, algunos de los efectos adversos de estas han sido comentadas en el capítulo 1. Estas condiciones afectan de diversas maneras, pero han sido clasificadas en tres: Absorción molecular, variaciones de índice de refracción y dispersión. La absorción y la dispersión son producto de las partículas suspendidas en la atmósfera y por lo general se modelan como atenuaciones en la potencia la onda óptica. Mientras que las variaciones de índice de refracción provocan variaciones de irradiancia, esparcimiento del haz, incluso perdida de la coherencia espacial, es decir, el haz es redirigido fuera del objetivo.

Absorción y dispersión

La absorción ocurre cuando una molécula de gas suspendida en algún medio absorbe un fotón y lo transforma en energía cinética, y es de esta manera la atmósfera se calienta. Cabe mencionar que la absorción es función de los componentes del medio al igual que la longitud de onda del fotón como previamente se había mencionado [7, 18].

La dispersión, a diferencia de la absorción, no es perdida de energía a través de la conversión en energía cinética, si no la re dirección de la misma al atravesar ciertas partículas suspendidas en el aire. La dispersión puede ser clasificada en función del tamaño de las partículas que la ocasionen:

- Dispersión de Rayleigh: Es provocada por moléculas aire y neblina que son más pequeñas con respecto a la longitud de onda en cuestión. Este tipo de dispersión obedece a la ley de Rayleigh que indica que el coeficiente de dispersión es proporcional a λ^{-4}, de modo que para longitudes de onda más pequeñas, la dispersión es mayor.

- Dispersión de Mie: Esta es provocada por partículas o moléculas de tamaño relativamente similar o mayor a la longitud de onda en cuestión. Este tipo de dispersión tiende a afectar en mayor medida que la dispersión de Rayleigh.

En la figura 3.2 se observa a la derecha la dispersión de Rayleigh, mientras que a la izquierda se observa la dispersión de Mie.

3.1.5. Turbulencia atmosférica

Esta es ocasionada por el cambio de temperatura existente entre la superficie de la tierra y el aire de la atmósfera. Durante el día, la superficie de la tierra en general

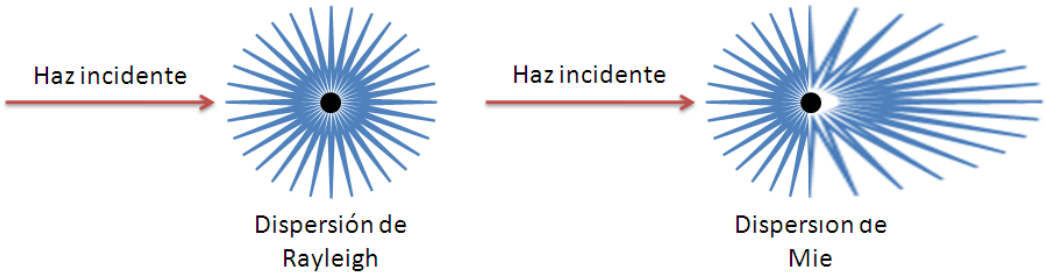


Figura 3.2: Derecha dispersión de Rayghleigh. A la izquierda dispersión de Mie.

suele estar más caliente que el aire en la atmósfera, esto ocasiona que los rayos de luz que atraviesan paralelamente la tierra se vean dirigidos hacia arriba. Mientras que en la noche sucede lo contrario. Este tipo de fenómenos producen distorsiones en el frente de onda, dispersando el haz, cambiando el centro de incidencia del mismo, y redistribuyendo la energía del haz a lo largo de una sección transversal provocando fluctuaciones en la irradiancia. La turbulencia atmosférica además de limitar la visibilidad, también afecta la coherencia espacial del haz observado mientras que este se propague a través de la atmósfera. Este fenómeno limita en gran medida la colimación o enfoque de un haz reduciendo en gran medida la potencia recibida del mismo. Adicionalmente las corrientes de aire crean mezclas de aire con diferentes gradientes de temperatura y por tanto diferentes índices de refracción. Este fenómeno ha sido ampliamente estudiado y es modelado probabilísticamente en ciertos rangos definidos por la estructura constante de índice de refracción C_n^2, la cual es una medida de la magnitud con la que el índice de refracción varía en una determinada trayectoria medida en $m^{-2/3}$. Esta estructura es usada comúnmente para definir el régimen de turbulencia que presenta un determinado medio. Los regímenes de turbulencia se denominan, turbulencia débil para valores de $C_n^2 < 10^{-17}m^{-2/3}$, mientras que para $C_n^2 > 10^{-13}m^{-2/3}$ se denomina turbulencia fuerte [25]. Entre los modelos utilizados se encuentran la distribución K y Gama-Gama para modelos de turbulencia de moderada a fuerte, mientras que la distribución log-normal

preferentemente en de turbulencia débil [22, 12, 30].

En base a estos estudios y modelos se ha buscado reproducir estas condiciones para efectos de experimentos controlados, como tal es el trabajo de Sánchez López, Mohammad Abtahi, Abtahi [39, 31, 1], quienes utilizan una caja capaz de generar turbulencia atmosférica al mezclar el aire caliente dentro de la caja con un flujo de aire frío proveniente del exterior de la misma. Este tipo de herramientas, una vez caracterizadas pueden ser utilizadas para efectos de pruebas y medir el desempeño de ciertas técnicas.

3.2. Consideraciones de las codificaciones de canal

Las codificaciones de canal tienen diversas propiedades y características, debiéndose estas a la naturaleza de las mismas. Como ejemplo de lo anterior tenemos que las codificaciones de bloques, que en general resultan ser sencillas de implementar proveyendo al sistema de un mejor desempeño ante los errores introducidos por el canal. Los códigos LDPC (Low-Density Parity Check) son un tipo de códigos de bloques inventados por Gallager en 1962, los cuales, debido a limitantes de procesamiento no habían sido considerados[13]. Este tipo de códigos han ganado popularidad debido a su prometedor desempeño ante la corrección de errores. Ahora bien, un parámetro más que resulta de interés en algunos sistemas de comunicaciones en tiempo real es la latencia. Esta se define como el tiempo mínimo requerido para la transmisión de un determinado mensaje, independientemente del tamaño del mismo, de tal manera que esta contempla el tiempo que lleva el adecuar el mensaje antes de transmitirlo, la transmisión del mismo y los procesos que conlleve el procesar el mensaje antes de poder entregarlo al destino [29, 45]. Esto nos lleva a analizar la naturaleza de los códigos de bloques, ya que estos exigen tener todo un bloque de datos por parte de la fuente para iniciar el proceso de codificación y de igual manera que todo el bloque haya llegado al decodificador para

poder realizar el proceso de decodificación, lo cual incrementa la latencia del sistema pudiendo esto llegar a ser prohibitivo para algunos sistemas. Maiya en su trabajo [29], compara el desempeño de los códigos convolucionales con los códigos LDPC en función de la latencia del proceso de decodificación, encontrando que para latencias bajas los códigos convolucionales tienen un mejor desempeño, mientras que para latencias altas los códigos LDPC tienen un mejor desempeño. De igual manera, Henh [16] indica que al comparar códigos LDPC con los códigos convolucionales en el proceso de decodificación, los códigos convolucionales son los que tienen una menor latencia. Zhu también realiza una comparación entre códigos de bloques, convolucionales y turbo códigos para diferentes tamaños de entrelazados en un canal que presenta turbulencia atmosférica débil ($\sigma_x = 0.2$). Para su estudio utiliza modulación en intensidad con detección directa (MI/DD) [50]. En general el desempeño de los códigos mejoró conforme el tamaño del entrelazado K se incrementaba hasta que KT se aproximaba a la duración del desvanecimiento τ_0, siendo T el tiempo de bit. Para los turbo códigos se observó que el desempeño seguía mejorando incluso después de este límite, no obstante la complejidad y el retraso que pudiera sufrir la codificación y decodificación sigue siendo un factor que podría ser prohibitivo en algunos sistemas. En la tabla 3.1 se observan los resultados de simulaciones de códigos de bloques (Hamming 7,4), códigos convolucional sistemáticos (code rate 1/3) y turbo códigos con entrelazado de K bits hechas por Zhu.

Codificación de canal	Complejidad	dBm ($P_e = 10^{-8}$)	Comentarios
Código bloques (Hamming 7,4)	Media	15.15	$T_b = 0.001\tau_0$
Códigos convolucionales	Media	11.71	$T_b = 0.001\tau_0$
Turbo Códigos $K = 100$	Alta	10.47	Entrelazado $K = 100$
Turbo Códigos $K = 10000$	Alta	9.07	Entrelazado $K = 10000$

Tabla 3.1: Comparación de codificaciones de canal en un canal con turbulencia atmosférica débil

3.3. Conclusión del capítulo

De la sección anterior observamos que los códigos convolucionales tienen una mejora de 3.44 dBm con respecto a los códigos de bloques, mientras que los turbo códigos con entrelazado de 100 y 10000 bits superan a su vez a los códigos convolucionales con 1.24 dBm y 2.64 dBm respectivamente. Si bien es verdad que los turbo códigos tienen un mejor desempeño, también requieren un mayor nivel de complejidad dado que implican el uso de otras codificaciones de canal concatenadas. De la sección anterior podemos concluir que una buena combinación entre desempeño en cuanto a corrección de errores y latencia son los códigos convolucionales. Estos pueden mejorar su desempeño en los canales que presenten desvanecimiento mediante la implementación de un entrelazador de profundidad K en función del tiempo de bit y el tiempo de desvanecimiento del canal en cuestión. Adicionalmente, si el canal óptico inalámbrico presentara un desvanecimiento muy fuerte (asociado un régimen de turbulencia moderado o fuerte), el uso de técnicas como la diversidad espacial pueden reducir estos niveles de desvanecimiento [33, 41]. Adicionalmente los códigos convolucionales también un buen punto de partida para la implementación de turbo códigos, los cuales, como se comentó en la sección anterior, tienen un mejor desempeño para la corrección de errores, siempre y cuando la complejidad y la latencia no sean un impedimento para el sistema en cuestión.

Capítulo 4

Implementación del codificador y decodificador.

4.1. Introducción

Un posible escenario para la implementación del sistema se ilustra en la figura 4.1, en el cual se observa una fuente de datos cualesquiera, un elemento codificador que proveerá al sistema un mejor desempeño, una etapa de modulación para la transmisión de datos a través del canal, el cual supone diversas perturbaciones incluida la turbulencia atmosférica, mientras que como receptor se supone una etapa de demodulación, una etapa de procesamiento que permita adecuar la señal para la siguiente etapa que es de decodificación, misma que permitirá obtener los datos previamente codificados. Cabe mencionar que a pesar de que el sistema presenta diversos tipos de perturbaciones, solo es de interés el desempeño del sistema ante turbulencia débil, debido a que existen técnicas y métodos complementarios para compensar esas perturbaciones o bien acotar los regímenes de turbulencia en cierta medida, como previamente se ha comentado en

los capítulos anteriores.

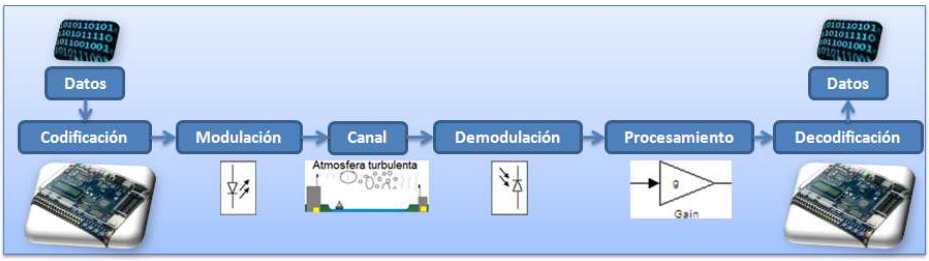


Figura 4.1: Diagrama a bloques del sistema a implementar.

4.1.1. Implementación general

En el capitulo anterior se definió el uso de un código convolucional como codificación de canal, no obstante son necesarios otros elementos para el desarrollo de un sistema integral y más funcional, como lo es un sistema de recuperación de reloj para poder establecer el momento preciso de muestreo de los datos, estos sistemas de recuperación de reloj suelen tomar un tiempo que no suele ser constante debido a diversos factores, por lo que es necesario un esquema de sincronización de trama, el cual permita diferenciar la trama de sincronía del inicio de transmisión de datos.

Todo el sistema de codificación ha sido implementado en una plataforma FPGA Cyclone II EP2C20F484C7, mientras que el sistema de decoficación ha sido implementado en una plataforma FPGA Cyclone II EP2C35F672C6, misma que tiene mayor capacidad en cuanto a unidades lógicas, esto debido a que la complejidad del sistema radica en el decodificador, y es este el que requiere una mayor cantidad de recursos como se ilustra en las siguientes secciones.

La etapa de sincronización en el decodificador ha sido implementada parcialmente en el dispositivo FPGA, ya que este requiere una señal de reloj a partir de la cual puede funcionar, es por ello que un circuito externo de recuperación de reloj ha sido

implementado, el cual se describirá con más detalle más adelante. Todo el comportamiento de los dispositivos FPGA ha sido descrito mediante VHDL, que es un lenguaje descriptivo portable a otras plataformas FPGA, para futuras aplicaciones y/o reciclar componentes. No obstante algunos elementos definidos son específicos de los FPGA utilizados y estos no pueden ser portados a otras plataformas o sistemas, por lo que una adecuación a esos elementos es necesaria. Otros elementos pueden ofrecer mejoras considerables al sistema, no obstante no se han implementado en esta instancia, no obstante se comentaran algunos en la sección de trabajo futuro.

En la figura 4.2 se ilustran los componentes de hardware que fueron utilizados para la implementación general. Estos son dos dispositivos FPGA, codificador y decodificador, además de un circuito integrado de amarre de fase (Phase-Locked Loop), que es un elemento que se encarga de sincronizar la fase de una señal de salida con respecto a una señal de entrada.

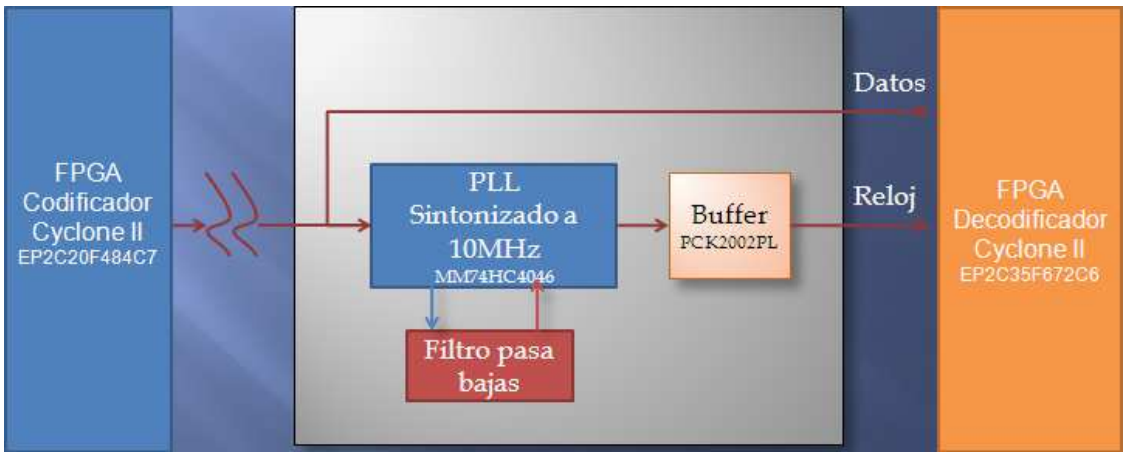


Figura 4.2: Elementos de hardware utilizados.

4.1.2. VHDL y FPGA

VHDL

VHDL (very-high-speed integrated circuits hardware description lenguaje o lenguaje de descripción de hardware de circuitos integrados de muy alta velocidad), es un lenguaje para la descripción de hardware que suele ser usado para modelar sistemas digitales a diferentes niveles, que pueden ir desde complejos algoritmos hasta programar compuertas individuales, o una combinación de ambos. Este lenguaje permite definir entidades que pueden ser reutilizadas cuantas veces sea conveniente dentro de otras entidades, lo que genera jerarquías dentro del sistema. Este lenguaje permite una programación concurrente, lo cual significa que varias instrucciones son ejecutadas simultáneamente, al igual que permite la programación secuencial, es decir, instrucciones una después de otra, todo esto en función de cómo se describa el hardware [4].

Características

- Este lenguaje suele ser utilizado para describir los componentes digitales modelados en sistemas CAD (Computer-Aided Desing), lo que permite el intercambio de información entre diversos sistemas CAD.
- Este no es un lenguaje orientado a un determinado tipo de tecnología, por lo que puede ser portada tanto de un tipo de hardware de un manufacturero a otro, o incluso de un modelo de algún sistema CAD a un determinado hardware, a excepción de algunos elementos específicos de un hardware o herramientas especiales.
- Soporta diseño de sistemas síncronos o asíncronos
- Lenguaje legible tanto para humanos como para maquinas.
- Maneja librerías publicas y el lenguaje no tiene propietario.

- Se pueden diseñar maquinas de estados finitos, complejos algoritmos, modelado de ecuaciones boleanas, diversos sistemas embebidos paralelos.
- Lenguaje estandarizado por la IEEE y ANSI.
- El uso de entidades, componentes, funciones, paquetes, permiten que el diseño a gran escala sea fácil.

VHDL es un lenguaje que permite describir el comportamiento de dispositivos digitales de una forma fácil y estandarizada, no obstante existen comportamientos que pueden ser modelados y simulados sin que necesariamente puedan ser implementados, debido a restricciones que pueda tener un hardware en particular. Como ejemplo, un modelo puede contemplar la detección de ambos flancos de una determinada señal para su correcto funcionamiento, lo cual no es directamente realizable por un determinado hardware.

FPGA

FPGA es el acrónimo de Field-Programmable Gate Array y es un dispositivo semiconductor que contiene bloques de lógica cuya interconexión y funcionalidad puede ser configurada mediante un lenguaje de descripción especializado después de ser manufacturado. Este dispositivo no está restringido a utilizar un determinado set de instrucciones, ya que sus componentes internos son reconfigurables, permitiendo así que se pueda obtener un sistema muy flexible. Estos dispositivos permiten implementar cualquier función que un circuito integrado de aplicación específica (application-specific integrated circuit ASIC) pueda desarrollar, mientras que a su vez puede ser actualizado sin importar que ya se encuentre instalado. Los FPGA de hoy en día, tienen integrado diversos elementos embebidos como SRAM, entradas/salidas de alta velocidad, mezcladores,

a los cuales se accede mediante un software específico del manufacturero. Las unidades lógicas pueden ser configuradas para realizar complejas funciones combinacionales. Adicionalmente, las unidades lógicas de la mayoría de los FPGA tienen elementos de memoria que facilitan la implementación de flip-flops o grandes elementos de memoria.

Actualmente los FPGA son utilizados para el rápido desarrollo de prototipos, mismos que posteriormente se convertirán en circuitos integrados discretos, o bien el prototipado de microcontroladores. También suelen ser utilizados para aplicaciones específicas en las que no existen comercialmente elementos que lleven a cabo una función determinada o bien, se pretenda realizar alguna optimización o prueba en particular.

4.1.3. Código de línea

Los códigos de línea son una forma en particular de representar la información en base a pulsos que dan a la señal transmitida ciertas características. Esta puede facilitar la transmisión de datos en función del sistema que se proponga y las características del mismo. Algunas de las características que suelen ser consideradas al momento de implementar una codificación de línea son:

- Presencia o ausencia de un nivel de DC u *offset*.
- Densidad espectral de potencia a 0 Hz, ya que las constantes no suelen representar información.
- Ancho de banda.
- Facilidad para obtener sincronía de reloj.

A continuación se muestra una tabla comparativa entre diferentes códigos de línea (tabla 4.1) y algunas de sus propiedades, mientras que en la figura 4.3 se observa la

forma de onda y el espectro producido por estas [14].

Código de línea	Obtención de señal de reloj	Primer nulo	Acoplamiento CA
Unipolar NRZ	Difícil	f_0	No
Unipolar RZ	Fácil	$2f_0$	No
Polar NRZ	Difícil	f_0	No
Polar RZ	Rectificado	$2f_0$	No
Dipolar OOK	Fácil	$2f_0$	Si
Manchester	Difícil	$2f_0$	Si

Tabla 4.1: Comparación entre códigos de línea

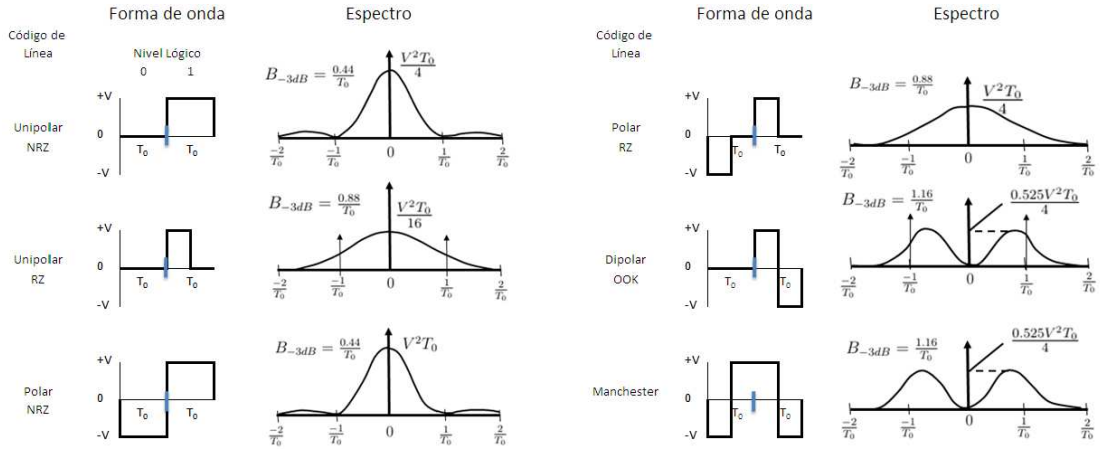


Figura 4.3: Formas de onda y espectro de diversos códigos de línea.

Dado que el sistema supone una transmisión óptica no se pueden utilizar señales con valores negativos, además dado que el sistema pretende un rápido desarrollo se ha elegido un código de línea del cual se pueda obtener la señal de reloj fácilmente, por lo que se ha elegido Unipolar Retorno Cero. Si bien este tipo de codificación duplica el ancho de banda requerido, el sistema de comunicación óptica inalámbrica dispone de un gran ancho de banda, ya que no convive con ningún otro sistema y el ancho de banda disponible es muy grande.

4.1.4. Esquema de recuperación de reloj

El código de línea unipolar retorno cero presenta una fuerte componente de reloj, tal y como se observa en el espectro de la señal que se ilustra en la figura 4.3, esto permite una fácil recuperación de la señal de reloj. En este caso, la recuperación de la señal de reloj se ha hecho mediante un circuito PLL el cual está formado por un comparador de fase, un oscilador y un filtro pasa bajas. Este circuito integrado genera una señal de fase idéntica a la señal entrante con el oscilador local a partir de un lazo de retroalimentación y un filtro pasa bajas. Esta señal generada presenta un ligero desfase ϕ con respecto a la señal de entrada que resulta ser nuestra señal de datos , mismo que es compensado por un PLL interno del FPGA. Este PLL interno, permite además de compensar el desfase ϕ, generar señales de frecuencia múltiplo de la señal de reloj recuperada mediante el circuito integrado que serán necesarias para la correcta decodificación del código de línea. El uso de un circuito integrado PLL se debe a que el PLL interno del FPGA tiene dificultades al tratar con señales cuya fase varíe abruptamente o haya ausencia de la señal de reloj. El proceso de decodificación del código de línea implica revisar el estado lógico de los datos recibidos a la misma velocidad que la señal de reloj, o bien, realizar una comparación en ambos flancos de la señal de reloj. Para algunos FPGA esto no es posible, es por ello que es necesaria la utilización de una señal del doble de frecuencia, para que en esta señal de se procesen las comparaciones en un mismo flanco, tal y como se ilustra en la figura 4.4.

El circuito integrado utilizado fue un PLL HC4046M cuyo esquemático se ilustra en la figura 4.6, mientras que la implementación física se muestra en la figura 4.5. Este PLL ha sido sintonizado a 10 MHz de acuerdo con la hoja de especificaciones provista por el fabricante con los elementos pasivos que se le han conectado. Los elementos PCK2002PL son buffers para proteger las salidas y entradas a los dispositivos FPGA.

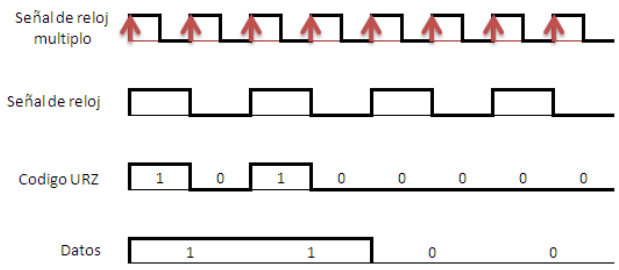


Figura 4.4: Forma de onda del código de línea URZ y la señal de reloj requerida para decodificar.

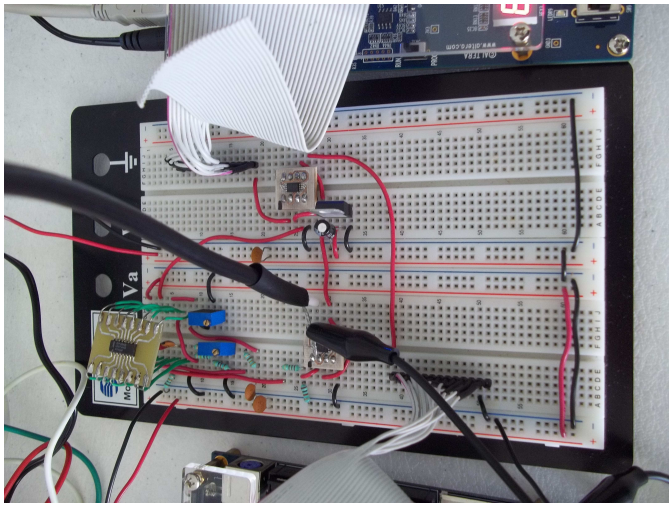


Figura 4.5: Implementación del sistema de recuperación de reloj.

4.2. Implementación del sistema

El sistema base considerado consta de codificación de canal, código de línea y una trama de sincronía para la recuperación de la señal de reloj implementados sobre una plataforma FPGA, utilizando VHDL como lenguaje de programación. Este lenguaje, como ya se ha comentado, es perfectamente portable a cualquier otro dispositivo FPGA, excepto en los recursos específicos de los FPGA Cyclon II *EP2C35F672C6* (decodificador) y , los cuales fueron utilizados para la implementación. La codificación de canal consta de un codificador y decodificador convolucional. Para el decodificador se ha uti-

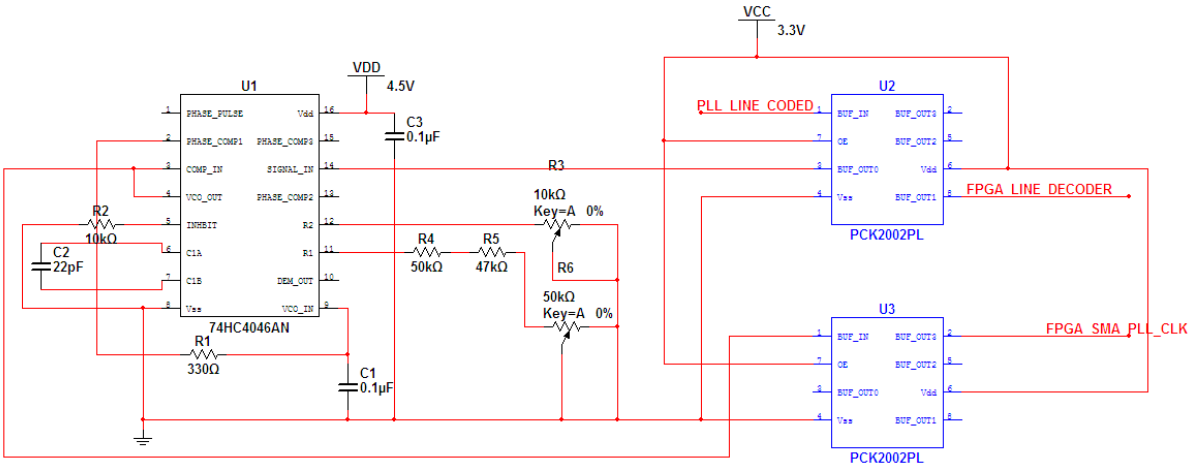


Figura 4.6: Esquemático de la implementación del sistema de recuperación de reloj.

lizado el algoritmo de Viterbi implementado con células ACS (Add-Compare-Select), dichas células ACS permiten asociar fácilmente las métricas de estado con los respectivos historiales de trayectorias dentro del diagrama de Trellis. El sistema utiliza una señal de reloj base de 10 MHz, la cual se divide y se multiplica en función de la operación que se realice. La selección de esta frecuencia se basa en el rango de operación de los elementos PLL, tanto circuito integrado HC4046M (0 Hz, a 14 MHz), como del PLL interno del FPGA (10 MHz a 200 MHz). Con esta señal de reloj base, se puede codificar datos a una velocidad de 5 Mbits/s, debido a que la tasa de codificación es $\frac{1}{2}$.

A continuación se hace una descripción de los componentes básicos para la implementación de este sistema en base a un código convolucional descrito por los vectores $V_1 = [111]$ $V_2 = [101]$.

4.2.1. Implementación general codificador y sus componentes

La entidad principal o *Top Entity* como se le suele llamar es **TopCoder** y es aquella que engloba a las demás y describe la interacción entre las mismas. Dicha entidad

considera como puertos las siguientes señales:

Entidad: **Codificador** (In: *Clk, Reset* Out: *CodLineOut*)

Clk: Proveniente del cristal en la tarjeta de desarrollo (50 MHz).

Reset: Utilizada para reiniciar el sistema.

CodeLineOut: Salida del sistema, ya sea la trama de sincronización o los datos codificados convolucionalmente.

El bloque **TopCoder** se puede observar en la figura 4.7.

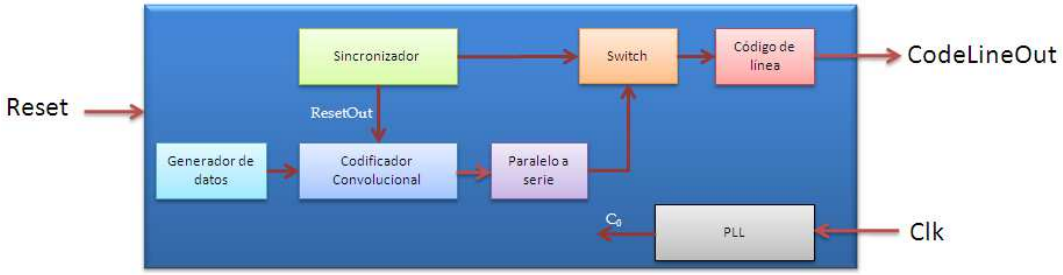


Figura 4.7: Top entity del sistema codificador.

Componentes

A continuación se da una descripción de los componentes utilizados en el Top Entity.

PLL:

Entidad: **PLL** (In: *areset, inclk0* Out: *c0, locked*)

Este bloque permite la generación de la señal de reloj de 10 MHz en el puerto *c0* a partir del cristal de la tarjeta de desarrollo de 50 MHz. Este componente tiene un tiempo de respuesta que es monitoreado con la señal *locked* y funciona como *reset* activo en bajo para el resto de los componentes. El puerto *areset* es únicamente un

puerto al *reset* del Top Entity.

Generador de datos:

Entidad: **GenDatos** (In: *Reset, Clk* Out: *Datos*)

Es una serie de n registros cuya finalidad es proveer una secuencia pseudoaleatoria de longitud $2^n - 1$ bits. Es implementado mediante una serie de registros retroalimentados tal y como se ilustra en la figura 4.8 [42].

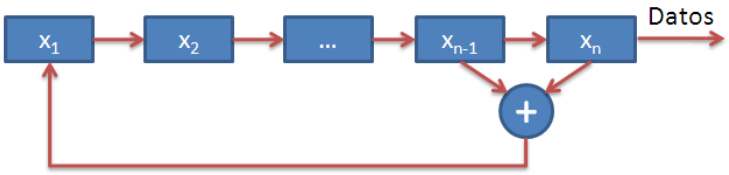


Figura 4.8: Generador de secuencia pseudoaleatoria de n bits.

Los registros utilizados son instancias de flip-flops tipo D (Entidad **Reg**), mismos que son inicializados en cero a excepción del registro X_1, que es inicializado en 1 (Entidad **Reg1**) para que este pueda dar inicio a la secuencia pseudoaleatoria.

Sincronizador:

Entidad: **Sincronizador** (In: *Reset, Clkwork* Out: *ResetOut, Data*)

Es un bloque encargado de enviar una trama de sincronización a través del puerto Data y avisar al bloque **CodConvV1** cuando puede este empezar a operar por medio de la señal *ResetOut*. La trama de sincronización consta de dos partes: Secuencia de 1's y secuencia de inicio. La secuencia de 1's tiene una duración de 2000 ciclos de reloj. Esta secuencia de unos es traducida por el bloque **CodigoLinea** como la señal de reloj durante dicho periodo, y provee tiempo suficiente al circuito integrado PLL (HC4046M) para recuperar la seña de reloj de 10 MHz y posteriormente el bloque **PLL** del decodificador genere la respectiva señal de reloj para sus operaciones. La

secuencia de inicio es una palabra de longitud de 13 bits que identifica cuando la señal de sincronización ha acabado, de manera que el decodificador al detectar esta secuencia sabe que los datos siguientes corresponden a los datos codificados. La palabra de 13 bits (1111100110101) es generada dentro del bloque Barker. Las secuencias Barker son una secuencia de $+1$'s y -1's que tienen una autocorrelación fuera del origen muy baja, no obstante para señales meramente positivas tal efecto no se cumple, de modo que para este caso, únicamente es una secuencia conocida que indica el inicio de los datos [15].

Switch:

Este no es propiamente un bloque implementado mediante una entidad, si no un segmento de código dentro del Top Entity que realiza el conmutado entre la secuencia de sincronización y los datos codificados.

Código de Línea:

Entidad: **CodigoLinea** (In: *Clkwork, dataIn* Out: *dataOut*)

Este bloque se encarga de la implementación del código de línea Unipolar Retorno a Cero, que si bien no es óptimo, presenta una facilidad para la recuperación de la señal de reloj. Básicamente este bloque realiza la operación AND entre los datos de entrada *dataIn* y la señal de reloj *Clkwork*, arrojando el resultado en el puerto de salida *dataOut*. La señal de reloj es de 10 MHz, mientras que los datos ingresados tienen una velocidad de 10 Mbits.

Codificador convolucional:

Entidad: **CodConvV1** (In: *clk, reset, dataIn* Out: *CodConv*)

Este bloque realiza el proceso de codificación convolucional utilizando tres registros de memoria y dos puntos de suma descritos por los vectores $V_1 = [111]$, $V_2 = [101]$. Esto implica que cada dato que ingrese por *dataIn* permanecerá en el codificador tres

ciclos de reloj *Clkwork* o bien 6 ciclos de *clk*, a la vez que producirá dos bits codificados en *CodConv* por cada ciclo de reloj *Clkwork* hasta salir. Para ello, este bloque utiliza una instancia de un divisor de frecuencia **DivFrecV1** que divide la frecuencia de la señal de reloj clk a la mitad. Los registros de memoria que almacenan los datos son instancias de flip-flops tipo D (Entidad **Reg**). En la figura 4.9 se ilustra esta relación entre la señal de reloj de entrada *clk* (de 10MHz), la señal de reloj *clkwork* (de 5MHz) producida por la instancia **DivFrecV1**, los datos de entrada *dataIn* (Bit 1, 2, 3...) y los datos de salida *CodConv* ($U_{11}U_{12}$, $U_{21}U_{22}$, $U_{31}U_{32}$... donde U_{11} y U_{12} son los bits codificados en un mismo ciclo de reloj *clkwork*). Esto es ilustrado en la figura 4.9.

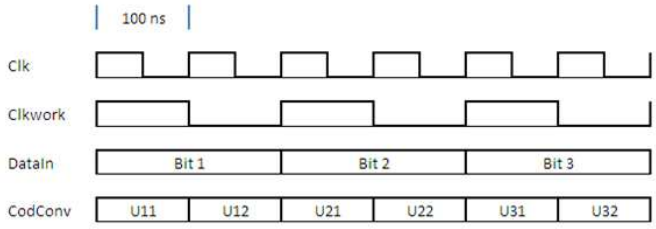


Figura 4.9: Señales dentro del bloque CodConvV1.

Convertidor de paralelo a serie:

Entidad: **ParallelToSerie** (In: *P2S, clk, reset* Out: *DataOut*)

Este bloque se encarga de convertir los datos de paralelos provenientes del codificador a datos serie para poder posteriormente ser codificados por el código de línea.

4.2.2. Implementación general decodificador y sus componentes

La entidad principal del decodificador es **TopDecoder**.

Entidad: **TopDecoder** (In: *Clk, DataIn, Reset* Out: *TestLineDecoded, TestDataIn,*

TestSincFlag, TestBitPack1, TestBitPack2, TestClk Buffer: *DataOutClk, DataOut*)

Clk: Es la entrada a la señal de reloj de 10 MHz recuperada por el circuito integrado(CI) PLL. Esta señal es posteriormente utilizada por la entidad **PLL** para generar la señal de reloj necesaria para la decodificación de línea.

DataIn: Puerto de entrada para los datos en forma serie, puede ser monitoreada por la señal de salida *TestDataIn*.

Reset: Utilizada para reiniciar el sistema.

DataOutClk: Señal de reloj para los datos decodificados.

DataOut: Señal de salida para los datos decodificados.

TestLineDecoded: Señal de prueba para monitorear los bits decodificados por el decodificador de línea.

TestDataIn: Señal de prueba para monitorear los datos de entrada.

TestSincFlag: Señal de prueba para monitorear la bandera de sincronización. Esta bandera es controlada por la entidad **DecoderBarker** y se utiliza como reset que da inicio a la entidad **DecConv**.

TestBitPack(1,2): Señales que permiten monitorear el arreglo de bits en paralelo que serán procesados por el decodificador convolucional (específicamente la entidad **DeltaXY**).

TestClk: Señal de prueba de la señal de reloj recuperada por el CI PLL.

Display7Segmentos: Salida para 4 Displays de 7 segmentos c/u (Activos en bajo para la tarjeta DE2 de Altera), estos son utilizados por un elemento verificador de bits decodificados correcta o erróneamente según la implementación elegida.

El bloque **TopDecoder** se puede observar en la figura 4.10.

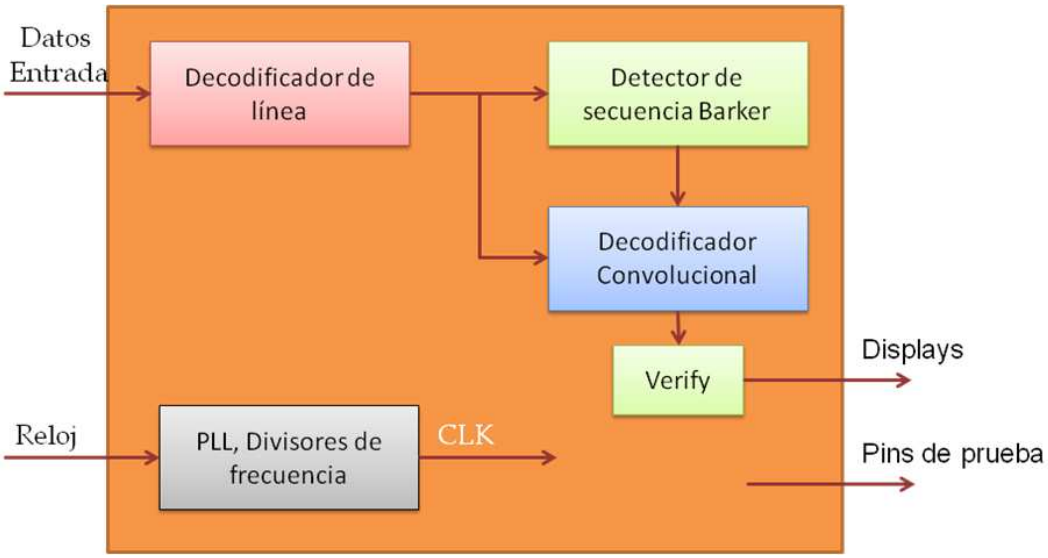


Figura 4.10: Top entity del sistema decodificador.

Componentes

PLL:

Entidad: **PLL** (In: *areset, inclk0* Out: *c0, locked*)

Este bloque es utilizado para generar una señal del doble de la frecuencia de reloj (genera una señal de 20 MHz a partir de una de 10 MHz), con la finalidad de poder decodificar el código de línea URZ, además de ajustar el desfase que pueda haber entre la señal de reloj y la señal de datos recibida por el FPGA. La señal de reloj producida por este elemento es entregada en el puerto *c0*. Este componente tiene un tiempo de respuesta que es monitoreado con la señal *locked* y funciona como *reset* activo en bajo para el resto de los componentes. El puerto *areset* es únicamente un puerto al *reset* del Top Entity.

Decodificador de línea:

Entidad: **LineDecoder** (In: *clk, reset, DataIn* Out: *DataOut*)

Este bloque se encarga de decodificar el código de línea, de tal manera que el resto de los bloques puedan trabajar directamente con sus respectivos datos.

Divisor de frecuencia:

Entidad: **DivFrecV1** (In: *ClkIn, reset* Out: *Clkwork*)

Se encarga de dividir la frecuencia de la señal de reloj obtenida por el PLL (20 MHz) y reducirla a (10 MHz). Esta señal de reloj es utilizada por el sincronizador (**DecoderBarker**) y el buffer de entrada (**BufferIn**).

Sincronizador:

Entidad: **DecoderBarker** (In: *clk, reset, DataIn* Out: *SincFlag*)

Este elemento se encarga de detectar la secuencia de sincronización o secuencia Barker utilizada para indicar el fin de la trama de sincronía y de este modo comenzar con la decodificación de los datos. Este componente tiene una señal de salida SincFlag que funciona como reset para el resto de los elementos.

Buffer de entrada:

Entidad: **BufferIn** (In: *BIn, clk, reset* Out: *BOut, ClkworkOut*)

Se encarga de convertir el código serie a código en paralelo para su fácil manejo dentro del decodificador. Además genera una señal *ClkworkOut* de la mitad de la frecuencia de la señal de reloj *clkwork* (5 MHz), esta señal indica los tiempos de operación relacionados a los datos.

Verificador:

Entidad: **Verify** (In: *Clk, reset, DecodedData* Out:*Display7Segmentos*)

Este elemento se encarga de sincronizar la secuencia decodificada con la secuencia generada en el codificador considerando el tiempo de decodificación, posteriormente contabiliza la cantidad de errores o aciertos en la decodificación y los muestra en cuatro displays de 7 segmentos activos en bajo, según haya sido configurado.

Decodificador:

Entidad: **Decoder** (In: *clkwork, reset, BitPack* Out: *DataOut, clwkout*)

Este elemento es el que mayor complejidad presenta de toda la implementación, por lo que procederemos a analizarlo más detalladamente junto con las entidades que lo conforman.

La arquitectura del decodificador está formada principalmente por cuatro entidades: **DeltaXY**, **ACS**, **PathHistory**. Estos tres elementos junto con los procesos realizados por la entidad **Decoder** permiten obtener los datos decodificados que serán revisados por la entidad **verify**. La arquitectura del decodificador se ilustra en la figura 4.11

Componente Delta XY:

Entidad **DeltaXY** (In: *X, Clkwork, Y, Reset* Out: *DeltaXY*)

Este componente se encarga de comparar la secuencia codificada recibida (X) contra la secuencia generada por los diversos trayectos en el diagrama de Trellis (Y), dando como salida la cantidad de bits en los que difieren ($DeltaXY$). Al existir ocho trayectos de entrada a los cuatro estados disponibles (a, b, c y d), se requieren ocho de estos componentes.

Célula Add-Compare-Select

Entidad **ACS** (In: *TaoIn1, TaoIn2, DeltaXY11, DeltaXY12, DeltaXY21, DeltaXY22, Reset, Clkwork, MetricMonitorFlag* Out: *TaoOut1, TaoOut2, PathSelect1, PathSelect2*)

Es el componente central del decodificador. Para cada estado realiza el cálculo de la trayectoria con menor métrica, esto en función de la información recibida del componente **DeltaXY** a través de las entradas *DeltaXY's* y las métricas anteriores *TaoIn1* y *TaoIn2*. Indica a las memorias **PathHistory** que historial de trayecto deben almacenar por medio de *PathSelect1* y *PathSelect2*.Pone a disposición de la entidad **Decoder** las métricas de estado para que elija de que memoria tomar el bit decodificado. Este componente puede hacer un ajuste de las métricas de estado cuando éstas están a punto de desbordarse. Esto es hecho mediante la entrada *MetricMonitorFlag*, misma que es controlada por la entidad **Decoder**. Este componente puede procesar los cálculos para dos estados a la vez.

Memoria para los trayectos:

Entidad **PathHistory** (In: *Clk, PathIn1, PathIn2, BitIn, PathSelect, Reset* Out: *DecodedBit* InOut: *PathHistoryOut*)

Es un componente formado por $5k$ registros de memoria (para $K = 3$ son 15) que almacena los bits producto de un trayecto entrando a un determinado estado con la menor métrica. Su primer registro siempre es llenado con el bit que produzca un trayecto hacia el estado asociado a la memoria, mientras que el resto de los registros son llenados con los bits productos de un trayecto con menor métrica indicado por el componente **ACS**. De esta memoria se toma su último valor como dato decodificado, siempre y cuando el estado asociado a esta memoria tenga la menor métrica de todas. Cada memoria está asociada a un estado de modo que son necesarias cuatro de estas.

Decisión de bit decodificado:

No es propiamente un componente, si no una funcionalidad de la entidad **Decoder** y es la etapa final del decodificador Viterbi y entrega los bits decodificados obtenidos de las memorias a su destino en función de las métricas de estado proporcionadas por

el componente **ACS**, para ello la entidad **Decoder** tiene a su disposición el último bit en los registros de cada una de las memorias y las métricas de estado proporcionadas por el componente **ACS**.

Verificador.

Entidad **Verify** (In: *Clk, Reset, DecodedData* Out: *Display7Segmentos*)

Este componente no tiene función dentro del proceso de decodificación, no obstante realiza la comparación entre la secuencia decodificada contra una copia del mensaje codificado en el transmisor. Este elemento considera el retrazo generado por el proceso de decodificación y los tiempos de procesamiento de la señal. Estos tiempos son 3 ciclos de reloj por procesos del sistema más $5K$ ciclos de reloj, que es el tiempo que le toma al decodificador decodificar el primer bit. Este componente puede hacer un conteo de los bits correctos o erróneos en una secuencia de 65,535 bits, que es lo que pueden desplegar 4 displays 7 segmentos en un formato hexadecimal.

4.2.3. Generación de codificadores y decodificadores

Una vez implementado un codificador y decodificador se buscó realizar la implementación de múltiples codificadores y decodificadores a partir de estos componentes. Específicamente el decodificador, debido a que el codificador es relativamente sencillo. Para ello se analizó el proceso de codificación y decodificación en busca alguna relación que permita predecir el comportamiento de su estructura y de esta manera poder utilizar los elementos de los que ya se disponen. Para variaciones en las conexiones de los puntos de suma no se observan cambios en la estructura del decodificador, solo se observa un ajuste de los valores producidos en los trayectos, no siendo así para variaciones de la constante de restricción o numero de memorias utilizadas. Para estos casos, el numero de estados crece exponencialmente, como se comentó en el capítulo 2. Este incremento

en la cantidad de estados viene junto con un cambio en las conexiones existentes entre ellos y se observó que se puede describir de la siguiente manera: A cada estado destino b_{2i-1} y b_{2i} le corresponde un estado origen a_i y $a_{i+\frac{n}{2}}$, para toda $i = 1, 2, 3...\frac{n}{2}$, donde n es el número de estados existentes. Hasta este momento se han encontrado referencias que reporten el uso de esta relación o que la describan. La relación entre los estados previamente descrita se ilustra en la figura 4.12.

Descripción del script Matlab

De la relación descrita previamente se implementó un script en Matlab, el cual recibe como entrada los vectores V_1 y V_2 que describen a un codificador convolucional de dos puntos de suma y K elementos de memoria. Estos vectores pueden ser proveídos por el usuario o bien seleccionados de un compendio de códigos optimizados [42]. El script llama a un conjunto de subfunciones capaces de crear los archivos de código VHDL con las entidades y configuración necesaria para el correcto funcionamiento de un codificador y decodificador seleccionado por el usuario. El script permite al usuario definir el número de registros de memoria que utilizará un generador de secuencias pseudoaleatorias de la forma ilustrada en la figura 4.8 como fuente de datos para el codificador, así como definir el conteo de bits decodificados correcta o incorrectamente en el decodificador, lo cual facilita el cálculo del BER de una determinada implementación en un canal especifico. El script también entrega alguna información relevante sobre las secuencias generadas tanto por el codificador como por el generador de secuencias pseudoaleatorias, mediante algunas variables de control, como lo son el promedio de 1's generados por el codificador con una determinada fuente de datos y el código convolucional resultante.

4.3. Conclusión del capítulo

En este capítulo se ha descrito la implementación de un codificador y decodificador, al igual que el desarrollo de una relación que describe la interacción entre los estados para el incremento de la constante de restricción K. De esta relación se ha implementado un script que permite la implementación de un codificador y decodificador con dos puntos de suma cualesquiera, mediante el ajuste de algunas variables de control. Este script permite el desarrollo y prueba de un sistema de comunicaciones en poco tiempo, dejando únicamente algunos elementos a seleccionar, como por ejemplo el canal, el tipo de modulación, o algunos otros elementos que puedan mejorar o habilitar un sistema en un determinado escenario. La implementación hecha por medio de bloques admite la inclusión o eliminación de elementos sin afectar al resto del sistema. Un ejemplo de esto pudiera ser la eliminación de la serialización de los datos para incluir una modulación que utilice símbolos no binarios.

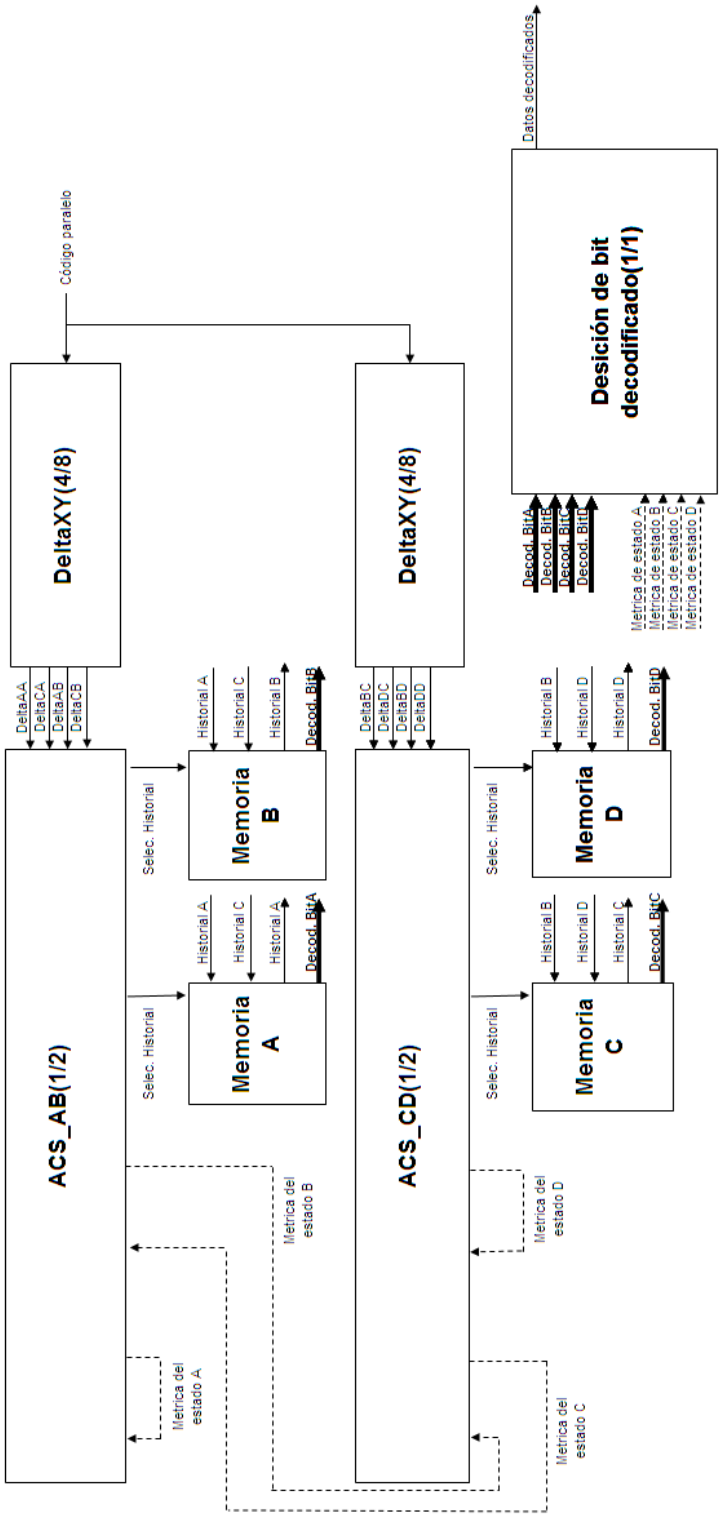


Figura 4.11: Implementación de la arquitectura del decodificador.

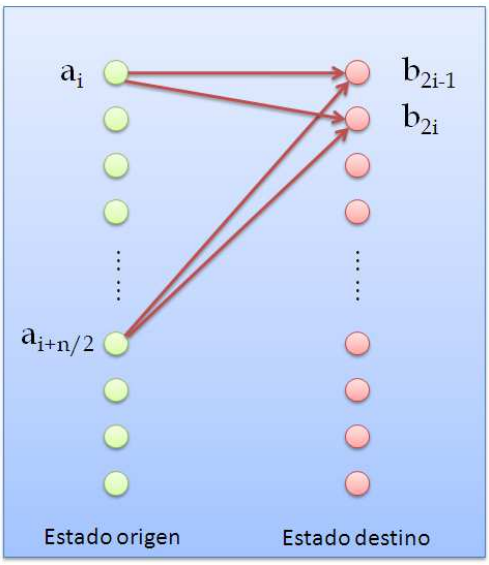


Figura 4.12: Relación existente para n estados origen y destino.

Capítulo 5

Resultados.

En este capítulo se presentan los resultados que muestran la correcta implementación de los codificadores y decodificadores, así como de sus componentes, mediante la implementación de un verificador en el decodificador y capturas de señales en tiempo en un osciloscopio Tekctronix (modelo). También se analiza el ancho de banda requerido por las etapas de codificación de canal y codificación de línea, con respecto a los datos a transmitir, el cual se ilustra con las capturas en un analizador de espectros Rohde & Schwars modelo FSH4 Spectrum Analizer. Se presenta además una evaluación de la eficiencia de la implementación en relación al consumo de unidades lógicas en el dispositivo FPGA, el cual es un indicador de la complejidad del sistema implementado.

5.1. Verificación de la implementación

Para la verificar la correcta implementación del codificador y decodificador son necesarias algunas pruebas bajo condiciones de ruido cero, es decir, una prueba en la cual los datos a transmitir no se vean perturbados de ninguna manera, de esta forma

podemos asegurar que cualquier desperfecto que apareciera sería únicamente producto de la implementación y no del canal en sí, para lo cual se utilizó el esquema ilustrado en la sección anterior en la figura 4.2.

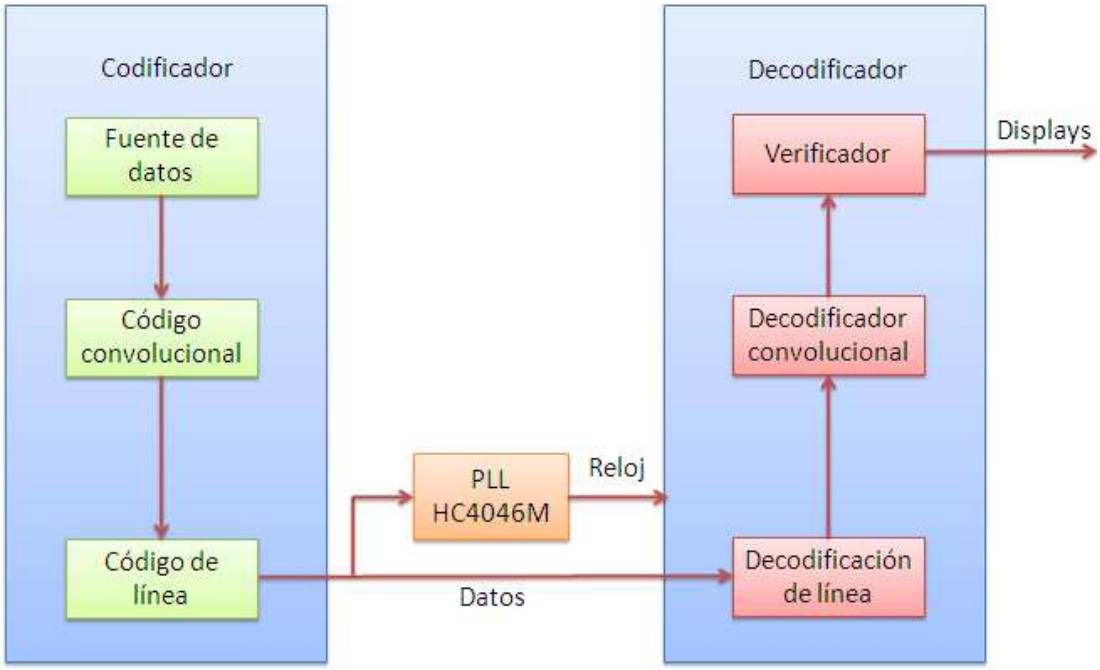


Figura 5.1: Esquema de prueba bajo condiciones de ruido cero.

Para la verificación de la correcta implementación se utilizó un elemento en el decodificador, el cual realiza una comparación entre la secuencia codificada y la decodificada considerando los ciclos de reloj que toma el proceso de decodificación. Este esquema queda expuesto en la figura 5.2.

Este esquema fue utilizado para la verificación de todos los codificadores y decodificadores presentados como óptimos [42] y que el script implementado facilita al usuario, excepto aquellos reportados en la sección 4 de este capítulo, debido a la falta de recursos en el dispositivo FPGA.

Adicionalmente se realizaron capturas en tiempo con un osciloscopio, para observar las diferentes etapas de la codificación. Dichas etapas son ilustradas en la figura 5.3. Las

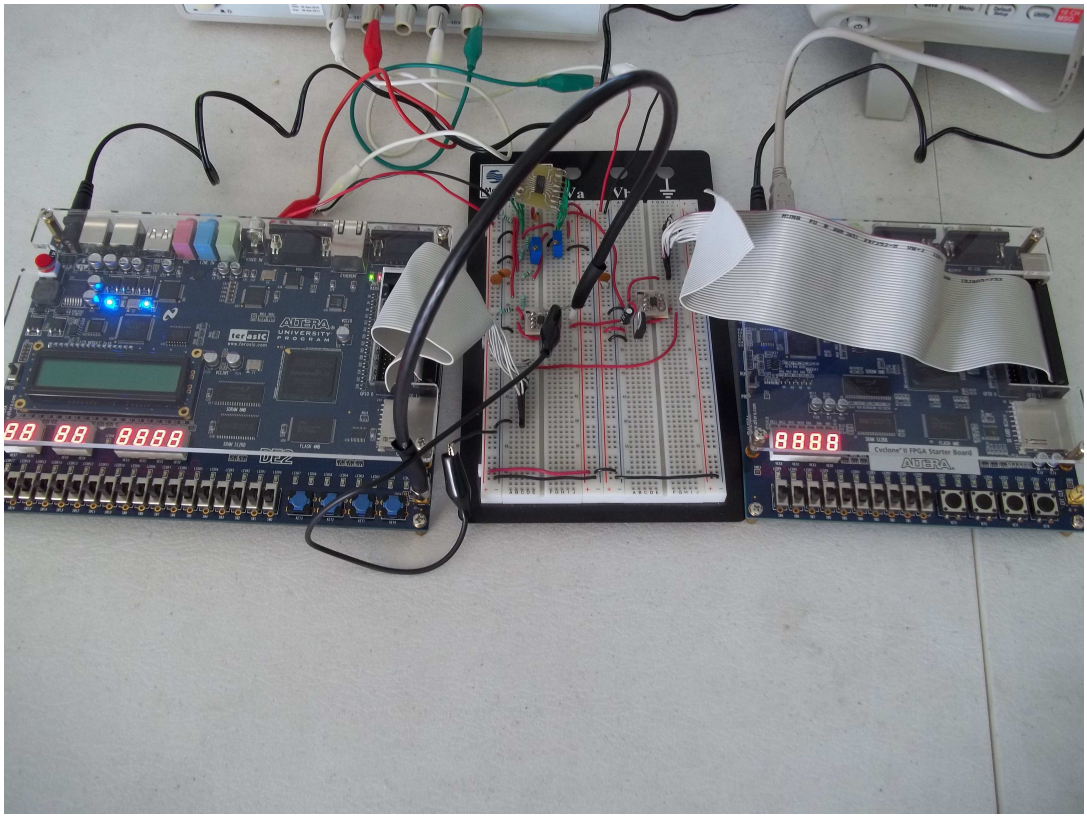


Figura 5.2: Implementación del sistema para pruebas bajo condiciones de ruido cero.

capturas ilustradas son con respecto a un codificador $K = 3$, $V_1 = [111]$ y $V_2 = [101]$, con un generador de 4 registros ($2^4 - 1 = 15$ bits). La selección de este codificador y generador fue con la finalidad de que se pudiera apreciar toda la secuencia codificada en una sola captura.

En las figuras 5.4, 5.5, 5.6, se muestran respectivamente: los datos generados en la fuente, la codificación convolucional de los datos, el código de línea asociado a los datos codificados convolucionalmente. Estas señales son obtenidas a partir de los puntos indicados en el esquema de verificación de formas de onda en tiempo. La señal amarilla es la señal en cuestión, mientras que la señal azul es el reloj de referencia de 10 MHz.

En la figura 5.7 se observa un desfase de 42 ns entre la señal de reloj recuperada

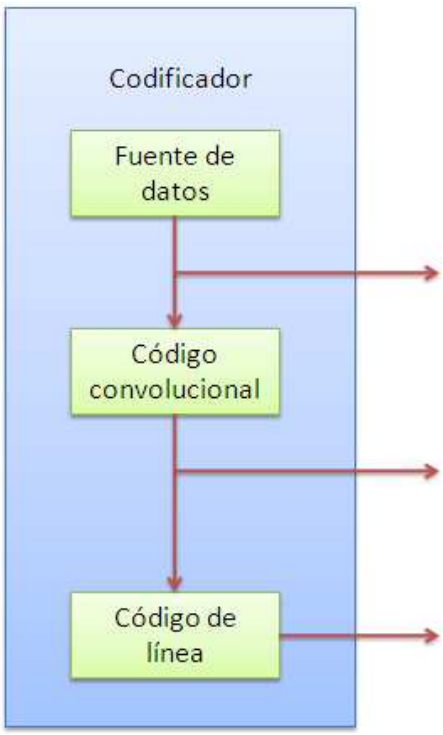


Figura 5.3: Esquema de verificación de formas de onda en tiempo.

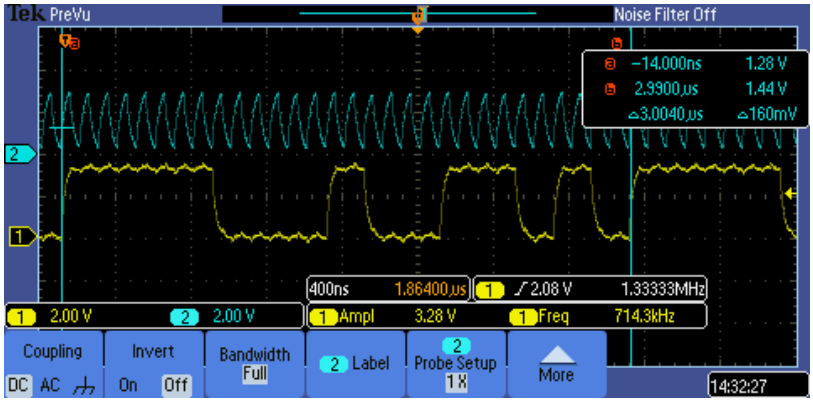


Figura 5.4: Captura de los datos generados en la fuente.

por el circuito integrado PLL y la señal de reloj asociada a los datos. Este desfase fue compensado por el PLL interno del FPGA para un correcto muestreo de los datos.

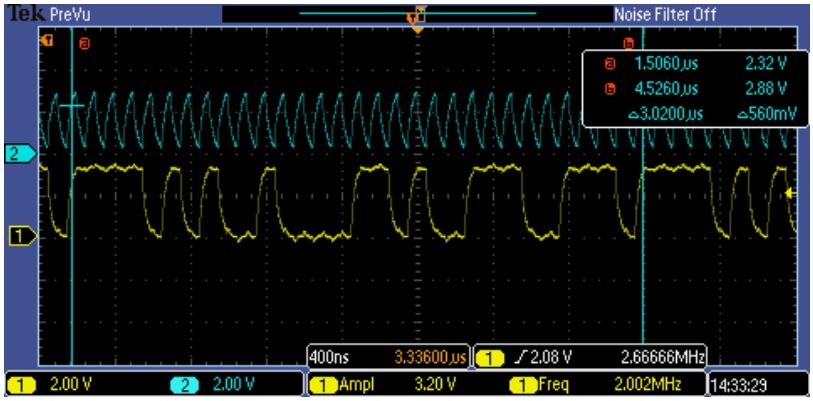


Figura 5.5: Codificación convolucional de los datos generados en la fuente.

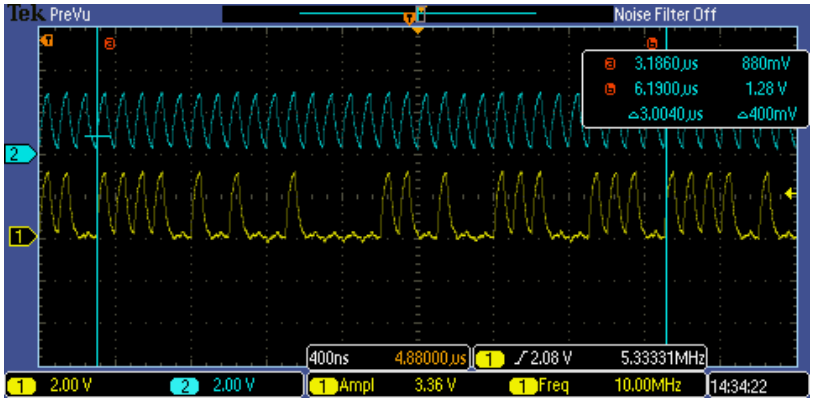


Figura 5.6: Código de línea URZ asociado a los datos codificados en el codificador convolucional.

5.1.1. Análisis de espectro.

Un análisis del espectro generado por las diversas etapas resulta importante para observar el uso del ancho de banda por parte de los diversos componentes. Este análisis es necesario si se van a transmitir con múltiples portadoras para evitar interferencia entre sí. En las figuras 5.1, 5.1, 5.1 se muestran respectivamente los espectros generados por: los datos generados en la fuente, la codificación convolucional de los datos, el código de línea asociado a los datos codificados convolucionalmente.

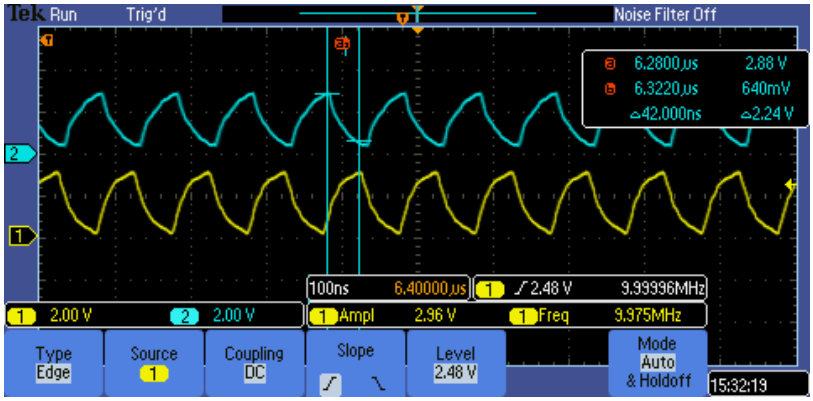


Figura 5.7: Desfase existente entre la señal de reloj recuperada y la señal de reloj asociada a los datos.

Figura 5.8: Espectro generado por los datos generados en la fuente.

De estas capturas observamos que el ancho de banda requerido por la fuente de datos es de 5 MHz, posteriormente la codificación convolucional duplica este ancho de banda a 10 MHz, esto debido a que la codificación convolucional utilizada genera dos bits por cada bit de datos codificados. En la captura del espectro de la codificación de línea observamos que el ancho de banda se duplica de nuevo de 10 MHz a 20 MHz,

Figura 5.9: Espectro generado por la codificación convolucional de los datos generados en la fuente.

Figura 5.10: Espectro generado por el código de línea URZ asociado a los datos codificados en el codificador convolucional.

esto debido a que cada bit es codificado en dos bits, producto de la multiplicación de la señal de datos codificados convolucionalmente con la señal de reloj.

Análisis de la recuperación de reloj.

De los códigos óptimos elegidos existe un caso particular de $K = 6$ descrito por los vectores $V_1 = [10111]$ y $V_2 = [11001]$, el cual ante un generador de secuencias pseudoaleatorias de 4 registros genera una secuencia con una baja distribución de 1's (0.2333). Ante una baja distribución de 1's el sistema no puede mantenerse sincronizado debido a la ausencia de la señal de reloj, lo cual provoca un mal muestreo de los datos en el sistema y por tanto un sin fin de errores. Esta ausencia de la señal de reloj provocada por la baja distribución de 1's se puede observar en la captura del espectro del código de línea ilustrado en la figura 5.11, la cual al ser comparada con el mismo código convolucional pero con un generador de 5 registros 5.12 se observa una disminución de $4.1dBm$ en la potencia de la señal de reloj.

5.2. Evaluación de la complejidad del sistema

La evaluación de la complejidad de la implementación fue hecha mediante variación de los parámetros de los códigos generados por el script hecho en Matlab. Variaciones en los vectores, no generan cambios significativos, al igual que variaciones en el incremento de registros para el generador de secuencias pseudoaleatorias, no obstante la complejidad se incrementó exponencialmente conforme se incrementó K. Los indicadores de la complejidad del sistema fueron la utilización de unidades lógicas y registros dedicados utilizados, los cuales se muestran en la tabla 5.1. Estos fueron calculados por el software Quartus II para un dispositivo FPGA Cyclone II EP2C35F672C6, el cual cuenta con 33,216 unidades lógicas, razón por la cual los codificadores con $K = 7, 8$ no pudieron

Figura 5.11: Espectro generado por la codificación convolucional $K = 6$ ($V_1 = [10111]$ y $V_2 = [11001]$) con generador de datos de 4 registros.

Figura 5.12: Espectro generado por la codificación convolucional $K = 6$ ($V_1 = [10111]$ y $V_2 = [11001]$) con generador de datos de 5 registros.

ser probados. El codificador con $K = 9$, no pudo ser compilado por el software, no obstante se obtuvo una estimación por medio de una regresión exponencial para tener un estimado del consumo de unidades lógicas y registros dedicados para este codificador, misma que se ilustra en las figuras 5.13 y 5.14.

K	Unidades lógicas utilizadas	Registros dedicados utilizados
3	559	168
4	1311	282
5	3521	557
6	10867	1173
7	35807	2550
8	134271	5590

Tabla 5.1: Incremento de la complejidad del sistema con respecto a K

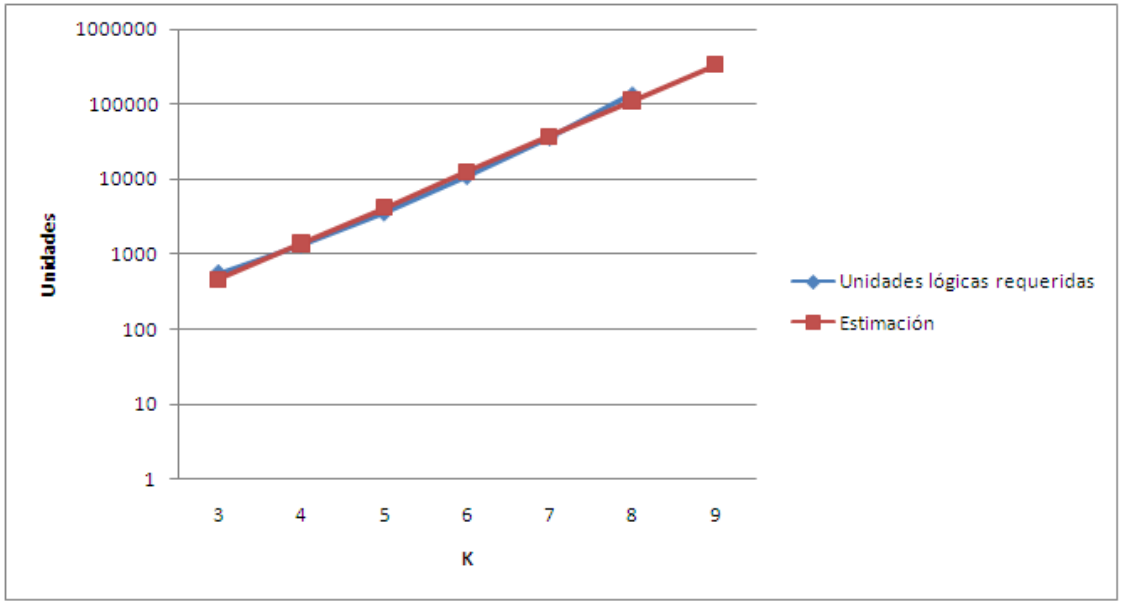


Figura 5.13: Utilización de unidades lógicas con respecto a K y estimación para $K = 9$.

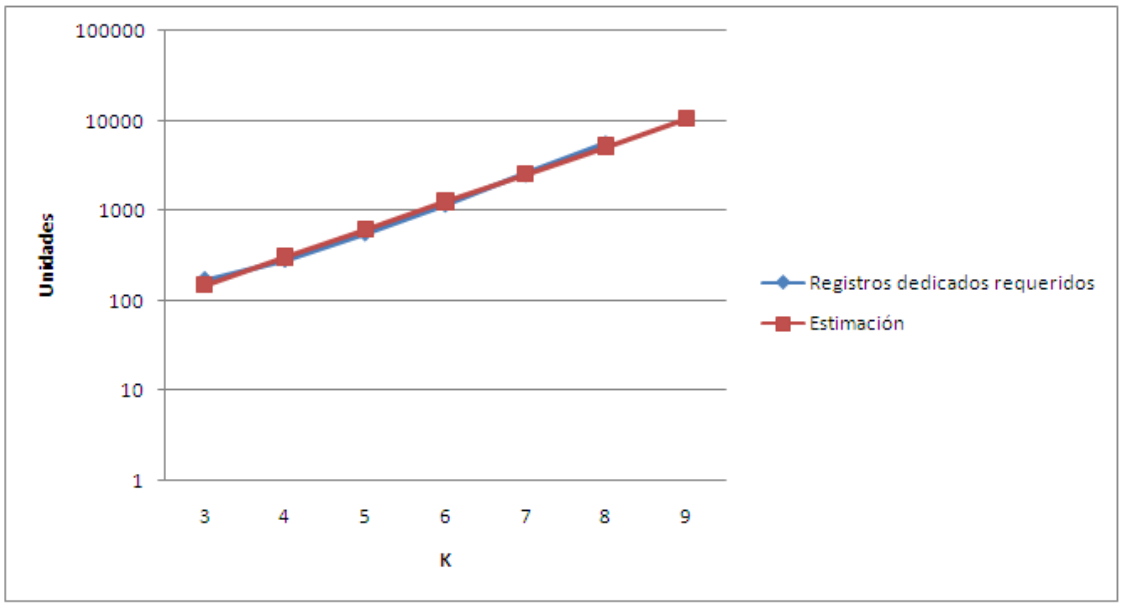


Figura 5.14: Utilización de registros dedicados con respecto a K y estimación para $K = 9$.

Capítulo 6

Conclusiones.

En este trabajo de tesis hemos analizado el uso de codificaciones de canal como una opción para habilitar enlaces que permitan dar solución al problema del cuello de botella y disponibilidad de ancho de banda en las telecomunicaciones.

Se ha analizado la opción del canal óptico inalámbrico como una posible solución a este problema, y se han observado los retos que esta implica, para lo cual se han propuestos las codificaciones de canal para poder habilitar este tipo de enlaces o algún otro con características similares. Se analizaron los diversos tipos de codificaciones de canal en busca de aquella que sea versátil para un amplio espectro de canales, y pueda ser pauta para diversas implementaciones que se adecuen al canal en cuestión. De este estudio se encontró factible el uso de codificaciones convolucionales, como se estableció en los **capítulos 2** y **3**. De este estudio y de una primera implementación se encontró una relación existente entre los trayectos de los estados al variar la constante de restricción, esta relación sirvió para el desarrollo de un script en Matlab el cual es capaz de generar códigos en VHDL para describir un codificador y decodificador de dos puntos de suma cualesquiera. Este script permitió implementar algunos de los códigos óptimos descritos en el **capitulo 2**, los cuales han sido probados bajo condiciones de ruido cero. De

estas implementaciones se pudo observar el uso de ancho del banda por parte de los códigos implementados, tanto de la codificación de canal como de la codificación de línea. También se observaron restricciones en la implementación por parte del hardware disponible, como lo fueron las frecuencias de trabajo de los circuitos integrados PLL y los PLL internos del FPGA, o los problemas de sincronía que tienen estos elementos ante una baja distribución de 1s, y por lo tanto ausencia parcial de señal de reloj en la señal transmitida. Adicionalmente se pudo verificar el correcto funcionamiento de los codificadores y decodificadores óptimos para $K = 3, 4, 5, 6$ y se corroboró la complejidad del sistema ante incrementos de la constante de restricción K.

6.1. Aportaciones

La principal aportación del presente trabajo es la obtención de codificadores y decodificadores funcionales, además de la descripción de la relación existente entre los trayectos de los estados y la implementación del script en Matlab que permite la creación del código VHDL para codificadores y decodificadores de dos puntos de suma cualesquiera mediante unas variables de control que definen al codificador/decodificador en cuestión, lo cual es de suma importancia para una rápida implementación y caracterización de sistemas de comunicaciones, los cuales tienen pueden tener aplicaciones académicas, científicas y/o industriales. En el ámbito académico puede ser utilizado para observar las mejoras que las codificaciones de canal en general pueden aportar a un sistema de comunicaciones, y en específico observar comportamientos particulares de los códigos convolucionales, como lo son los errores catastróficos que no son posibles observar salvo algunos casos específicos que no suelen ser implementados. En el ámbito científico puede ser utilizado para efectos experimentales en diversos canales, o con variaciones ya sea modificando, agregando y/o remplazando las entidades existentes, y

una vez optimizadas para un determinado canal pueden llegar a ser implementables en un escenario real. Adicionalmente la implementación de este sistema permitió observar las limitantes de hardware y observaciones para el sistema de recuperación de reloj que futuras implementaciones deben considerar.

6.2. Trabajo a futuro

En base a las observaciones, al trabajo y a las investigaciones realizadas en este trabajo de tesis se consideran pertinentes los siguientes puntos como trabajo a futuro:

- Evaluar el desempeño del sistema en un canal específico bajo condiciones controladas.
- Implementar un esquema de recuperación de reloj más versátil e independiente de la distribución de 1's generados por alguna etapa anterior, como lo puede ser el envío de una trama de sincronización cada determinado tiempo con la finalidad de mantener el sistema sincronizado.
- Implementar las adecuaciones necesarias para algunos canales específicos, como lo pueden ser entrelazadores, diversidad espacial.
- Implementar esquemas de modulación que permitan hacer un uso más eficiente del ancho de banda y/o de la potencia transmitida, como por ejemplo una modulación PPM.
- Implementar bloques flexibles de entrelazado para la futura implementación de códigos concatenados.

Bibliografía

[1] M. Abtahi, P. Lemieux, W. Mathlouthi, L. A. Rusch, S. Member, and A. Abstract. Suppression of Turbulence-Induced Scintillation in Free-Space Optical Communication Systems Using Saturated Optical Amplifiers. 24(12):4966–4973, 2006.

[2] S. Arnon. Optimization of Urban Wireless Communications Systems. *IEEE Transactions on Wireless Communications.*, 2003.

[3] M. Awan, E. Leitgeb, F. Nadeem, M. Khan, and C. Capsoni. A New Method of Predicting Continental Fog Attenuations for Terrestrial Optical Wireless Link. In *2009 Third International Conference on Next Generation Mobile Applications, Services and Technologies*, pages 245–250. IEEE, Sept. 2009.

[4] J. Bhasker. *A VHDL Primer*. Prentice h edition, 1999.

[5] R. E. Blahut. *Algebraic Codes for Data Transmission*. Cambridge University Press, Cambridge, 2003.

[6] J. Castineira Moreira and P. Guy Farrell. *Essentials of Error-Control Coding*. 2006.

[7] C. P. Colvero, M. C. R. Cordeiro, and J. P. von der Weid. FSO Systems: Rain, Drizzle, Fog, Haze Attenuation at Different Optical Windows Propagation. pages 563–568, 2007.

[8] F. M. Davidson and Y. T. Koh. Interleaved convolutional coding for the turbulent atmospheric optical communication channel. *IEEE Transactions on Communications*, 36(9):993–1003, 1988.

[9] E. Elebeoba and S. National. Optimal Generators for a Systematic Block Code Model of Prokaryotic Translation Initiation. *Engineering in Medicine and Biology Society, 2003. Proceedings of the 25th Annual International Conference of the IEEE*, pages 3858–3860, 2003.

[10] J. H. Franz and V. Jain. *Optical communications, components and systems.* CRC Press, 2000.

[11] D. L. Fried. Optical resolution through a randomly inhomogeneus medium for very long and very short exposures. *Journal Optical Society of America*, 1966.

[12] R. M. Gagliardi and S. Karp. *Optical communications.* 1995.

[13] R. G. Gallager. Low-Density Codes. 1962.

[14] I. Glover and P. Grant. *Digital Communications.* Prentice h edition, 1998.

[15] S. Golomb and R. Scholtz. Generalized Barker Sequences. *IEEE Transactions on Information Theory*, 11(4):533 – 537, 1965.

[16] T. Hehn and J. Huber. LDPC codes and convolutional codes with equal structural delay: a comparison. *IEEE Transactions on Communications*, 57(6):1683–1692, 2009.

[17] W. C. Huffman and V. Pless. *Fundamentals of Error-Correcting Codes.* Cambridge University Press, Cambridge, 2003.

[18] M. Ijaz, Z. Ghassemlooy, H. L. Minh, S. Rajbhandari, J. Perez, and A. Gholami. Bit Error Rate Measurement of Free Space Optical Communication Links under Laboratory- Controlled Fog Conditions. pages 52–55, 2011.

[19] S. W. Jhon. Line of sight wave propagation through the turbulent atmosphere. *Wireless Communications, IEEE Transactions on*, 2(4), 1968.

[20] Z. Jia, Q. Zhu, and F. Ao. Atmospheric Attenuation Analysis in the FSO Link. *Communication Technology, 2006. ICCT '06. International Conference on*, 2006.

[21] M. S. Khan, S. S. Muhammad, M. S. Awan, E. Leitgeb, M. Grabner, and V. Kvicera. Validating Relationship between Aerosol's Liquid Water Content and Optical Attenuation for Terrestrial FSO Links. *11th International Conference on Telecommunications - ConTEL 2011*, pages 195–198, 2011.

[22] K. Kiasaleh. Performance of coherent dpsk free-space optical communication systems in k-distributed turbulence. *IEEE Transactions on Communications*, 54(4):pp. 604 – 607, 2006.

[23] I. I. Kim, R. Stieger, J. A. Koontz, C. Moursund, M. Barclay, P. Adhikari, J. Schuster, and E. Korevaar. Wireless optical transmission of fast ethernet, FDDI, ATM and ESCON protocol data using the TerraLink laser communications system. *Society of Photo-Optical Instrumentation Engineers*, 1998.

[24] S. Lambert and W. Casey. *Laser communications in space.* Artech House Publishers, 1995.

[25] R. L. P. Larry C. Andrews. *Laser Beam Propagation Through Random Media.* 1998.

[26] E. Leitgeb, M. S. Khan, and M. Loeschnigg. Analysis of the Influence of the Particle Surface Area (PSA) for Optical Wireless Links under Fog Conditions. pages 3–7, 2012.

[27] J. Li and M. Uysal. Optical wireless communications: system model, capacity and coding. *Vehicular Technology Conference, 2003. VTC 2003-Fall. 2003 IEEE 58th*, 1, 2003.

[28] Q. Liu, C. Qiao, G. Mitchell, and S. Stanton. Optical wireless communication networks for first- and last-mile broadband access. *Journal of Optical Networking*, 4, 2005.

[29] S. V. Maiya, D. J. Costello, and T. E. Fuja. Low Latency Coding : Convolutional Codes vs . LDPC Codes. *IEEE Transactions on Communications*, 60(5):1215–1225, 2012.

[30] J. R. W. Mclaren, Thomas John C., Mackintosh Jessica L., Mudge Kerry A., Grant Kenneth J., Clare Bradley A., and C. W. G. Comparison of probability density functions for analyzing irradiance statistics due to atmospheric turbulence. *Applied Optics*, 51(25):pp. 5996–6002, 2012.

[31] L. A. R. Mohammad Abtahi. MITIGATING OF SCINTILLATION NOISE IN FSO COMMUNICATION LINKS USING SATURATED OPTICAL AMPLIFIERS. *Military Communications Conference, 2006. MILCOM 2006*, pages 1–5, 2006.

[32] S. S. Muhammad. A unified approach for Channel Modeling of Terrestrial FSO Links. pages 522–527, 2010.

[33] S. Navidpour, M. Uysal, and M. Kavehrad. BER Performance of Free-Space Optical Transmission with Spatial Diversity. *IEEE Transactions on Wireless Communications*, 6(8):2813–2819, Aug. 2007.

[34] A. Neubauer, J. Freudenberger, and V. Kühn. *Coding Theory Algorithms, Architectures and Applications.* 2007.

[35] J. P. Odenwalder. *Optimal Decoding of Convolutional Codes.* Ph. d., UCLA, 1970.

[36] G. Osche. *Optical detection theory for laser applications.* 2002.

[37] J. G. Proakis and M. Salehi. *Fundamentals of Digital Communications.* 4th edition.

[38] J. G. Proakis and M. Salehi. *Communication systems engineering.* 2001.

[39] J. d. D. Sanchez Lopez, A. Arvizu Mondragon, J. I. Nieto Hipolito, and H. Cervantes de Avila. DISEÑO CONSTRUCCIÓN Y CARACTERIZACIÓN DE CÁMARA PARA LA GENERACIÓN DE TURBULENCIA ATMOSFÉRICA PARA EXPERIMENTOS DE COMUNICACIONES ÓPTICAS INALÁMBRICAS. *ELECTRO*, 32:154–159, 2009.

[40] K. Shaik. Atmospheric propagation effects relevant to optical communications. *TDA Progress Report*, 1988.

[41] E. J. Shin, S. Member, and V. W. S. Chan. Optical Communication Over the Turbulent Atmospheric Channel Using Spatial Diversity. *Global Telecommunications Conference, 2002*, pages 2055–2060, 2002.

[42] B. Sklar. *Digital Communications Fundamentals and Applications.* Prentice Hall, 2nd edition, 2001.

[43] M. Tatarko, . Ovseník, and J. Turán. Availability and Reliability of FSO links Estimation from Measured Fog Parameters. pages 192–195, 2012.

[44] A. Wheelon. *Electromagnetic scintillation Vol, II. Weak Scattering.* Cambridge University Press, Cambridge, 2003.

[45] Wikipedia.org. Latency (engineering).

[46] Wikipedia.org. Telescopio Espacial Hubble.

[47] H. A. Willebrand and B. S. Ghuman. *Free space optics: enabling optical connectivity in today's networks.* 2001.

[48] S. A. Zabidi, W. A. Khateeb, R. Islam, and A. W. Naji. The Effect of Weather on Free Space Optics Communication (FSO) Under Tropical Weather Conditions and a Proposed Setup for Measurement. *International Conference on Computer and Communication Engineering (ICCCE 2010)*, (May):11–13, 2010.

[49] X. Zhu and J. Kahn. Free space optical communication through atmospheric turbulence channels. *IEEE Transactions on Communications*, 2002.

[50] X. Zhu, J. M. Kahn, and C. Sciences. Performance Bounds for Coded Free-Space Optical Communications through Atmospheric Turbulence Channels. *IEEE Transactions on Communications*, 51(8), 2003.