INSTITUTO POLITÉCNICO NACIONAL
CENTRO DE INVESTIGACIÓN EN COMPUTACIÓN
SECRETARÍA DE INVESTIGACIÓN Y PROSGRADO

Laboratorio de inteligencia artificial y procesamiento digital de señales

Implementación de códigos correctores de errores para un módem para la comunicación sobre la red eléctrica

TESIS

Que para obtener el grado de
Maestro en Ciencias en Ingeniería en Cómputo
con opción en sistemas digitales

Presenta
Ing. Alfonso Martínez Cruz

Directores de tesis
Dr. Ricardo Barrón Fernández
Dr. José Luis Oropeza Rodríguez

Zacatenco, México D.F., Diciembre de 2011

Contenido

Índice de figuras

Índice de tablas

Agradecimientos

GRACIAS
A mis padres Josefina y Gabriel por darme la oportunidad de crecer, desarrollarme y cumplir mis metas.

GRACIAS
A mis directores de tesis: Dr. Ricardo y Dr. José Luis por guiarme y compartir sus experiencias durante mis estudios de maestría.

GRACIAS
Al Instituto Politécnico Nacional por permitirme realizar mi formación durante mis estudios de maestría.

GRACIAS
A mis profesores durante mi estancia en el Centro de Investigación en Computación por compartir su conocimiento y formar parte de mi preparación en mis estudios de maestría.

GRACIAS
A mis compañeros por compartir sus consejos, experiencias y por su apoyo durante mis estudios.

GRACIAS
Al Consejo Nacional de Ciencia y Tecnología, CONACyT, por el apoyo económico proporcionado durante la realización de mis estudios de maestría.

Alfonso Martínez Cruz

Resumen

El desarrollo de los turbo códigos en las pasadas dos décadas, ha resultado en una mejora significativa en el desempeño de los sistemas de comunicación PLC. Todo ello es posible debido a la considerable reducción en el tamaño de los dispositivos, el desarrollo de circuitos integrados más complejos con soporte para la codificación de canal y la implementación de técnicas de modulación más robustas. Actualmente, algunos de los mayores problemas investigados están relacionados con la implementación de arquitecturas eficientes para operaciones de los algoritmos de decodificación, el desarrollo de técnicas para un eficiente manejo del consumo de potencia y la reducción en la latencia del hardware utilizado.

La aplicación de los turbo códigos en los sistemas de comunicación, ha sido caracterizado por su uso en condiciones difíciles para un código convencional. Algunas de estas condiciones incluyen la presencia de altos niveles de ruido en el canal, lo cual requiere dispositivos que minimicen el consumo de potencia de sistemas en los cuales la transmisión de información tiene un alto costo.

En esta tesis se presenta la implementación del algoritmo Max-Log-MAP en el DSP TMS320C6416T. A su vez, se reportan los resultados obtenidos contaminando con ruido gaussiano e impulsivo los datos enviados al dispositivo receptor (DSP), así como el comportamiento del decodificador para diferentes valores de SNR (relación señal a ruido). La propuesta incluye esencialmente el diseño de la implementación del decodificador, el intercalador, mecanismo "puncturing" y decodificación de datos. La implementación presentada en esta tesis muestra los beneficios de un código de alto desempeño en una aplicación real.

Con base en las pruebas realizadas en un canal con ruido Gaussiano e Impulsivo se observó que al incrementar la longitud del intercalador, se incrementó el desempeño en las curvas BER. Además, se observó que con un turbo codificador RSC(5,7) los resultados son muy cercanos a la configuración RSC(17,15) y a su vez el codificador RSC(31,27) generó los mejores resultados. Con base en ello, es posible reducir los recursos requeridos en hardware para implementar un turbo código para la aplicación en un canal PLC.

Abstract

The development of turbo codes over the past two decades, has resulted in significant performance improvement of PLC communication systems, considerable reduction of the device size, development of more complex chips with channel coding support and implementation of more robust modulation techniques. Today, some of the major research problems are related to the implementation of efficient architectures for decoding algorithms, development of techniques for efficient power consumption management, and latency reduction in the hardware used.

The application of turbo codes in communication systems has been characterized for use in difficult conditions for a conventional code. Some of these conditions include the presence of high noise levels in the channel, which require devices to minimize systems power consumption in which the transmission of information has a high cost. In this thesis we present the implementation of the Max-Log-MAP algorithm in TMS320C6416T DSP. We report the results obtained with Gaussian and impulsive noise contaminating the data sent to the receiving device (DSP) as well as the behavior of the decoder for different SNR (signal/noise relation). Specifically the proposal includes the design implementation of the encoder, interleaver, "puncturing" mechanism and decoding data.

The implementation presented in this thesis shows the benefits of a high-performance code in a real application. Based on tests performed on a channel with Gaussian and impulsive noise was observed that increasing the length of the interleaver, the performance increase BER curves. It was further observed that with a turbo encoder type RSC(5,7) the results are very close to the RSC(17,15) configuration, also RSC(31,27) encoder generated the best results. On this basis it is possible to reduce hardware resources required to implement a turbo code for the application in the PLC channel.

Capítulo 1

Introducción

La importancia de los sistemas de comunicación en la vida diaria representa un factor vital en el desarrollo tecnológico de la humanidad, la comunicación a través de distintos medios (por ejemplo fibra óptica, cable telefónico, señales de microondas), ha conformado la llegada de distintos servicios a lugares que parecía casi imposible llegar. De igual forma, la necesidad de la transmisión de señales digitales a través de estos medios ha sido una parte fundamental para la aparición de un dispositivo conocido como módem, el cual se encarga de la transmisión y recepción de los datos a través de un medio de comunicación [1].

La red eléctrica no fue inicialmente desarrollada para servir como medio de comunicación, sin embargo, debido a la amplia cobertura y su bajo costo de instalación, representa un medio atractivo para ser utilizado como canal de comunicación [2, 3].

Durante el proceso de la transmisión de información digital a través de un canal de comunicaciones, es inevitable que se produzcan errores en la información recibida debido a diversos factores tales como: perturbaciones, ruido e interferencias [4]. Debido a ésto, resulta necesario implementar códigos que puedan recuperar la información original, a pesar de los problemas generados durante su transmisión. Actualmente en los sistemas de comunicación se utilizan códigos que agregan información redundante a la información original, con el objetivo de poder recuperar dicha información. Un código que sólo puede detectar errores en la información recibida se conoce como código de detección de errores, a diferencia de uno que no sólo puede detectar errores sino también puede corregirlos y es conocido como código de detección y corrección de errores [5].

La decisión de aplicar la detección o corrección de errores en el diseño de un código, depende de las características de la aplicación o sistema. Cuando un sistema de comunicación es capaz de realizar una transmisión full duplex (esto es, una transmisión

en la cual, la fuente y el destino se pueden comunicar al mismo tiempo, y en dos formas, por ejemplo como es el caso de la conexión telefónica). En este caso el código puede ser diseñado para la detección de errores por que la corrección puede ser realizada solicitando una repetición en la información. Estos esquemas son conocidos como esquemas de solicitud automática de repetición (ARQ) [6, 7].

En un sistema ARQ existe la posibilidad de solicitar la retransmisión de la información de un mensaje dado. Hay otro caso en los sistemas de comunicación para los cuales el modo full-duplex no es posible. Un ejemplo de ellos es un sistema de comunicación conocido como paginación, es decir un envío de caracteres alfanúmericos como mensajes de texto para un usuario móvil. En este tipo de sistema de comunicación el tipo de corrección de errores utilizado es conocido como corrección de errores hacia delante (FEC). A su vez se define un código corrector de errores de alto desempeño como aquel código que permite minimizar el número de errores aprovechando gran parte de la capacidad del canal en la transmisión de información.

La utilización de los códigos correctores de errores de alto desempeño se realiza generalmente en sistemas de comunicación en los cuales el reenvío de un mensaje tiene un alto costo, por ejemplo en las comunicaciones satélitales, aeroespaciales, grabación de disco compacto, transmisión de equipos remotos con batería, televisión digital, sistemas inalambricos, llamadas en teléfonos celulares, entre otros [8].

En el caso del canal de la red eléctrica de baja tensión (PLC) a pesar de utilizarse eficientes esquemas de modulación como OFDM, QPSK, los cuales permiten alcanzar un gran ancho de banda en la transmisión de la información minimizando problemas de atenuación y ruido, los problemas de conexión y desconexión de cargas en el canal pueden generar picos y perturbaciones de gran duración [9]; los cuales pueden provocar que una trama de datos tenga que ser desechada completamente debido a la cantidad de errores en los datos. Para ello es necesario adoptar un sistema de corrección de errores hacia delante (FEC) que permita minimizar estos problemas [10].

Dentro de los códigos correctores de errores de alto desempeño han destacado los turbo códigos y los códigos de chequeo de paridad de baja densidad (LDPC). Siendo ampliamente utilizados dado que se han obtenido muy buenos resultados en el aprovechamiento del canal de comunicación para un número variable del tamaño de tramas de datos trasmitidos [6].

Para obtener códigos que obtuvieran resultados cercanos al límite téorico propuesto por Claude Elwood Shannon, el mismo propuso que las tramas de datos codificados debían ser muy largas, lo cual eleva la complejidad en la decodificación en forma exponencial ya que hay que elegir de entre 2^{n-1} palabras posibles, cuál fue la trama enviada [11]. A pesar de que se tuvieron ideas para tener códigos de bloques con la estructura suficiente para ser

decodificados para grandes longitudes de tramas, no se contaba con algoritmos eficientes de decodificación que pudieran optimizar la capacidad del canal. Fue hasta la aparición de los turbo códigos y los códigos de chequeo de paridad de baja densidad (LDPC) que se lograron obtener muy buenos resultados. Estos códigos se han caracterizado por presentar un comportamiento seudo-aleatorio en la codificación de las tramas y, a su vez, empleando sofisticados algoritmos de decodificación, poder recuperar la trama enviada a pesar de incrementar considerablemente el tamaño o longitud de ésta [12].

1.1. Antecedentes

1.1.1. Códigos correctores de errores

Como se mencionó en la introducción de este capítulo con el descubrimiento de los turbo códigos y los códigos de chequeo de paridad de baja densidad (LDPC) se obtuvieron resultados muy cercanos al límite teórico propuesto por Claude Elwood Shannon, tomando en consideración la capacidad teórica de un canal de comunicación, la cual, es la máxima velocidad a la cual se pueden transmitir datos desde un punto a otro a través de un canal de comunicación y poder recuperar desde el punto receptor el mensaje originalmente transmitido [11, 12]. El descubrimiento de técnicas más eficientes de corrección de errores motivó a los investigadores al estudio y aplicación de nuevos algoritmos de codificación y decodificación, así como mejoras, modificaciones e implementaciones de los algoritmos ya propuestos [13].

Actualmente los turbo códigos y los códigos de chequeo de paridad de baja densidad se han utilizado en forma exitosa con otras técnicas populares de comunicación como por ejemplo: la detección multiusuario, múltiple entrada y múltiple salida (MIMO), y la multiplexación por división de frecuencias ortogonales (OFDM). Esta última técnica es capaz de establecer altas velocidades de transmisión digital en canales selectivos en frecuencia. También, de acuerdo a la literatura, se puede mencionar que los turbo códigos han sido utilizados en distintas aplicaciones como son módems ADSL, comunicación espacial, grabación en discos duros, obteniendo muy buenos resultados.[1] En la figura 1.1 se muestran una aplicación de códigos correctores de errores en comunicación satelital y en televisión digital. Uno de los algoritmos ampliamente utilizado para la decodificación de los turbo códigos, es el algoritmo de máxima probabilidad a posteriori (MAP), conocido también como algoritmo BCJR, por las siglas de sus autores [14]. Este algoritmo realiza la decodificación por medio de un número de iteraciones que, aunque aumentan la latencia del

[1] Keattisak Sripimanwat, Turbo Code Applications, A Journey from a Paper to Realization,Springer,National Electronics and Computer Technology Center (NECTEC),Pathumthani, Thailand,2005.

procesamiento de los datos, logran disminuir en forma considerable la potencia utilizada en la transmisión [4, 15, 16].

Dentro de las implementaciones realizadas con turbo códigos, los sistemas han utilizado versiones con diferentes propósitos, por ejemplo para tratar de minimizar la cantidad de energía utilizada para la transmisión, resolver problemas de altos niveles de ruido, atenuación, variación en frecuencia o tiempo. Es por ello que en el presente trabajo de investigación se realizará la implementación de una versión básica de un turbo código en un sistema de comunicación para el canal de la red eléctrica (PLC) con el objetivo de minimizar el número de errores generados por perturbaciones externas.

Figura 1.1. Algunas aplicaciones de códigos correctores de errores, en comunicación satelital y televisión digital.

1.2. Motivación

Los códigos correctores de errores de alto desempeño se han utilizado en diferentes aplicaciones de un sistema de comunicación como son:

- Comunicaciones satélitales
- Sistemas inalámbricos
- Transmisiones remotas de equipos con batería
- Televisión digital

Estos códigos de corrección de errores hacia delante (FEC) han generado excelentes resultados, muy cercanos al límite téorico de Shannon, por lo cual se han convertido en una interesante área de investigación en aplicaciones en donde no es conveniente el reenvío de mensajes o existe una gran cantidad de ruido, el cual provoca una alta tasa de

errores. Dadas las características del canal PLC, resulta interesante aplicar estas técnicas de corrección para minimizar los errores producidos, y además analizar los resultados obtenidos en condiciones reales en dicho canal de comunicación. En la figura 1.2 se aprecia una aplicación PLC en interior (*indoor*).

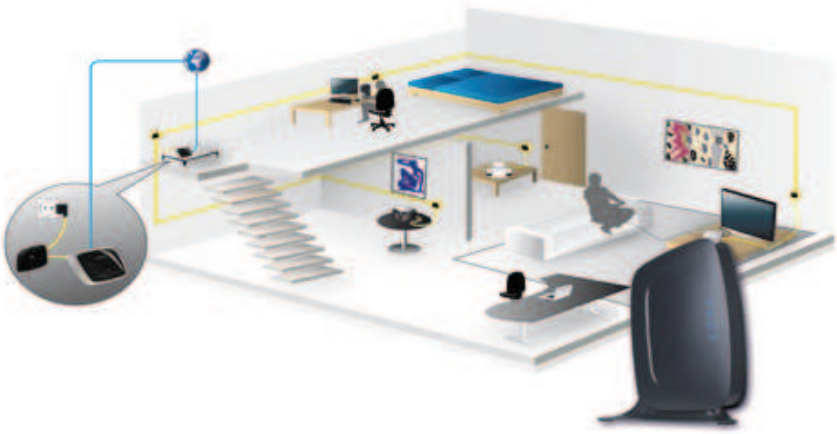

Figura 1.2. Aplicación PLC en interior, empleando la red eléctrica.

1.3. Planteamiento del problema

El canal de la red eléctrica presenta características como ser variante en el tiempo y en la frecuencia además de ser un canal sin memoria. Actualmente no se ha determinado su máxima capacidad en la transmisión de datos. Además, la mayoría de los sistemas implementados presentan patentes y la información no se encuentra totalmente abierta para el diseño y mejora de nuevos sistemas, en los cuales puedan ser probados algoritmos de corrección de errores de alto desempeño. En la actualidad no se cuenta con una biblioteca de códigos disponible que en su conjunto conformen un módulo corrector de errores basado en turbo códigos para el hardware específico.

Debido a que la información transmitida a través del canal de la red eléctrica llega distorsionada por el ruido (impulsivo: periódico y asíncrono, de fondo coloreado, de banda angosta), dadas las características del canal. El problema a resolver consiste en recuperar en forma correcta los datos del lado receptor, utilizando para ello, un código corrector de errores de alto desempeño, brindando así robustez y confiabilidad en la recepción de la información, tomando como fundamento científico las ventajas y beneficios expuestos en

la parte de antecedentes y motivación.

En la figura 1.3 se muestra un sistema de comunicación a través de la línea eléctrica, en donde en forma ilustrativa, los datos se ven afectados por el ruido, por lo cual la importancia reside en poder recuperar la información a pesar de contener errores y con ello reducir el consumo de energía utilizado.

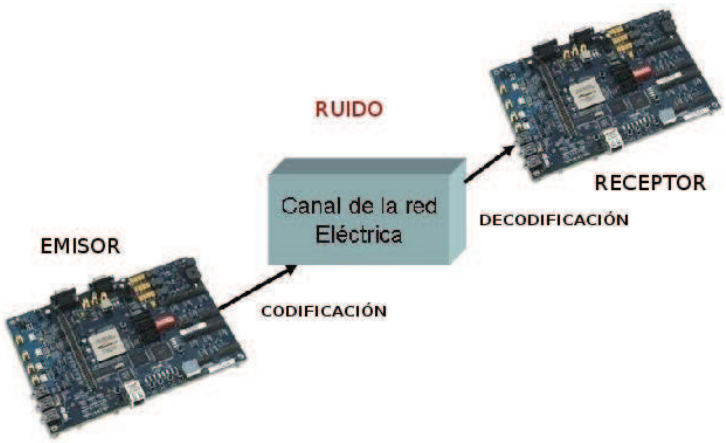

Figura 1.3. Comunicación a tráves de un canal PLC.

1.4. Justificación

La realización del presente trabajo permitirá a los investigadores, tener referencia de la simulación e implementación práctica del hardware así como los algoritmos utilizados para la realización de futuros proyectos y también servirá como plataforma de desarrollo para la implementación de códigos correctores de errores propuestos. También la presente tesis permitirá tener un desarrollo tecnológico abierto ya que en la mayoría de los casos, los desarrollos no se encuentran totalmente disponibles para su comprensión y modificaciones o para mejoras futuras.

1.5. Hipótesis

Dados los buenos resultados de la aplicación de los códigos correctores de errores de alto desempeño en sistemas de comunicación en donde existen altos niveles de ruido y perturbaciones. Se propone en la presente tesis utilizar un código de alto desempeño para el

canal de la red eléctrica PLC con el objetivo de minimizar el número de errores producidos en dicho canal de comunicación, realizando para ello su implementación en hardware (DSP) comúnmente utilizado para tal fin.

1.6. Objetivos

1.6.1. Objetivo general

Desarrollar un módulo corrector de errores considerando las características de la red eléctrica que permita la correcta transmisión y recepción de datos basándose en un esquema de turbo código.

1.6.2. Objetivos específicos

- Estructurar la plataforma básica de hardware para la comunicación.
- Desarrollar una biblioteca de rutinas para analizar las funciones básicas del módulo.
- Simular e implementar el módulo corrector de errores en un DSP y analizar los resultados obtenidos.

1.7. Método de investigación y desarrollo utilizado

En el desarrollo del presente trabajo se analizará el desempeño de un código corrector de errores para un canal PLC realizando simulaciones en C y MATLAB. También se llevará a cabo la implementación de dichos algoritmos en el procesador digital de señales incluido en la plataforma de desarrollo TMS320C6416T DSK, comparando su funcionamiento con base en los resultados obtenidos en las simulaciones realizadas. Se analizará su desempeño variando algunos parámetros básicos propios de dicho código corrector.

La metodología a seguir se basa en la simulación de los módulos programados utilizando las herramientas anteriormente mencionadas, y en caso de que los algoritmos sean correctos se agregan a los demás elementos componentes del código corrector, y en caso contrario se revisan las rutinas hasta lograr un correcto funcionamiento. Una vez implementado el módulo en su conjunto, se prueba su desempeño bajo diferentes condiciones. Todo lo anterior se sustenta sobre el método de investigación hipotético deductivo. En la figura 1.4 se puede observar la metodología utilizada en la realización del presente trabajo de investigación.

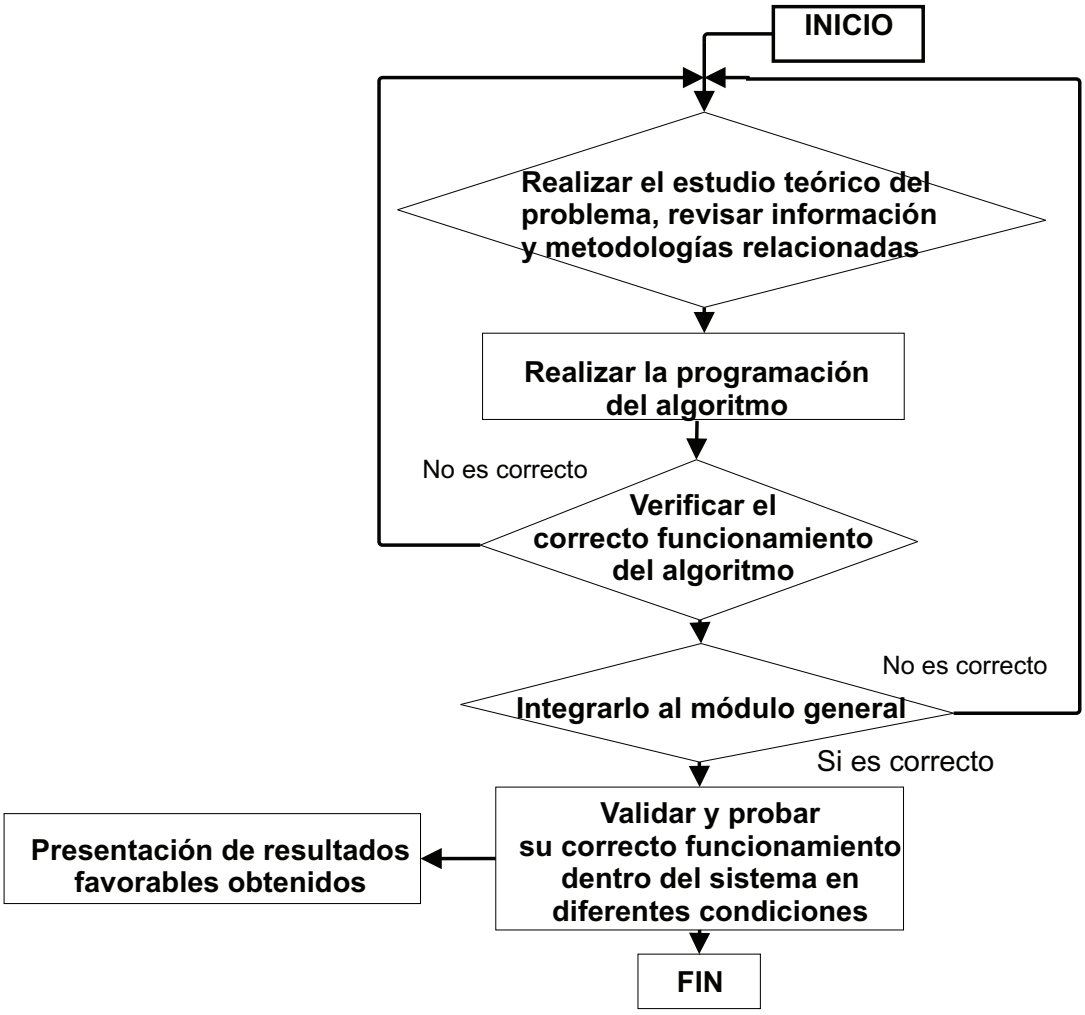

Figura 1.4. Metodología utilizada en la investigación

1.8. Alcances del trabajo

En el presente trabajo se ilustra la programación, simulación e implementación de un código corrector de errores de alto desempeño para un canal PLC, así como el desarrollo de una plataforma básica de comunicación para analizar el módulo de corrección de errores, y con esto poder garantizar la correcta recuperación de los datos en un canal de comunicación, como el de la red eléctrica. A su vez, se estudian y se muestran a detalle los instrumentos de hardware y software necesarios para la programación y validación de los algoritmos del turbo código utilizado.

Se realiza el estudio del estado del arte de la problemática y aplicación de dichos códigos correctores. Se realiza la programación de los códigos para los diferentes elementos que conforman el bloque transmisor y receptor en el esquema de la corrección de errores. Así como una biblioteca de códigos para el módulo de corrección de errores. Finalmente se realizan pruebas dentro del hardware variando diferentes parámetros del módulo corrector de errores bajo diferentes condiciones, validando los resultados obtenidos.

1.9. Contribuciones

1.- **Un módulo simulador** de un código de corrección de errores de alto desempeño para un canal PLC.

2.- **Una plataforma básica de comunicación**, en la cual se pueden probar en forma práctica algoritmos correctores de errores propuestos.

3.- **Una biblioteca de funciones de código abierto** que constituyen el módulo corrector de errores.

4.- **Una implementación del módulo de corrección de errores** que al utilizar un simulador del canal PLC, permite conjeturar que tendrá un correcto funcionamiento cuando se utilice en un canal de la red eléctrica.

1.10. Conferencias en congresos internacionales

- **Low complexity Turbo Code Specification for Power-line Communication (PLC)**. Alfonso Martínez-Cruz, Ricardo Barrón-Fernández, José Luis Oropeza-Rodríguez, Gerardo A. Laguna-Sánchez. Center for Computing Research. 2011 IEEE Electronics, Robotics and Automotive Mechanics Conference. CERMA 2011.

- **Low Complexity Turbo code implementation in DSP TMS320C6416T.** Alfonso Martínez-Cruz, RicardoBarrón Fernández, José Luis Oropeza Rodríguez. 7o Congreso Internacional:TENDENCIAS TÉCNOLOGICAS EN COMPUTACIÓN. Centro de Innovación y Desarrollo Tecnológico en Cómputo, 2011.

1.11. Estructura de la tesis

La estructura de la tesis es la siguiente:

- **Capítulo 1:** Se presenta una introducción acerca de la importancia de los sistemas de comunicación así como los antecedentes y la importancia de la utilización de códigos correctores para minimizar los errores. También se exponen los fundamentos que conforman el presente trabajo de investigación entre los cuales se encuentran: el planteamiento del problema, los objetivos, el alcance, las aportaciones; entre otros.

- **Capítulo 2:** Se describe en forma breve, el estado del arte actual con respecto a la utilización de los códigos correctores de errores de alto desempeño, enfocándose en los turbo códigos en los sistemas de comunicación, así como los avances en la tecnología empleada y las investigaciones realizadas. Se presentan distintas aplicaciones de los turbo códigos. En la parte final se presenta un resumen con los aspectos relevantes tratados en el capítulo.

- **Capítulo 3:** Se describen las partes esenciales que componen un turbo código, así como la estructura general de un turbo codificador y principales características. Se describen los principales algoritmos utilizados en la decodificación de los turbo códigos como son: Algoritmo de máxima probabilidad a posteriori (MAP) y el algoritmo Max-Log-MAP [5, 17]. Se explican a detalle los fundamentos teóricos del sistema propuesto y las características de los algoritmos utilizados.

- **Capítulo 4:** Se presenta el módulo propuesto. En este capítulo se describen cada uno de los elementos componentes de dicho sistema así como las características de la implementación del módulo corrector de errores de alto desempeño realizada.

- **Capítulo 5:** Se muestran y analizan los resultados obtenidos bajo las condiciones propuestas en el sistema implementado. Así como el entorno bajo el cual se llevaron a cabo dichas pruebas. Se reportan las curvas BER en un canal con ruido gaussiano y ruido impulsivo así como el comportamiento del algoritmo de decodificación implementado. Se presentan las gráficas con los valores calculados de probabilidad en

diferentes iteraciones en el algoritmo. Además se muestran los resultados obtenidos en la decodificación de imágenes con ruido gaussiano.

- Se presentan las conclusiones así como los trabajos futuros de acuerdo a los resultados obtenidos y el desarrollo presentado en los capítulos anteriores.

1.12. Resumen del capítulo

En el presente capítulo se describió una breve introducción a los códigos correctores de errores así como a la importancia de los sistemas de comunicación en la actualidad. Ya que gracias a la aplicación de técnicas avanzadas de procesamiento digital de señales, así como de esquemas de corrección de errores hacia adelante (turbo códigos y códigos LDPC), se ha logrado diseñar mejores equipos de comunicación.

También se presentaron los fundamentos que para realizar este trabajo de investigación tales como: el planteamiento del problema, la justificación y la hipótesis. Además se explicó la metodología utilizada, los alcances y las contribuciones de la investigación.

Además se explicaron los objetivos de este trabajo de investigación, entre los cuales se encuentra desarrollar una biblioteca de rutinas para analizar las funciones básicas del módulo corrector de errores implementado en el DSP. Finalmente se describió la estructura general del presente trabajo por cada uno de los capítulos. En el siguiente capítulo se presentará el estado del arte relacionado con los turbo códigos, así como, su aplicación en un canal PLC.

Capítulo 2

Estado del arte

En este capítulo se describirá el estado del arte relacionado con la historia de los turbo códigos así como algunas de sus aplicaciones actualmente. También se mencionará la importancia de la utilización de los códigos correctores de errores en los sistemas de comunicación. Se comentará acerca de los estudios realizados de la aplicación de los turbo códigos en el canal PLC. Finalmente se dará un resumen del presente capítulo.

2.1. Códigos correctores de errores FEC

En el año de 1948 Claude E. Shannon dio a conocer su teoría de comunicación en la cual propuso un límite teórico para trasmitir datos a través de un canal de comunicaciones. También propuso la utilización de códigos aleatorios para poder aprovechar de mejor forma un canal de comunicación dado [18]. Desde entonces ingenieros, investigadores y matemáticos han buscado construir códigos que sean buenos para la corrección de errores y al mismo tiempo puedan ser implementados, es decir que no sean tan complejos. En el año de 1950 R. W. Hamming presentó los códigos de corrección de errores conocidos como códigos Hamming [19]. Para el año de 1963 Robert G. Gallager descubre los códigos de chequeo de paridad de baja densidad [20] los cuales, en ese entonces, no fueron muy promovidos a pesar de mostrar buenos resultados debido a que su implementación requería de dispositivos complejos para ese entonces.

Por la década de 1970 se realizaron investigaciones pioneras subsecuentes con el objetivo de construir familias de códigos de longitudes más grandes y que fueran capaces de eficientar el uso del canal. Fue en los años de la década de los 80s cuando se da un paso de la teoría a la implementación práctica con el desarrollo de codificadores y decodificadores diseñados para sistemas de comunicación con lo cual inicia una etapa de desarrollo tecnológico. Gracias

a la invención de nuevos códigos y técnicas de modulación y transmisión se han obtenido resultados exitosos, y muy cercanos al límite teórico investigado por Shannon.

Entre los códigos descubiertos a lo largo de los años se encuentran: los códigos de bloques, códigos de Hamming, códigos convolucionales, y el algoritmo Viterbi, Bose, los códigos Chaudhuri y Hocquenghem (BCH), los códigos Reed-Solomon (RS) y la modulación codificada Trellis (TCM) hasta la aparición de los turbo códigos en los años de la década de los 90s y el redescubrimiento de los códigos de chequeo de paridad de baja densidad (LDPC) por D. J. C. Mackay and R. M. Neal en el año de 1995 [21].

En el año de 1993 se realizó en Genova Suiza la conferencia internacional anual en comunicaciones. En este evento Claude Berrou, Alain Glavieux y Punya Thitimajshima presentaron la invención de un nuevo esquema de corrección de errores. Este esquema presentó mejores resultados que las técnicas actualmente existentes y una mucho mejor ganancia en la codificación [14]. Fue mostrada una gráfica en simulación en la cual su desempeño fue cercano a la capacidad teórica del canal. Ésta presentó una probabilidad de error de 10^{-5} para un valor de SNR de 0.7 dB. Estos resultados inspiraron a los investigadores a proponer nuevos esquemas de codificación, mejoras y simplificación de los códigos existentes para resolver los problemas en la comunicación. En la figura 2.1 podemos apreciar los códigos correctores de errores actualmente más utilizados.

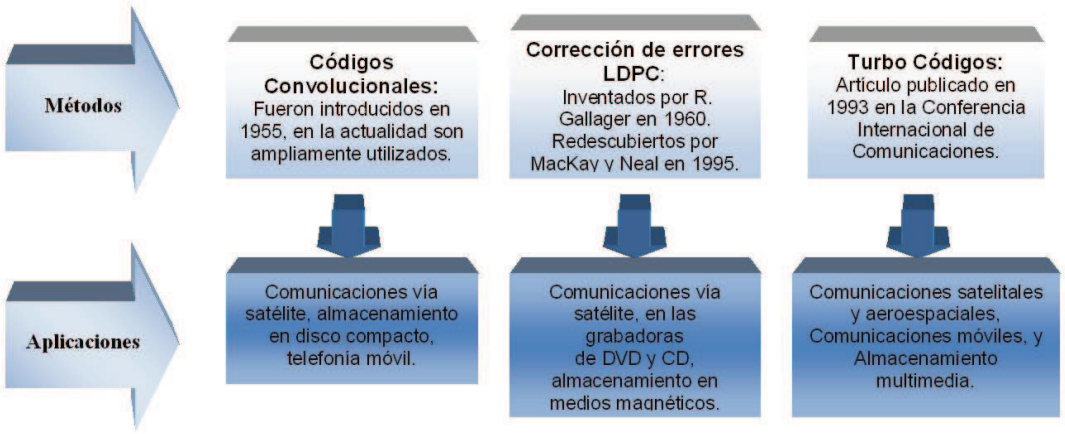

Figura 2.1. Códigos correctores más utilizados actualmente.

Hay dos publicaciones importantes realizadas a partir del descubrimiento de los turbo códigos. La primera de ellas es "Recursive Systematic Convolutional codes and application to parallel concatenation", que fue presentada en la conferencia IEEE Globecom 1995 por Thitimajshima. Sin embargo, un año después fue publicado otro artículo llamado

"Near optimum error correcting coding and decoding: turbo-codes" (en transactions and communications IEEE). Este artículo fue publicado en octubre de 1996 y escrito por Claude Berrou y Alan Glavieux.

Desde la aparición de los turbo códigos inició una nueva etapa de desarrollo e investigación de nuevas técnicas que mejoraron el desempeño de los sistemas de comunicación combinando métodos existentes. Este hecho no solamente impactó a la comunidad tecnológica sino en el ámbito social, económico, y académico. La realización de diferentes trabajos relacionados han generado más de 400 patentes. Por lo cual ha sido considerado un núcleo tecnológico en los productos de comunicación [8]. En la tabla 2.1 se muestra el estado del arte relacionado con la historia de los turbo códigos.

Tabla 2.1. Antecedentes de los turbo códigos.

Año	Evento presentado
1986	Claude Berrou, Alain Glavieux, and Patrick Adde iniciaron el trabajo en Soft-Output Viterbi Algorithm (SOVA) y una retroalimentación estuvo basada en los artículos de G. Battail (1987) and J. Hagenauer.
1988	Patente (número de aplicación : Número de patente) FR91 05279 : FR 2 675968 (Decodificación de máxima probabilidad, inventor: C. Berrou y P. Adde), 23 de abril.
1990	Patente FR91 05280 : FR2 675 971 (Turbo Códigos - Inventor : C. Berrou), 23 de abril.
1992	Patente US 870,483 : US 5,406,570 (Decodificación ML - Inventor : C. Berrou y P. Adde), 16 de abril.
1992	Patente EP92 460011.7 : EP 0 511 139 (Decodificación ML - Inventor : C. Berrou y P. Adde), 22 de abril.
1992	Patente US 870,614 : US 5,446,747 (códigos turbo - Inventor : C. Berrou), 16 de abril.
1992	Patente EP92 460013.3 : EP 0 511 141 (códigos turbo - Inventor : C. Berrou), 22 de abril.
1993	Near Shannon limit error correcting coding and decoding: Turbo-Codes por Claude Berrou, Alain Glavieux y Punya Thitimajshima fue presentado en ICC'93 en Genova con aplicación de patente no. FR91 05279, EP92 460011.7 y US 870,483 (Decodificación ML).
1994	Códigos convolucionales sistemáticos recursivos y aplicación para concatenación paralela por Punya Thitimajshima fue publicada en Globecom'95.
1996	Near Optimum Error Correcting Coding and Decoding : Turbo-Codes por Claude Berrou y Alain Glavieux fue publicada en IEEE Transactions on Communications en octubre.
1996	El premio IEEE Stephen O. Rice (Mejor artículo en IEEE Trans. Commun.) fue entregado a Claude Berrou y Alain Glavieux.
1996	El premio IEEE Information Theory Society fue entregado a Claude Berrou y Alain Glavieux por su publicación en IEEE Trans. Commun. in 1996.
1998	Claude Berrou, Alain Glavieux, y Punya Thitimajshima recibieron el premio Golden Jubilee por innovación tecnológica por la invención de los Turbo códigos en agosto.
2003	Claude Berrou y Alain Glavieux recibieron la medalla IEEE Richard W. Hamming por la invención de los turbo códigos, los cuales han revolucionado las comunicaciones digitales.
2004	10 años, aniversario por la invención de los turbo códigos (1993-2003).

Al ser presentados los turbo códigos por sus autores se publicaron las patentes con números 91 05279 (Francia), 92 460011.7 (Europa), y 07/870,483 (USA). Los primeros dos números fueron titulados en francés como: *"Procédé de décodage dún code convolutif*

à maximum de vraisemblance et pondération des décision, et décodeur correspondants" para Francia y Europa. Para Estados Unidos se tituló: *"Method for a maximum likelihood decoding of a convolutional code with decision weighting, and corresponding decoder"*. Sin embargo la primera patente fue hecha en Francia y se tituló: *"Procéde de codage correcteur dérreurs à aumoins deux codages convolutifs systématiques en parallèle, procédé de décodage itératif, module de décodage et décodeur correspondants"* ó *"Error-correction coding method with at least two systematic convolutional coding in parallel, corresponding iterative decoding method, decoding module and decoder"*. La primera fue realizada en Francia el 23 de abril de 1991 (número 91 05280). Esta patente aplica para los 20 años despúes de haberse expedido, para su uso se requiere permiso, licencia, o términos de agradecimientos. Sin embargo para los países en los cuales no fue realizada se encuentra disponible para su posible exploración comercial [8]. En la figura 2.2 se puede apreciar dicha patente.

Institut National de la Propriété Industrielle (INPI), France	
Title of invention : Proédé de codage correcteur d'erreurs à au moins deux codages convolutifs systématiques en parallèle, procédé de décodage itératif, module de décodage et décodeur correspondants	
Inventor	Claude Berrou
Assignee	France Telecom and Telediffusion de France S.A.
Application number	91 05280
Patent number	2675971
Filing date	April 23, 1991
European Patent Office(EPO), Europe	
Title of invention : Proédé de codage correcteur d'erreurs à au moins deux codages convolutifs systématiques en parallèle, procédé de décodage itératif, module de décodage et décodeur correspondants	
Inventor	Claude Berrou
Assignee	France Telecom and Telediffusion de France S.A.
Application number	92 460013.3
Patent number	0 511141
Filing date	April 22, 1992
United States Patent Office (USPTO), USA	
Title of invention : Error-Correction coding method with at least two systematic convolutional coding in parallel, corresponding iterative decoding method, decoding module and decoder	
Inventor	Claude Berrou
Assignee	France Telecom and Telediffusion de France S.A.
Application number	870614
Patent number	5446747
Filing date	April 16, 1992

Figura 2.2. Patente de turbo códigos.

En la época actual los esfuerzos se enfocan a resolver varios problemas en las implementaciones de códigos correctores de errores entre los cuales se encuentran: disminuir la complejidad en la decodificación, mejorar la ganancia en la codificación, asociar los métodos de codificación a las técnicas actuales de comunicación, entre otros.

2.1.1. Aplicaciones de turbo códigos

Aunque muchas veces no lo percibimos, la codificación de la información se encuentra a nuestro alrededor, desde la telefonía celular, la grabación en algún medio de almacenamiento o la recepción de señal en la televisión. Desde la aparición de los códigos correctores de errores hacia delante y los resultados que se obtuvieron en simulaciones, una de las prioridades ha sido implementar dichos códigos en distintas aplicaciones con el objetivo de poder asegurar el manejo de la información a altas velocidades y en diferentes medios. En la década de los años 90s con el descubrimiento de los turbo códigos y los códigos de chequeo de paridad de baja densidad (LDPC), inicio una nueva etapa de desarrollo en la implementación de estas técnicas junto con otras para resolver y mejorar soluciones en los medios de comunicación. En el caso de los turbo códigos se necesitaron simplificar y reducir los algoritmos inicialmente propuestos y a la vez diseñar arquitecturas orientadas a éstos.

En la actualidad algunas aplicaciones en donde se utilizan los turbo códigos son: comunicaciones satelitales y aeroespaciales, comunicaciones móviles e inalámbricas, almacenamiento multimedia, módems ADSL, comunicaciones sobre fibra óptica, televisión digital, telemetría y comunicaciones en el espacio libre. En la figura 2.3 se pueden observar algunos de los dispositivos donde se aplican los turbo códigos en la telefonía digital.

Las aplicaciones de los turbo códigos se han caracterizado por ser utilizadas en condiciones complicadas para un código con bajo desempeño. Algunas de estas condiciones son: la presencia de medios de comunicación ruidosos (variantes en el tiempo, frecuencia), sistemas de comunicación en los cuales la retransmisión de información tiene un alto costo, dispositivos que requieren minimizar el consumo de energía (por ejemplo que utilizan baterías). Debido al proceso iterativo para su decodificación, este código corrector ha tenido algunas restricciones sobre todo en sistemas en donde se requiere procesamiento en tiempo real y en que la latencia es un factor fundamental, por ejemplo transmisiones ópticas.

El impacto tecnológico de la aparición de los turbo códigos trajo consigo la reducción en el tamaño de las antenas, paneles solares, el tamaño de las baterías utilizadas para dispositivos de comunicación, tamaño de dispositivos móviles; así como el desarrollo de circuitos integrados más complejos y de equipos de comunicación. Debido a ello muchas compañías se han interesado por su desarrollo e implementación para el diseño de mejores

Figura 2.3. Aplicación de turbo códigos en la telefonía digital.

productos y reducción de costos [22].

Cada año se realiza el *" Simposium internacional de turbo códigos y procesamiento iterativo de información "*, el cual tiene como objetivo dar a conocer el estado actual y las investigaciones realizadas de los métodos iterativos y de su aplicación en la teoría de la codificación de la información y de las comunicaciones digitales. Las dos principales técnicas de codificación actuales (turbo códigos y LDPC) son estudiadas así como variantes y métodos iterativos de codificación [23].

2.1.2. Comparación de las características de turbo códigos

En la época actual los dos tipos de códigos más utilizados en los sistemas de comunicaciones modernos son los turbo códigos y los códigos de chequeo de paridad de baja densidad (LDPC). Estos códigos se caracterizan por presentar un comportamiento seudo-aleatorio en la construcción de los bloques de datos, así como también en utilizar métodos iterativos en la decodificación de la información.

Los turbo códigos presentan varios parámetros que influyen en su desempeño y a la vez son adecuados para longitudes de bloques de datos constantes y sus aplicaciones son diversas en la codificación de información. En el presente trabajo se utilizó un esquema basado en turbo códigos tomando en consideración su gran desempeño BER y que el canal de la red eléctrica puede presentar mayores niveles de ruido que en un canal de comunicación común. En la tabla 2.2 se muestran las principales ventajas y desventajas de

los turbo códigos y códigos LDPC (Low Density Parity Check).

Tabla 2.2. Comparación de ventajas y desventajas de turbo códigos y códigos LDPC.

Tipo de código	Ventajas	Desventajas
Turbo códigos	En la actualidad están posicionados como los mejores. Presentan aplicaciones incluso fuera del ámbito de transmisión de datos. Incrementan la tasa de datos sin necesidad de incrementar la potencia de transmisión. Son adecuados para aplicaciones de restricción de la longitud de datos y bloques intermedios.	Relativa alta complejidad de decodificación, así como alta latencia.
Códigos LDPC (Low Density Parity Check Codes)	Su desempeño BER es cercano al de los turbo códigos. Versiones modificadas presentan cada vez un mejor desempeño. Su complejidad en la decodificación es menor. Su BER es muy bueno si la longitud de código es lo suficientemente grande ($n = 10,000$). Tiene ausencia de patente.	Pueden llegar a necesitarse más iteraciones en la decodificación por lo cual la complejidad puede aumentar.

2.2. Desarrollo de la tecnología digital

El desarrollo creciente de la tecnología CMOS digital ha permitido el diseño de circuitos integrados cada vez más potentes sobre los cuales ha sido posible realizar la implementación de algoritmos complejos que anteriormente no eran posibles de realizar. La fabricación de plataformas de desarrollo de hardware y software más complejas, como microprocesadores, procesadores embebidos, FPGAs, DSPs; entre otros, ha generado el desarrollo de nuevos y mejores sistemas de comunicación.

En el proceso de desarrollo de los dispositivos para los sistemas de comunicación ha sido necesario implementar algoritmos cada vez más complejos con los cuales se obtengan aplicaciones más eficientes y a su vez mejoren el consumo de energía, latencia, disipación de energía; entre otras. En el caso de los procesadores digitales de señales (DSP) además de ser utilizados para el procesamiento de señales son empleados en detección y corrección de errores. Lo anterior se ha llevado a cabo con el objetivo de incrementar la velocidad de procesamiento en los datos. Además de su gran flexibilidad, la cual ha hecho que estos dispositivos sean muy utilizados como parte fundamental en los sistemas de comunicación.

En el caso de los turbo códigos hay varios aspectos que los han hecho interesantes para su implementación, por ejemplo: los algoritmos utilizados para su decodificación son

complejos para las arquitecturas de hardware y su utilización requiere gran espacio en memoria RAM y con ello problemas de latencia. También se puede mencionar que entre los principales problemas que se están resolviendo e implementando se encuentran los siguientes:

- Construcción de arquitecturas eficientes para las operaciones relacionadas con los algoritmos de decodificación.
- Diseño de arquitecturas que satisfagan las aplicaciones, dependientes del costo, beneficio, consumo de potencia y latencia.
- Construcción de técnicas para la reducción y el manejo del consumo de energía.
- Diseño de arquitecturas que estén libres de colisiones en cuanto a paralelización de los algoritmos.

Para disminuir el tiempo de latencia y a la vez incrementar la capacidad de procesamiento se han dado desarrollado coprocesadores para realizar la turbo codificación, tal es el caso de algunos DSPs de la compañía Texas Instruments. También se han fabricado circuitos integrados específicos para realizar esta tarea de acuerdo a la aplicación utilizada. En el caso de la tecnología PLC hay varias compañías que desarrollan circuitos integrados orientados a esta tecnología cuyo objetivo es ser parte fundamental en el diseño de equipos y poder ofrecer soluciones potentes y de bajo costo integrando varias técnicas utilizadas en este medio de comunicación. Entre ellas se encuentran: DS2, SPiDCOM, Intellon, Siconect, gigle, xeline, MARVELL; entre otras. Gracias al gran desarrollo a nivel semiconductor estos circuitos integrados han podido integrar soluciones potentes a bajo costo tal es el caso del circuito integrado SPC310 de la compañía SPiDCOM, el cual puede trabajar a velocidades de hasta 200 Mbps, utilizando modulación OFDM, turbo código FEC avanzado y adaptación de canal [24]. En la figura 2.4 se muestra el circuito integrado Aitana desarrollado por la compañía DS2 el cual fue diseñado para módem PLC que puede alcanzar velocidades de 200 Mbps utilizando como esquema de modulación OFDM. Este circuito integrado permite la configuración en frecuencia programable (2-32 MHz).

2.2.1. Estándares

Con el desarrollo de dispositivos PLC comerciales se han dado necesidades en forma similar a las del desarrollo de otras tecnologías sobre un medio de comunicación. Entre ellas se encuentran la creación de estándares que permitan la convivencia de los dispositivos PLC creados por diferentes compañías, comunicación sin interferencias con señales en otros medios de comunicación, flexibilidad en complementación con otras tecnologías.

Figura 2.4. Chip Aitana de la compañía DS2.

Para la regulación de la tecnología PLC se han constituido organizaciones que puedan dar impulso a su comercialización, desarrollo e investigación. En el año 2000 varias empresas decidieron formar un grupo llamado *Home Plug Powerline Alliance*, el cual, se constituyó de un grupo de compañías que impulsan el desarrollo de esta tecnología. Actualmente este grupo ha creado estándares, certificaciones y especificaciones para el desarrollo de equipos que utilizan dicha tecnología. Lo anterior con el objetivo de tener compatibilidad entre equipos y a la vez poder brindar un buen servicio a través de este canal de comunicación. Entre sus estándares se encuentran: Homeplug 1.0, Homeplug BPL y Homeplug AV, los cuales han sido tomados por diferentes fabricantes de semiconductores y equipos PLC [25]. El primer estándar ANSI para esta tecnología de banda ancha fue "TIA-1113 standard", este estándar se basa ampliamente en las especificaciones de Homeplug 1.0 y está definido a 14 Mbps basado en OFDM [26].

En el año de 2005, IEEE formó un grupo de trabajo para establecer un nuevo estándar, el cual, sirviera para unificar la tecnología PLC de banda ancha y a la vez permitiera a los fabricantes diseñar equipos con características compatibles y que a la vez brindarán un mejor servicio a los usuarios. Finalmente en enero del 2010, IEEE publicó el estándar IEEE P1901. El estándar puede utilizarse para todos los dispositivos PLC que trabajen en banda ancha, incluyendo a los equipos que sean utilizados como de primera y última milla. Teniendo como restricción un distancia de 1,500 mts., entre ellos. Las aplicaciones pueden ser distintas: redes LAN, aplicaciones de baja energía, aplicaciones sobre vehículos, aplicaciones de distribución; entre otras. El tipo de modulación puede ser OFDM con IFFT utilizando turbo códigos y la modulación OFDM con wavelets utilizando códigos de corrección convolucionales y códigos de chequeo de paridad de baja densidad (LDPC) [27].

El estándar se encuentra limitado a la capa física y a la subcapa de enlace en el modelo OSI. El módulo corrector de errores utiliza grupos de octetos llamados bloques físicos (PBs) de 520 y 136 octetos de longitud. El turbo codificador puede utilizar una tasa de código

de 1/2, 16/21 ó 16/18. Varias compañías vendedoras de semiconductores han anunciado planes para producir circuitos integrados basados en el estándar IEEE P1901 [28]. Otros de los estándares publicados por IEEE son: IEEE P1675, IEEE P1775 [25]. La tabla 2.3 presenta algunos de los estándares para la tecnología PLC.

Tabla 2.3. Estándares relacionados con la tecnología PLC.

Nombre	Frecuencia utilizada
HomePlug GP	10MBits/s
HomePlug 1.0	14MBits/s
HomePlug 1.1	85MBits/s
UPA (DS2)	T85MBits/s
UPA-HD (DS2)	200MBits/s
HomePlug AV	200MBits/s
HD-PLC (CEPCA)	200MBits/s
IEEE P1901	200MBits/s

Existen otros grupos que también se han encargado de dar impulso al desarrollo de la tecnología PLC entre los cuales se encuentran: *POWERNET*, *Universal Powerline Association (UPA)*, *CEPCA*, *OPERA (Open PLC European Research Alliance)*, *Consumer Electronics Powerline Communications Alliance (CEPCA)*. Estos grupos también han buscado la convivencia y mejora de los dispositivos relacionados con esta tecnología, estableciendo especificaciones y estándares que ayuden a mejorar las capas constituyentes de estos equipos. La figura 2.5 muestra las principales compañías y estándares de la tecnología PLC a través del tiempo.

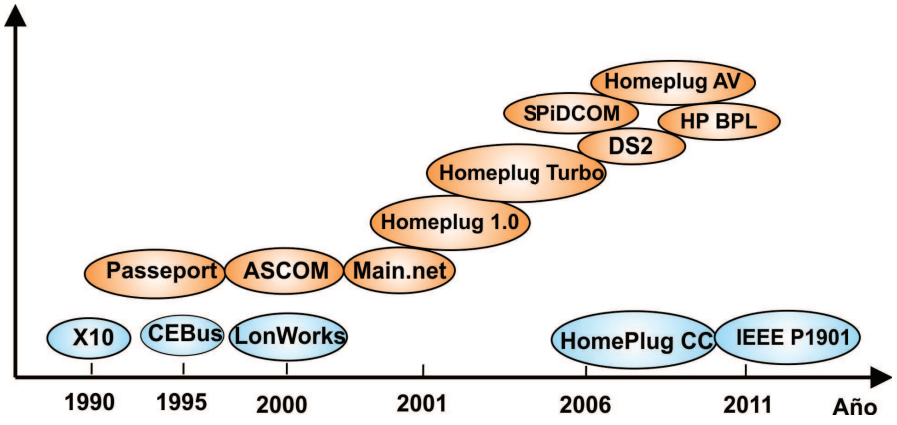

Figura 2.5. Compañias y estándares a través del tiempo

2.3. Estudios realizados de aplicación de turbo códigos en el canal PLC

Se han realizado varios estudios de la aplicación de turbo códigos para un canal PLC, en el cual se han utilizado con diferentes técnicas de modulación entre las cuales se encuentran: QPSK, TDMA, CSMA, OFDM; entre otras. De acuerdo a las investigaciones realizadas la complementación de OFDM con esta técnica de codificación de información ha generado muy buenos resultados en cuanto al aprovechamiento del ancho de banda y a la vez una mayor robustez a los errores provocados por las características propias de dicho canal.

La variación de los diferentes parámetros de los turbo códigos ha sido utilizada para estudiar su comportamiento dentro de este canal de comunicación buscando su mejor funcionamiento bajo diferentes condiciones de ruido y variación en tiempo y frecuencia en el canal. Encontrando que su comportamiento varía de acuerdo a la fluctuación de la varianza del ruido [29].

Otras investigaciones han utilizado variaciones y adaptaciones de algoritmos que puedan disminuir los efectos del ruido generado en el canal y a su vez permitan mejorar la tasa de errores (BER). Dado que el ruido impulsivo generado en un canal PLC es uno de los factores que degradan en forma considerable el desempeño de las técnicas de corrección de errores, varias investigaciones han tratado de disminuir sus efectos utilizando modificación de algoritmos [30].

Se puede mencionar que en las investigaciones del comportamiento de los turbo códigos en este canal ha sido importante considerar el modelo seleccionado para representar dicho medio y por lo tanto las características del ruido, las cuales afectan directamente el comportamiento del sistema propuesto. También con la creación de estándares se han realizado esquemas que, cumpliendo con ellos, permiten además estudiar su comportamiento dadas las condiciones propias del medio de comunicación, con base en el número de iteraciones en el algoritmo, y tamaño de la trama de datos; entre otros [31].

Con respecto al canal PLC las investigaciones han mostrado que este medio de comunicación es complicado de modelar y el cual presenta varias características que lo hacen un canal ruidoso. Algunas de estas características son [32]:

- Respuesta variante en el tiempo y en la frecuencia.
- Una función de transferencia que depende de la topología de la red, la localización, cargas conectadas, condiciones físicas.
- Varios tipos de ruido: ruido impulsivo periódico, ruido impulsivo aleatorio, ruido de fondo coloreado, ruido de banda angosta.

En el canal PLC además del ruido Gaussiano existe ruido impulsivo, por lo cual, se han realizado investigaciones que estudian el ruido en este medio de comunicación [33]. Esta es una de las características por la que al tratar de encontrar mejores soluciones para disminuir la tasa de errores BER en el canal PLC, se han realizado pruebas incluyendo diferentes escenarios, los cuales permitan mostrar los factores que degradan el desempeño del método corrector de errores en dicho medio, ya que es importante en la realización de simulaciones e implementaciones prácticas [3].

Otros aspectos que se han considerado son la impedancia y la atenuación; los cuales son factores que también influyen en la trasmisión de datos y, a la vez, limitan la distancia entre equipos así como el número de repetidores utilizados, etcétera.

Se han incluido los turbo códigos en varios estándares para el canal PLC utilizando modulación OFDM y esto ha impulsado su aplicación para este tipo de canal gracias a su alto desempeño en condiciones con alto nivel de ruido. En la tabla 2.4 se muestra el estado de arte de aplicaciones PLC y turbo códigos, así como varios artículos importantes.

Tabla 2.4. Estado del arte relacionado a códigos correctores de errores en un canal PLC.

Comunicaciones	Aplicaciones PLC	Turbo códigos
C. E. Shannon, **A Mathematical Theory of Communication**, The Bell System Technical Journal, Vol. 27, July, October, 1948.	J. L. Carmona y F. J. Cañete y Cortes y L. Diez. **A DMT módem prototype for broadband PLC.** IEEE, 2008.	Claude berrou and Alain Glavieux, **"Near Optimum Error Correcting Coding and Decoding:Turbo-Codes"**, IEEE, transacctions of communications, vol 44, No. 10, pp. 1064- 1070, octuber 1996.
Gerardo Abel Laguna, **Aplicación de la transformada de ondeleta para la estimación de canales PLC**, Instituto Politécnico Nacional, Centro de Investigación en Computación, Zacatenco, México, D. F., 14 de junio de 2010.	H. A. Garcia-Baleon y V. Alarcon-Aquino, **A Power-Line Communication Módem based OFDM**, Department of Computing, Electronics and Mechatronics, Universidad de las Americas, Puebla, 2009.	M. C. Rafael Antonio Márquez Ramírez y M. C. Pablo Manrique Ramírez. **Técnicas de Codificación Turbo para el Control de Errores en un Canal de Comunicación.** CIC, México 2009.
Manfred Zimmermann and Klaus Dostert. **An Analysis of the Broadband Noise Scenario in Powerline Networks.**, International Symposium on Powerline Communications and its Applications, Limerick, Ireland. April 2000.	Luis Simões y José A. B. Gerald. **A Communication System for Power Lines.**, Proc. 5th Conference on Telecommunications, Tomar, Portugal, April 2005.	Tadahiro WADA. **A study on the performance of turbo coding for noise environments in power lines.** IEEE,2003.
	Rastislav Róka, **The Design of a PLC modem and its Implementation using FPGA circuits**, Journal of electrical engineering, vol. 60, No. 1, 2009, pp 43-47, Department of Telecommunications, Faculty of Electrical Engineering and Information Technology, Slovak University of Technology, Ilkovicova 3, 812 19 Bratislava, Slovakia,2009.	Daisuke Umehara, Hiroshi Yamaguchi and Yoshiteru Morihiro, **Turbo Decoding in Impulsive Noise Environment**, IEEE Communications Society, Globecom, Japan 2004
		L. Guerrieri, P. Bisaglia, G. DellÁmico, E. Guerrini, **Performance of the turbo coded HomePlug AV system over power-line channels**, Dora Spa, STMicroelectronics Group, Via Lavoratori Vittime del Col du Mont 24, 11100 Aosta, Italy, 2007

2.4. Resumen del capítulo

En la primera parte del presente capítulo se describió una breve reseña del desarrollo de los códigos correctores de errores hasta el descubrimiento de los turbo códigos. Con lo cual se inició un gran desarrollo en el diseño de los sistemas de codificación de información, ya que se obtuvieron mejores resultados que las técnicas existentes anteriormente. También se mencionaron algunas de las principales aplicaciones en la actualidad de este método de corrección de errores, ya que debido a su adaptabilidad en diferentes entornos, ha permitido su aplicación en un amplio número de aplicaciones.

También se explica el desarrollo de los dispositivos que utilizan la tecnología PLC, en los cuales, se han aplicado distintas técnicas y algoritmos de comunicaciones cada vez más complejos que han mejorado el rendimiento de los equipos de comunicación. Además, se describen algunos grupos que se han creado para impulsar y crear estándares para esta tecnología. Tal es el caso de Homeplug, un grupo que ha creado diferentes estándares y especificaciones que permiten la convivencia y el desarrollo de dispositivos creados por diferentes fabricantes y a la vez un mejor servicio a los clientes. Los diferentes grupos han tomado en cuenta la velocidad de transmisión así como características del canal de comunicación para realizar dichas especificaciones.

Con respecto a las perspectivas de la tecnología PLC, se puede mencionar ha tenido mayor impacto en Europa, Asia y Estados Unidos. Ya que muchas compañías pioneras que han impulsado su desarrollo, se ubican en estas regiones además del avance tecnológico que históricamente han tenido con respecto a otros lugares.

Es importante hacer mención de los estudios realizados con la codificación de la información en el canal PLC, ya que en el caso de los turbo códigos, las investigaciones e implementaciones realizadas han obtenido buenos resultados e incluso éstos han sido incluidos en estándares para este medio de comunicación, impulsando aun más su utilización e investigación para esta tecnología. Cabe resaltar que se han realizado modificaciones y variaciones de los parámetros de esta técnica de corrección de errores con el objetivo de mejorar el desempeño en condiciones propias del canal PLC. En el siguiente capítulo se presentarán los fundamentos teóricos del módulo corrector de errores propuesto.

Capítulo 3

Fundamentos teóricos del módulo corrector de errores propuesto

En el presente capítulo se describirán las características principales de los elementos que conforman un esquema de turbo código general. También se comentará acerca de los códigos convolucionales, así como los tipos principales de codificadores. Se mencionarán las características del hardware utilizado. En la ultima parte, se describirá el tipo de ruido en el canal PLC y el metódo Monte Carlo, el cual, será utilizado en este trabajo para realizar las simulaciones con el módulo corrector de errores implementado y obtener las curvas BER.

3.1. Introducción

Desde la aparición de los turbo códigos y los buenos resultados obtenidos en simulaciones realizadas en un canal de comunicaciones con ruido, gran parte de los problemas planteados se han enfocado en realizar implementaciones adecuadas al tipo de aplicación en el sistema de comunicaciones utilizado, así como la cantidad de recursos disponibles en el hardware utilizado.

Existe literatura especializada en el estudio de los turbo códigos, la cual describe cada uno de los componentes principales que conforman el codificador y decodificador de dicha técnica de corrección de errores. Sin embargo la mayoría de las implementaciones han sido patentadas y en su mayoría no se cuenta con código abierto de dichas implementaciones.

Entre los componentes más importantes de un turbo código se encuentran el intercalador y des-intercalador, los cuales pueden ser descritos como una permutación seudo-aleatoria de los símbolos de la información. Aunque para un número pequeño de

datos su complejidad de procesamiento es relativamente baja, al incrementar dicha cantidad también se incrementa la complejidad y la latencia del sistema, sin embargo el desempeño del código corrector es mejor.

Para realizar el estudio de un esquema de turbo codificación es importante tener en cuenta un conjunto de elementos de los codificadores convolucionales entre los cuales se encuentran: su representación, los tipos principales y construcción en paralelo. Además de estudiar los principales tipos de algoritmos de decodificación como son: MAP-BCJR y Max-log-MAP.

Con respecto al canal de comunicación PLC es importante tomar en cuenta el tipo de ruido generado, las características de variación en tiempo y frecuencia, tipo de modulación utilizada, frecuencia de transmisión. De acuerdo a la literatura y los estudios realizados, no es posible modelar el ruido en este canal como solo ruido gaussiano ya que son cuatro los tipos de ruido encontrados: ruido de banda angosta, ruido de fondo coloreado y ruido impulsivo sincrono/asíncrono con la fuente principal.

3.2. Códigos convolucionales

Una de las técnicas principales de corrección de errores son los códigos convolucionales. Esta técnica de corrección de errores consiste en recibir k palabras de entrada de código y construir n palabras de salida, las cuales no dependen sólo de la entrada actual sino también de los últimos K segmentos en el codificador donde K es conocida como la memoria del codificador. Entre más grande sea el valor de la memoria K más alta sera la complejidad de la decodificación y mayor la capacidad de corrección de errores [6].

La operación realizada en el bloque de datos de entrada puede ser vista como una operación de convolución o un filtrado. Los registros inicialmente se encuentran inicializados a cero y entonces la secuencia de datos de entrada se desplaza hacia la derecha en un instante de tiempo “i” y con base en el nuevo dato que ingresa se determina el nuevo estado del codificador y así sucesivamente hasta el fin del bloque de datos de entrada. Una vez finalizado el bloque de datos de entrada se ingresan K ceros para reinicializar los registros del codificador a su estado original. En la figura 3.1 se puede ver un codificador convolucional formado por un nivel de memoria 2 y 4 estados.

Los códigos convolucionales suelen ser preferidos en implementaciones prácticas en hardware con respecto a los códigos de bloque ya que se han obtenido mejores resultados y los algoritmos de decisión suave han sido más fácilmente diseñados para su decodificación [5]. Un código convolucional con parametros n, k y K puede ser denotado como $C_{conv} = (n, k, K)$ en donde “n” es el número se símbolos generados para una palabra de entrada

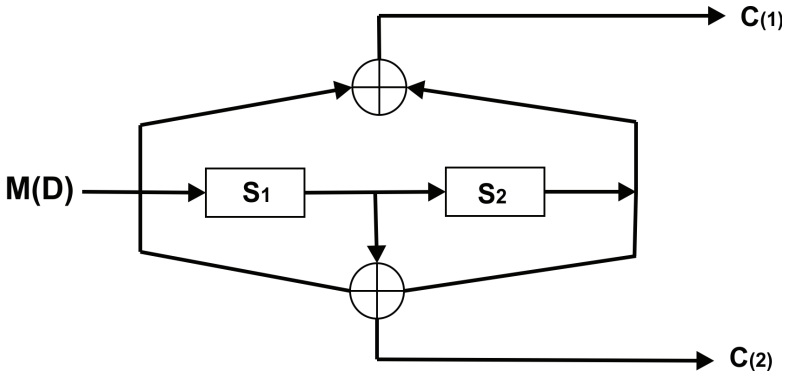

Figura 3.1. Codificador convolucional de 4 estados.

de "k" símbolos. Esta relación es conocida como tasa de datos de un código convolucional (k/n).

La estructura de un código convolucional puede ser vista como una máquina secuencial de estados finitos (FSSM), la cual es analizada con base en una matriz de función de transferencia polinomial $G(D) = \frac{P(D)}{Q(D)}$. Este análisis se lleva a cabo en el dominio D, conocido como dominio de retardo en el cual la entrada y salida adoptan expresiones polinomiales que pueden ser denotadas como $M(D)$ y $C(D)$ respectivamente. En el dominio D los exponentes y subíndices determinan la posición del elemento dentro de la secuencia de datos de entrada. Por ejemplo teniendo la secuencia de entrada: $m^l = (m_0^l, m_1^l, m_2^l, ...)$ será representada como: $M_{(D)}^l = m_0^l + m_1^l D + m_2^l D^2 ....$ Generalmente se suele utilizar la notación octal para representar los polinomios que forman un turbo codificador. Por ejemplo en el codificador convolucional recursivo sistemático de la figura 3.2 al tomar los puntos de conexión a_0, a_3 y a_4 de la parte superior se forma el polinomio $1 + D^3 + D^4$. Al agrupar los elementos de derecha a izquierda (D^4 menos significativo y 1 más significativo) se tiene que es equivalente a 10011 en representación binaria y en formato octal a 23. En forma similar al tomar las conexiones a_0, a_1, a_2 y a_4 de la parte inferior, se construye el polinomio $1 + D + D^2 + D^4$, tomando como valor menos significativo a más significativo el orden de derecha a izquierda, tenemos el valor 11101 en formato binario, lo cual equivale a 35 en formato octal.

Un codificador que sólo tiene entradas polinomiales en la función de transferencia se conoce como "codificador feedforward" o "codificador FIR". Un codificador que tiene funciones racionales en su matriz de función de transferencia es conocido como "codificador feedback" ó "codificador IIR".

Un codificador convolucional puede ser visto como una máquina secuencial de estados finitos ya que está compuesto por un conjunto de registros de memoria que almacenan un

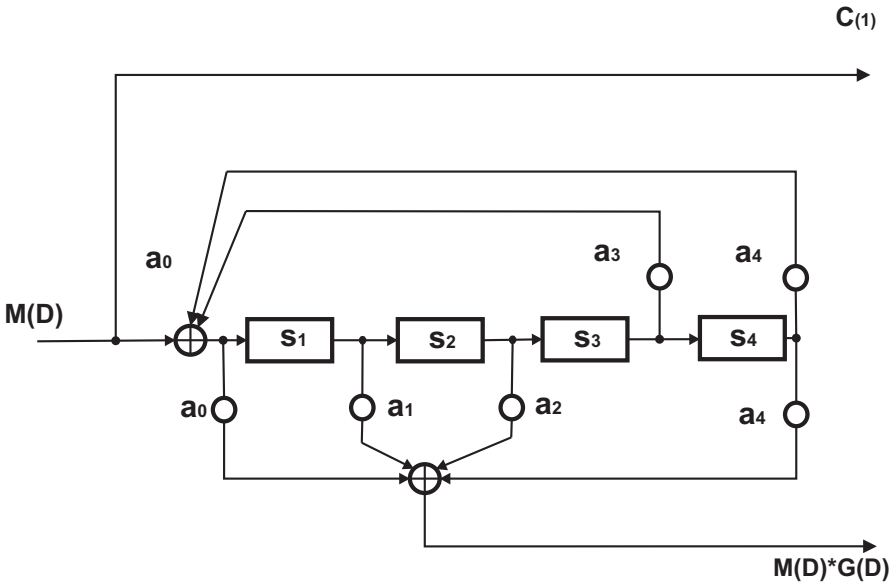

Figura 3.2. Codificador convolucional sistématico recursivo.

dato en un instante de tiempo en el cual ingresa un nuevo dato a la entrada del codificador. Una forma de representación gráfica de estos codificadores es un diagrama de estados, el cual es muy útil para observar que estados pueden seguirse a partir de un estado dado así como el número total de los estados del codificador.

En la figura 3.3 se muestra el diagrama de estados del código convolucional de la figura 3.1, en donde la dirección de las flechas indican el nuevo estado del codificador y sus índices muestran la salida en base al dato de entrada.

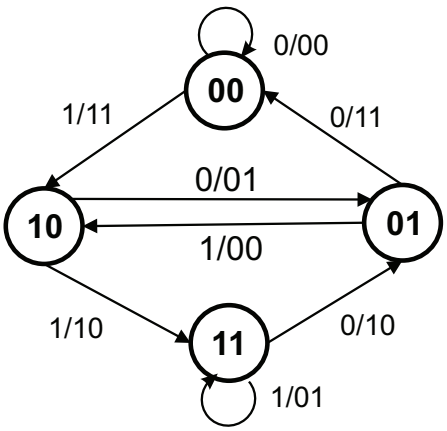

Figura 3.3. Diagrama de estados del codificador convolucional.

Una forma de representación conocida como diagrama de Trellis consiste en poner los puntos del conjunto de estados posibles que puede tomar el codificador convolucional y trazar las rutas que se generan en base al estado actual en los registros de memoria y al valor de la nueva entrada.

Esta forma de representación es muy útil sobre todo para el trazo de la ruta total que genera una trama de datos de entrada a traves del codificador. En la figura 3.4 se muestra la representación del diagrama de Trellis del codificador convolucional 3.1. Esta forma de representación además de ayudar a mostrar la codificación de los datos de entrada permite seguir los estados posibles en el proceso de decodificación de la información recibida.

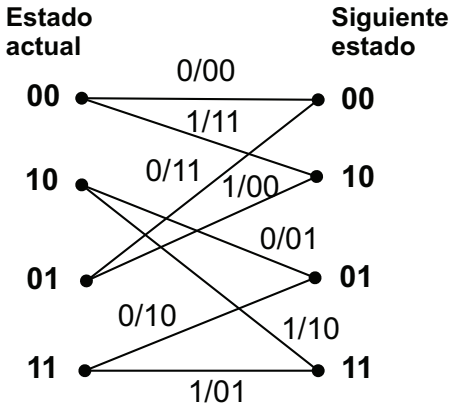

Figura 3.4. Diagrama de Trellis para un código convolucional.

Un codificador convolucional que se encuentra construido con una retroalimentación hacia atrás en el punto de suma es conocido como codificador convolucional recursivo (RCC).

3.2.1. Codificadores convolucionales no recursivos

En este tipo de codificadores convolucionales también conocidos como codificador "feedforward" la información original es afectada por una matriz generadora y el decodificador necesita tener dicha matriz para poder recuperar la información a partir de los datos enviados por el dispositivo transmisor. En la figura 3.5 se muestra la estructura de un codificador convolucional no recursivo.

La matriz generadora para un código convolucional tiene la forma general mostrada en la ecuación 3.1.

$$G(D) = \frac{C(D)}{M(D)} = a_0 + a_1 D + a_2 D^2 + ... + a_n D^n \tag{3.1}$$

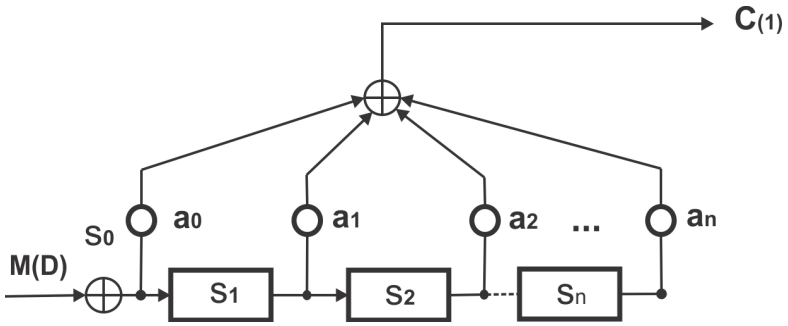

Figura 3.5. Estructura de un codificador convolucional no sistemático.

3.2.2. Codificadores convolucionales sistemáticos recursivos

En los codificadores sistemáticos recursivos la información puede ser obtenida en base a la palabra de código de salida ya que la información no es modificada por la matriz generadora y gracias a ello no es necesario tener una función de transferencia en el receptor para la recuperación de dicha información. En la figura 3.6 se muestra la estructura general de un codificador convolucional sistemático recursivo.

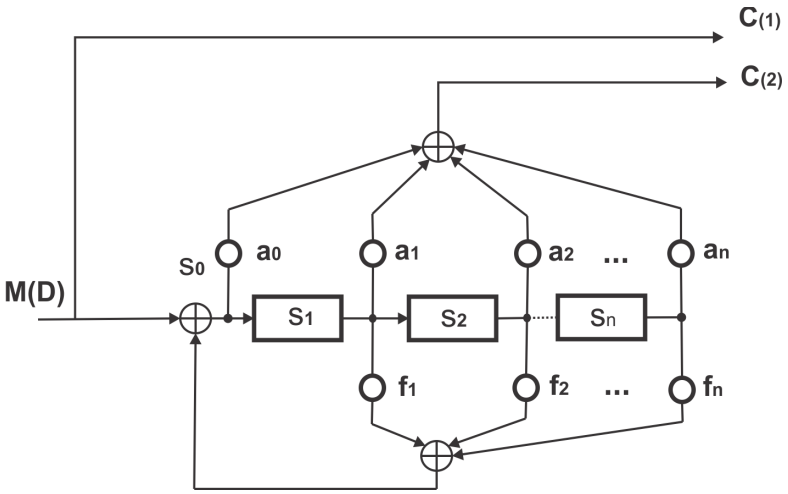

Figura 3.6. Codificador convolucional sistemático recursivo.

Un código sistemático recursivo tiene la matriz generadora de la forma que se muestra en expresión 3.2. La expresión 3.3 representa la forma simplificada de la matriz generadora de un código RSC.

$$\begin{bmatrix} 1 & 0 & ... & 0 & G_1^{k+1}(D) & G_1^{k+2}(D) & G_1^{n}(D) \\ 0 & 1 & ... & 0 & G_2^{k+1}(D) & G_2^{k+2}(D) & G_2^{n}(D) \\ . & . & & . & . & . & . \\ . & . & & . & . & . & . \\ . & . & & . & . & . & . \\ 0 & 0 & ... & 1 & G_k^{k+1}(D) & G_k^{k+2}(D) & G_k^{n}(D) \end{bmatrix} \tag{3.2}$$

$$G(D) = \frac{C(D)}{M(D)} = \begin{bmatrix} 1 & \frac{a_0+a_1D+a_2D^2+...+a_nD^n}{1+f_1D+f_2D^2+...+f_nD^n} \end{bmatrix} \tag{3.3}$$

3.3. Turbo codificador

Un turbo codificador está compuesto por dos codificadores convolucionales unidos en paralelo con un intercalador y una etapa de perforación a la salida. Los codificadores convolucionales utilizados suelen ser codificadores convolucionales recursivos sistemáticos y a la vez idénticos entre ellos.

El bloque intercalador consiste en la permutación de los datos de entrada en un orden seudoaleatorio con el objetivo de aleatorizar la secuencia de datos y que la secuencia de entrada en el segundo codificador sea muy diferente a la secuencia de datos original.

El bloque "puncturing" o mecanismo de perforación, consiste en eliminar algunos bits de paridad a la salida de cada codificador alternadamente con el objetivo de reducir la tasa de código, es decir disminuir el número de símbolos de salida generados con respecto a los datos de entrada, aunque a veces esto reduzca el desempeño BER en el código. Debido a esta estructura en el codificador la secuencia de salida estará compuesta por los datos de la información original y los bits de paridad generados por los dos codificadores convolucionales recursivos sistemáticos.

La unión de dos codificadores convolucionales sistemáticos recursivos en paralelo en conjunción con un intercalador produce un código con pocas palabras de bajo peso. Este código no necesariamente tendrá una distancia libre grande. En la figura 3.7 se muestra un diagrama de bloques de la estructura general de un codificador turbo.

El intercalador y los codificadores RSC que componen el codificador turbo son elementos esenciales en el desempeño del código. Con el incremento en la longitud del intercalador se han obtenido mejores resultados en el desempeño BER. Aunque este incremento en la longitud no incrementa mucho la complejidad de la codificación, si

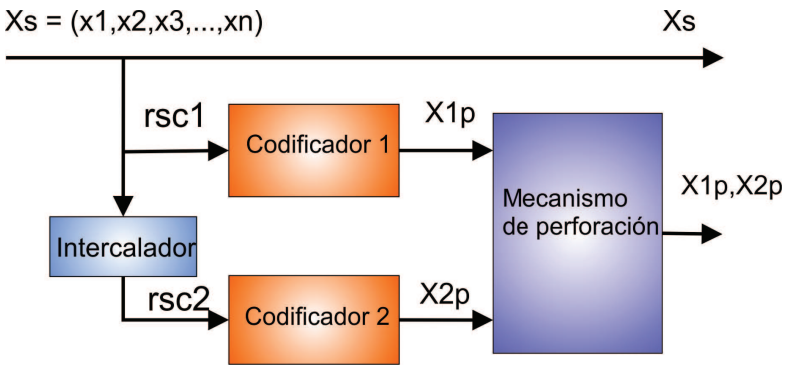

Figura 3.7. Estructura general de un turbo codificador.

incrementa la latencia, debido a ello resulta fundamental en el tipo de aplicación utilizada. La estructura de los codificadores RSC puede parecer relativamente simple pero su unión en paralelo consigue buenos resultados utilizando algoritmos iterativos de decodificación [6].

3.3.1. Intercalador

El intercalador o permutador consiste en tomar un bloque de "N" datos de entrada y reordenarlos en un orden seudoaleatorio, el cual debe ser conocido en el codificador y el decodificador. El objetivo del intercalador es descorrelacionar las entradas a los codificadores constituyentes y también que los bits de chequeo de paridad sean independientes. Esto mejora la probabilidad de corregir un patrón de error por alguno de los decodificadores y reduce la cantidad de errores en ráfaga generados en la transmisión de la información.

De acuerdo a la literatura [34] se puede definir al intercalador como: dado el conjunto $I = (1, ..., i, ..., N)$ e $I_B = (b(1), ..., b(i), ..., b(N))$, donde I es un conjunto de enteros positivos, e I_B es el conjunto de índices de las posiciones de los elementos de I en algún orden intercalado. El intercalador es la función de mapeo:

$$B(I_B \rightarrow I) : j = b(i), i, j \in I \tag{3.4}$$

El proceso inverso al intercalador de símbolos se conoce como des-intercalador y en forma similar a la expresión anterior se puede definir como dado el conjunto $I = (1, ..., i, ..., N)$ e $I_{B^{-1}} = \left(b^{-1}(1), ..., b^{-1}(i), ..., b^{-1}(N)\right)$, donde I es un conjunto de enteros positivos, e I_B es el conjunto de índices de las posiciones de los elementos de I en orden

des-intercalado. El des-intercalador está dado por la función:

$$B^{-1}\left(I_{B^{-1}} \rightarrow I\right) : i = b^{-1}\left(j\right), i, j \in I. \tag{3.5}$$

La secuencia de N datos de entrada debe ser por lo regular más grande que 1000, debido al incremento en el desempeño del turbo código [35]. Cuando un intercalador es utilizado contra errores en ráfaga, éste es diseñado para convertir los patrones de error que contienen datos continuos seriales con errores en patrones más aleatorios. Esta distribución es realizada entre muchos vectores de código. Los errores en ráfaga son característicos de algunos canales, por ejemplo el canal inalámbrico. También podemos mencionar que ocurren en códigos concatenados, donde el propio decodificador puede sobrecargarse con errores, es decir, puede pasar una ráfaga de errores a la salida del decodificador.

Existen diferentes tipos de intercaladores entre los cuales se encuentran:

- Intercaladores de bloque.
- Intercaladores convolucionales.
- Intercaladores aleatorios.
- Intercaladores lineales.
- Intercaladores S-random.

Un intercalador de tipo bloque consiste en un arreglo en forma matricial de tamaño $M_1 \times N_1$, en el cual los datos de entrada son escritos en cada una de las filas de dicho arreglo y después éstos son leídos en formato columna. Todos los elementos de la matriz deben ser llenados con un dato y la operación de permutar los datos en esta forma generará un retardo de $M_1 \times N_1$ intervalos. Este tipo de intercalador realizará la separación de los datos en un patrón menor a M_1 pero al menos de N_1 símbolos [6]. Un intercalador de tipo bloque debe tener un número de filas M_1 más grande que la longitud de los errores en ráfaga experimentados. El número de columnas N_1 generalmente es seleccionado equivalente o mayor a la longitud del bloque de datos decodificado. De esta forma una ráfaga de N_1 errores podría producir solamente un error por un vector de código. Se ha observado que cuando $M_1 = N_1$ siendo ambos valores impares el desempeño del intercalador es mejor. En la figura 3.8 se muestra la forma general de un intercalador tipo bloque.

Un intercalador convolucional se compone de N registros multiplexados de forma que cada registro consecutivo contiene L símbolos más que el registro anterior. El primer registro no tiene ningún retardo y los datos son instantáneamente multiplexados, a

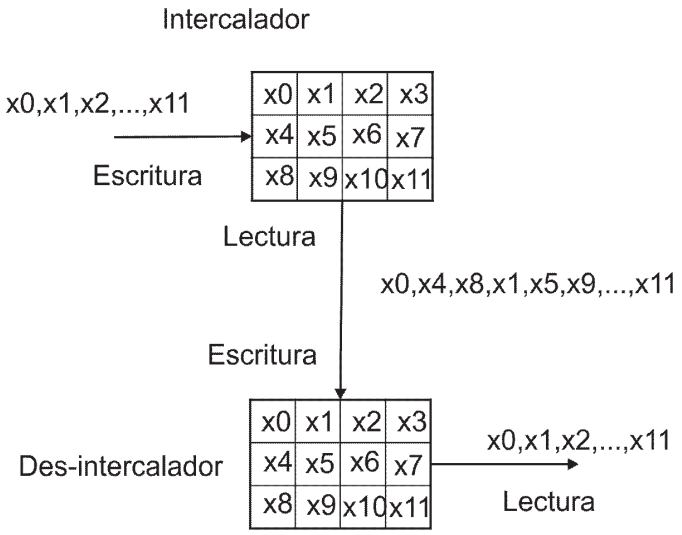

Figura 3.8. Estructura general de un intercalador de tipo bloque.

diferencia de los otros registros, los cuales son multiplexados en cada instante de tiempo por lo cual este tipo de intercalador también es conocido como intercalador multiplexado.

El intercalador aleatorio se construye en forma similar a un intercalador tipo bloque con la diferencia de que las posiciones se generan en forma seudo aleatoria. El requerimiento de memoria de este tipo de intercalador es de $M_1 \times N_1$ símbolos debido a que se requieren dos intercaladores, uno para escribir y otro para leer los datos el requerimiento de memoria será de $2M_1 \times N_1$ símbolos.

En el esquema de un turbo código, el intercalador tiene un papel importante ya que de acuerdo a la literatura se ha obtenido mejor desempeño en el BER cuando la longitud del intercalador es incrementada. Además el intercalador de tipo bloque se comporta mejor que un intercalador seudo-aleatorio para longitudes de bloques de datos pequeñas y en longitudes grandes se da el caso contrario [6].

En este trabajo se utilizó un intercalador de tipo S-Random, el cual permuta un conjunto de elementos en localizaciones que se encuentran apartadas al menos S posiciones entre los elementos contiguos. El intercalador es construido de acuerdo al algoritmo 1.

Algoritmo 1 Algoritmo para contrucción de intercalador tipo S-Random

1: Seleccione aleatoriamente el primer elemento de la permutación $\pi_1 \in \{1, 2, ..., n\}$.
2: Para cada elemento posterior de la permutación, $\pi_i, i = 2, 3, ..., n$ se selecciona aleatoriamente un número desde el conjunto $\{1, 2, ..., n\}$ y lo comparamos con la selección previa S $\pi_1, \pi_2, ..., \pi_S$
3: Si los valores absolutos de las distancias entre la selección entre la selección actual y todas las selecciones previas S son más grandes que S (el criterio S-random) la selección es aceptada. En otro caso, volvemos a seleccionar otro número hasta que el criterio S-Random se cumpla.
4: El proceso termina cuando cada elemento de la permutación satisface el criterio S-random.

3.3.2. Mecanismo de perforación (Puncturing)

El mecanismo "puncturing" o de perforación consiste en eliminar alternadamente en cada instante de tiempo los bits de paridad de los codificadores RSC.

Su objetivo principal es incrementar la tasa de código, en otras palabras disminuir la longitud de la trama de datos enviada. Aunque esto puede disminuir el desempeño del código corrector, permite disminuir la latencia. Generalmente este mecanismo es representado con la siguiente matriz en la expresión 3.6:

$$P = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{3.6}$$

Por ejemplo consideremos un codificador turbo de la figura 3.9 con la matriz generadora de la expresión 3.7:

$$P = \begin{bmatrix} 1 & \frac{1}{1+D^2} \end{bmatrix} \tag{3.7}$$

El intercalador consiste en la secuencia mostrada en la expresión 3.8:

$$\Pi = \{8, 3, 7, 6, 9, 0, 2, 5, 1, 4\} \tag{3.8}$$

Dada la secuencia de entrada: $X_s = [1, 1, 0, 0, 1, 0, 1, 0, 1, 1] = v^{(0)}$.

La salida del primer codificador será: $X_{1p} = [1, 1, 1, 1, 0, 1, 1, 1, 0, 0]$.

La secuencia intercalada será: $X'_s = [1, 0, 0, 1, 1, 1, 0, 0, 1, 1]$

La salida del segundo codificador es: $X_{2p} = [1, 0, 1, 1, 0, 0, 0, 0, 1, 1]$.

Al multiplexar las tres tramas de datos juntas obtendremos la siguiente secuencia de salida: $v = [1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1]$.

Si se aplica la perforación a los bits de paridad para tener una tasa de código de $1/2$ obtenemos la siguiente secuencia: $v = [1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1]$.

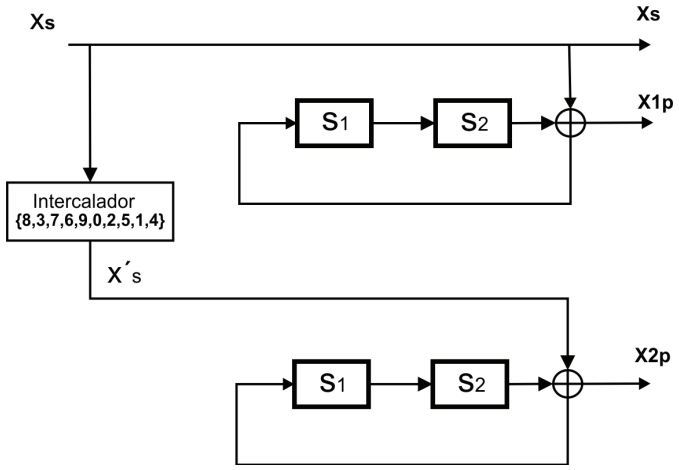

Figura 3.9. Codificador turbo con intercalador aleatorio.

En la figura 3.10 se muestra la perforación de los bits de redundancia realizada a una trama codificada.

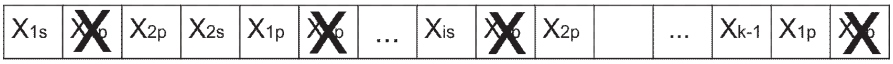

Figura 3.10. Mecanismo de perforación (puncturing) en una trama codificada para obtener una tasa de 1/2.

3.3.3. Análisis del turbo codificador

Uno de los principales objetivos al construir un turbo código es el poder utilizar los mejores codificadores componentes para poder máximizar la distancia libre del código generado. Para valores grandes de la relación señal a ruido (E_b/N_0) es importante poder máximizar la separación entre palabras de código, sin embargo para relaciones bajas es más importante optimizar la distribución de peso de las palabras de código [35]. De acuerdo a la descripción de cada uno de los componentes del turbo codificador realizada en las secciones anteriores, se puede observar que estos componentes son lineales por lo cual la estructura del codificador es lineal. Es decir se puede tomar la secuencia de código de salida a la secuencia todos ceros y a la vez se puede asumir que la secuencia todos ceros puede ser transmitida. Incluso si se aplica el mecanismo de perforación o "puncturing" a los bits de paridad de la salida de los codificadores RSC no se pierde linealidad ya que las posiciones perforadas son siempre las mismas. De acuerdo a [35] considerando la palabra de código y una k^{th} palabra de código para alguna $k \in 1, 2, ..., 2^N - 1$. El decodificador podrá elegir la k^{th} palabra de código con una probabilidad de: $Q\left(\sqrt{2d_k r E_b/N_0}\right)$, donde r es la tasa de

código y d_k es el peso de la k^{th} palabra de código. Tomando en cuenta estas dos palabras de código la tasa de errores está dada por la expresión 3.9.

$$P_b(k|0) = w_k \cdot \frac{1}{N} \cdot Q\left(\sqrt{\frac{2d_k r E_b}{N_0}}\right) =$$
$$\frac{w_k}{N} Q\left(\sqrt{\frac{2rd_k E_b}{N_0}}\right) \quad \left(\frac{bits \;\; con \;\; errores}{bits \;\; de \;\; datos}\right) \tag{3.9}$$

Donde w_k es el peso de la k^{th} palabra de código.

Incluyendo todas las $2^N - 1$ posibles palabras de código tenemos:

$$P_b = P_b(k \in 1, 2, ..., 2^N - 1|0)$$
$$\textstyle\sum_{k=1}^{2^N-1} P_b(k|0)$$

$$= \sum_{k=1}^{2^N-1} \frac{w_k}{N} Q\left(\sqrt{\frac{2rd_{wv}E_b}{N_0}}\right) \tag{3.10}$$

Reorganizando los términos se tiene:

$$P_b \leq \sum_{w=1}^{N} \sum_{v=1}^{\binom{N}{w}} \frac{w}{N} Q\left(\sqrt{\frac{2rd_{wv}E_b}{N_0}}\right) \tag{3.11}$$

Una palabra de código está formada por los bits de información más los bits de paridad generados por los codificadores RSC. Cuando una secuencia de datos de peso unitario ingresa al primer codificador, dada su característica de recursividad, éste generará una palabra de código de mucho mayor peso, a diferencia de un codificador convolucional no recursivo, el cual podría generar una palabra de bajo peso incluso a pesar de tener un intercalador a la entrada.

Una de las propiedades de un código RSC es que presenta una respuesta impulsiva infinita, por lo cual para que la secuencia a la salida del codificador posea un peso finito es necesario terminar en el estado inicial. Para poder forzar a la secuencia de código a terminar con un peso fijo es necesario regresar al codificador al estado cero [36].

Cuando se tiene una secuencia de datos de entrada, la cual produce una secuencia de bajo peso a la salida del codificador, el intercalador alterará la secuencia de entrada produciendo un patrón de datos diferente haciendo poco probable que una secuencia de salida de peso mínimo sea combinada con otra de peso similar. Una de las características

en los codificadores RSC es que protegen la generación de palabras de bajo peso, lo cual no puede ser evitado por el intercalador dentro del esquema del codificador turbo.

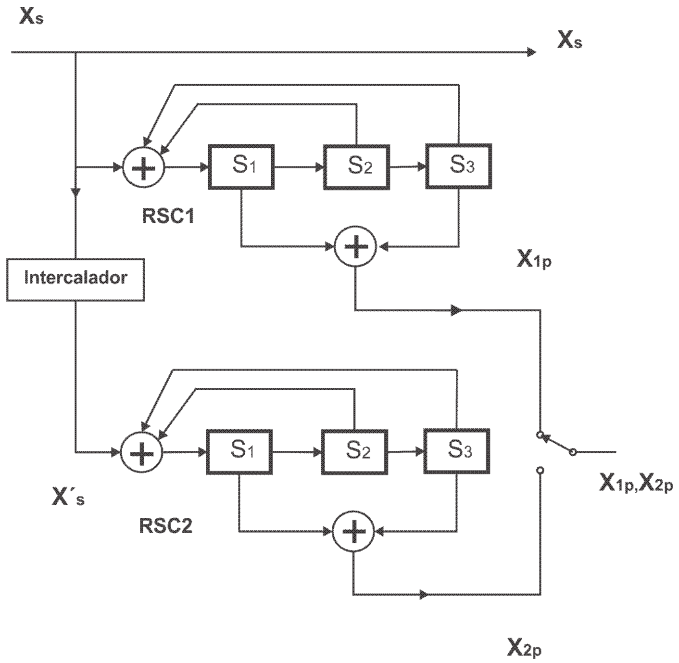

Figura 3.11. Estructura de un turbo codificador RSC(13,15).

3.4. Turbo decodificador

La estructura general del turbo decodificador se encuentra formada por dos decodificadores componentes MAP (Máxima probabilidad a posteriori) en serie los cuales se encuentran separados por una etapa de des-intercalado e intercalado de la información entre ellos. Teniendo a su vez una retroalimentación de la información de salida del segundo decodificador hacia el primero (razón por la que fueron llamados turbo códigos), lo cual produce una ganancia en la decodificación al tener un cierto número de iteraciones.

El algoritmo utilizado para la decodificación iterativa es el algoritmo modificado BCJR de máxima probabilidad a posteriori (MAP) en honor a sus creadores: Bahl, Cocke, Jelinek y Raviv o algoritmo “forward-backward” publicado en el año 1974. Aunque en ese entonces la importancia del algoritmo no fue muy resaltada debido a que presentaba el mismo grado de complejidad que otros algoritmos de decodificación iterativa, fue hasta 1993 cuando fue utilizado por Berrou, Glavieux y Thithimajshima en 1996 para la decodificación de los turbo códigos y se mostraron sus verdaderos alcances [14]. En la figura 3.12 se muestra la

estructura general de un turbo decodificador utilizando el algoritmo BCJR-MAP.

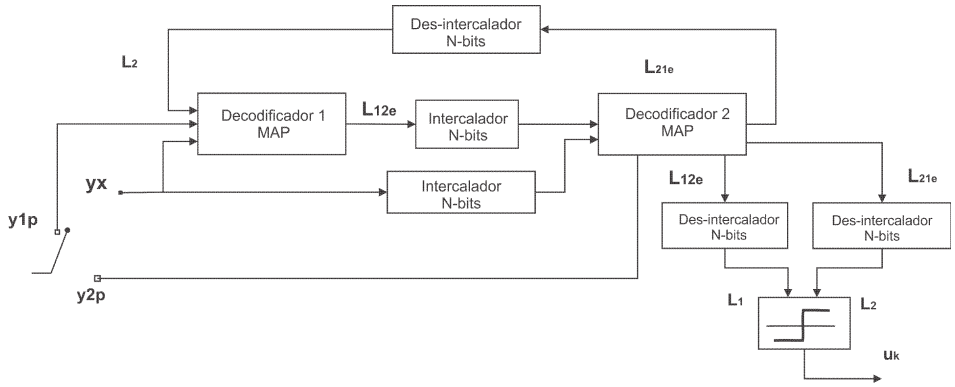

Figura 3.12. Turbo decodificador BCJR-MAP

3.5. Teorema de Bayes

El teorema de Bayes enunciado por Thomas Bayes en 1763 [37] expresa la probabilidad condicional de un evento aleatorio A dado B en términos de la distribución de probabilidad condicional del evento B dado A y la distribución de probabilidad marginal de sólo A.

TEOREMA **1 (Teorema de Bayes)** *Sea* $A_1, A_2, A_3, ..., A_i, ..., A_n$ *un conjunto de sucesos mutuamente excluyentes y exhaustivos, y tales que la probabilidad de cada uno de ellos es distinta de cero. Sea* B *un suceso cualquiera del que se conocen las probabilidades condicionales* $P(B|A_i)$*. Entonces, la probabilidad* $P(A_i|B)$ *viene dada por la expresión:*

$$P(A_i|B) = \frac{P(B|A_i) * P(A_i)}{P(B)} \tag{3.12}$$

Por ejemplo supongamos que realizamos n experimentos, en los cuales tenemos dos caras de una moneda: A y B y conocemos ciertos valores producidos en los experimentos.

$$BYBBBYBYYBBB...$$
$$AAAXAXXAAAXX...$$
$$.....n$$

De acuerdo a la literatura la probabilidad de un evento se define como:

$$Probabilidad\ \ del\ \ evento\ \ E = \frac{Casos\ \ favorables\ \ del\ \ evento\ \ E}{Casos\ \ posibles\ \ del\ \ evento\ \ E} \tag{3.13}$$

Tomando en consideración lo anterior, se tiene que la probabilidad de que caiga la cara A será igual al número de veces que ocurrió A entre el número total de experimentos realizados n.

$$P(A) = \frac{\#A}{n} \tag{3.14}$$

En forma similar la probabilidad de que ocurra B se presenta en la siguiente expresión:

$$P(B) = \frac{\#B}{n} \tag{3.15}$$

Tomando en consideración lo anterior se tiene que la probabilidad de que ocurra B dado que ocurrió A será igual a la relación del número de veces que ocurrió A dado que ocurrió B entre la probabilidad de que ocurra A.

$$P(B/A) = \frac{\#(A,B)/n}{\#A/n} = \frac{P(A,B)}{P(A)} \tag{3.16}$$

$$P(B/A) = \frac{P(A,B)/P(A)}{\#A/n} = \frac{P(A,B)}{\sum_{i=1}^{n}(B_i,A)} \tag{3.17}$$

3.6. El logaritmo de la relación de probabilidad

La relación de probabilidad logarítmica (LLR) para un bit b_i es denotada como $L(b_i)$, y definida como el logaritmo natural del cociente entre el número de probabilidades de que el bit sea 1 ó que sea 0. Dado que éste es un número con signo, puede ser utilizado como representativo del símbolo, el cual está siendo estimado y a la vez es muy conveniente para definir el cociente de probabilidad de que un bit sea un +1 o un -1 cuando se utiliza el formato polar. Es similar a saber cuando una decisión es tomada sobre una señal transmitida con un alfabeto ±1 en lugar del alfabeto 0,1. Esta estimación se define como:

$$L(b_i = \ln \frac{P(b_i = +1)}{P(b_i = -1)} \tag{3.18}$$

3.6.1. Algoritmo de máxima probabilidad a posteriori (BCJR-MAP)

El algoritmo de máxima probabilidad a posteriori fue el algoritmo utilizado por Berrou, Glavieux, y Thithimajshima para la decodificación, debido a su proceso iterativo y a la estimación compartida de dos decodificadores lo que origina muy buenos resultados [14].

La etapa de decodificación de un turbo código se realiza utilizando el algoritmo MAP, el cual, miminiza la probabilidad de error usando la secuencia recibida de entrada para identificar el bit más probable en cada estado del diagrama de Trellis. Por ejemplo se podrán utilizar las siguientes representaciones para los pares de símbolos transmitidos/recibidos.

$$y_k = x_k, p_k \quad y = y_{1K} = y_1, y_2, ..., y_K; \quad y'_k = x'_k, p'_k \quad y' = y'_{1K} = y'_1, y'_2, ..., y'_K \tag{3.19}$$

La secuencia completa de símbolos puede dividirse en tres partes: pasado, presente y futuro:

$$y = \underbrace{y_1, y_2, ..., y_{k-1}}_{y_{<k}} \quad y_k \quad \underbrace{y_{k+1}, ..., y_N}_{y_{>k}} = y_{<k} \quad y_k \quad y_{>k} \tag{3.20}$$

Las decisiones suaves MAP están definidas como el logaritmo de relación de verosimilitud:

$$L_{map}(x_k) = \log \left[\frac{P_r(x_k = +1)|y'}{P_r(x_k = -1|y')} \right] \tag{3.21}$$

De acuerdo al algoritmo MAP para un código convolucional se puede expresar la relación de máxima probabilidad en términos de Trellis:

$$\frac{P_r(x_k = +1|y')}{P_r(x_k = -1|y')} = \frac{\sum_{(s',s)\in S^+} P_r(S_{k-1} = s', S_k = s|y')}{\sum_{(s',s)\in S^-} P_r(S_{k-1} = s', S_k = s|y')} = \frac{\sum_{(s',s)\in S^+} P_r(S_{k-1} = s', S_k = s, y')}{\sum_{(s',s)\in S^-} P_r(S_{k-1} = s', S_k = s, y')} \tag{3.22}$$

Donde el numerador de la suma son todas las transiciones posibles de estado asociadas con un bit de valor '1', y en el denominador son todas la posibles transiciones asociadas con un bit de valor '0'. El valor $P_r(S_{k-1} = s', S_k = s, y')$ representa la probabilidad de recibir una secuencia de n bits y', siendo s' el estado anterior en el instante $k-1$ y s el estado actual en el instante k. Este término puede ser calculado como el producto de tres probabilidades:

$$P_r(S_{k-1} = s', S_k = s, y') = \alpha_{k-1}(s')\gamma_k(s', s)\beta_k(s) \tag{3.23}$$

Donde:

$$\alpha_{k-1}(s') = P(s', y_{<k})$$
$$\gamma_k(s', s) = P(y_k, s|s')$$
$$\beta_k(s) = P(y_{>k}, |s)$$

Tomando como referencia el instante k, las probabilidades α, β y γ se asocian con el pasado, el presente, y el futuro de la secuencia de entrada y, respectivamente.

Se puede definir a $\alpha_k(s', s) = P(y_k, s|s')$ como la probabilidad condicional de que el símbolo recibido es y_k en el instante k y el estado actual es $S_K = S$ conociendo con anterioridad que el estado anterior fue $S_{k-1} = s'$ por lo cual de acuerdo al teorema de Bayes se tiene:

$$\alpha_k(s', s) = P(y_k|x_k)P(u_k) \tag{3.24}$$

Para el caso de un canal con ruido gaussiano AWGN la expresión anterior se puede calcular como:

$$\gamma_k(s', s) = C_k e^{u_k L(u_k)/2} exp(\frac{L_c}{2} \sum_{n}^{l=1} x_{kl} \cdot y_{kl}) \tag{3.25}$$

Donde el término $L_c = 4 \cdot a \cdot R_c \frac{E_b}{N_0}$. $\frac{N_0}{2}$ es la densidad de potencia espectral del ruido, a es la amplitud del canal (para canales sin atenuación $a = 1$), E_c y E_b son la energía transmitida por bit codificado y bit de mensaje, respectivamente y R_c es la tasa de código.

- La información a priori. También llamada información intrínseca. Representa a la información disponible de los bits recibidos antes de que ocurra una decodificación. En el proceso iterativo del algoritmo de decodificación ésta es generada en la segunda iteración por el segundo decodificador.

- La información sistemática. Es la información explicita disponible de la información recibida. En otras palabras es la probabilidad a posteriori.

- La información extrínseca. Es la información generada por el decodificador, basado en la secuencia recibida y en la información a priori disponible.

De acuerdo al algoritmo las probabilidades α y β pueden calcularse:

$$\alpha_k(s) = \sum_{s'} \alpha_{k-1}(s')\gamma_k(s', s) \tag{3.26}$$

$$\beta_{k-1}(s') = \sum_{s} \beta_k(s)\alpha_k(s', s) \tag{3.27}$$

3.6.2. Algoritmo Max-log-MAP

Existen varias limitaciones para la implementación del algoritmo de Máxima probabilidad a posteriori (BCJR MAP) en hardware entre las cuales se encuentran:

- El algoritmo requiere una estimación precisa de la varianza del ruido y su desempeño es muy sensitivo a la relación señal a ruido SNR.

- El algoritmo trabaja con números reales por lo cual la representación de las variables requiere de al menos entre 16 y 24 bits en la representación en punto fijo.

- El gran número de multiplicaciones y operaciones exponenciales demanda gran cantidad de recursos y a la vez incrementa la latencia en el sistema.

Debido a estos problemas se han realizado simplificaciones al algoritmo con el objetivo de poder tener resultados lo más cercanos posibles con menor complejidad en el tipo de operaciones y a la vez una menor latencia en el sistema. El algoritmo Max-log-MAP [5, 17] fue resultado de aplicar propiedades y simplificaciones a las operaciones del algoritmo MAP trabajando para ello en el dominio logarítmico.

Definiendo:

$$A_t\left(p\right) = ln\left(\alpha_t\left(p\right)\right) \tag{3.28}$$

$$B_t\left(q\right) = ln\left(\beta_t\left(q\right)\right) \tag{3.29}$$

$$\Gamma_t\left(p,q\right) = ln\left(\gamma_t\left(p,q\right)\right) \tag{3.30}$$

Tomando en cuenta el logaritmo de una suma de números se tiene la siguiente propiedad [5]:

$$\ln\left(A+B+C+...\right) \approx \ln(max(A,B,C...)) \tag{3.31}$$

$$\ln\left(\sum_i e^{x_i}\right) \approx max(x_i) \tag{3.32}$$

Utilizando la propiedad anterior para aproximar el logaritmo de una suma de números se tiene:

$$A_{t+1}(q) = \ln\left(A_{t+1}(q)\right) = \ln\left(\sum_{p=0}^{Q-1} \alpha_t(p)\gamma_t(p,q)\right) = \ln\left(\sum_{p=0}^{Q-1} exp\left[A_t(p) + \Gamma_t(p,q)\right]\right) \tag{3.33}$$

$$A_{t+1}(q) \approx max_{p\in[0,1,\ldots,Q-1]}\left(A_t(p) + \Gamma_t(p,q)\right) \tag{3.34}$$

$$B_t(p) \approx max_{p\in[0,1,\ldots,Q-1]}\left(\Gamma_t(p,q) + B_{t+1}(p)\right) \tag{3.35}$$

$$\Gamma_t(p,q) = \tilde{x}^{(p,q)}\frac{\lambda_{(p,t)}}{2} + \frac{L_c}{2}\tilde{x}^{(p,q)}r_t^{(0)} + \frac{L_c}{2}\tilde{v}^{(1,p,q)}r_t^{(1)} \tag{3.36}$$

$$\lambda(x_t|r) = \log\frac{\sum_{(p,q)\in S_1}\alpha_t(p)\,\gamma_t(p,q)\,\beta_{t+1}(q)}{\sum_{(p,q)\in S_0}\alpha_t(p)\,\gamma_t(p,q)\,\beta_{t+1}(q)} \tag{3.37}$$

$$\lambda(x_t|r) = \log\frac{\sum_{(p,q)\in S_1}A_t(q) + \Gamma_t(p,q) + \beta_{t+1}(q)}{\sum_{(p,q)\in S_0}A_t(q) + \Gamma_t(p,q) + \beta_{t+1}(q)} \tag{3.38}$$

$$\lambda(x_t|r) \approx max_{(p,q)\in S_1}\left(A_t(q) + \Gamma_t(p,q) + \beta_{t+1}(q)\right) - max_{(p,q)\in S_0}\left(A_t(q) + \Gamma_t(p,q) + \beta_{t+1}(q)\right) \tag{3.39}$$

$$\tilde{\lambda}_1(d_k) \approx max_{alls,s'}\left(\tilde{\gamma}_1\left(R_k,s',s\right) + \tilde{\alpha}_{k-1}\left(s'\right) + \tilde{\beta}_k(s)\right) - max_{alls,s'}\left(\tilde{\gamma}_0\left(R_k,s',s\right) + \tilde{\alpha}_{k-1}\left(s'\right) + \tilde{\beta}_k(s)\right) \tag{3.40}$$

Los pasos del algoritmo Max-log-MAP se muestran en el algoritmo 2.

Algoritmo 2 Max-log-MAP

1: Inicialización $A_t(p) = \begin{cases} 0 & is \quad s=0 \\ \log\left(\frac{1}{M}\right) & in \quad other \quad case \end{cases}$

$B_t(q) = \log\left(\frac{1}{M}\right)$ *para todo s.*
$L_{21e} = 0$ *para todo s.*
$L_{12e} = 0$ *para todo s.*
$L_c = 1$

2: **for** $C = 1 \rightarrow Iteraciones$ **do**

3: Calcular:
$\Gamma_t(p,q) = \tilde{x}^{(p,q)} \frac{\lambda_{(p,t)}}{2} + \frac{L_c}{2} \tilde{x}^{(p,q)} r_t^{(0)} + \frac{L_c}{2} \tilde{v}^{(1,p,q)} r_t^{(1)}$
$A_{t+1}(q) = max_{p \in [0,1,\ldots,Q-1]} (A_t(p) + \Gamma_t(p,q))$
$B_t(p) = max_{p \in [0,1,\ldots,Q-1]} (\Gamma_t(p,q) + B_{t+1}(p))$

4: **for** $k = 1 \rightarrow N$ **do**

5: Calcular:

$$\lambda(x_t|r) = max_{(p,q) \in S_1} (A_t(q) + \Gamma_t(p,q) + \beta_{t+1}(q)) - max_{(p,q) \in S_0} (A_t(q) + \Gamma_t(p,q) + \beta_{t+1}(q))$$

6: **end for**
7: **for** $k = 1 \rightarrow N$ **do**
8: $L_{12e}[k] = \lambda[k]$
9: **end for**
10: $Intercalador(L_{12e})$
$Intercalador(mesginput)$
11: **if** $C = 1$ **then**
12: **for** $k = 1 \rightarrow M$ **do**
13: $\beta_t(q) = max_{p \in [0,1,\ldots,Q-1]} (A_t(p) + \Gamma_t(p,q))$
14: **end for**
15: **end if**
Repetir los pasos del (3) al (9) para el decodificador 2.
16: $Des - intercalador(mesginput)$
$Des - intercalador(L_{21e})$
$Des - intercalador(L_{12e})$
17: **for** $k = 1 \rightarrow N$ **do**
18: **if** $(L_c \times mesginput[k] + L_{12e}[k] + L_{21e}[k]) > 0$ **then**
19: $mesgout[k] = 1$
20: **else**
21: $mesgout[k] = 0$
22: **end if**
23: **end for**
24: **end for**

3.7. Módulo TMS320C6416T DSK

En el presente trabajo de investigación se utilizó el módulo TMS320C6416T DSK ya que cuenta con el procesador digital de señales TMS320C6416T de la serie TMS320C64x, el cual, posee una arquitectura capaz de realizar operaciones de multiplicación así como sumas y restas con registros de 32 bits, las cuales, son las operaciones más utilizadas durante el procesamiento de los algoritmos utilizados. También se puede mencionar que de acuerdo a [38], esta familia de procesadores tiene un alto desempeño en aplicaciones tales como: estaciones base inalámbricas, módems multi-linea, sistemas de radares, aplicaciones de audio y video, entre otras.

El módulo C6416T DSK se muestra en la figura 3.13, el cual, es una plataforma de desarrollo que permite a los usuarios evaluar y desarrollar aplicaciones para la familia TI C64xx. La DSK también sirve como un hardware de referencia de diseño para el DSP TMS320C6416T [39].

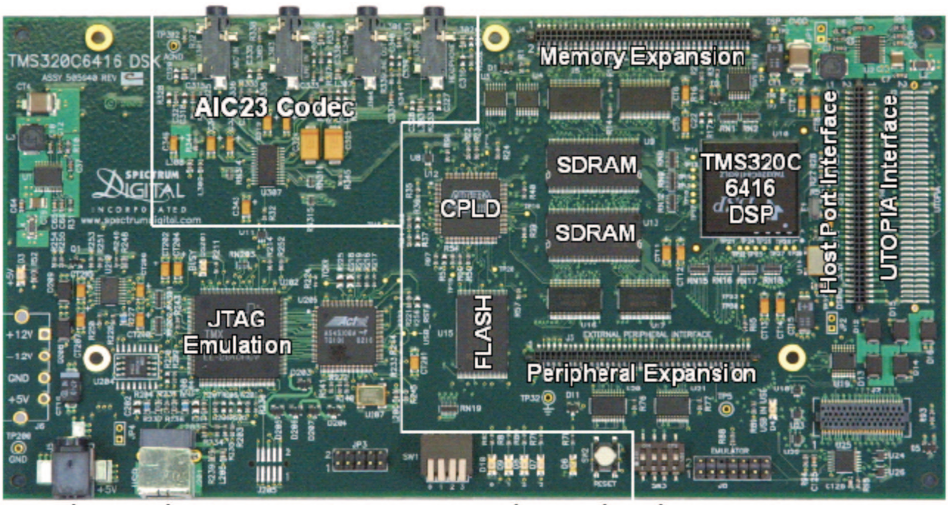

Figura 3.13. Módulo TMS320C6416T DSK.

El DSP se conecta a periféricos externos a través de dos buses, el EMIFA de un ancho de 64 bits y el EMIFB de un ancho de 8 bits. La SDRAM, la flash y el CPLD son conectados a uno de los buses. EMIFA es también conectada a una tarjeta hija de expansión la cual se usa cuando se conecta una tercera tarjeta. Un dispositivo lógico programable (CPPLD) es utilizado para implementar la lógica de pegamento que une los componentes de la tarjeta. El CPLD también tiene un registro basado en la interfaz de usuario que carga la configuración de usuario a la tarjeta leyendo y escribiendo los registros del CPLD. El Code Composer Studio se comunica con la tarjeta DSK a través del emulador JTAG con la interfaz USB.

La DSK puede además ser utilizada con el emulador a través del conector externo JTAG. Las características de esta plataforma de desarrollo son las siguientes:

- Un procesador digital de señales opera a 1 Ghz.
- Un codec stereo AIC23.
- 16 Mbytes de SDRAM.
- 512 Mbytes de memoria flash no volátil.
- 4 LEDS y DIP switch accesibles al usuario.
- Software de configuración a traves de los registros en el CPLD.
- Opciones de configuración de arranque y selección de reloj de entrada.
- Emulación JTAG por medio de un emulador del mismo nombre en la tarjeta con interfaz host USB o emulador externo.
- Voltaje de alimentación de +5 volts.

3.7.1. Procesador digital de señales TMS320C6416T

El procesador digital de señales TMS320C6416 está basado en una avanzada arquitectura veloz de muy larga palabra de instrucción (VLIW), la cual es muy útil para algoritmos numéricos; La memoria de programa interna está estructurada de un total de ocho instrucciones que pueden ser ejecutadas en un solo ciclo [40]. Por ejemplo con una tasa de reloj de 1 Ghz, el C6416 es capaz de ejecutar ocho instrucciones de 32 bits cada 1/(1 GHz) o 1 ns. Las características del C6416 incluyen 1056 kB o memoria interna (32 kB así como L1P y L1D en caché y 1024 kB en la memoria L2 mostrada entre el espacio de programa y de datos), 8 unidades funcionales o de ejecución compuestas de 6 ALUs y dos unidades multiplicadoras, un bus de direcciones de 32 bits para direccionar 4 GB (gigabytes), y dos grupos de registros de propósito general del 32 bits.[2]

3.7.2. Mapeo de memoria de TMS320C6416T DSK

La familia c64xx de DSPs tiene un gran espacio de direccionamiento. El código de programa y de datos puede ser colocado en algún lugar del espacio de direcciones unificado. Las direcciones son siempre de 32 bits de ancho. El mapeo de memoria muestra a la

[2] *Rulph Chanssaing,Digital Signal Processing and Applications with the tms320c6713 and tms320c6416 dsk,Ed. John Wiley, Sons,2008*

izquierda el espacio de memoria direccionable de un procesador genérico 6416T con detalles de cómo es utilizada cada región a la derecha. Por omisión, la memoria interna se sitúa al inicio del espacio direccionable. Las porciones de memoria pueden ser remapeadas por medio de software como la caché L2 así como la RAM fija. Cada EMIF (interfaz de memoria externa) tiene 4 regiones direccionables llamadas espacios disponibles del circuito integrado (CE0-CE3). La SDRAM ocupa CE0 de EMIFA mientras el CPLD y la memoria flash son mapeadas a CE0 y CE1 del respectivo EMIFB. La tarjeta hija utiliza las regiones CE2 y CE3 del EMIFA.

La figura 3.14 muestra el mapa de memoria útil en la plataforma TMS320C6416T DSK. El código programado se encuentra mapeado en las regiones de memoria:

- Memoria Interna: Se encuentra el código de programa incluyendo variables y stack.

- SDRAM: Para arreglos de almacenamiento globales y de gran longitud.

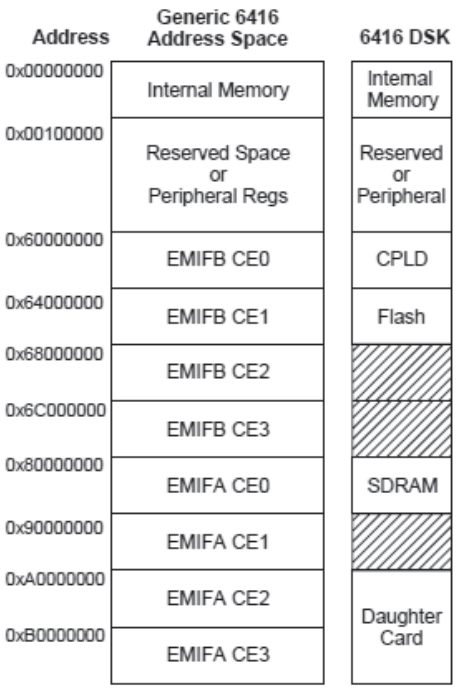

Figura 3.14. Mapa de memoria de TMS320C6416T DSK.

3.8. Canal de comunicaciones PLC

Como se mencionó en el capítulo anterior una de las razones que dificultan la transmisión de información a través de este medio de comunicación son las condiciones de ruido que existen. Dichas condiciones no son constantes y varían dependiendo de la topologia, del lugar y el tiempo en que se realiza la comunicación.

En el canal de comunicaciones PLC se pueden encontrar 4 tipos de ruido [41, 42]:

- Ruido de fondo coloreado (*colored background noise*). Es causado principalmente por la suma de varias fuentes de ruido con baja potencia, tales como equipos electrónicos utilizados en edificios y casas habitación.

- Ruido de banda angosta (*narrow-band noise*). Es causado por interferencias con estaciones de radio AM, FM y radio aficionados.

- Ruido impulsivo periódico (*periodic impulsive noise*). Es causado por la conmutación de fuentes de alimentación conectadas a dicho canal.

- Ruido impulsivo aleatorio (*random impulsive noise*). Es causado por la conexión y desconexión de cargas o dispositivos eléctricos en el canal.

De acuerdo a literatura [43, 44] la forma más adecuada para representar el ruido en un canal PLC es la unión del ruido blanco gaussiano aditivo con el ruido impulsivo. El modelo de ruido considerado más adecuado para la representación es el modelo de ruido blanco aditivo de clase A (AWCN) o también conocido como ruido de clase A de Middleton. Se considera que el ruido impulsivo es el que afecta en mayor grado la comunicación a traves del canal, ya que ocurre en forma aleatoria y puede tener una duración desde microsegundos hasta milisegundos, lo cual puede ser excesivamente grande si se transmite información a altas frecuencias [45].

3.8.1. Ruido blanco gaussiano aditivo (AWGN)

El ruido gaussiano es provocado por interferencias en el canal, por componentes electrónicos o amplificadores en el dispositivo receptor en la comunicación. El modelo de canal con ruido AWGN es aquel en el cual a la señal transmitida en el canal se la adiciona ruido gaussiano y se puede representar mediante la expresión 3.41 [5].

$$r_t = s_m(t) + n(t) \tag{3.41}$$

Donde s_m es la señal transmitida y $n(t)$ es un proceso de ruido gaussiano. La figura 3.15 muestra el esquema de este modelo de canal. Cuando se desea agregar atenuación a este modelo de canal se utiliza la expresión 3.42 donde α es el factor de atenuación.

$$r_t = \alpha s(t) + n(t) \tag{3.42}$$

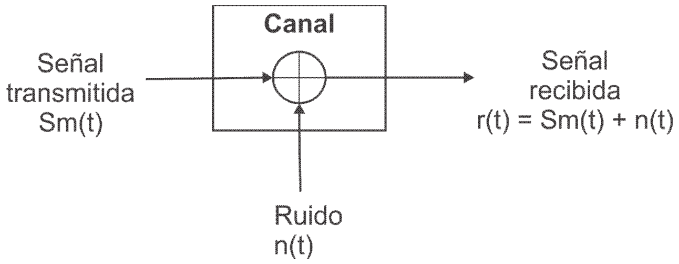

Figura 3.15. Modelo del canal AWGN.

3.8.2. Ruido impulsivo

El ruido impulsivo en el canal PLC se puede dividir en sincrono y asíncrono. Es considerado como la principal fuente generadora de errores en un canal PLC debido a su alta densidad espectral (PSD). Éste suele aparecer en intervalos de tiempo corto y con una alta duración con respecto a la frecuencia de la señal de información. Dichos impulsos pueden tener una duración de desde microsegundos hasta milisegundos. El ruido impulsivo periódico sincrono puede ocurrir en la red con voltaje alterno en 50 Hz en un rango de frecuencias entre 50 y 100 Hz. Las principales fuentes de este de ruido son los rectificadores de voltaje que utilizan diodos y los dimmers. En la figura 3.16 se observa una señal de ruido impulsivo periódico sincrono con periodo de 10 mseg en una frecuencia de 100 Hz. Este tipo de ruido puede verse como una señal envolvente rectangular, la cual, tiene un ancho y también un intervalo o periodo en el que se repite.

De acuerdo a la literatura [3] se han realizado más estudios relacionados con modelos para este tipo de ruido en el dominio del tiempo. El ruido impulsivo puede expresarse como una envolvente rectangular que tiene amplitud, ancho y periodo de pulso. Con ayuda de una función impulso $imp(t)$ con ancho y amplitud unitarios tenemos la expresión 3.43 [41].

$$n_{imp}(t) = \sum_{i=i}^{N} A_i \cdot imp\left(\frac{t - t_{arr,i}}{t_{w,i}}\right) \tag{3.43}$$

donde t_w es el ancho del impulso, $t_{arr,i}$ es el tiempo de arribo, t_d es la distancia

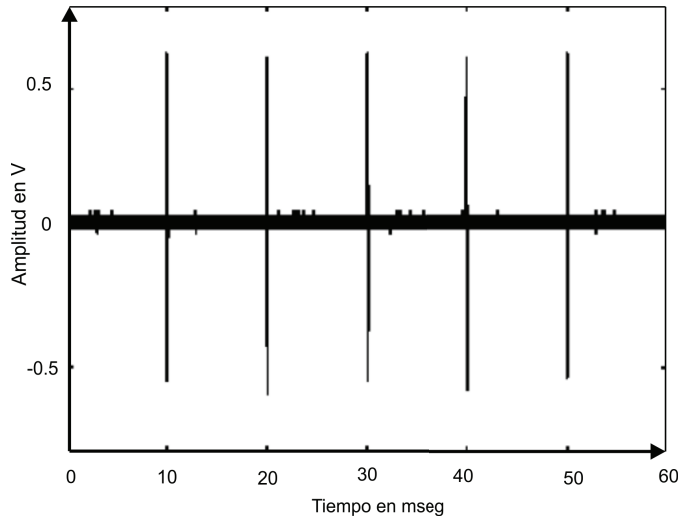

Figura 3.16. Gráfica de señal con ruido impulsivo periódico síncrono en un canal PLC.

al impulso y T_{IAT} es el tiempo de inter-arribo entre dos impulsos. Estas variables son aleatorias y presentan propiedades estadísticas. La distancia entre dos impulsos de ruido generado puede ser descrita con la expresión 3.44.

$$T_{IAT} = t_w + t_d = t_{arr,i+1} - t_{arr,i} \tag{3.44}$$

La tasa del impulso puede ser expresada mediante la relación 3.45

$$\frac{N_{imp}}{T_{win}} \tag{3.45}$$

donde N_{imp} es el número impulsos generados durante una ventana T_{win}.

En el presente trabajo de investigación se utilizó ruido Gaussiano y también el modelo AWCN (utilizado en [45]) también conocido como ruido de clase A de Middleton, el cual es una combinación de ruido aditivo blanco más ruido impulsivo. En este modelo una muestra de ruido $n(m)$ puede ser obtenida de acuerdo a la expresión 3.46.

$$n(m) = x_G(m) + y(m)\sqrt{K_m} \tag{3.46}$$

En donde $x_G(m)$ representa el ruido de fondo Gaussiano con media cero y varianza σ_g^2, $y(m)$ es otra secuencia Gaussiana con media cero y varianza $\frac{\sigma_I^2}{A}$ que representa la potencia de ruido impulsivo; A representa el tiempo promedio entre impulsos arribados, σ_I^2 es la potencia promedio del ruido impulsivo, σ_g^2 es la potencia promedio de la componente

de ruido Gaussiano. y K_m es una variable aleatoria de Poisson, la cual, representa el tráfico impulsivo, con una pdf caracterizada por A. Todas las variables son estadísticamente independientes.

3.9. Método Monte Carlo

El método Monte Carlo fue utilizado a mediados de 1940 cuando la primera computadora electrónica fue construida. En forma más precisa, fueron John Von Newmann y Stanislaw Ulam quienes trabajaron en una idea para trabajar con números aleatorios generados por una computadora para resolver problemas encontrados en el desarrollo de la bomba atómica [46].

El nombre Monte Carlo fue usado como título en el artículo de 1949 por Metropolis y Ulam, referido al famoso casino en Mónaco donde la aleatoriedad fue utilizada en forma repetitiva. Este método proporciona soluciones aproximadas a una gran cantidad de problemas matemáticos.

Se puede mencionar que utiliza la generación aleatoria para estudiar el comportamiento de sistemas que poseen componentes que se comportan en forma aleatoria. En otras palabras consiste en simular en la computadora, el comportamiento del sistema mediante la generación aleatoria de variables describiendo el comportamiento de sus componentes. El método genera las probabilidades de ocurrencia de un evento determinado.

Una aplicación sencilla del método Monte Carlo puede ser la estimación del área de una curva cerrada dentro de un cuadrado. Por ejemplo una cuarta parte de circunferencia dentro de un cuadrado, puede servir para el cálculo del valor de π. Para ello se realiza un muestreo uniforme de puntos dentro del cuadrado, al incrementar la cantidad de puntos se divide la cantidad de puntos que caen dentro de la curva entre la cantidad total de puntos muestreados. Esta relación es aproximada a la relación del área de la curva cerrada entre el área del cuadrado en el que se encuentra inscrita dicha curva.

En el presente trabajo su utilizará el método de simulación Monte Carlo para evaluar el desempeño del módulo corrector de errores propuesto y para obtener las curvas BER. En la figura 3.17 se puede observar un esquema general de la aplicación del método Monte Carlo en un sistema de comunicaciones [47].

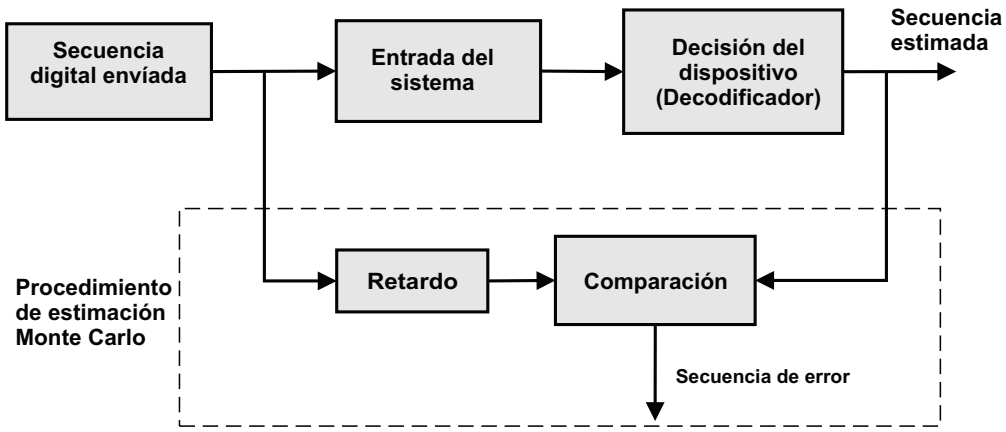

Figura 3.17. Diagrama de bloques del método de simulación Monte Carlo en un sistema de comunicaciones.

3.10. Resumen del capítulo

En el presente capítulo se describieron las características básicas de un código convolucional, por ejemplo tamaño de memoria, número de estados; entre otras. También se explicaron sus principales formas de representación, entre las que se encuentran: diagrama de Trellis, de estados, representación en el dominio de retardo D. Estas formas de representación son útiles para el manejo de los algoritmos tanto en la codificación como la decodificación de los turbo códigos.

Con respecto a la codificación, cabe resaltar que el turbo codificador está formado por la concatenación en paralelo de dos codificadores sistemáticos recursivos, un bloque intercalador y un bloque "puncturing" o mecanismo de perforación. Dada la estructura formada por los componentes anteriores, una de las características es la generación de pocas palabras de código con bajo peso y un alto grado de independencia entre la información de redundancia generada por codificadores.

Otra etapa importante es la decodificación, en la cual el algoritmo originalmente usado fue el algoritmo BCJR-MAP. Este algoritmo calcula la relación de máxima probabilidad a posteriori para cada bit de la secuencia recibida. La estructura general de un turbo decodificador se compone de dos decodificadores unidos por una retroalimentación entre la salida del segundo decodificador y la entrada del primero. Esta unión permite obtener una ganancia en la decodificación basándose en la información auxiliar proporcionada entre los dos decodificadores.

Debido a la complejidad computacional requerida por el algoritmo BCJR-MAP se desarrollaron variantes que utilizarán menos recursos y proporcionarán resultados

aproximados. Entre estos algoritmos se encuentra el algoritmo Max-log-MAP. El cual ha sido utilizado en las implementaciones de los turbo códigos y ha permitido su integración en dispositivos más pequeños, rápidos y de menor costo.

El canal PLC posee características que lo han hecho un área interesante de investigación. Entre ellas se encuentran que a diferencia de otros canales de comunicación, éste no puede ser modelado como un canal únicamente con ruido Gaussiano ya que según diferentes estudios realizados se han observado que existen 4 principales tipos de ruidos los cuales son: ruido de banda angosta, ruido fondo coloreado, ruido impulsivo sincrono/asíncrono con respecto a la fuente principal y ruido impulsivo asíncrono.

Los temas presentados en este capítulo conforman los principios básicos de estudio para turbo códigos así como las características de ruido en un canal PLC. En el siguiente capítulo se presentará el sistema de comunicación implementado para realización de las pruebas del módulo corrector de errores en el DSP.

Capítulo 4

Arquitectura y funcionalidad del módulo corrector de errores propuesto

En este capítulo se describirán las características del módulo corrector de errores implementado, así como el sistema de comunicación propuesto para la realización de las simulaciones. También se explicarán la implementación de los algoritmos en el DSP y la biblioteca de códigos que conforman el módulo corrector de errores. Además se describirán las características de la interfaz gráfica programada en Matlab y sus funciones principales. Finalmente se dará un resumen del capítulo.

4.1. Introducción

Para el desarrollo de la presente investigación se ha construido un esquema de turbo código con una estructura del codificador de $RSC\,(7,5)$, $RSC\,(17,15)$ y $RSC\,(31,27)$. A su vez el decodificador utiliza el algoritmo Max-log-MAP. Para incrementar la tasa de código se implementó un módulo "puncturing" a la salida del codificador, los datos son intercalados con el objetivo de reducir la cantidad de errores generados en ráfaga. El tipo de intercalador utilizado es S-random, seudo-aleatorio y de tipo bloque que puede tener un tamaño de $N = 1000$ y el cual puede ser ampliado hasta un tamaño de $N = 4160$.

Ya que la arquitectura del procesador digital de señales TMS320C6416T es de punto fijo, las operaciones son realizadas en formato Q15, es decir se utiliza la conversión a este formato para representar los números en punto flotante. Para el almacenamiento de los datos utilizados en el algoritmo, se utilizaron arreglos en memoria IRAM y SDRAM.

Debido a las restricciones en los recursos del hardware, los parámetros del turbo código son restringidos en cuanto a la longitud del bloque de datos entre 500 y 4160 bits en el codificador y el decodificador.

El algoritmo de decodificación implementado en el DSP es el algoritmo Max-log-MAP [5, 17]. El cual fue utilizado con representación de punto fijo en el formato Q15 para las operaciones realizadas. La etapa de "puncturing" implementada genera una tasa de código de 1/2 a la salida del codificador. Lo cual permite incrementar la tasa de código pero a la vez incrementa el valor de umbral para la convergencia del algoritmo.

Para utilizar condiciones similares a las que se pueden tener en un canal PLC se utilizó el simulador transreceptor OFDM diseñado por [45] y una versión de turbo código similar programada en la PC, la cual fue agregada a dicho simulador para analizar el funcionamiento del código corrector variando diferentes parámetros como son: la longitud del intercalador, tipo de códificador, número de iteraciones; entre otros.

Con base en las pruebas realizadas, la latencia generada en el DSP y la literatura [4] se seleccionaron los parámetros para realizar las pruebas con el simulador en un canal con ruido Gaussiano e impulsivo como se explicará en el capítulo 5. Por lo cual en el presente capítulo se describen a detalle las características de la implementación de dicho módulo corrector de errores.

4.2. Diagrama de bloques del sistema propuesto

El sistema propuesto se compone del módulo corrector de errores implementado en la plataforma de desarrollo DSP TMS320C6416T DSK, así como también de la interfaz diseñada en Matlab con una conexión por medio del puerto serial entre ellos. La interfaz se encarga de almacenar los resultados entregados por el DSP y mostrarlos en forma gráfica. Ademas contiene los algoritmos para poder simular y observar los resultados en la PC de forma independiente.

Por su parte se implementaron dos bibliotecas de funciones que conforman en su conjunto el módulo corrector de errores en el DSP. En el procesador digital de señales se encuentra el codificador y decodificador implementado. El dispositivo puede funcionar como transmisor y receptor codificando y decodificando la información. En la figura 4.1 se muestra el esquema del sistema propuesto.

El turbo código contenido en el módulo corrector de errores está formado por un módulo transmisor, el cual a su vez contiene los bloques: intercalador, codificador RSC 1, codificador RSC 2 y un mecanismo de perforación. Por su parte el módulo receptor se compone de los siguientes bloques: un decodificador 1, un decodificador 2, un intercalador

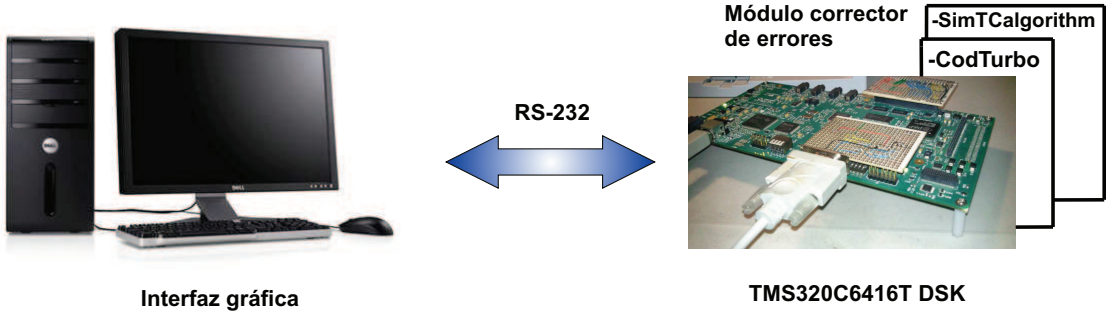

Figura 4.1. Esquema del sistema implementado.

y un des-intercalador. El algoritmo utilizado para realizar la decodificación de los datos recibidos es el algoritmo Max-Log-MAP. En la figura 4.2 se muestra el diagrama de bloques del codificador y decodificador.

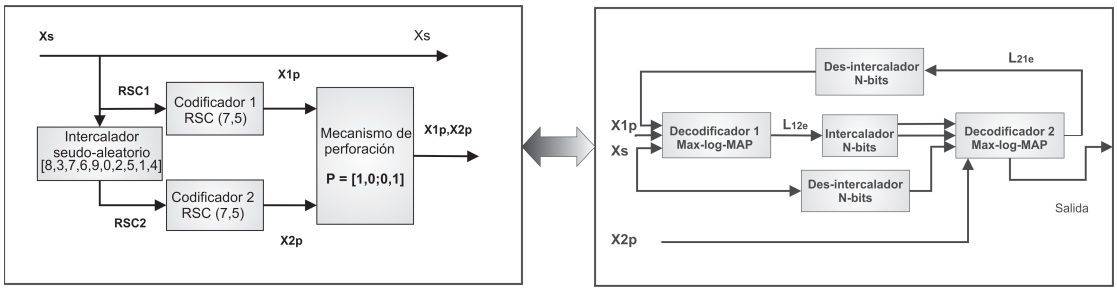

Figura 4.2. Diagrama de bloques del módulo corrector de errores implementado.

La interfaz gráfica permite analizar las curvas BER generadas por el DSP al procesar un conjunto de datos o realizar una simulación Monte Carlo. También permite realizar la codificación de imágenes binarias y recuperar dichas imágenes desde del DSP. A su vez contiene la versión similar de los algoritmos implementados en el DSP con el objetivo de analizar su comportamiento.

Las características del sistema implementado para la comunicación entre el DSP y la PC se presentan en la tabla 4.1.

Tabla 4.1. Características del sistema implementado.

TMS320C6416T DSK	▪ Frecuencia de operación hasta 1 GigaHertz. ▪ 16 Mbytes de memoria sincrona SDRAM. ▪ 512 Kbytes memoria flash no volátil. ▪ Voltaje de alimentación 5 volts. ▪ Codec estéreo AIC23.
Computadora	▪ Procesador: Intel core 2 Quad a 2.5 Gigahertz. ▪ Memoria RAM: 2 GB. ▪ Sistema operativo: Windows xp.
Interfaz serial	▪ 115,200 bps

4.3. Implementación del codificador

El módulo de codificación de canal implementado, consiste de dos codificadores convolucionales sistemáticos recursivos (RSC) con una estructura idéntica entre ellos (descrita en el capítulo 3 en formato octal) RSC(7,5), la cual también puede ser configurada a RSC(17,15) y RSC(31,27) respectivamente.

En la figura 4.3 se muestra la estructura general del turbo codificador implementado y utilizado en el código corrector de errores en el DSP.

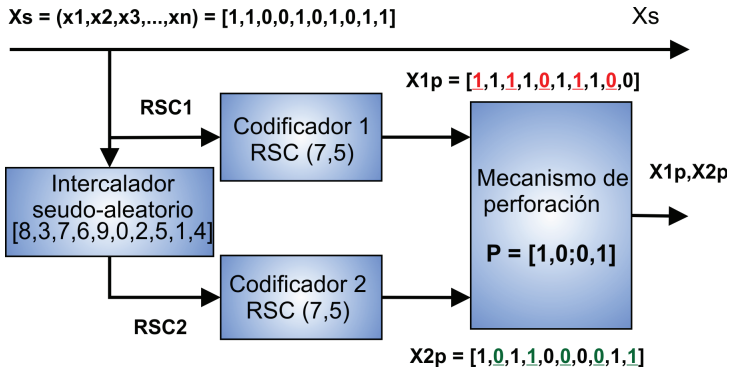

Figura 4.3. Diagrama de bloques del turbo códificador implementado.

Aunque la estructura del codificador es de complejidad relativamente baja para la implementación, no es el mismo caso en el lado receptor. Por ejemplo, para el caso del codificador RSC(7,5) con 4 estados y 2 símbolos de memoria, es necesario almacenar los valores de las probabilidades α, β y γ para cada uno de los estados en arreglos en memoria. Ya que son necesarios para realizar los cálculos en cada iteración en el algoritmo de decodificación. La matriz generadora para el codificador RSC(7,5) se observa en la expresión 4.1.

$$G(D) = \frac{C(D)}{M(D)} = \left[\begin{array}{cc} 1 & \frac{1+D^2}{1+D+D^2} \end{array} \right] = \left[\begin{array}{cc} 1 & \frac{5}{7} \end{array} \right] \tag{4.1}$$

Como se puede observar, la forma de la matriz generadora contiene cada una de las conexiones entre los estados del codificador. El numerador contiene las conexiones de la parte inferior y el denominador representa las conexiones de la parte superior de dicho codificador. Por ejemplo, en el polinomio $1+D+D^2$ del denominador, al tomar los elementos de derecha a izquierda (menos significativo a más significativo) se tiene que es equivalente a 111 en representación binaria, lo que a su vez es equivalente a 7 en formato octal. Por otro lado, de forma similar, tomando los elementos del polinomio en el numerador, se tiene que $1+D^2 = 101$ en formato binario, lo que a su vez equivale a 5 en formato octal. En la figura 4.4 se muestra la estructura del codificador RSC(7,5), la cual se compone de dos líneas de retroalimentación hacia atrás y dos líneas unidas por un punto de suma, en conjunto para generar los bits de redundancia.

En la expresión 4.2 se presenta la matriz generadora para el codificador RSC $(17, 15)_{octal}$. Tomando el polinomio $1 + D + D^2 + D^3$ del denominador de la fracción y tomando los elementos de derecha a izquierda (de menos a más significativo) se tiene que equivale a 1111 en formato binario, lo que a su vez es equivalente a 17 en formato octal. De forma similar tomando los elementos el polinomio $1 + D + D^3$ del numerador de derecha a izquierda (de menos a más significativo) se tiene que equivale a 1101 en formato binario, lo que equivale a 15 en formato octal.

Es importante señalar que al incrementar el número de estados y la longitud de memoria del codificador, se incrementan los recursos requeridos en la memoria del hardware utilizado para almacenar las probabilidades para el bloque de datos a la entrada. La figura 4.5 muestra la estructura para este codificador.

$$G(D) = \frac{C(D)}{M(D)} = \left[\begin{array}{cc} 1 & \frac{1+D+D^3}{1+D+D^2+D^3} \end{array} \right] = \left[\begin{array}{cc} 1 & \frac{15}{17} \end{array} \right] \tag{4.2}$$

Ya que la complejidad de la estructura del codificador es relativamente baja, se puede implementar en las plataformas actuales de hardware. Por ejemplo, en el caso del

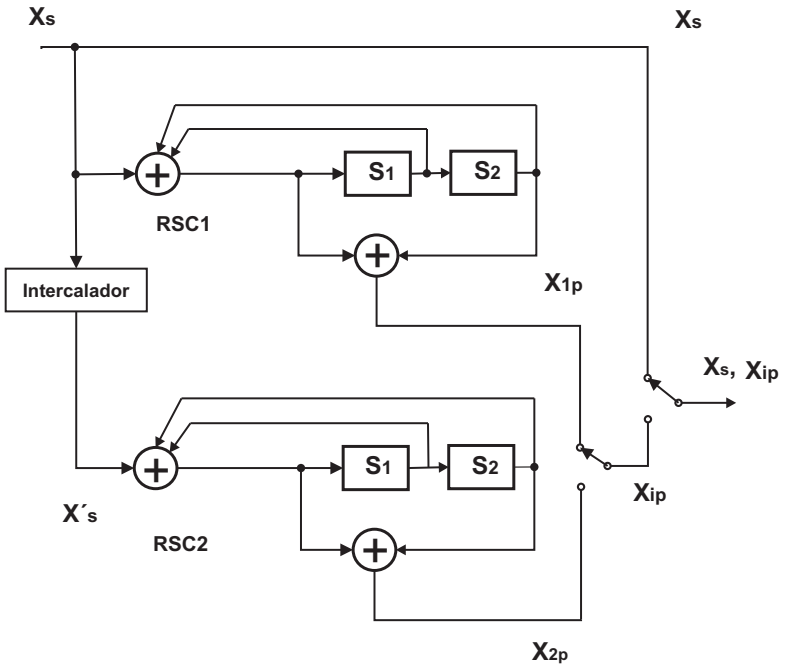

Figura 4.4. Estructura del codificador RSC(7, 5).

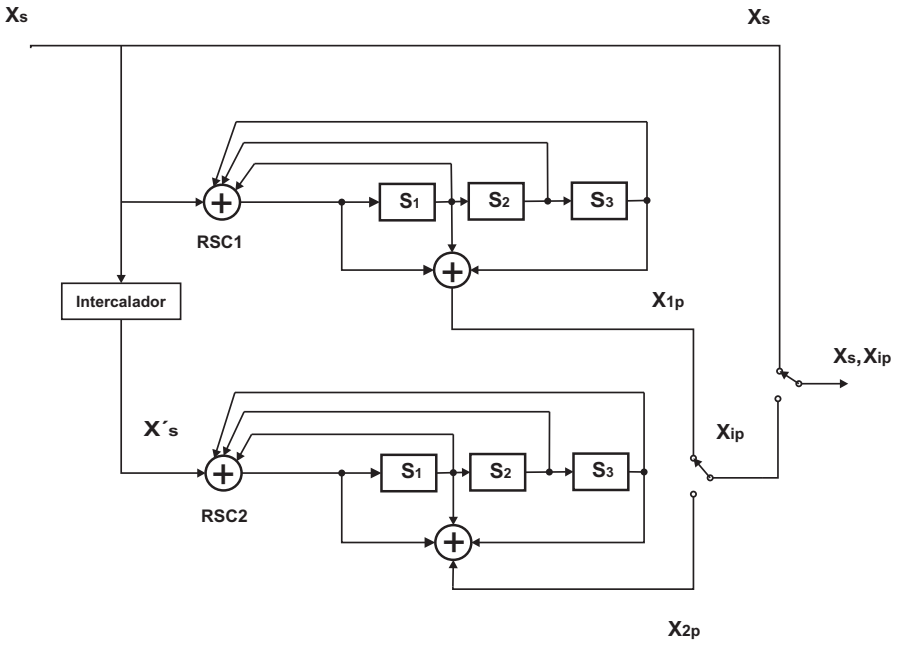

Figura 4.5. Estructura del codificador RSC(17, 15).

codificador RSC(7,5) dado que el número de estados del codificador es S=4 y el tamaño de memoria es M=2, resulta factible su implementación en dispositivos de hardware, por ejemplo un procesador digital de señales (DSP).

En la figura 4.6 se muestra el codificador RSC (31,27) implementado. La estructura de este codificador se compone de 16 estados así como 4 registros de memoria. La expresión 4.3 representa la matriz generadora para dicho codificador.

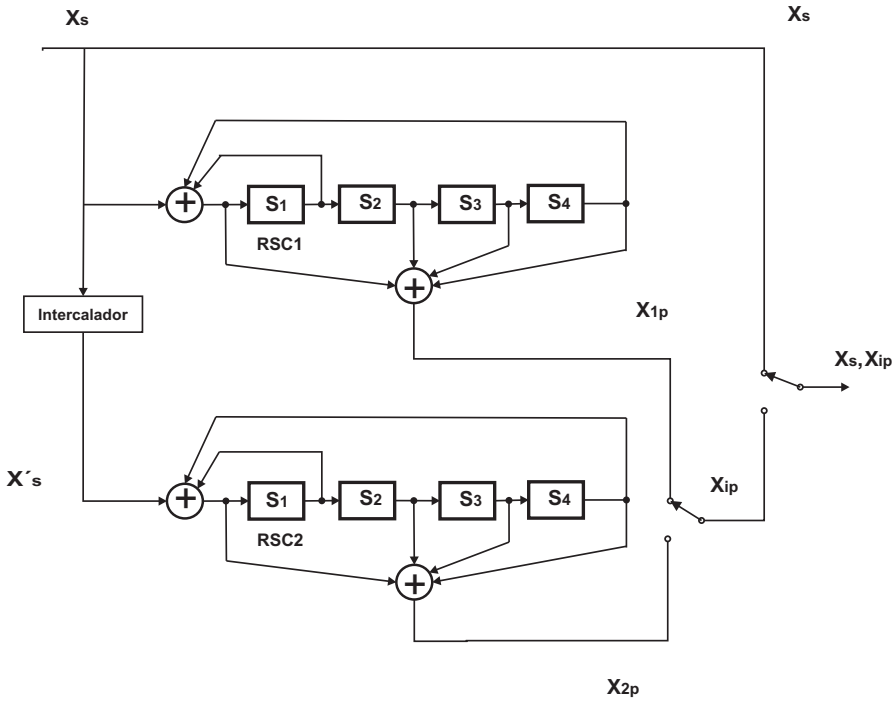

Figura 4.6. Estructura del codificador RSC(31, 27).

La matriz generadora para el codificador de la figura 4.6 se presenta en la expresión 4.3. Si se toma el polinomio $1+D^2+D^3+D^4$ del numerador y se toman de derecha a izquierda (de menos significativo a más significativo) será igual a 10111 en representación binaria y a 27 en formato octal. A su vez el polinomio $1+D+D^4$ del denominador será igual a 11001 en representación binaria y a 31 en formato octal.

$$G(D) = \frac{C(D)}{M(D)} = \left[\begin{array}{cc} 1 & \frac{1+D^2+D^3+D^4}{1+D+D^4} \end{array}\right] = \left[\begin{array}{cc} 1 & \frac{27}{31} \end{array}\right] \tag{4.3}$$

4.3.1. Módulo intercalador y mecanismo de perforación

El intercalador implementado puede ser de tipo S-random, seudo-aleatorio y de tipo bloque con tamaño igual a la longitud del bloque de datos de entrada N. El cual de acuerdo a las investigaciones realizadas produce un alto desempeño del BER al incrementar la longitud de la trama de datos. Para la implementación, en el caso del intercalador de tipo S-random, se utilizó el algoritmo S-random descrito en el capítulo 3 y se generaron dos tablas, una para el intercalador y otra para el des-intercalador las cuales fueron almacenadas en el codificador y el decodificador. A continuación se presenta un ejemplo de un intercalador y un des-intercalador de tipo S-random con una longitud igual a 20 y una separación de al menos $S = 3$ entre elementos consecutivos:

$$B\left(I_B \rightarrow I\right) = [1, 5, 8, 15, 12, 19, 4, 9, 13, 17, 2, 6, 10, 18, 3, 14, 7, 11, 0, 16]. \tag{4.4}$$

$$B^{-1}\left(I_{B^{-1}} \rightarrow I\right) = [18, 0, 10, 14, 6, 1, 11, 16, 2, 7, 12, 17, 4, 8, 15, 3, 19, 9, 13, 5]. \tag{4.5}$$

El mecanismo "puncturing" utilizado se compone básicamente de una matriz cuadrada de 2×2 dada en la expresión 4.6 que perfora los bits de paridad alternadamente a la salida de los codificadores RSC. En el esquema de turbo código en el DSP se utilizó dicho mecanismo.

$$P = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{4.6}$$

4.3.2. Construcción de los bloques de datos

Con el esquema del codificador, la construcción de una trama de datos a enviar se realizará de la siguiente forma: primeramente a los bits del bloque de entrada se le agregan n-k bits con valor cero para que la secuencia pueda finalizar en el estado inicial cero visto en el diagrama de Trellis. Este método es generalmente conocido como "zero-tail" y con él se logra obtener un bloque de datos para ser decodificado.

Una vez que el bloque de datos de entrada es procesado por el primer codificador RSC 1 se genera la secuencia de bits de redundancia X_{1p}. Para el caso del segundo codificador RSC el bloque de bits es permutado por el intercalador y después es ingresado a dicho codificador generando la secuencia de bits de paridad X_{2p}. La tasa típica de este esquema es de 1/3 utilizando el bloque completo de bits a la salida del turbo codificador y las dos secuencias de bits de redundancia generados. Sin embargo, si se utiliza la técnica de

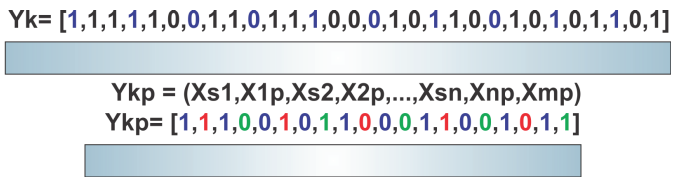

Figura 4.7. Trama construida por el turbo codificador con tasa de código 1/2.

"puncturing" la tasa de bits a la salida será de 1/2.

La figura 4.7 muestra la trama a la salida del turbo códificador utilizando el mecanismo de perforación junto con una trama construida sin utilizar dicho mecanismo. La secuencia perforada contiene los bits sistemáticos o de información y la mitad de los bits de redundancia generados por cada codificador RSC. Con este esquema se incrementa la tasa de código, es decir, disminuye la cantidad de información redundante enviada.

4.4. Implementación del decodificador

Para la implementación del decodificador se utilizó el algoritmo Max-Log-MAP [5, 17]. Dado que se trata de una aproximación del algoritmo BCJR-MAP original se tienen resultados con menos precisión y por lo cual un desempeño más bajo, sin embargo, la complejidad en el procesamiento disminuye considerablemente.

En la figura 4.8 se muestra en un diagrama de Trellis, la selección de la trayectoria más probable hacia el siguiente estado conforme al valor del dato recibido. Las líneas punteadas indican las trayectorias posibles cuando el dato tiene un valor de cero y con líneas continuas para valores de uno. Del lado izquierdo se puede observar que se tienen dos estados $A_{k-1}(0)$ y $A_{K-1}(1)$ y el siguiente estado $A_k(2)$. De esta forma calculando las probabilidades de las dos posibles rutas, cuando el dato de entrada toma el valor de cero $\Gamma(0,2)$ y $\Gamma(1,2)$, se elige la ruta más probable o más corta $(\Gamma(1,2))$ y ésta es almacenada. En el sentido de delante hacia atrás tenemos los estados $B_k(1)$ y $B_k(3)$ y el estado anterior $B_{k-1}(3)$ en donde se calculan los valores de las dos rutas $\Gamma(3,1)$ y $\Gamma(3,3)$, las cuales, son las rutas más cortas para los valores del dato de 1 y 0 respectivamente.

La probabilidad $L(U_k|y)$, es el resultado de la diferencia de los máximos valores de $A_{K-1}(s') + \Gamma_k(s',s) + B_k(s)$ cuando el dato toma el valor de 0 y cuando toma valor de 1.

El esquema del decodificador implementado en el DSP TMS320C6416T es presentado en la figura 4.9. Como se describió en el capítulo 3, un turbo decodificador está compuesto por dos *decodificadores componentes* conectados en serie, los cuales, tienen la información a priori a la entrada así como, la información recibida y la información extrínseca recibida desde el otro decodificador. Esta información es inicializada a cero para la primera iteración.

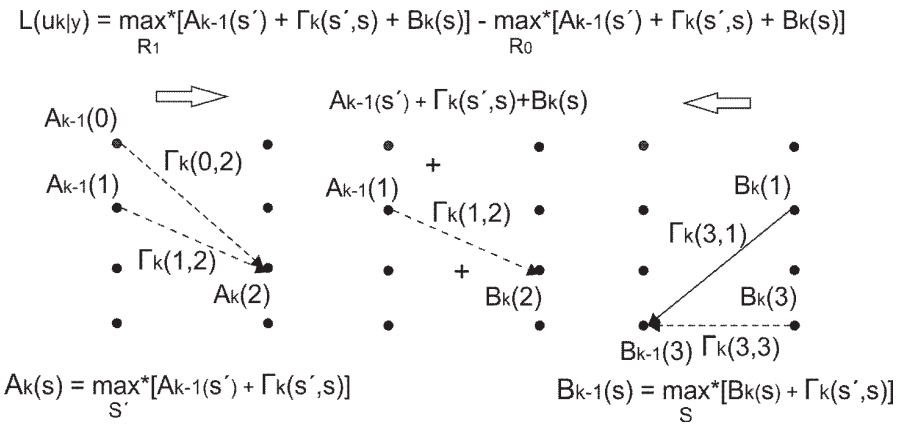

Figura 4.8. Diagrama de Trellis algoritmo Max-Log-MAP.

El principio de la decodificación de la información está basado en un número de iteraciones determinado, en el cual, la información obtenida por cada uno de los decodificadores es consultada para determinar con una decisión dura (0,1), el valor del dato recibido.

Para el presente esquema, se consideró un turbo código binario con una tasa de código 1/2 y para el caso de perforación (puncturing) se insertaron valores nulos en el receptor en las posiciones de los bits perforados de acuerdo a la matriz de "puncturing".

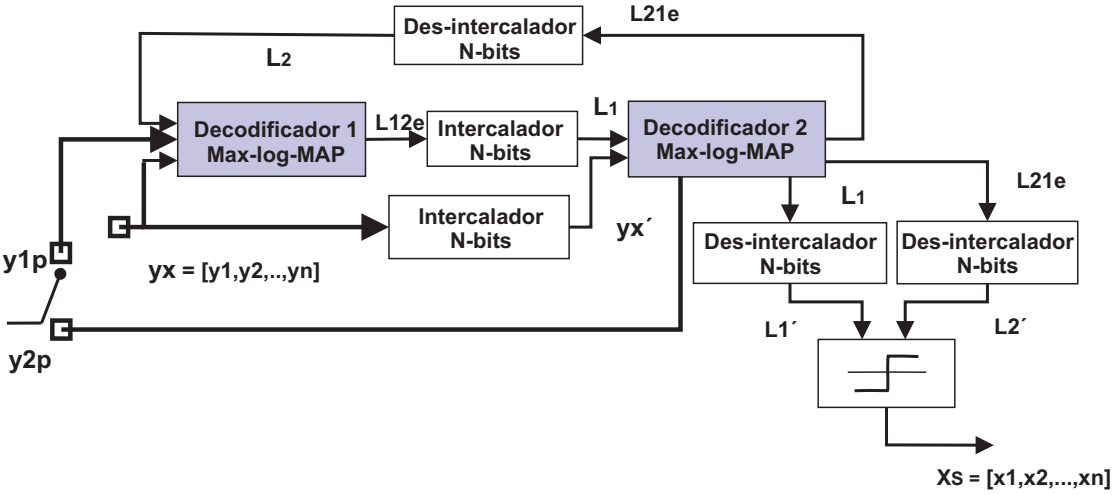

Figura 4.9. Decodificador implementado Max-Log-MAP.

Es importante señalar que la información extrínseca proporcionada para cada uno de los decodificadores es multiplicada por un factor de entre 0 y 1 en cada iteración en el algoritmo. Esto con el objetivo de mantener la estabilidad en la retroalimentación de la información calculada. Por ejemplo, en la primera iteración este parámetro puede tener un

valor de inicial de 0.7 y se puede ir incrementando con un valor proporcional (por ejemplo: $\Delta = \frac{1-0,7}{ITERACIONES}$) hasta finalizar en 1 en la iteración final.

Algunos parámetros en el algoritmo son fundamentales en el desempeño del turbo código. Entre éstos se encuentran: el número de iteraciones, el cual, en muchas ocasiones suele configurarse entre 6 y 10. El número de iteraciones se seleccciona de acuerdo a la latencia soportada por la aplicación y a la velocidad del hardware utilizado.

En el caso de la longitud del intercalador es conveniente utilizar un tamaño mayor a 1000 pues el incremento en el desempeño del turbo código es proporcional al tamaño o longitud de este componente. Además de acuerdo al tipo de intercalador utilizado se alcanza un buen desempeño. En el caso de la presente implementación se ha seleccionado un intercalador de tipo S-random pues mostró mejor comportamiento para las longitudes utilizadas en el intercalador.

En la figura 4.10 se muestra el esquema basado en el funcionamiento iterativo del turbo decodificador. Para describirlo hay que tomar en cuenta: los dos decodificadores componentes, la información del canal que contienen los bits de redundancia y la información sistemática, la información extrínseca que es la información a la salida de cada decodificador y a la vez es la información a priori para el otro.

En la primera iteración el primer decodificador toma la información útil y la redundante, así como la información a priori inicializada a cero y calcula la información extrínseca. El segundo codificador toma esta información y de igual manera los bits de información útil y redundante y calcula información de salida. Si es el caso que se alcanzó la iteración de parada o convergencia, se calcula la información dura con base en los resultados obtenidos por los dos decodificadores.

La figura 4.11 muestra el principio utilizado para realizar la turbo decodificación, en el cual, cada decodificador componente, utiliza la información extrínseca producida por el otro decodificador como información a priori para refinar la probabilidad calculada a su salida. De forma similar, el otro decodificador componente utiliza la información de este decodificador como para calcular la información extrínseca a su salida y finalmente este proceso se repite iterativamente.

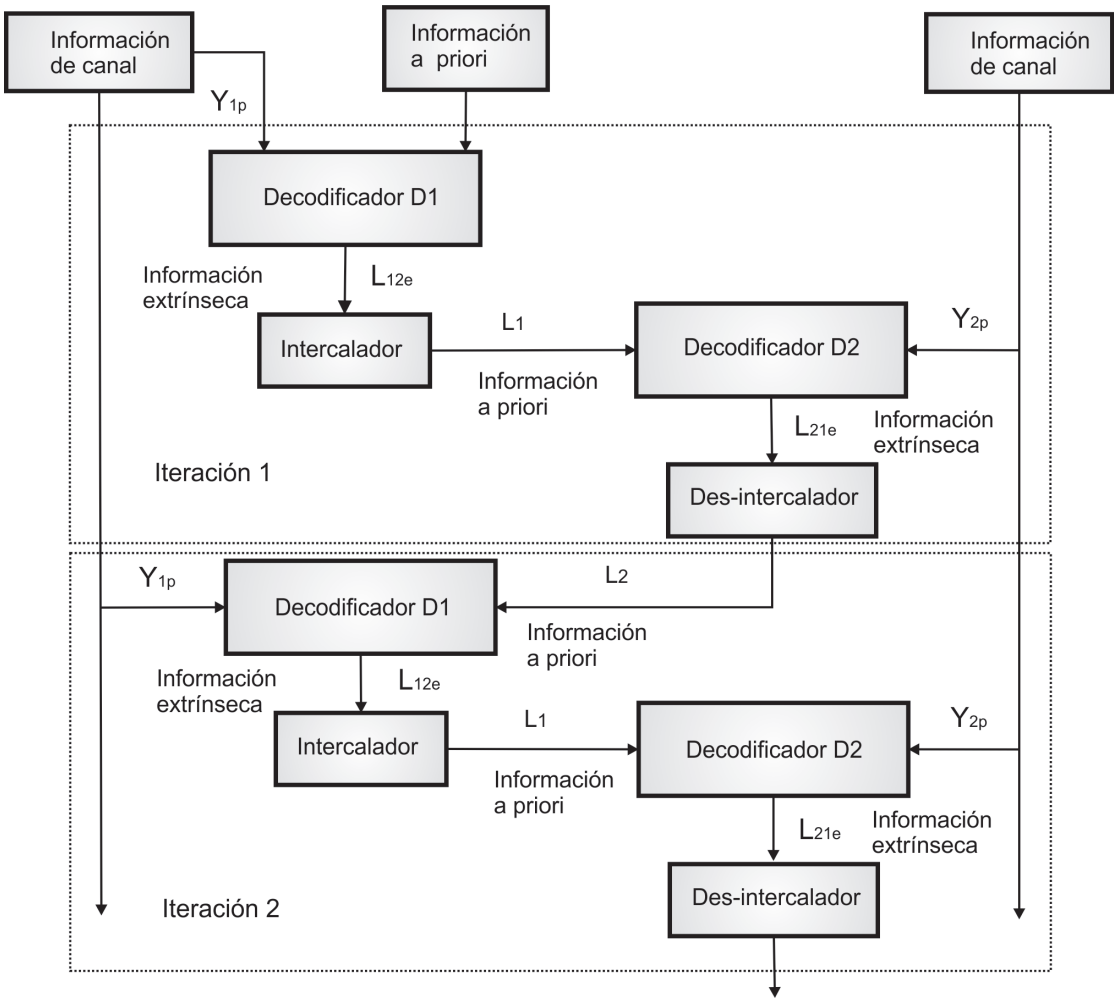

Figura 4.10. Diagrama de flujo del procedimiento de decodificación.

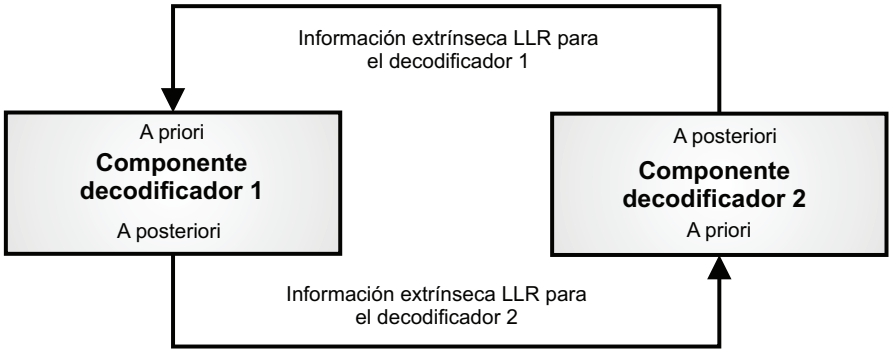

Figura 4.11. Principio de decodificación en el turbo código implementado.

4.5. Programación de los algoritmos en el DSP

Para la programación de los algoritmos en el DSP se utilizó el editor de código Code Composer v3.1. Este editor está diseñado para programar, simular, y comunicar varias series de procesadores digitales de señales de la compañía Texas Instruments. En la figura 4.12 se muestra el proceso que realiza el editor para generar el código ejecutable y realizar la programación de un algoritmo en el DSP. La programación del código fuente puede realizarse en lenguaje C o C++, así como en lenguaje emsamblador. Con estos archivos se realiza la compilación y optimización de código para después generar el código en emsamblador correspondiente. Después se realiza la generación de los archivos objeto (.obj) para que finalmente el enlazador genere el código ejecutable y se realice la programación en el dispositivo.

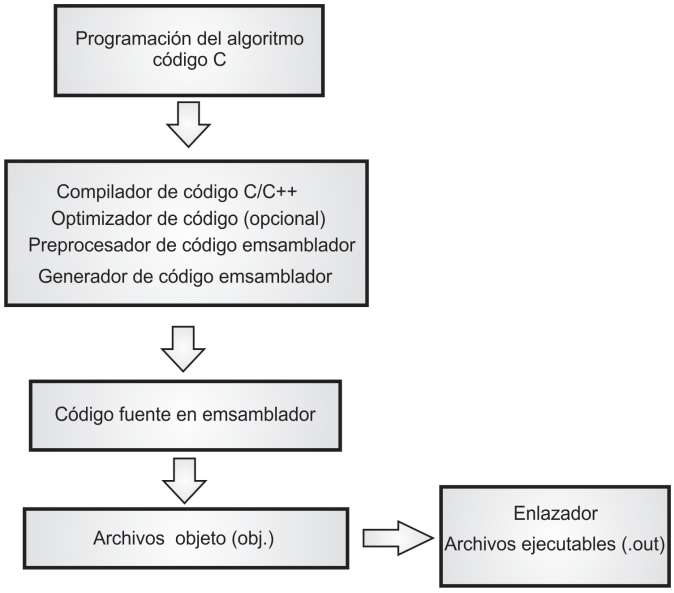

Figura 4.12. Diagrama de flujo del proceso para la implementación de los algoritmos en el DSP.

4.6. Biblioteca de códigos del módulo corrector de errores

Como parte de este trabajo de investigación se implementaron en el DSP dos bibliotecas de funciones (SimTCalgorithm y CodTurbo) para poder construir y realizar pruebas con diferentes esquemas en el turbo código. Las dos bibliotecas corresponden a los elementos que conforman el codificador y decodificador así como las funciones requeridas para realizar la simulación Monte Carlo y la decodificación de bloques de datos respectivamente.

Las bibliotecas de códigos implementadas en el DSP se componen de los elementos que se muestran en la tabla 4.2. Estas bibliotecas permiten tener tres esquemas diferentes en el turbo codificador. También permiten generar ruido Gaussiano de acuerdo a un valor de SNR dado. Se pueden tener tres tipos de intercalador: seudo-aleatorio, S-random y de tipo bloque. En el decodificador se pueden variar el número de iteraciones, así como la longitud de bloque de datos a decodificar.

Tabla 4.2. Elementos que conforman la biblioteca de códigos en el DSP

Elemento	**Nombre de la función**
Simulación Monte Carlo	SimTCiteraciones
Decodificación bloque binario	DecTurbo
Decodificación con ruido	decode_TC_Q15
Creación de mensaje binario	createmesg
Intercalador seudo-aleatorio	createinterleave_rand
Intercalador tipo bloque	createinterleave_block
Codificación de un codificador RSC	encode
Intercalador de un bloque de datos	interleave
Desintercalador de un bloque de datos	deinterleave
Mecanismo de perforación	puncture
Calculo de la potencia del ruido	calc_No
Algoritmo Max-Log-MAP	decodemaxlogmap
Ruido Gaussiano	gaussian2
Tipo de codificador	createtable
Turbo codifica un bloque de datos	encodeTCframe

4.7. Descripción de la interfaz serial

Para la construcción de la interfaz serial en el DSP se utilizó el módulo de puerto serial con almacenamiento multicanal (McBSP) que se encuentra dentro de este dispositivo. La funcionalidad para interfazar este módulo a una interfaz UART fue implementada mediante software. Para poder utilizar el generador interno de tasa de muestreo como el mismo reloj, el McBSP fue configurado para recibir y transmitir un bit UART como una palabra de 16 bits. Es decir para transmitir un bit, éste debe ser expandido a un ancho 16 bits y al momento de recibir una palabra de 16 bits, ésta debe ser comprimida a un solo bit. Además se debe tomar el tamaño necesario de la trama de datos.

La conexión del módulo McBSP a una interfaz UART se muestra en la figura 4.13. La línea de transmisión del UART se encuentra conectada a las líneas de entrada de datos

y de sincronización del módulo ya que la línea de transmisión en el dispositivo UART contiene tanto los datos como la sincronización. A su vez, la línea de transmisión del McBSP está conectada a la entrada de datos del UART. Además de la interfaz universal asíncrona receptor transmisor (UART) implementada, se encuentra conectado un hardware adicional (MAX233) para convertir los niveles de voltaje típicos de la comunicación RS-232.

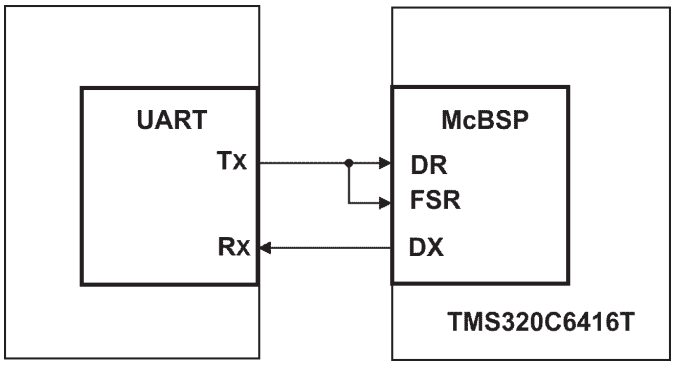

Figura 4.13. Implementación utilizando el puerto serial.

La transferencia de la información entre el McBSP y los buffers de entrada y salida se lleva a cabo mediante acceso directo a memoria (DMA). La lectura o escritura de los datos se lleva a cabo mediante bloques de tamaño configurable. Para realizar la transmisión de un bloque de datos cada uno de los bits es convertido en una palabra de 16. Por ejemplo un 1 equivale a 0xFFFF y un 0 a 0x0000. Una vez que cada dato es convertido al formato apropiado, éste es puesto en el buffer de transmisión hasta completar el bloque deseado. Al bloque de datos a transmitir se le agrega el dato de inicio y fin de trama en las posiciones correspondientes. Después, este bloque de datos es transferido al buffer transmisor del McBSP por medio de EDMA y entonces es transmitido utilizando el generador sincrono de trama, el cual transmite continuamente uno a uno cada uno de los datos. En la figura 4.14 se muestra la conversión realizada a un conjunto de bits que se desea transmitir utilizando el formato previamente descrito.

En la figura 4.15 se puede observar la interfaz implementada en el DSP TMS320C6416T. Dicha interfaz permite la conexión al software diseñado para el almacenamiento y presentación de los resultados por un puerto de comunicación independiente al utilizado para la programación y simulación en la PC. La figura 4.16 muestra la conexión de la tarjeta DSK TMS320C6416T con la PC.

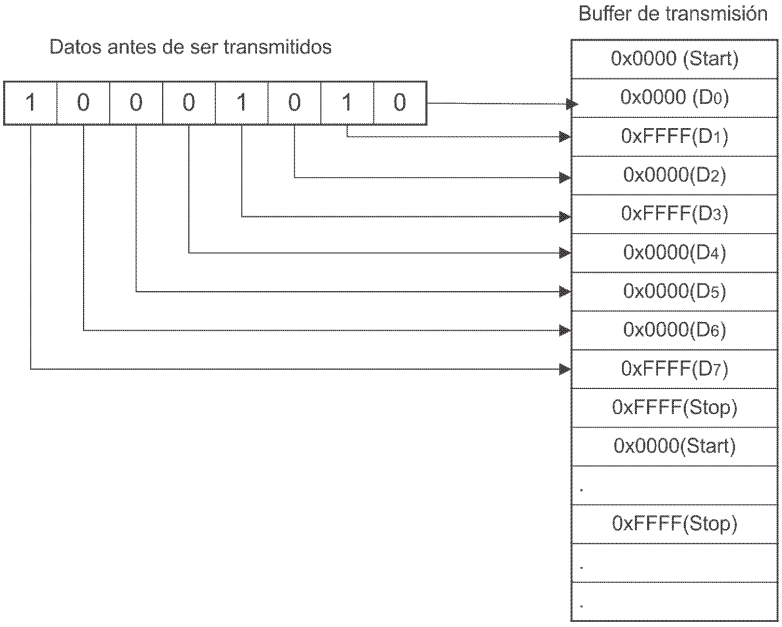

Figura 4.14. Procesamiento de un bloque de datos por el buffer transmisor.

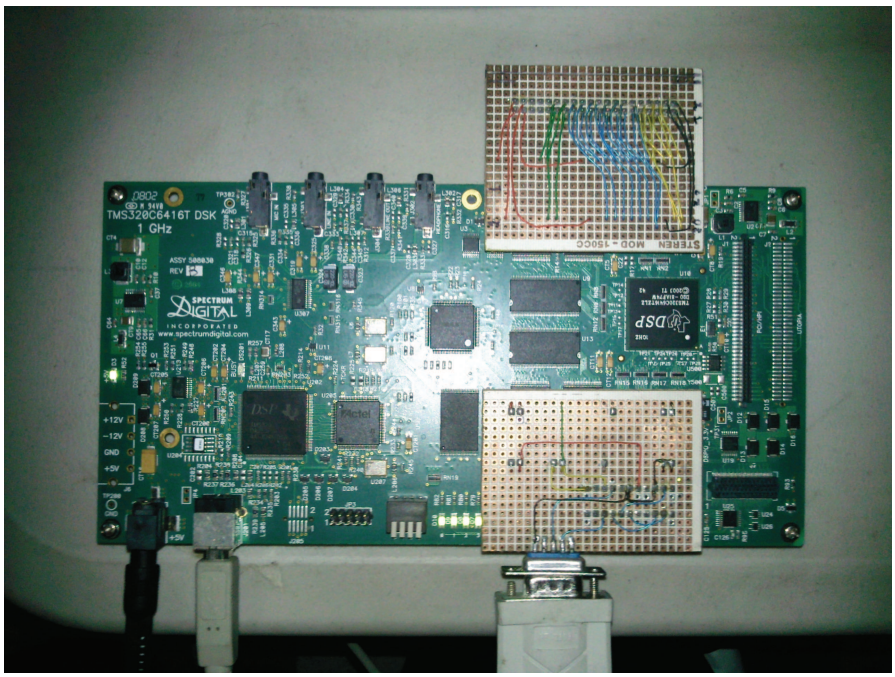

Figura 4.15. TMS320C6416T DSK con interfaz implementada.

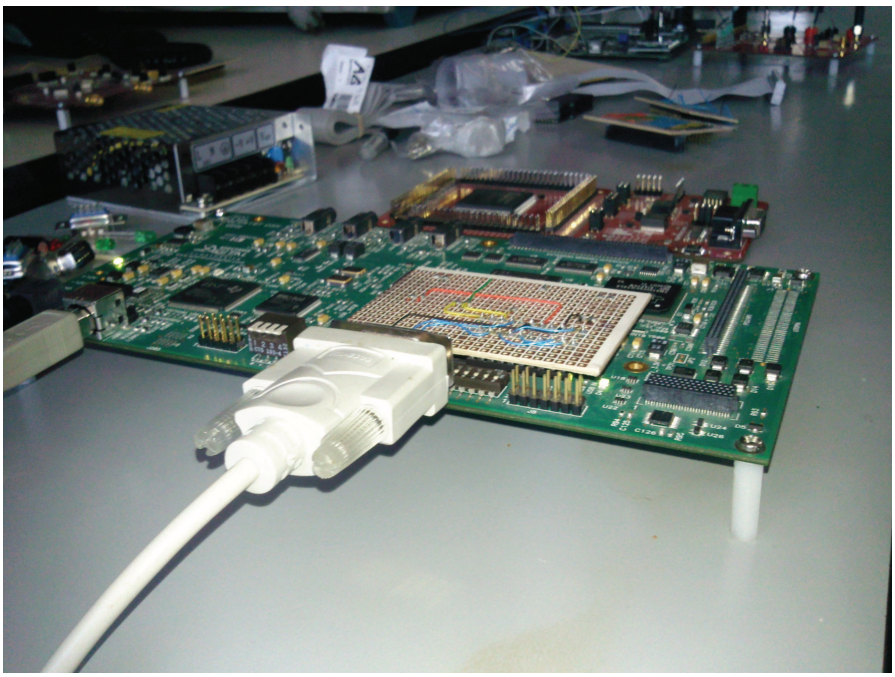

Figura 4.16. Interfaz serial implementada en el módulo TMS320C6416T DSK.

4.7.1. Controlador EDMA

El controlador EDMA transfiere datos entre regiones en el mapeo de memoria sin intervención de la CPU. Es decir esto ocurre en segundo plano de la operación de la CPU. Además maneja todas las transferencias de datos entre el nivel L2 de la memoria caché y los dispositivos periféricos del DSP TMS320C6416T. También este controlador permite mover datos a y desde algún espacio de memoria direccionable, incluyendo la memoria interna (nivel L2 RAM), periféricos y memoria externa. Cuando se inicia una transferencia de datos se genera un evento EDMA el cual es capturado por un registro de eventos. Si ocurren eventos simultáneamente, éstos son procesados por un codificador de eventos. A su vez los parámetros del evento generado son almacenados en la RAM de eventos y procesados por el hardware generador de direcciones el cual direcciona el EMIF o los periféricos para realizar la transacción de lectura o escritura. El controlador EDMA tiene la capacidad de realizar rápidas y eficientes transferencias mediante la aceptación de un rápido acceso directo a memoria (QDMA) desde la CPU. Las transferencias QDMA son mejores en aplicaciones donde se requieren rápidas transferencias de datos. Tales como peticiones de datos en "lazos apretados" de un algoritmo. [3]

[3] *User's Guide, TMS320C6000 Peripherals Reference Guide, SPRU190D, Texas Instruments, February, 2001*

El controlador DMA puede accesar a alguna de las regiones mapeadas en memoria:

- Memoria de datos en el circuito integrado.
- Memoria de programa en el circuito integrado cuando es mapeada dentro del espacio de memoria que está siendo usado como memoria caché.
- Expansión de memoria vía EMIF.

Cada uno de los parámetros del controlador EDMA están organizados en seis palabras de 32 bits o 24 bytes. El acceso a los parámetros. El controlador EDMA comprende:

- Registros para procesamiento de interrupciones y eventos.
- Codificador de eventos.
- Parámetros RAM.
- Hardware para generación de direcciones.

4.7.2. Tipos de transferencias EDMA

El EDMA proporciona dos tipos de transferencias de datos: transferencias de una dimensión (1D) y de dos dimensiones (2D). El número de dimensiones de una transferencia determina la estructura de una trama de datos. Para una transferencia de 1D las tramas de datos están compuestas de un número de elementos individuales. En una transferencia 2D los bloques de datos están compuestos de un número de arreglos, cada uno de los cuales está compuesto de elementos individuales.

Las transferencias en 1D se enfocan a elementos individuales. Cada trama de datos a ser transferida tiene una sola dimensión asociada a ésta, indicando el número de elementos por trama. Los canales EDMA pueden ser configurados para transferir múltiples tramas pero cada una es manejada individualmente. Esta transferencia puede ser considerada como de dos dimensiones pero con la segunda dimensión con un valor de 1.

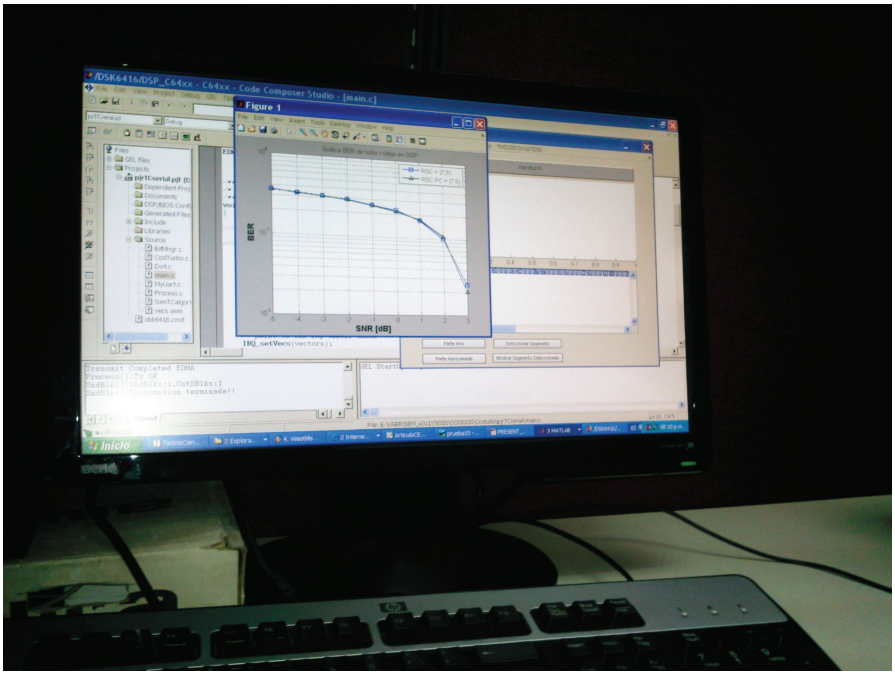

Figura 4.17. Interfaz gráfica con comunicación por el puerto RS-232.

4.8. Ruido gaussiano

Para contaminar los datos de entrada al módulo corrector de errores, se utilizó el algoritmo conocido como "transformada de Box-Muller" descubierto por George Edward Pelham Box and Mervin Edgar Muller en 1958 [48]. Este algoritmo requiere baja complejidad en el hardware para ser implementado. Genera valores seudo-aleatorios en forma aproximada a la función gaussiana. El primer paso del algoritmo consiste en generar dos valores independientes x_1 y x_2 de una variable uniformemente distribuida dentro de un intervalo $[0, 1]$. En el quinto paso las variables y_1 y y_2 son derivadas de la funcion $f(x) = \sqrt{\frac{-2,0*\ln w}{w}}$. Generándose así del resultado de la multiplicación $y = n * f(x)$. El algoritmo 3 presenta los pasos del algoritmo de la transformada "Box-Muller".

En el algoritmo anterior $ranf()$ es una fuente generadora de números aleatorios uniforme distribuidos entre [0,1]; y1, y2 son números aleatorios gaussianos independientes. Este algoritmo fue implementado en el DSP TMS320C6416T y en la PC para contaminar los datos a la entrada del modulo corrector de errores y analizar los resultados obtenidos.

Algoritmo 3 Algoritmo de la transformada de Box-Muller

1: Inicialization
2: **while** $W >= 1{,}0$ **do**
3: Calculamos:
$x1 = 2{,}0 * ranf() - 1{,}0$
$x1 = 2{,}0 * ranf() - 1{,}0$
$W = x1 * x1 + x2 * x2$
$w >= 1{,}0$
4: **end while**
5: $W = sqrt((-2{,}0 * ln(w))/w)$
6: $y1 = x1 * w$
7: $y2 = x2 * w$

4.9. Software para la comunicación con el módulo implementado

Para analizar los resultados generados en el DSP TMS320C6416T, se diseñó una interfaz con el software Matlab vR2009a. Esta interfaz mantiene comunicación con el DSP por medio de la interfaz UART implementada en dicho hardware. Este puerto es independiente al puerto de programación de la tarjeta.

La interfaz diseñada se encarga básicamente de realizar las siguientes funciones:

- Enviar bloques de datos contaminados con ruido Gaussiano al DSP.
- Solicitar ejecución de la simulación Monte Carlo en el DSP utilizando el algoritmo Max-Log-MAP.
- Solicitar ejecución de la simulación Monte Carlo para los codificadores: RSC(7,5), RSC(17,15), RSC(31,27).
- Almacenar en archivo los resultados obtenidos y mostrar dichos resultados en forma gráfica, los cuales fueron generados en el DSP y en la PC.
- Enviar bloques de datos binarios turbo codificados al DSP.
- Leer imágenes binarias, codificarlas en bloques, transmitirlos al DSP y recuperar la imagen decodificada final.

La pantalla principal de la interfaz diseñada contiene un menú que posee cada una de las opciones mencionadas arriba para solicitar al DSP el comando correspondiente. Una vez que el DSP recibe el bloque de datos y el comando, éste inicia con la ejecución de

la prueba correspondiente. Al finalizar la prueba el dispositivo envía los resultados a la interfaz para que sean procesados y mostrados en forma gráfica.

Es importante mencionar que la interfaz contiene los mismos algoritmos que fueron implementados en el DSP pero en formato de punto flotante. Lo cual permite comparar los resultados obtenidos tanto en la PC como en dicho dispositivo y analizar las diferencias.

La interfaz facilita el análisis de los resultados con el objetivo de poder mostrar el comportamiento de los algoritmos que forman parte de la biblioteca del módulo corrector de errores. Entre los resultados mostrados en forma gráfica se encuentran:

- Curva BER generada con los resultados de la simulación Monte Carlo en el DSP.
- Curva BER generada con los resultados de la simulación en la PC.
- Número de errores a la entrada y salida del decodificador en el DSP utilizando al algoritmo Max-Log-MAP.
- Número de errores a la salida del decodificador en el DSP y en la PC.
- Imagen con ruido original, con ruido Gaussiano y decodificada desde el DSP.

En la imagen 4.18 se muestran algunos de los resultados presentados por la interfaz después de ser generados al ejecutar el método Monte Carlo el DSP y en la PC para la transmisión de bloques de bits. Entre dichos resultados se muestra la gráfica de barras de los errores a la entrada y a la salida del decodificador así como los errores finales con base en la relación señal a ruido (SNR).

Al utilizar el esquema de turbo código implementado es importante tomar en cuenta los requerimientos del sistema de comunicación en el cual se necesita utilizar dicho esquema ya que el número de iteraciones así como el factor de normalización en la información extrínseca son parámetros importantes para optimizar y mejorar el desempeño en dicho sistema.

La figura 4.19 muestra los resultados presentados por la interfaz gráfica diseñada después de realizar la decodificación de una imagen en el DSP y ser comparada con la imagen originalmente y la imagen con ruido.

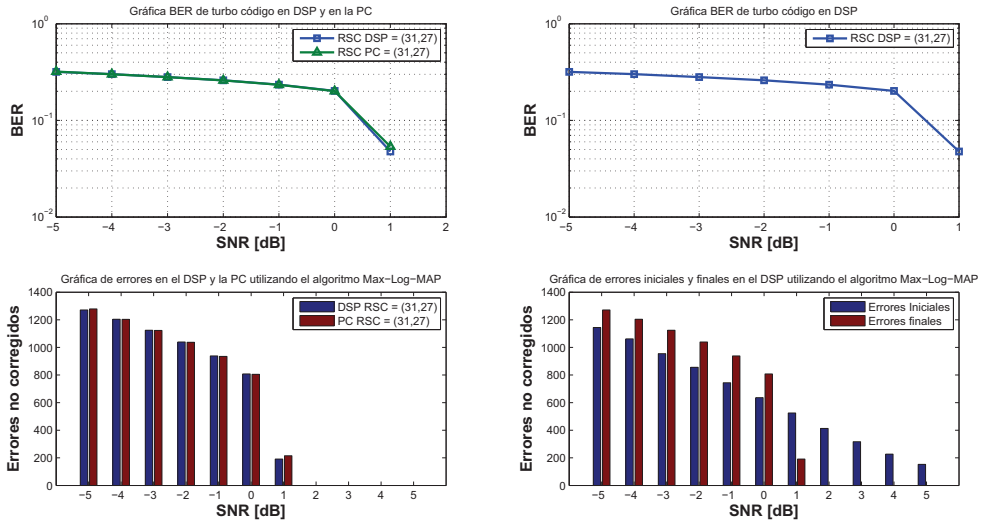

Figura 4.18. Gráficas presentadas por la interfaz diseñada en Matlab.

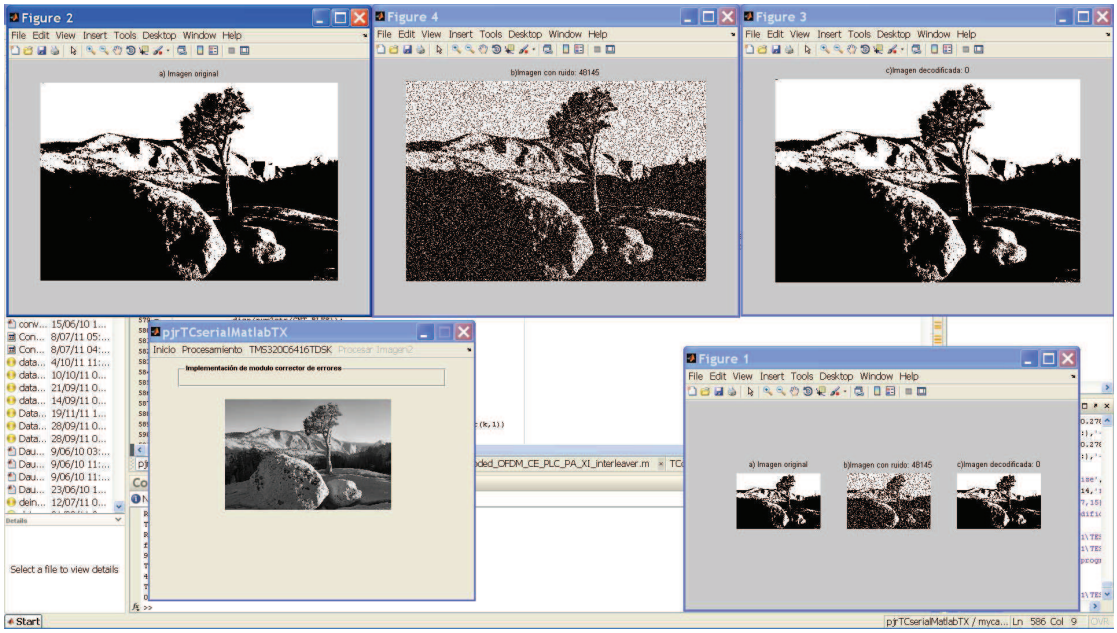

Figura 4.19. Resultados presentados en la interfaz gráfica al decodificar una imagen en el DSP.

4.10. Resumen del capítulo

En el presente capítulo se explicaron a detalle las características del módulo corrector de errores implementado en el DSP TMS320C6416T así como el diagrama de bloques del sistema propuesto. El tipo de turbo codificador puede utilizarse en tres configuraciones diferentes: RSC(7,5), RSC(17,15) y RSC(31,27). Por su parte el decodificador utiliza el algoritmo Max-Log-MAP para la decodificación de la información.

También se describieron los algoritmos para la generación de ruido, los cuales fueron utilizados para contaminar los datos de entrada. El algoritmo implementado en el DSP para la generación de ruido Gaussiano fue "la transformada Box-Muller" ya que requiere pocos recursos en hardware y produce una secuencia seudo-aleatoria aproximada con distribución Gaussiana.

Como se mencionó, la interfaz de usuario diseñada en Matlab envía la información al DSP y a la vez recibe, almacena y presenta los resultados en forma gráfica obtenidos después de la decodificación de la información. También muestra las curvas BER de la simulación Monte Carlo utilizando el algoritmo Max-Log-MAP. Para establecer la comunicación entre la interfaz gráfica y la plataforma de desarrollo TMS320C6416T dsk se implementó una interfaz UART en el DSP y se construyó el hardware requerido para establecer la comunicación por medio de un puerto RS-232.

El módulo corrector de errores implementado en el DSP puede funcionar como transmisor y receptor. Los bloques de datos enviados desde la PC son decodificados en el DSP y la información obtenida es devuelta para su análisis. El desempeño del turbo código implementado se puede incrementar modificando algunos parámetros en los algoritmos programados como son: número de iteraciones, longitud del intercalador, tipo de codificador. En el capítulo 5 se presentarán las pruebas y los resultados obtenidos al realizar las simulaciones con el módulo corrector de errores en el DSP.

Capítulo 5

Pruebas y resultados

En este capítulo se presentarán los resultados obtenidos al realizar simulaciones Monte Carlo con el módulo corrector de errores implementado en el DSP, primeramente en un canal con ruido Gaussiano y después en un canal con ruido Gaussiano e impulsivo. También se presentarán los resultados al decodificar imágenes en formato binario utilizando el sistema de comunicación implementado en un canal con ruido Gaussiano. En la última sección se dará un resumen del capítulo.

5.1. Introducción

En una implementación en hardware de un código corrector de errores es importante distinguir y evaluar algunas características que permiten validar el correcto funcionamiento del módulo implementado para su desempeño dentro de un sistema de comunicación. Entre ellas se encuentran:

- Tasa de código generada a la salida del codificador.
- Tiempo en la decodificación del bloque de datos en el receptor.
- Desempeño del código con variación en el tamaño del intercalador utilizado (para el caso de un turbo código).
- Comportamiento de las curvas BER con variación de la relación señal a ruido SNR.

Para realizar las pruebas, con el módulo corrector de errores implementado, se utilizó una interfaz que establece la comunicación entre la PC y el DSP TMS320C6416T DSK desarrollada en MATLAB debido a su rápido desarrollo, esto con el objetivo de comparar el desempeño de los algoritmos en ambas partes.

Con el módulo implementado en el procesador digital de señales TMS320C6416T se realizaron pruebas de desempeño realizando simulaciones Monte Carlo con aplicación de ruido Gaussiano, codificación y envío de bloques de datos desde la PC al DSP, la turbo codificación de imágenes y envío en bloques de datos al DSP para su decodificación, así como pruebas de tiempo de procesamiento del algoritmo MAX-Log-MAP [5, 17].

En la primera sección se presentan los valores de la información extrínseca con base en el número de iteraciones del algoritmo de decodificación en un canal con ruido Gaussiano. Despúes se presentarán los resultados obtenidos al realizar simulaciones utilizando el metódo Monte Carlo con el turbo código implementado en el DSP y la versión similar programada en la PC. El método para realizar la transmisión, contaminación y recepción de los bits transmitidos fue elegido con base en la literatura [4]. Las longitudes de bloque utilizadas fueron elegidas con base en la referencia anterior y al estándar IEEE P1901 para el caso de las longitudes de 1088 y 4160 bits. En el caso de las pruebas realizadas, los datos enviados son mapeados a formato antipodal $(+1, -1)$ y después son contaminados con ruido Gaussiano de acuerdo al valor de la SNR. Una vez contaminados son tomados por el decodificador y finalmente se comparan el mensaje original y el mensaje a la salida del decodificador y se contabilizan los errores finales.

5.2. Comportamiento del algoritmo Max-Log-MAP en el DSP TMS320C6416T

Como se describió en el capítulo 3, el comportamiento del algoritmo de decodificación Max-Log-MAP depende de varios parámetros, entre ellos, la longitud del bloque de datos a decodificar, ya que la convergencia en el algoritmo es un factor importante para alcanzar un incremento significativo en el desempeño del mismo.

Con el módulo de corrección de errores implementado, se realizaron pruebas analizando la información extrínseca proporcionada por los dos decodificadores. A continuación se muestran los resultados de la decodificación de los datos de entrada recibidos. Estos datos se contaminaron con ruido gaussiano utilizando el algoritmo de la transformada de Box-Muller descrito en el capítulo 4. Los parámetros utilizados para la realización de estos experimentos se muestran en la tabla 5.1. Para el caso del codificador RSC(31,27) se tomaron 20 datos de entrada para poder observar con mayor facilidad el comportamiento del algoritmo con base en el número de la iteración actual. El codificador utilizado en la figura 5.1 tiene 16 estados y cuatro registros de memoria (es decir con estructura RSC(31,27)). La tabla 5.2 muestra el intercalador y des-intercalador utilizados en el codificador y el decodificador.

Tabla 5.1. Parámetros del módulo para resultados mostrados en la figura 5.1

Parámetro	**valor**
Tipo de intercalador	S-random
Tamaño de intercalador	20
Número de estados	16
Longitud de memoria	4
Número de iteraciones	8
Tipo de ruido	AWGN
Relación SNR	1.2 dB
Tasa de código	1/2
Número total de bloques transmitidos	1
Total de datos transmitidos	20

Tabla 5.2. Intercalador y des-intercalador para resultados mostrados en la figura 5.2

Intercalador	**Des-intercalador**
1	18
5	0
8	10
15	14
12	6
19	1
4	11
9	16
13	2
17	7
2	12
6	17
10	4
18	8
3	15
14	3
7	19
11	9
0	13
16	5

La figura 5.1 muestra los valores de la información extrínseca para cada dato, a la entrada, recibido. Como se puede ver al incrementar el número de iteración, la decisión para elegir el valor del dato recibido es más confiable y con ello se logra obtener el valor más aproximado al valor original. Para inicializar el codificador al estado cero, los últimos cuatro datos del bloque son puestos a cero en la codificación.

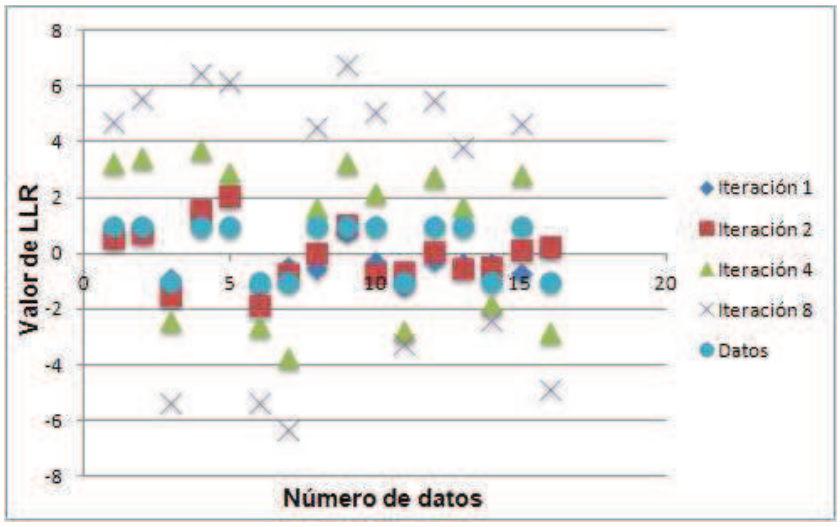

Figura 5.1. Valores de la información extrínseca LLR en el codificador RSC(31,27) para diferente número de iteraciones.

Los resultados de los valores de la información extrínseca LLR para el codificador con estructura RSC(17,15) se muestran en la figura 5.2 y 5.3 para 0 y 1 dB respectivamente.

Se puede observar que al inicio, la información resultante calculada (LLR) presenta una rápida aproximación para poder encontrar el valor de los símbolos de la información originalmente enviados, sin embargo, al incrementar el número de iteraciones, los valores de la información LLR para cada símbolo se incrementan y son más confiables para elegir los valores correctos de cada bit de la información. Ya que en este caso la longitud del bloque de datos es pequeña, la convergencia del algoritmo es muy rápida. La figura 5.2 muestra la comparación de los valores de la información extrínseca para una relación señal a ruido de 1 dB.

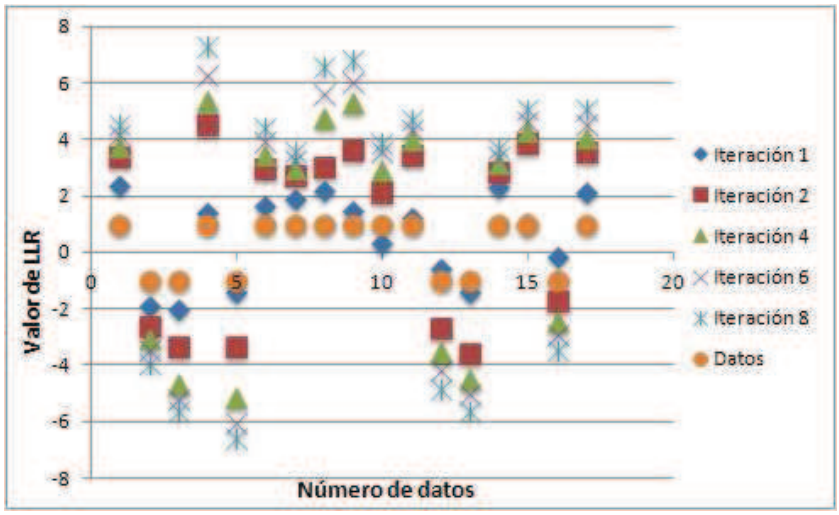

Figura 5.2. Valores de la información extrínseca LLR en el codificador RSC(17,15) con base en el número de iteraciones.

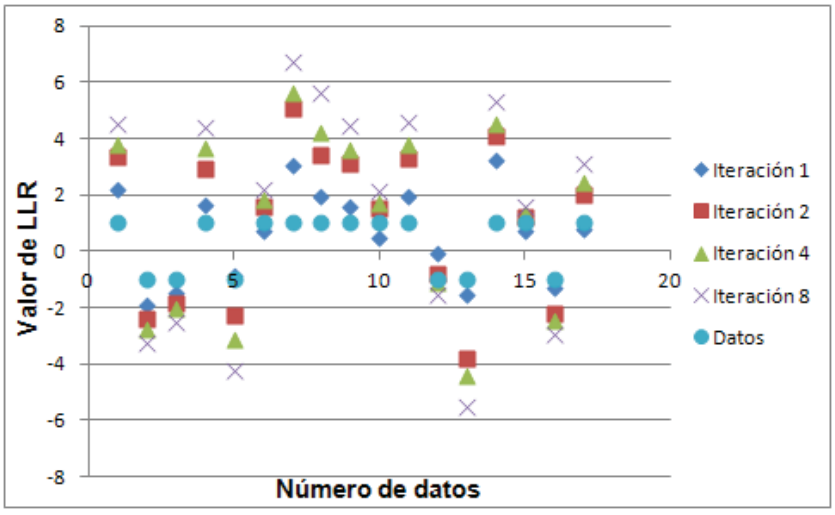

Figura 5.3. Valores de la información extrínseca LLR para diferentes iteraciones en el codificador RSC(17,15).

5.3. Tiempos de ejecución del turbo código implementado

Los tiempos de ejecución de la codificación y decodificación en ciclos de CPU para diferente tamaño de bloque de datos del turbo código implementado en el DSP se muestran en las tablas 5.3 y 5.4 respectivamente. La tasa de código utilizada fue de 1/2. A su vez la tabla 5.5 presenta la cantidad de memoria utilizada por la biblioteca de códigos y arreglos para el almacenamiento de variables. Tomando el número de ciclos utilizados para procesar 4000 bits se tiene una tasa de trasferencia de datos de aproximadamente 1.4 Mps.

Tabla 5.3. Tiempos de ejecución de la codificación de un bloque de datos en el codificador RSC(31,27)

Longitud de bloque (bits)	**RSC(7,5)**	**RSC(17,15)**	**RSC(31,27)**	**mseg,** $F_{clock} = 1$ **GHz**
500	702590	703675	703267	0.703 mseg
1000	1404053	1405660	1404758	1.4 mseg
2000	2808371	2808323	2807382	2.8 mseg
4000	5613633	5615541	5615203	5.6 mseg

Tabla 5.4. Tiempos de ejecución de la decodificación de un bloque de datos en el codificador RSC(7,5) para una iteración

Longitud de bloque (bits)	**Num. Ciclos CPU**	**mseg,** $F_{clock} = 1$ **GHz**
500	6607258	6.6 mseg
1000	13215194	13.2 mseg
2000	26437584	26.4 mseg
4000	52871964	52.9 mseg

Tabla 5.5. Cantidad de memoria utilizada por la biblioteca de códigos implementada

Memoria	**Cantidad (Hex)**	**Cantidad (Dec)**
Memoria IRAM	0x483fc	295,932
Memoria SRAM	0x4553e0	2,543,456

5.4. Pruebas con simulaciones Monte Carlo en el DSP TMS320C6416T

Para obtener las curvas de la tasa de errores binarios (BER) del turbo código implementado, se realizaron simulaciones Monte Carlo tanto en el DSP como en la PC, con el objetivo de verificar el comportamiento de los algoritmos y comparar las curvas BER generadas por ambas partes.

La tabla 5.6 muestra los parámetros utilizados para realizar la simulación Monte Carlo con un codificador RSC(31,27) de 16 estados y 4 registros de memoria. El tipo de intercalador utilizado fue S-random con una longitud de 4000 y S = 40.

El escenario utilizado para realizar este experimento fue un canal contaminado con ruido Gaussiano, el cual, fue generado por el algoritmo de la transformada de "Box-Muller" descrito en el capítulo 4. Dicha simulación consistió en la transmisión de 300000 bits codificados en bloques de 4000 bits de información útil. Dado que se utilizó la técnica de perforación, por cada bit de información se agregó un bit de información redundante (tasa de código = 1/2).

Para obtener la curva BER se modificó la SNR en un rango de -5 dB a 5dB con un incremento equivalente a 1 dB. La tabla 5.6 contiene los parámetros utilizados en el módulo corrector.

Tabla 5.6. Parámetros del módulo para resultados mostrados en la figura 5.4

Parámetro	**valor**
Tipo de intercalador	S-random
Tamaño de intercalador (bits)	4000
Número de estados	16
Longitud de memoria	4
Número de iteraciones	1,2,3,4
Tipo de ruido	AWGN
Relación SNR	-5,5 dB
Tasa de código	1/2
Número total de bloques transmitidos	75
Total de datos transmitidos	300000

En la figura 5.4 se puede observar que con una iteración en el algoritmo de decodificación, el desempeño es un relativamente bajo, comparado con un número mayor de iteración, ya que al tener más de tres iteraciones, la cantidad de errores disminuye

considerablemente y a partir de 1 dB se tiene un umbral de convergencia en el algoritmo de decodificación.

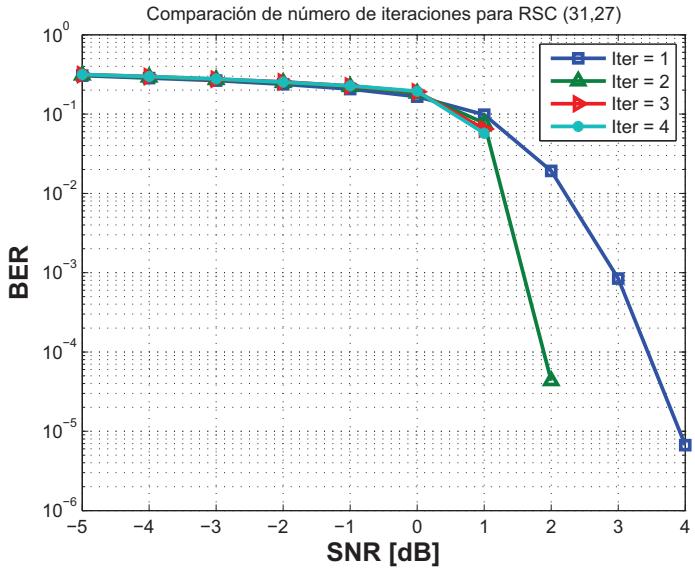

Figura 5.4. Curvas BER para diferentes iteraciones con un codificador RSC(31,27).

Para analizar el desempeño del módulo corrector implementado en el DSP, se comparó la cantidad de errores a la entrada y a la salida del decodificador en el DSP y en la PC, utilizando un codificador RSC(31,27), una longitud del intercalador de 4000, y 8 iteraciones.

Se puede observar en la tabla 5.7 que para una SNR menor a 1 dB aún se tienen errores a la salida del decodificador ya que no se alcanza el valor de umbral de convergencia del decodificador. A partir de este valor podemos observar que la cantidad de errores a la salida del decodificador es cero.

La figura 5.5 muestra la cantidad de errores a la salida del decodificador en la PC y en el DSP. Como se puede observar, existe una pequeña diferencia entre la cantidad de errores a la entrada de ambos codificadores, esto es debido a que el ruido fue generado en forma seudoaleatoria (ruido Guassiano). La cantidad de errores a la salida del decodificador es cercana en ambos casos a pesar de que la implementación en el DSP es en formato de punto fijo Q15.

La figura 5.6 muestra los errores a la entrada y salida del decodificador implementado en el DSP. Como se puede observar, para una SNR menor a 1 dB, los errores a la salida del

Tabla 5.7. Errores a la entrada y salida del decodificador en el DSP y en la PC utilizando un codificador RSC(31,27).

SNR (dB)	Input DSP	Out DSP	Input PC	Out PC
-5	1153	1275	1150	1278
-4	1058	1201	1059	1199
-3	956	1120	959	1124
-2	848	1031	850	1028
-1	742	932	749	936
0	636	802	630	800
1	526	174	526	194
2	415	0	416	0
3	318	0	318	0
4	228	0	229	0
5	153	0	153	0

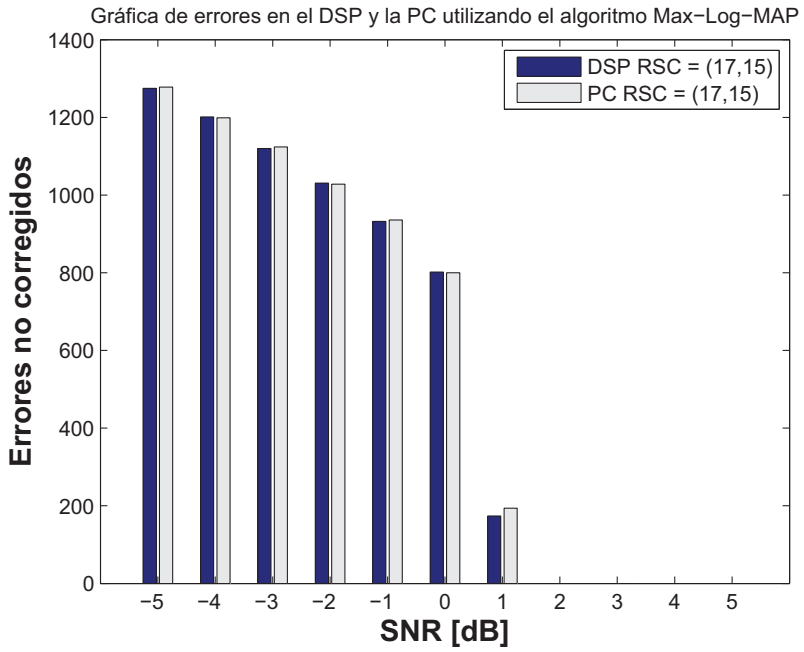

Figura 5.5. Errores a la salida del decodificador en la PC y el DSP.

decodificador son incluso mayores que los errores a la entrada del mismo. Esto se debe a que el algoritmo de decodificación aun no alcanza su umbral de convergencia, sin embargo, al incrementar la SNR la cantidad de errores disminuye hasta ser cero.

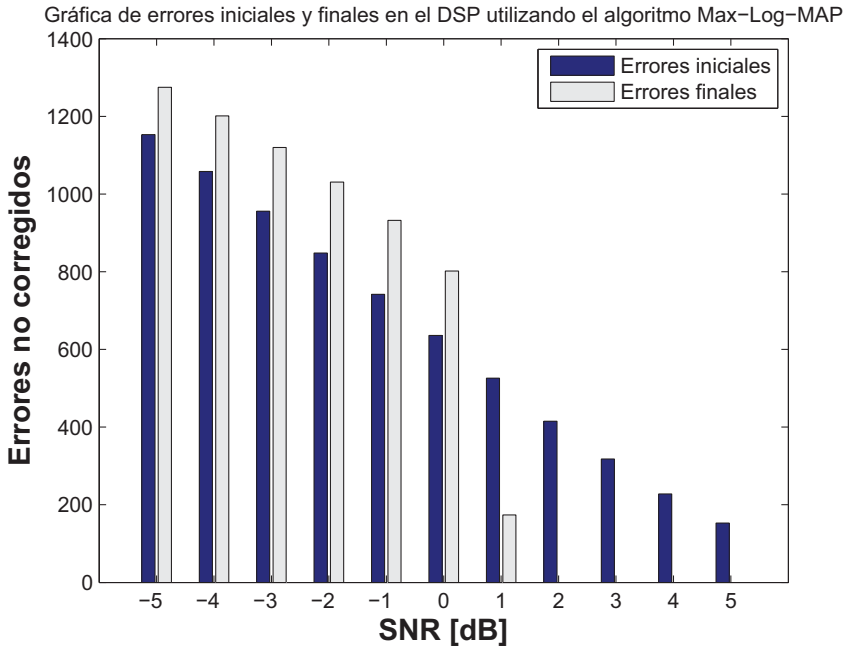

Figura 5.6. Errores a la entrada y a la salida del decodificador en el DSP.

La tabla 5.8 contiene los parámetros utilizados en el turbo código implementado al utilizar el codificador RSC(17,15). El número de estados del codificador es 8 y 3 registros de memoria. El tamaño de bloque como en el caso anterior fue de 4000 bits, una tasa de código de 1/2.

En la figura 5.7 se muestra la curva BER obtenida en el DSP y en la PC con los parámetros indicados anteriormente. Es importante mencionar que un valor aproximado a 1dB representa el umbral en el cual el turbo código genera un incremento considerable en la cantidad de errores corregidos a la salida del decodificador.

La gráfica de barras 5.8 presenta los errores a la entrada y a la salida del decodificador en el DSP. Se puede observar que en los valores iniciales de la relación SNR la cantidad de errores a la salida del decodificador es mayor que la de entrada. La diferencia entre estas cantidades va disminuyendo conforme se incrementa el valor de la SNR hasta que el

Tabla 5.8. Parámetros del módulo para resultados mostrados en la figura 5.7

Parámetro	**valor**
Tipo de intercalador	S-random
Tamaño de intercalador (bits)	4,000
Estados	8
Longitud de memoria	3
Iteraciones	8
Tipo de ruido	AWGN
Relación SNR	-5 , 5 dB
Tasa de código	1/2
Bloques transmitidos	75
Bits transmitidos	300,000

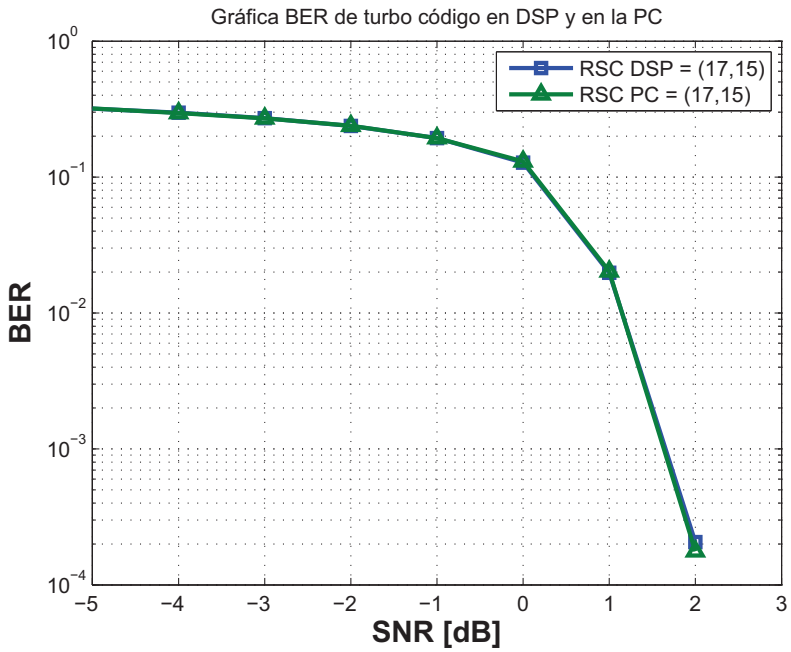

Figura 5.7. Curva BER obtenida de la simulación Monte Carlo en el DSP y en la PC.

algoritmo logra converger, corrigiendo así mayor cantidad de errores hasta lograr recuperar la información originalmente transmitida.

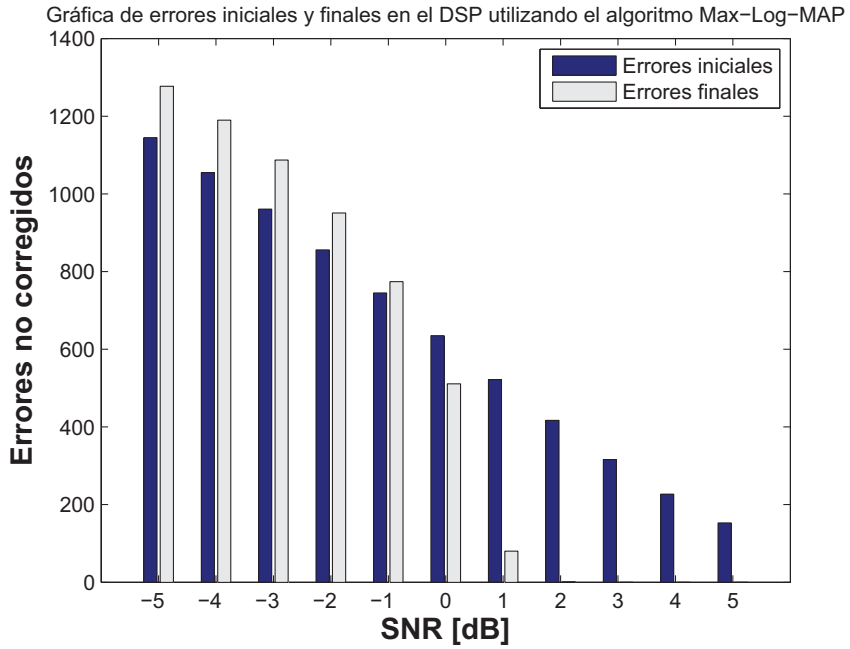

Figura 5.8. Errores iniciales y finales en el decodificador con parámetros de la tabla 5.8.

En la gráfica 5.9 se presentan los errores finales después de utilizar el algoritmo Max-Log-MAP implementado en el DSP y también con la versión similar del decodificador en la PC. Se puede observar que el desempeño de ambas versiones del decodificador es similar. La representación utilizada para las variables en la versión de turbo código utilizada en la PC fue en punto flotante a diferencia del DSP (En el que se utilizó una representación en punto fijo Q15). En la tabla 5.9 se presentan la cantidad de errores a la entrada y a la salida del decodificador en el DSP y en la PC para diferentes valores de SNR.

La tabla 5.9 presenta los valores de la cantidad de errores a la entrada y salida del decodificador implementado en el DSP y el programado en la PC. Se puede observar que el desempeño es muy cercano y a partir del valor de SNR de 1 dB la cantidad de errores a la salida disminuye considerablemente hasta alcanzar el valor de cero.

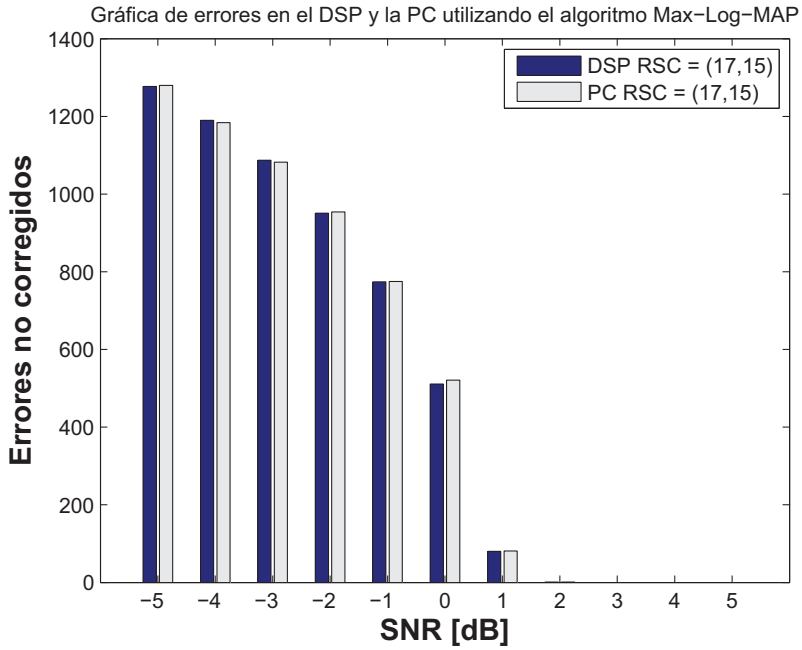

Figura 5.9. Errores en el decodificador en el DSP y en la PC utilizando los parámetros de la tabla 5.8.

Tabla 5.9. Errores a la entrada y salida del decodificador en el DSP y en la PC.

SNR (dB)	Input DSP	Out DSP	Input PC	Out PC
-5	1145	1277	1152	1280
-4	1055	1190	1057	1184
-3	961	1087	956	1082
-2	856	951	855	954
-1	745	774	743	775
0	635	511	637	521
1	522	80	522	81
2	417	1	417	1
3	316	0	317	0
4	227	0	226	0
5	153	0	152	0

5.5. Desempeño del turbo código con diferentes esquemas del codificador

Como se describió en el capítulo 4, la biblioteca del módulo corrector de errores implementado puede tener tres diferentes configuraciones en el codificador. Estas configuraciones varían de acuerdo a la matriz generadora, es decir en el número de estados así como el número de registros de memoria del codificador.

Se realizaron pruebas con el objetivo de comparar las curvas BER de cada configuración en el codificador para un canal con ruido gaussiano. La tabla 5.10 contiene los parámetros utilizados para la realización del experimento

Tabla 5.10. Parámetros del módulo para resultados de las curvas en la figura 5.10

Parámetro	**valor**
Tipo de intercalador	S-random
Tamaño de intercalador (bits)	4,000
Estados	4,8,16
Longitud de memoria	2,3,4
Iteraciones	8
Tipo de ruido	AWGN
Relación SNR	-5 , 5 dB
Tasa de código	1/2
Bloques transmitidos	75
Bits transmitidos	300,000

La figura 5.10 presenta las curvas BER de los tres codificadores implementados en el DSP. Como se puede observar en la gráfica, al incrementar el número de estados en el codificador y el número de registros de memoria, se incrementa el desempeño del turbo código en el DSP. Esto se debe a que se tienen en cuenta mayor cantidad de estados anteriores para poder recuperar el valor original del dato recibido. Es importante mencionar que aunque se mejora el desempeño en el turbo código, se incrementa la latencia y los recursos requeridos en memoria para el almacenamiento de las probabilidades necesarias en la decisión dura de cada uno de los bits recibidos.

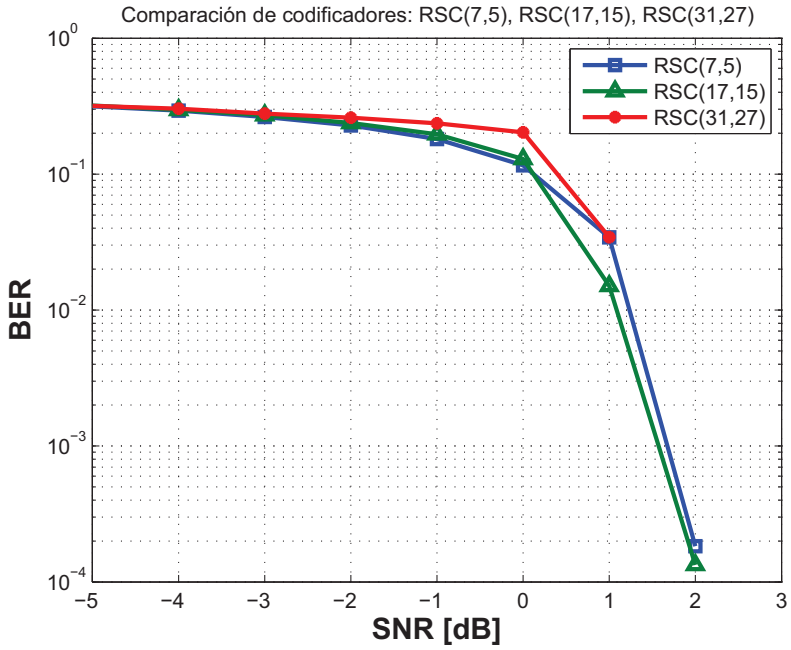

Figura 5.10. Comparación de curvas BER de los codificadores: RSC(7,5), RSC(17,15) y RSC(31,27).

Tabla 5.11. Comparación de los errores a la entrada y salida del decodificador en el DSP al utilizar tres tipos de codificadores: RSC(7,5), RSC(17,15), RSC(31,27).

SNR (dB)	Input DSP	Out DSP	Input DSP	Out DSP	Input DSP	Out DSP
-5	1147	1260	1150	1271	1144	1271
-4	1065	1171	1058	1185	1062	1204
-3	961	1052	959	1080	955	1124
-2	863	913	855	953	856	1039
-1	748	725	749	787	744	938
0	630	465	634	514	636	808
1	526	138	523	77	525	174
2	410	1	416	1	413	0
3	315	0	319	0	317	0
4	227	0	228	0	227	0
5	151	0	149	0	153	0

5.6. Desempeño del algoritmo Max-Log-MAP con diferente número de iteraciones en el DSP TMS320C6416T

El algoritmo de decodificación utiliza la información a la salida de los dos decodificadores componentes para efectuar finalmente la decisión dura de cada uno de los datos recibidos. Con el incremento en el número de iteraciones, es posible incrementar considerablemente la convergencia del algoritmo y por consecuencia el número de errores corregidos. Por lo cual se realizaron experimentos con dos esquemas del codificador: RSC(17,15) y RSC(31,27) utilizando el algoritmo Max-Log-MAP en el decodificador. De esta forma se obtuvieron las curvas BER por cada simulación utilizando un número fijo de bits totales transmitidos así como un número máximo de iteraciones en el decodificador. La tabla 5.12 presenta los parámetros utilizados para realizar la simulación Monte Carlo en el DSP para diferente número de iteraciones.

Tabla 5.12. Parámetros del módulo corrector para resultados mostrados en la figura 5.11

Parámetro	**valor**
Tipo de intercalador	S-random
Longitud de intercalador (bits)	4,000
Estados	16
Memoria	4
Iteraciones	1,2,3,4,6,8
Tipo de ruido	AWGN
Relación SNR	-5 , 5 dB
Tasa de código	1/2
Bloques transmitidos	75
Bits transmitidos	300,000

En la figura 5.11 se presentan las curvas BER para el codificador RSC(31,27) con diferente número de iteraciones. Como se puede observar en la primera iteración el BER es relativamente bajo, sin embargo, después de la segunda iteración el desempeño mejora considerablemente y en las siguientes iteraciones se tiene un incremento en el desempeño y una menor probabilidad de error.

Para obtener los resultados que se muestran en la gráfica 5.12 se utilizaron los parámetros de la tabla 5.13 para configurar el módulo de corrección de errores.

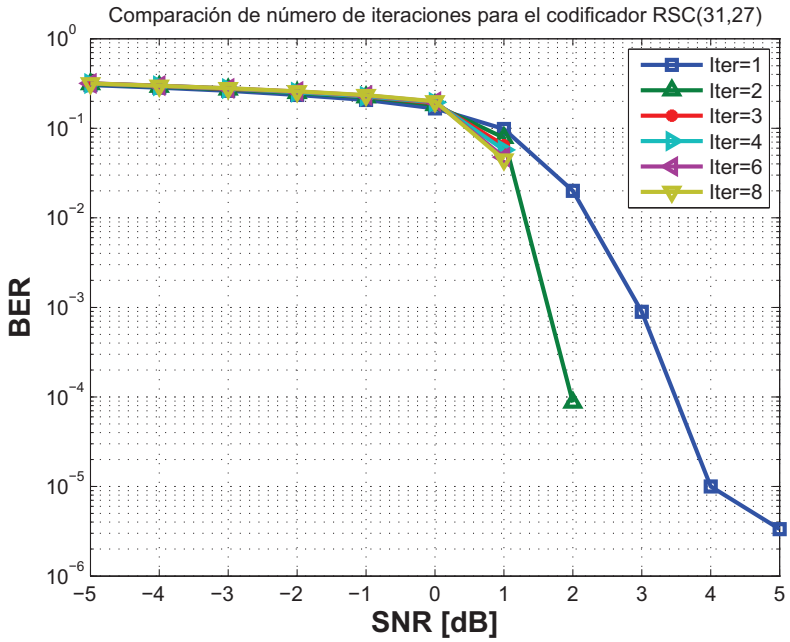

Figura 5.11. Curvas BER para diferente número de iteraciones del codificador RSC(31,27).

Tabla 5.13. Parámetros del módulo para resultados mostrados en la figura 5.12

Parámetro	**valor**
Tipo de intercalador	seudo-aleatorio
Longitud de intercalador (bits)	4,000
Estados	8
Longitud de memoria	3
Iteraciones	1,2,4,6,8
Tipo de ruido	AWGN
Relación SNR	-5 , 5 dB
Tasa de código	1/2
Bloques transmitidos	75
Bits transmitidos	300,000

Como se puede observar, al incrementar el número de iteraciones se obtiene una tasa más baja de errores. Para este caso, el desempeño es menor que en el caso del codificador RSC(31,27).

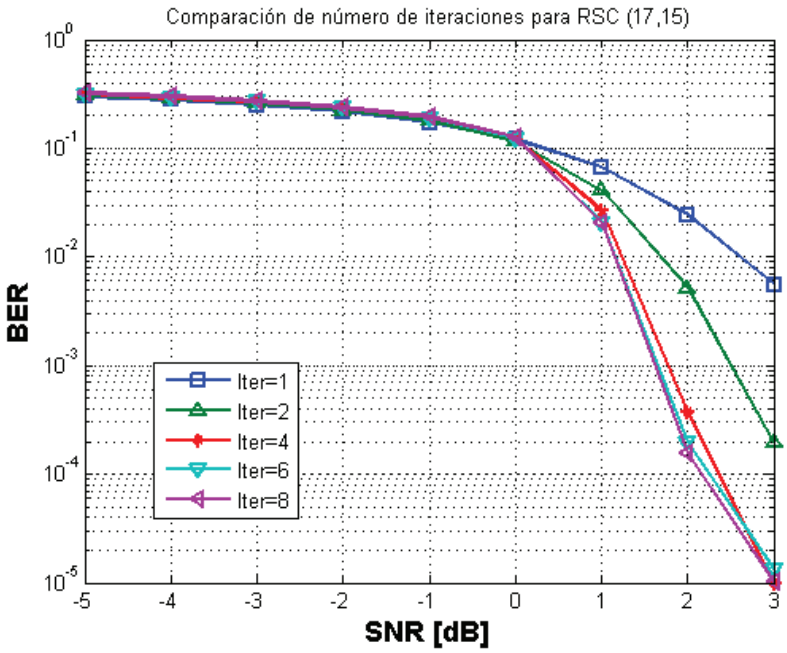

Figura 5.12. Gráfica BER del codificador RSC(17,15) con diferente número de iteraciones en un canal con ruido Gaussiano.

5.7. Simulaciones del algoritmo Max-Log-MAP variando la longitud del intercalador

Una de las características en la turbo codificación de un bloque de datos es el grado de aleatoriedad del orden en los datos que forman la trama. Ésta es incrementada al aumentar la longitud del intercalador y como consecuencia el tamaño total del bloque de datos a enviar.

Con el objetivo de analizar el comportamiento del turbo código implementado en el DSP con diferentes longitudes de bloque de datos, se realizaron pruebas con diferentes longitudes del intercalador y des-intercalador utilizando la misma matriz generadora en el codificador en un canal con ruido gaussiano. Para el experimento se utilizaron longitudes del intercalador de: 1000, 2000, 3000, 4000 bits. Para aplicaciones prácticas es importante procesar bloques con longitudes que permitan el procesamiento y almacenamiento en los

dispositivos de hardware existentes.

La figura 5.13 muestra las curvas BER para el codificador RSC(17,15) utilizando diferentes valores de longitud del intercalador en un canal con ruido Gaussiano.

Tabla 5.14. Parámetros del módulo para resultados mostrados en la figura 5.13

Parámetro	**valor**
Tipo de intercalador	seudo-aleatorio
Longitud de intercalador (bits)	1000,2000,3000,4000
Estados	8
Longitud de memoria	3
Iteraciones	8
Tipo de ruido	AWGN
Relación SNR	-5 , 5 dB
Tasa de código	1/2
Bloques transmitidos	300,150,100,75
Bits transmitidos	300,000

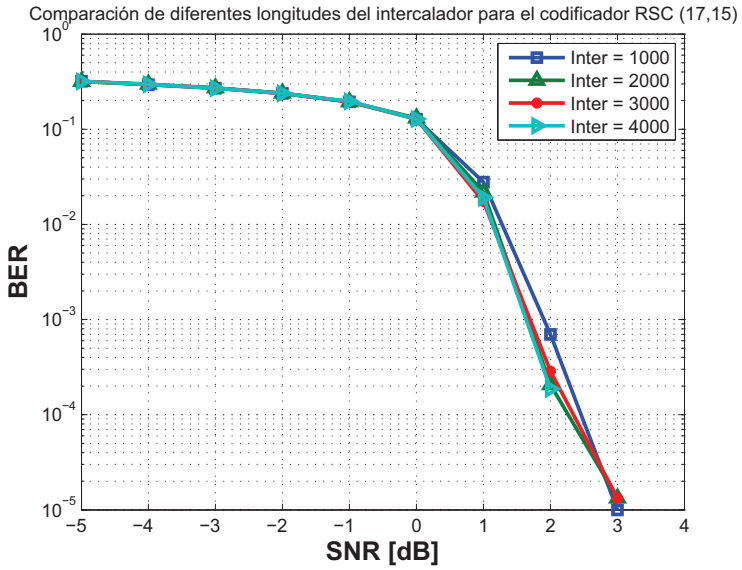

Figura 5.13. Curvas BER para el codificador RSC(17,15) con diferente longitud en el intercalador.

5.8. Simulaciones del turbo codificador en el DSP TMS320C6416T (Longitud de bloque 4160 bits)

Se realizaron pruebas utilizando el codificador RSC(17,15) implementado en el DSP. Para ello se utilizó una longitud de bloque de 4160 bits y diferente número de iteraciones en un canal con ruido gaussiano. Los parámetros utilizados se muestran en la tabla 5.15 y la curva BER obtenida se puede obervar en la figura 5.14.

Tabla 5.15. Parámetros del módulo corrector para resultados mostrados en la figura 5.14

Parámetro	**valor**
Tipo de intercalador	S-random
Longitud de intercalador (bits)	4,160
Estados	8
Memoria	3
Iteraciones	1,2,4,6
Tipo de ruido	AWGN
Relación SNR	-5 , 5 dB
Tasa de código	1/2
Bloques transmitidos	72
Bits transmitidos	300,000

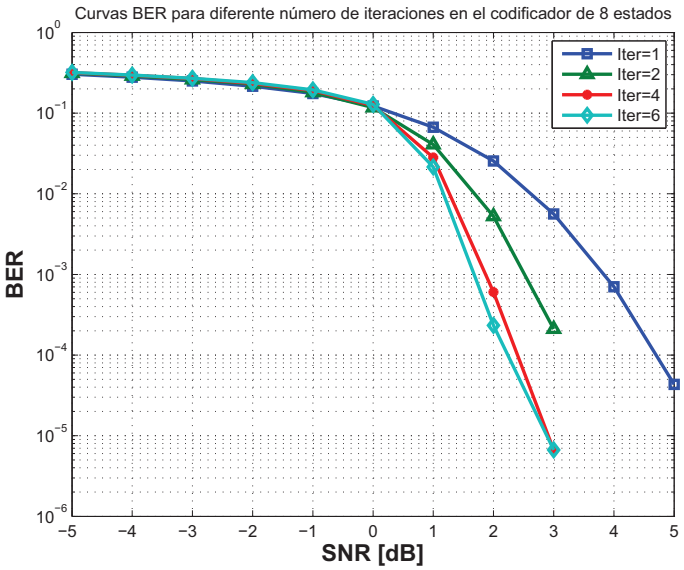

Figura 5.14. Curvas BER del codificador RSC(17, 15) en el DSP al utilizar una longitud de bloque de 4160 bits similar al estándar IEEE 1901 en un canal con ruido Gaussiano.

Además, se realizaron pruebas utilizando como criterio de parada 8 iteraciones en el decodificador y parámetros similares a los utilizados en la tabla 5.15 con modulación BPSK. En la figura 5.15 se muestra la curva BER obtenida con el turbo codificador al utilizar una longitud de 4160 bits en el intercalador.

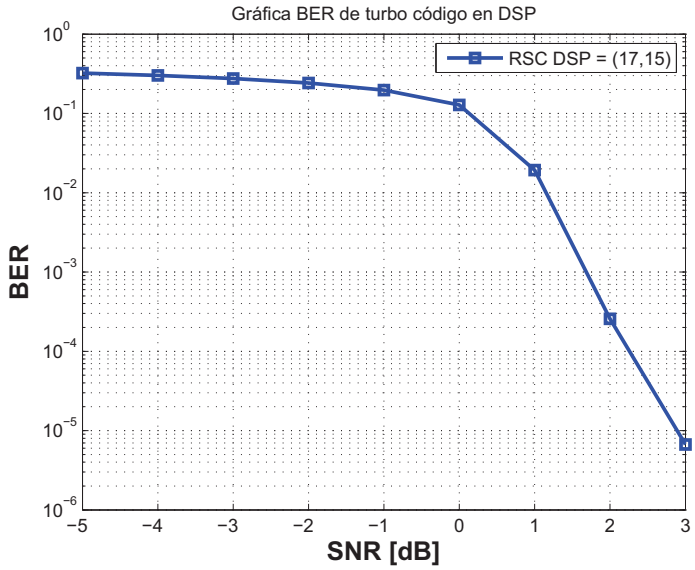

Figura 5.15. Curva BER del turbo codificador RSC(17,15) en el DSP con una longitud de intercalador de 4160 bits en un canal con ruido Gaussiano.

En la figura 5.16 se muestran los errores a la entrada y a la salida del decodificador en el DSP. Se puede observar un incremento considerable en la cantidad de errores corregidos que a partir de 1 dB en el nivel de la SNR.

En la tabla 5.16 se presentan la cantidad de errores a la entrada y a la salida del turbo decodificador con la versión implementada en el DSP y en la PC para diferentes valores de SNR.

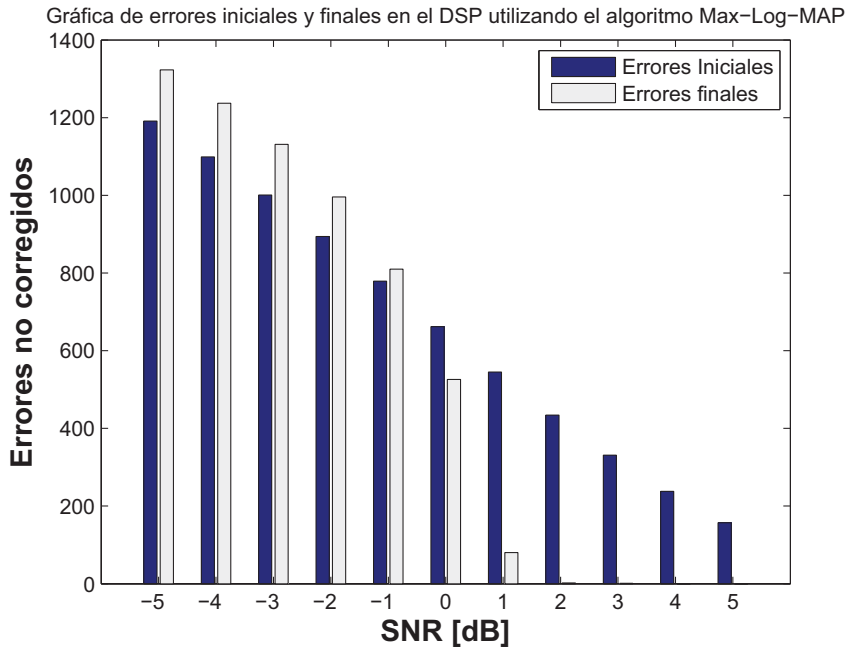

Figura 5.16. Errores a la entrada y a la salida del decodificador RSC(17,15) con longitud de intercalador de 4160 bits.

Tabla 5.16. Errores a la entrada y salida del decodificador en el DSP y en la PC utilizando un codificador RSC(17,15) con longitud del intercalador de 4160 bits.

SNR (dB)	Input DSP	Out DSP	Input PC	Out PC
-5	1191	1323	1199	1329
-4	1099	1237	1096	1228
-3	1001	1131	1000	1125
-2	894	996	893	999
-1	779	810	774	806
0	662	526	660	533
1	545	80	544	77
2	434	2	433	1
3	331	1	329	0
4	238	0	236	0
5	157	0	160	0

5.9. Pruebas del algoritmo Max-Log-MAP con ruido impulsivo

El ruido impulsivo es muy común en los canales de comunicación donde la señal de información comparte dicho medio con otras señales y en donde se ve perturbada por interferencias externas. Para realizar las pruebas con ruido impulsivo se realizaron simulaciones Monte Carlo con bloques de datos seudo-aleatorios contaminados con ruido generado por el simulador del canal PLC diseñado en [45]. Esto para poder tener las condiciones en un canal PLC y con variación de la relación señal a ruido.

En la figura 5.17 se muestra el diagrama de bloques del transreceptor OFDM utilizado para realizar los experimentos con ruido para el módulo corrector de errores. Este esquema incluye: mapeo QAM, buffer serial/paralelo, modulación OFDM y prefijo cíclico. Para complementar el bloque decodificador, el receptor incluye: eliminación del prefijo cíclico, sincronización, demodulación OFDM, estimación de canal, buffer paralelo/serial, y demapeo QAM.

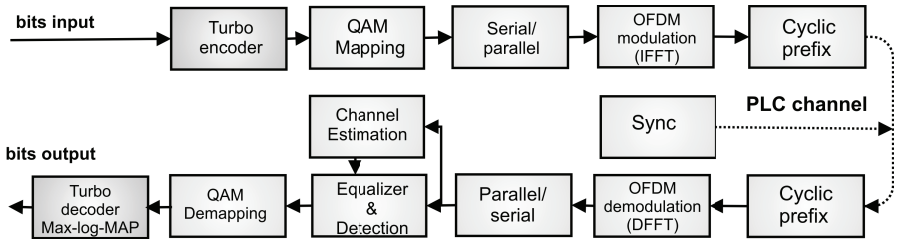

Figura 5.17. Transreceptor OFDM utilizado para obtener las muestras con ruido impulsivo.

El primer experimento se realizó utilizando la versión de turbo código programada en la PC agregada al esquema transreceptor OFDM, y se utilizaron los tres turbo codificadores: RSC(7,5), RSC(17,15) y RSC(31,27). Esto con el objetivo de comparar su desempeño en un canal PLC con 10 impulsos por ciclo. La tabla 5.17 presenta los parámetros seleccionados para obtener las curvas de la figura 5.18.

Como se puede observar en la figura 5.18, el desempeño del codificador RSC(31,27) es mejor que los codificadores (RSC(7,5) y RSC(17,15)) y a partir de 2 dB de la SNR no se presentaron errores, a diferencia de los otros dos casos. En este experimento, el canal PLC se contiene en su mayoría ruido Gaussiano y solo 10 impulsos de ruido impulsivo por ciclo.

El segundo experimento consistió en comparar el comportamiento del codificador RSC(31,27) con diferentes longitudes del intercalador. La tabla 5.18 contiene los parámetros

Tabla 5.17. Especificación general para el transreceptor OFDM y el turbo código para resultados de la figura 5.18.

Parámetros	Valor
Frecuencia del sistema: $F_s = B$	30 MHz
Tiempo de muestreo: $t_s = 1/f_s$	33.3 nseg
Subportadoras: N	64
Modulación portadora:	2QAM
Tasa de código:	1/2
Longitud de intercalador (bits):	4000
Codificadores:	(7,5),(17,15),(31,27)

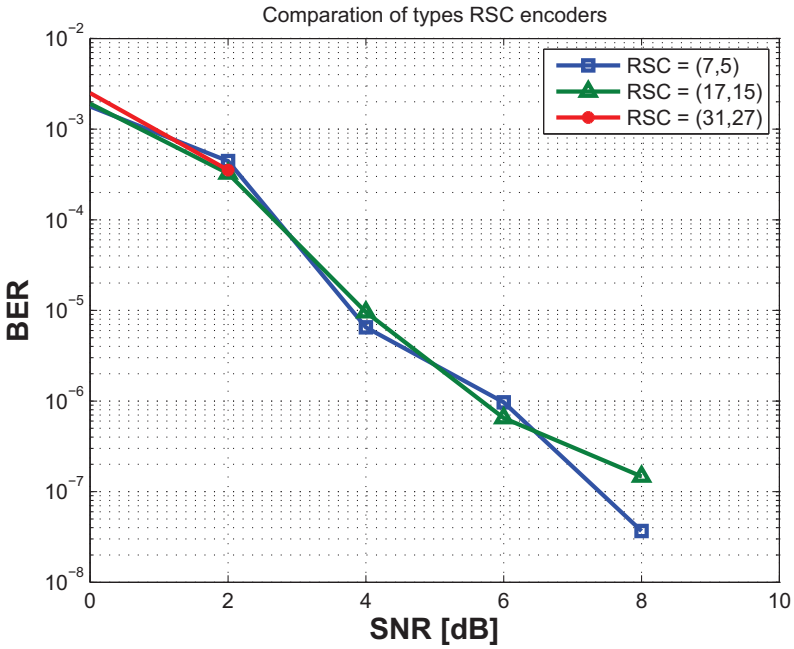

Figura 5.18. Curvas BER generadas en un canal ciclo-estacionario con ruido Gaussiano e impulsivo.

utilizados en el sistema transreceptor OFDM para la realización de las pruebas con el módulo corrector. Como se puede ver se utilizó una longitud de 4000 bits en el intercalador y una tasa de código de 1/2.

Tabla 5.18. Especificación general para el transreceptor OFDM y el turbo código para resultados de la figura 5.19.

Parámetros	**Valor**
Frecuencia del sistema: $F_s = B$	30 MHz
Tiempo de muestreo: $t_s = 1/f_s$	33.3 nseg
Subportadoras: N	64
Modulación portadora:	2QAM
Tasa de código:	1/2
Longitud de intercalador (bits):	4000

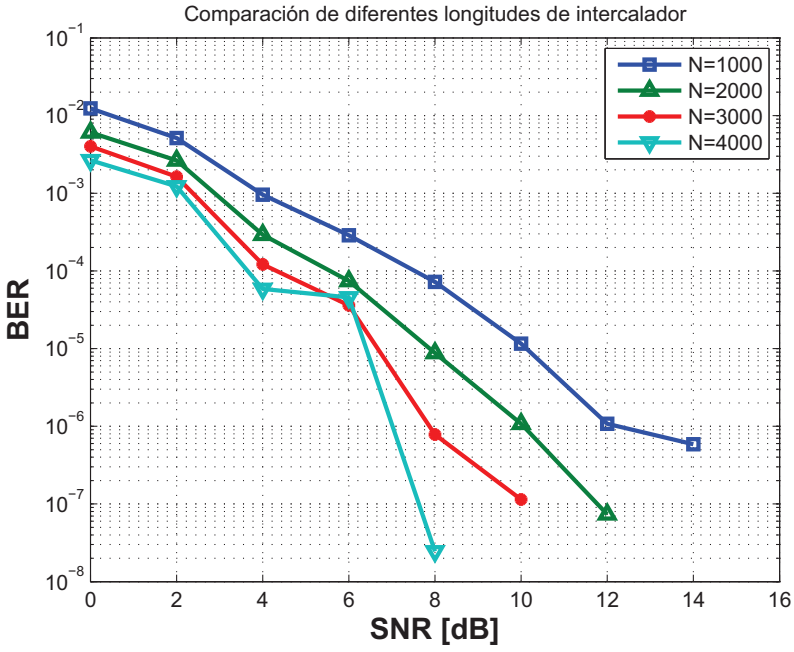

Figura 5.19. Curvas BER generadas por los tres codificadores en un canal PLC con ruido impulsivo.

En la figura 5.19 se muestran los resultados obtenidos al realizar las pruebas con un codificador RSC(31,27) con diferentes longitudes en el intercalador. Para este experimento

se utilizó un canal PLC ciclo-estacionario con 100 impulsos de ruido impulsivo por ciclo. Comparando las curvas generadas se observa que el desempeño se incrementa con respecto a la longitud. En implementaciones prácticas la longitud del intercalador suele utilizarse con valores de acuerdo a la latencia soportada por la aplicación y a los recursos del hardware utilizado. Debido a esto, para la realización de este experimento se seleccionaron longitudes de 1000, 2000, 3000, y 4000 bits. Se puede observar que el codificador con estructura RSC(31,27) presenta mejores resultados, ya que alcanza una probabilidad de error de 10^{-7}. Sin embargo debido a que posee mayor cantidad de estados de memoria, la latencia en el sistema es mayor ya que se incrementa la cantidad de accesos a memoria para el almacenamiento de las variables en la decodificación.

La figura 5.20 muestra las curvas BER obtenidas al utilizar tres diferentes turbo codificadores en un canal PLC ciclo-estacionario con ruido gaussiano e impulsivo con una densidad de 100 impulsos por ciclo con un mapeo 4-QAM. El mejor desempeño fue obtenido por el codificador RSC(31,27). En el caso del codificador RSC(7,5) se observó que su desempeño es muy cercano al codificador RSC(17,15) lo cual disminuye los recursos requeridos en hardware para poder lograr buenos resultados aún en estas condiciones.

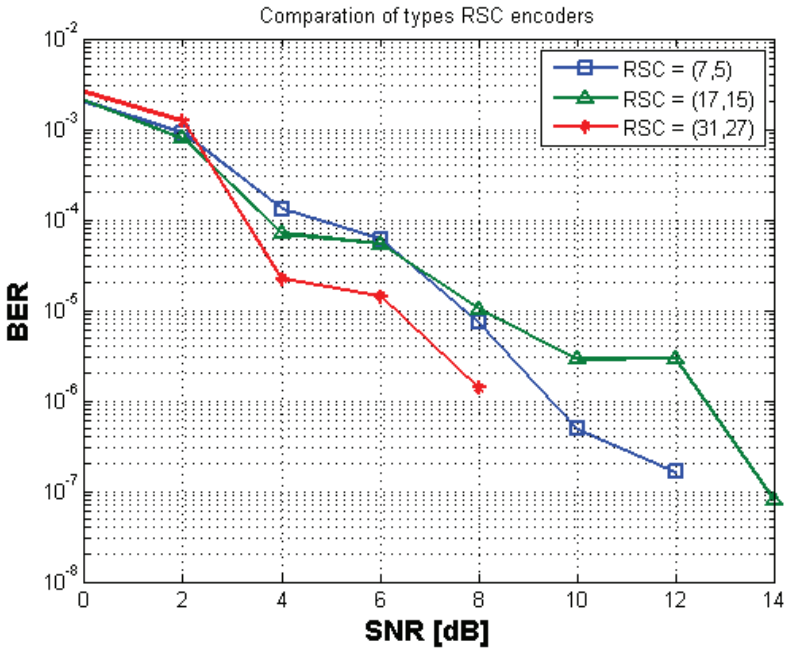

Figura 5.20. Curvas BER generadas por los tres codificadores en un canal PLC con 100 impulsos de ruido impulsivo por ciclo.

También se realizaron experimentos con longitudes en el intercalador de 1088 y 4160

bits con base en la especificación IEEE P1901 [27], esto con el objetivo de analizar los resultados y tener referencia de los parámetros utilizados.

Los parámetros utilizados para llevar a cabo dichos experimentos se muestran en la tabla 5.19 y la curva BER obtenida se presenta en la figura 5.21 para un canal PLC ciclo-estacionario con 100 impulsos de ruido impulsivo por ciclo.

Tabla 5.19. Especificación general para el transreceptor OFDM y el turbo código para resultados de la figura 5.21.

Parámetros	Valor
Frecuencia del sistema: $F_s = B$	30 MHz
Tiempo de muestreo: $t_s = 1/f_s$	33.3 nseg
Subportadoras: N	64
Modulación portadora:	BPSK
Tasa de código:	1/2
Longitud de intercalador (bits):	1088,4160
Codificador:	(17,15)

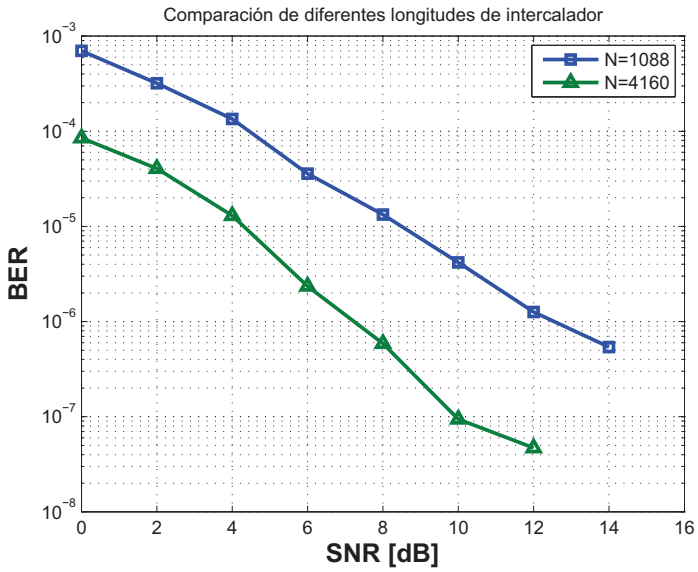

Figura 5.21. Curvas BER obtenidas utilizando el codificador RSC(17,15) con longitud de bloque de 1088 y 4160 bits en un canal PLC con 100 impulsos de ruido impulsivo por ciclo.

Por otro lado se realizaron experimentos utilizando los tres tipos de codificadores implementados en el DSP con una longitud en el intercalador de 4160 bits en canal PLC con

100 impulsos de ruido impulsivo por ciclo. Se utilizó modulación BPSK. Las curvas BER obtenidas se muestran en la figura 5.22. Como se puede observar a pesar de existir gran cantidad de ruido en el canal, se obtuvieron buenos resultados. Para el caso del codificador RSC(31,27), se obtuvo un mejor desempeño, sin embargo, incrementa la latencia en la implementación práctica.

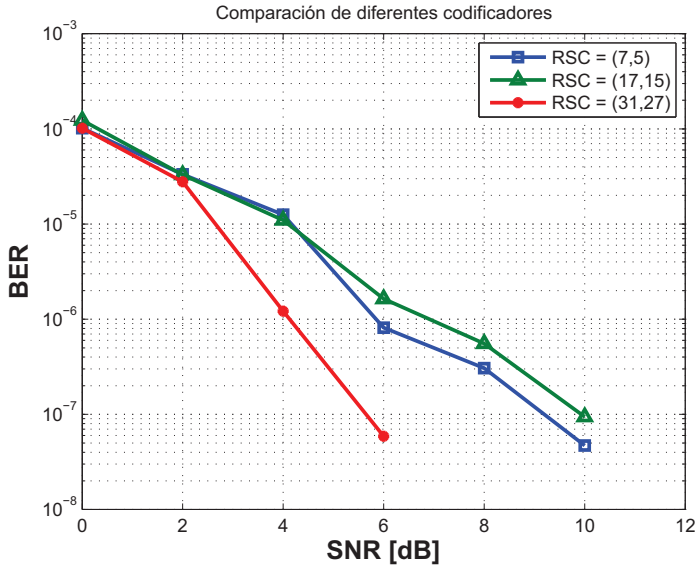

Figura 5.22. Curvas BER obtenidas utilizando los tres codificadores con longitud de bloque de 4160 bits en un canal PLC con 100 impulsos de ruido impulsivo por ciclo.

La figura 5.23 muestra la curva BER obtenida al utilizar el codificador RSC(17,15) variando las longitudes del intercalador y utilizando un mapeo 16-QAM para un canal PLC con 10 impulsos de ruido impulsivo por ciclo. Para este experimento se utilizaron los parámetros de la tabla 5.20. Como se puede observar, al utilizar la longitud de bloque de 4160 bits el desempeño se incrementa y mejora considerablemente la curva BER.

Tabla 5.20. Especificación general para el transreceptor OFDM y el turbo código para resultados de la figura 5.23.

Parámetros	Valor
Frecuencia del sistema: $F_s = B$	30 MHz
Tiempo de muestreo: $t_s = 1/f_s$	33.3 nseg
Subportadoras: N	64
Modulación portadora:	16-QAM
Tasa de código:	1/2
Longitud de intercalador (bits):	1088,4160
Codificador:	(17,15)

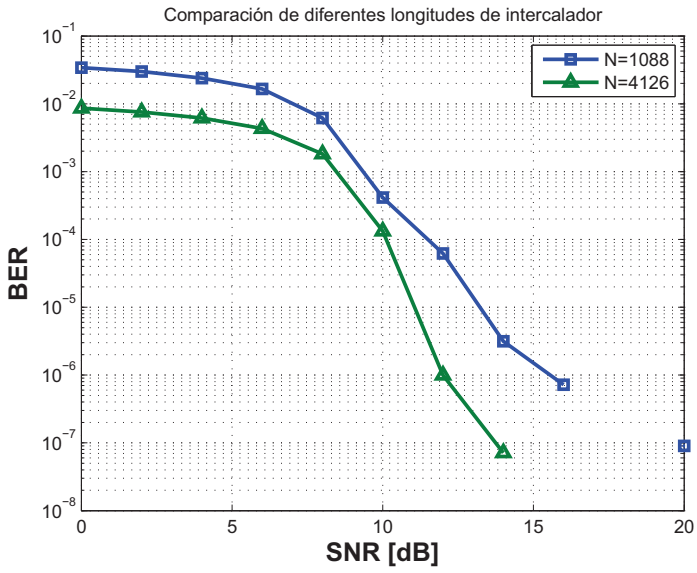

Figura 5.23. Curva BER obtenida con el turbo codificador RSC(17,15) con mapeo 16-QAM.

5.10. Pruebas con imágenes procesadas en el módulo corrector

Se realizaron pruebas utilizando la interfaz diseñada en Matlab como transmisor y la plataforma TMS320C6416T DSK como receptor para la decodificación de imágenes. La comunicación se realizó utilizando el puerto serie. Para ello se construyó una interfaz que permitiera conectar la plataforma TMS320C6416T DSK con la PC, la cual, fue descrita en el capítulo 4.

Para realizar dichas pruebas se utilizó un intercalador de tipo "S-random", para lo cual se generaron dos tablas con los datos para el intercalador y des-intercalador, las cuales, fueron utilizadas tanto en el codificador como en el decodificador respectivamente. Los bloques de datos fueron codificados por un turbo codificador con estructura $\mathrm{RSC}(31, 27)_{octal}$ con una tasa de código de $1/2$, y fueron enviados al DSP. Antes de que los datos fueran decodificados en el DSP, éstos fueron contaminados con ruido gaussiano, el cual, fue generado dentro de este dispositivo conforme al algoritmo "Box-Muller" descrito en el capítulo anterior.

De esta forma, los datos fueron decodificados utilizando como criterio de parada 8 iteraciones. Una vez obtenida la información corregida a la salida con base en la decisión dura del decodificador, dicho mensaje fue devuelto a la PC para ser almacenado y reconstruir la imagen procesada por el DSP. A continuación se presentan los pasos realizados para llevar a cabo dichas pruebas.

1.- Se codificaron las imágenes en formato binario en bloques de 4000 bits de información.

2.- Se agregó ruido Gaussiano.

3.- Se decodificaron cada uno de los bloques en el DSP.

4.- Una vez decodificado cada bloque fue devuelto a la interfaz en la PC.

5.- Se comparó la imagen original con la imagen recuperada.

Para realizar los experimentos y obtener los resultados de la tabla 5.22 al decodificar imágenes se utilizaron los parámetros de la tabla 5.21.

En la tabla 5.22 se muestran los resultados obtenidos de la decodificación de una imagen de tamaño 240 Kbytes para diferentes valores de SNR. La primera columna contiene los valores de la relación señal a ruido, en la segunda columna se presentan los errores totales

Tabla 5.21. Parámetros del módulo para resultados mostrados en la tabla 5.22

Parámetro	valor
Tipo de intercalador	S-random
Longitud de intercalador (bits)	4000
Distancia S	40
Estados	16
Longitud de memoria	4
Iteraciones	8
Tipo de ruido	AWGN
Relación SNR	0 - 5 dB
Tasa de código	1/2
Bloques transmitidos	60
Bits transmitidos	240000

en la imagen contaminada con ruido Gaussiano y en la tercera los errores finales después de aplicar el turbo código en el DSP.

Para realizar las pruebas, cada pixel fue convertido a un valor binario (0,1). De esta forma, la imagen fue almacena en la interfaz en Matlab y se codificaron bloques de 8000 bits con una tasa de código de 1/2 (4000 bits de información y 4000 bits de información redundante). Una vez codificado cada bloque fue enviado al DSP, los datos fueron contaminados con ruido Gaussiano y después decodificados en el DSP, finalmente la información fue devuelta a la PC. Al terminar, se contabilizaron los errores iniciales y finales en la imagen y se mostrarón las imágenes original, con ruido y la imagen decodificada.

Tabla 5.22. Resultados de las pruebas de la decodificación de una imagen con diferentes valores de SNR

SNR	Errores iniciales	Errores finales
1 dB	51230	16920
1.2 dB	49716	2042
1.25 dB	48726	0
1.28 dB	48385	0
1.3 dB	48025	0
1.4 dB	47005	0
2 dB	40950	0
3 dB	30842	0
4 dB	22110	0
5 dB	14662	0

En la figura 5.24 se puede observar la imagen contaminada con ruido Gaussiano y la imagen recupera en el DSP. A pesar de contener una gran cantidad de errores, el turbo código es capaz de recuperar la imagen original.

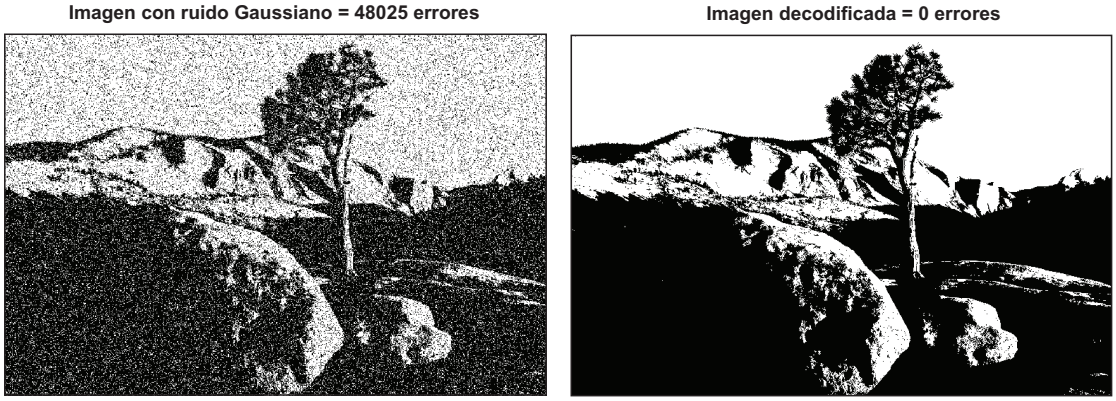

Figura 5.24. Experimento 1: Decodificación de imagen con ruido Gaussiano (48025 errores) para un valor de SNR de 1.3 dB

En la figura 5.25 se muestra el resultado de la decodificación de una imagen de 261 Kbytes en formato binario. Como se puede observar la imagen no presenta errores después de ser decodificada en el DSP.

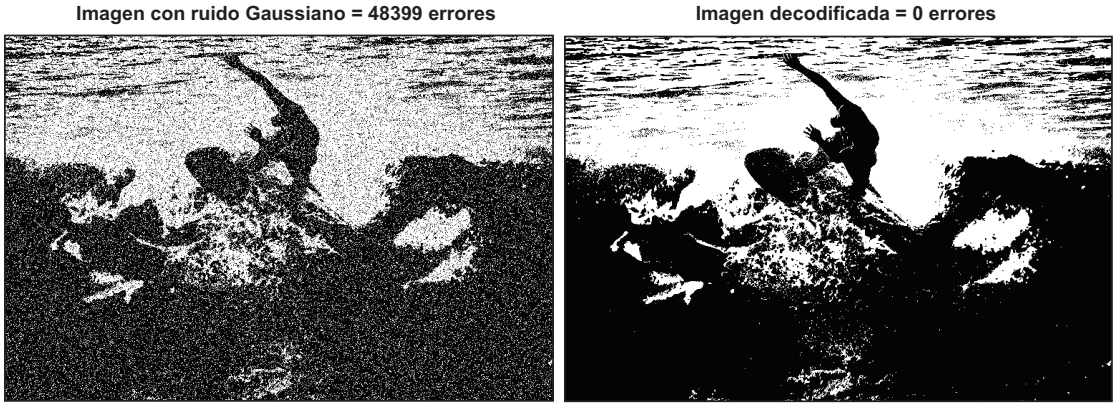

Figura 5.25. Experimento 2: Decodificación de imagen con ruido Gaussiano (48399 errores) para un valor de SNR de 1.3 dB

Las figuras 5.26 y 5.27 presentan la imagen original, la imagen con ruido Gaussiano y la imagen decodificada por el módulo corrector en el DSP para una SNR de 1.3 dB.

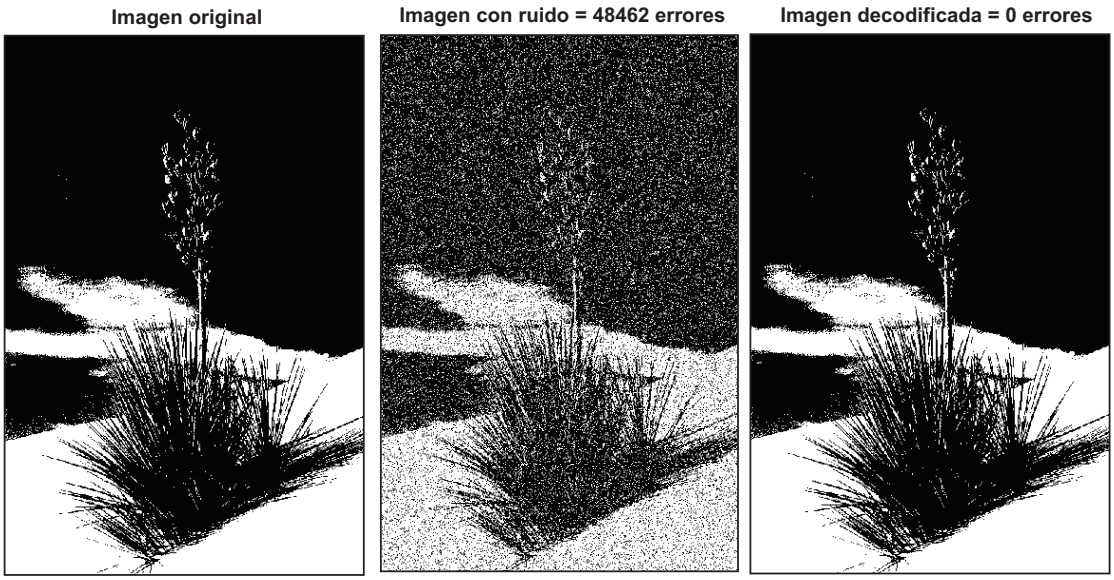

Figura 5.26. Experimento 3: Comparación: a) Imagen original, b) Imagen con ruido (48462 errores), c) Imagen recuperada desde el DSP.

Figura 5.27. Experimento 4: Comparación: a) Imagen original, b) Imagen con ruido (48164 errores), c) Imagen recuperada desde el DSP.

5.11. Resumen del capítulo

En este capítulo se presentaron los experimentos y los resultados obtenidos al realizar las simulaciones con el módulo corrector de errores implementado en el DSP. En la primera sección se mostraron las gráficas de los valores de la información extrínseca con base en el incremento del número de iteraciones en el algoritmo de decodificación. Después, se presentaron los resultados obtenidos al realizar las simulaciones al utilizar el método Monte Carlo primeramente en un canal con ruido gaussiano y después para un canal con ruido impulsivo y gaussiano. Para las pruebas realizadas se utilizaron los tres tipos de codificador implementados: RSC(7,5), RSC(17,15), y RSC(31,27). Las longitudes de bloque en bits utilizadas en la primera sección son: 1000, 2000, 3000, 4000. Las cuales de acuerdo a [4] son las más comúnmente utilizadas en implementaciones en hardware. También se realizaron pruebas con longitudes de: 1088, 4160 bits, las cuales de acuerdo a [27] son utilizadas para aplicaciones en el canal PLC. Con base en los resultados obtenidos, se puede mencionar que se obtuvieron mejores resultados al incrementar la longitud de bloque de datos codificados así como al incrementar el número de estados en el codificador. Para las pruebas con ruido impulsivo se utilizo un simulador diseñado en [45]. Se agrego la versión de turbo código similar implementada en Matlab y se utilizó modulación BPSK y 16-QAM, las cuales, son incluidas dentro del estándar IEEE P1901 [27]. Los resultados obtenidos muestran un mejor desempeño para la longitud de bloque de 4126 bits con respecto a 1088 bits. Así como para la modulación BPSK con respecto a 16-QAM. Al incrementar el número de impulsos de ruido impulsivo en el canal PLC se observó una disminución en el desempeño de la cantidad de errores corregidos debido a que se incrementaron la cantidad de errores por bloque en los datos recibidos. Es importante mencionar que para decodificar la información recibida, se calcula la relación de probabilidad por cada símbolo recibido y su vez es ingresada como información suave a la entrada del decodificador. En la última parte del capítulo se presentan los resultados obtenidos al decodificar imágenes en formato binario, utilizando para ello, la interfaz diseñada en Matlab como transmisor y el DSP como receptor. Con base en los resultados obtenidos se puede mencionar que al utilizar un codificador RSC(31,27) con una longitud de bloque de 4000 bits se se obtuvieron buenos resultados a partir de 1.25 dB en donde se recupero la imagen original sin errores. En el siguiente capítulo se presentaran las conclusiones con base en las pruebas y los resultados obtenidos en este trabajo de investigación.

Capítulo 6

Conclusiones y trabajos futuros

En el presente capítulo se presentarán las conclusiones obtenidas después de realizar el presente trabajo de investigación. A la vez se mencionarán los trabajos futuros propuestos para la mejora y aplicación del módulo corrector de errores implementado. Finalmente se presentarán las aportaciones de la investigación y las publicaciones realizadas en congresos internacionales.

6.1. Conclusiones

En el presente trabajo se estructuró la plataforma básica de hardware para la comunicación entre la tarjeta TMS320C6416T y la PC, y con ello se realizó la simulación del módulo de corrección de errores implementado. Dicha plataforma se compone de una interfaz serial UART a una velocidad de 115,200 bps, conectada a la tarjeta TMS320C6416T DSK y una interfaz gráfica diseñada en Matlab. Lo cual permite enviar y recibir la información desde el DSP para poder ser analizada y almacenada en archivo.

Se desarrolló una biblioteca compuesta por 30 rutinas en el DSP para analizar las funciones básicas del módulo corrector de errores. Entre estas rutinas se encuentran:

- Codificación de bloques de datos.
- Decodificación de bloques de datos recibidos.
- Generación de ruido Gaussiano.
- Construcción del intercalador.

Se programó el módulo corrector de errores utilizando un turbo código como esquema principal en el DSP. Para ello se programaron cada uno de los componentes del turbo codificador y decodificador para constituir así en conjunto dicho código corrector.

Se simuló e implementó el turbo código programado utilizando como medio auxiliar la PC para analizar el correcto funcionamiento de los algoritmos implementados en el DSP. A su vez se realizaron simulaciones Monte Carlo variando la relación señal a ruido y se compararon las curvas BER obtenidas en la PC y en el DSP utilizando los mismos parámetros en el turbo código. Con base en los resultados presentados en el capítulo 5 se observó una mínima diferencia entre ellos. Además se observó que con un tipo de turbo codificador RSC(31,27) se obtiene un umbral de convergencia de 1.3 dB.

- Con las pruebas realizadas en un canal con ruido Gaussiano e impulsivo se observó que el codificador RSC(31,27) tuvo el mejor desempeño.
- Se observó que al incrementar la longitud del intercalador, se incrementó el desempeño en las curvas BER.
- Se observó que al utilizar el turbo codificador RSC(31,27) se obtiene un umbral de convergencia de 1.3 dB.
- Al utilizar un intercalador de tipo S-random se observó un mejor desempeño en el módulo corrector de errores.

Para tener las condiciones de ruido en un canal PLC se utilizó el transreceptor OFDM diseñado en [45]. A este simulador se agregó el módulo corrector programado. Con este esquema se variaron los diferentes parámetros de dicho módulo corrector y se analizaron los resultados obtenidos. De esta forma se encontró que al incrementar la longitud del intercalador, se incremento el desempeño en las curvas BER. Además se observó que con un tipo de turbo codificador RSC(5,7) los resultados son muy cercanos a la configuración RSC(17,15) y los mejores resultados fueron obtenidos utilizando la configuración RSC(31,27).

Las longitudes del intercalador analizadas fueron de 1000, 1088, 2000, 3000, 4000 y 4160 bits con tres diferentes esquemas en el turbo codificador. El número de iteraciones utilizado fue de entre 1 y 10. Además con base en los experimentos realizados con longitudes en el intercalador de 1088 y 4160 bits (de cuerdo al estándar IEEE P1901 [27]) se observó un incremento considerable en el desempeño del algoritmo de decodificación a partir de 4 iteraciones. A su vez, al utilizar modulación BPSK y 16-QAM se obtuvieron mejores resultados con la longitud de 4160 bits en el intercalador para el codificador RSC(17,15). Finalmente al comparar los tres codificadores RSC implementados se obtuvieron mejores resultados con el codificador RSC(31,27).

Entre los aportes de este trabajo de tesis se encuentran:

1.- **Un módulo simulador** de un código de corrección de errores de alto desempeño para un canal PLC.

2.- **Una plataforma básica de comunicación**, en la cual se pueden probar en forma práctica algoritmos correctores de errores propuestos.

3.- **Una biblioteca de funciones de código abierto** que constituyen el módulo corrector de errores.

4.- **Una implementación del módulo de corrección de errores** que al utilizar un simulador del canal PLC, permite conjeturar que tendrá un correcto funcionamiento cuando se utilice en un canal de la red eléctrica.

Con base en todo lo anterior podemos afirmar que los objetivos formulados en el capítulo 1, se alcanzaron completamente.

6.2. Trabajos futuros

Con base en el desarrollo de esta investigación y los resultados reportados y analizados se puede mencionar que entre los trabajos futuros se encuentran:

1.- Dadas las características de latencia en el turbo código y el gran uso de memoria para el almacenamiento de variables, se propone la paralelización de los algoritmos de codificación y decodificación, ya que ésto podría reducir la latencia y a la vez mejorar el desempeño del módulo corrector de errores.

2.- La integración del módulo corrector de errores en la plataforma que constituye un módem para el canal de la red eléctrica y la realización de pruebas en dicho sistema.

3.- La implementación de los códigos de chequeo de paridad de baja densidad (LDPC) en la plataforma propuesta.

Dado que el esquema de turbo código utilizado posee diferentes elementos que influyen en su desempeño, queda como trabajo futuro analizar diferentes tipos y características en diferentes elementos como lo son: el intercalador y el codificador, con el objetivo de proponer mejoras y modificaciones a los algoritmos.

6.3. Publicaciones

Durante del desarrollo de la presente investigación se publicaron los siguientes artículos:

- **Low complexity Turbo Code Specification for Power-line Communication (PLC)**. Alfonso Martínez-Cruz, Ricardo Barrón-Fernández, José Luis Oropeza-Rodríguez, Gerardo A. Laguna-Sánchez. Center for Computing Research. 2011 IEEE Electronics, Robotics and Automotive Mechanics Conference. CERMA 2011.

- **Low Complexity Turbo code implementation in DSP TMS320C6416T.** Alfonso Martínez-Cruz, Ricardo Barrón Fernández, José Luis Oropeza Rodríguez. Center for Computing Research. 7o Congreso Internacional: TENDENCIAS TÉCNOLOGICAS EN COMPUTACIÓN. Centro de Innovación y Desarrollo Tecnológico en Cómputo, 2011.

Referencias bibliográficas

[1] J. M. Huidobro, *Todo sobre Comunicaciones*, 1st ed. Paraninfo, 1998.

[2] A. S. y. Z. W. Stefano Galli, "For the grid and through the grid: The role of power line communications in the smart grid," *IEEE*, pp. 1 – 22, 2010.

[3] G. Laguna and R. Barrón, "Survey on indoor power line communication channel modeling," *Artificial Intelligence Laboratory National Polytechnic Institute, Electronics, Robotics and Automotive Mechanics Conference, IEEE*, pp. 163 – 168, 2008.

[4] C. Berrou, *Codes and Turbo Codes*, 2nd ed. Springer-Verlag, 2010.

[5] T. K. moon, *Error correction coding, mathematical methods and algorithms*, 1st ed. Wiley Interscience & Sons, 2005.

[6] J. C. M. y Patrick Guy Farrell, *Essentials of error-control coding*, 1st ed. John Wiley & Sons, 2006.

[7] C. Schlegel, *Trellis Coding*, 1st ed. The Institute of Electrical and Electronics Engineers, 1997.

[8] K. Sripimanwat, *Turbo Code Applications,A Journey from a Paper to Realization*, 1st ed. Springer, National Electronics and Computer Technology Center (NECTEC), 2005.

[9] M. Zimmermann and K. Dostert, "An analysis of the broadband noise scenario in powerline networks," *School of Electrical and Computer Engineering, IEEE*, vol. 44, no. 1, pp. 249 – 258, 2002.

[10] T. C. Chuah, "Adaptive robust turbo equalization for power-line communications," *IEEE Transactions on power delivery*, vol. 22, no. 4, pp. 2172 – 2179, 2007.

[11] S. J. Johnson, *Iterative error correction, Turbo, Low-Density Parity-Check and Repeat-Accumulate Codes*, 1st ed. Cambridge university press, 2010.

[12] A. Abbasfar, *Turbo-like Codes, Design for High Speed Decoding*, 1st ed. Springer, 2007.

[13] E. Guizzo, "Closing in on the perfect code," *IEEE Spectrum*, pp. 36 – 42, March 2004.

[14] C. Berrou and A. Glavieux, "Near optimum error correcting coding and decoding: Turbo-codes," *IEEE, transacctions of communications*, vol. 44, no. 10, pp. 1064 – 1070, 1996.

[15] R. H., *The Art of Error Correcting Coding*, 1st ed. John Wiley & Sons Ltd, 2006.

[16] W. Henk, *Iterative Receiver Design*, 1st ed. Cambridge University Press, 2004.

[17] H. R. Sadjadpour, "Maximum a posteriori decoding algorithms for turbo codes," *AT & T research-Shannon Labs, Florham Park, NJ*, vol. 4045, no. 1, pp. 73 – 83, 2000.

[18] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, no. 1, pp. 101 – 111, 1948.

[19] R. W. Hamming, "Error detecting and error correcting codes," *The Bell system technical journal*, vol. 26, no. 2, pp. 147 – 161, April, 1950.

[20] R. G. Gallager, "Low-density parity-check codes," *Cambridge, MA: MIT Press*, 1963.

[21] T. M. y Shuji Maekawa, "Field test of the world first 200mbps plc modems," *Sumitomo Electric Industries, LTD, Shimaya, Konohana-ku, Osaka, Japan*, pp. 5330 – 5332, 2010.

[22] Crussière, "M. Ètude et o'timisation de communications à haut-débit sur lignes d'énergie: exploitation de la combinaison ofdm/cdma." Ph.D. dissertation, PhD thesis, Institut National des sciences Appliquées de Rennes (INSA), Rennes, Francia, 2005.

[23] B. TELECOM. (2011, December) International symposium on turbo codes and iterative information processing. [Online]. Available: http://conferences.telecom-bretagne.eu/turbocodes

[24] S. technologies. (2011, 3 de diciembre) Anywire broadband communications. [Online]. Available: http://www.spidcom.com/en.

[25] T. WADA, "A study on the performance of turbo coding for noise environments in power lines," *Department of electrical and electronic engineering, Shizuoka Univeristy Johoku 3-5-1, Hamamatsu, Japan*, pp. 3071 – 3075, 2003.

[26] G. D. L. Guerrieri, P. Bisaglia and E. Guerrini, "Performance of the turbo coded homeplug av system over power-line channels," *Faculty of engineering, Multimedia University, Jalan multimedia, 63100 Cyberjaya, Selangor, Malaysia, Dora Spa, STMicroelectronics Group, Via Lavoratori Vittime del Col du Mont 24, 11100 Aosta, Italy*, pp. 138 – 143, 2007.

[27] S. association IEEE, *IEEE Standard for Broadband over Power Line Networks: Medium Acess Control and Physical Layer Specifications*, 1st ed. IEEE Communications Society, 2010.

[28] C. ©2011 HomePlug Powerline Alliance. (2011, 3 de diciembre) Powerline networking. [Online]. Available: http://www.homeplug.org/tech/.

[29] D. J. M. y R. M. Neal, "Good codes based on very sparse matrices," *5a conferencia en criptografía y codificación, lectura notas en ciencia en computación, C. Bold, Ed. Springer*, vol. 1025, pp. 100 – 111, 1995.

[30] S. Z. R. y. M. E.-S. Muhammad Salman Yousuf, "Power line communications: An overview - part ii," *Dept of Electrical Engineering, KFUPM Dhahran, Saudi Arabia*, pp. 1 – 6, 2006.

[31] H. S. L. y. T. C. C. M. Shafieipour, "Robust tubo coded ofdm trasceiver for power-lines channels," *Faculty of engineering, Multimedia University, Jalan multimedia, 63100 Cyberjaya, Selangor, Malaysia*, vol. 7, no. 19, pp. 1416 – 1422, 2010.

[32] S. Z. R. y. M. E.-S. Muhammad Salman Yousuf, "Power line communications: An overview - part i," *IEEE*, pp. 218 – 222, 2008.

[33] H. Y. Daisuke Umehara and Y. Morihiro, "Turbo decoding in impulsive noise environment," *IEEE Communications Society, Globecom, Japan*, pp. 194 – 198, 2004.

[34] L. C. P. y. F. J. Branka Vucetic, Yongghui Li, "Recent advances in turbo code design and theory, ieee," *Cambridge, MA: MIT Press*, vol. 95, no. 6, pp. 1323 – 1344, 2007.

[35] W. E. Ryan, "A turbo code tutorial," *New Mexico State University*, pp. 1 – 9, 1998.

[36] B. Sklar, "Fundamentals of turbo codes," pp. 1 – 30, March 15, 2002.

[37] T. Bayes, "An essay towards solving a problem in the doctrine of chances," *Philosophical Transactions of the Royal Society of London*, pp. 370 – 418, 1763.

[38] I. Berkeley Design Technology, "A bdti analysis of the texas instruments tms320c64x," pp. 1 – 4, 2004.

[39] S. D. Inc., "Tms320c6416t dsk technical reference," *DSP Development Systems, Texas Instruments*, pp. 1 – 56, November, 2004.

[40] T. Instruments, "Tms320c6414t, tms320c6415t, tms6416t fixed-point digital signal processors," *Datasheets*, pp. 1 – 142, November 2003.

[41] M. Zimmermann and K. Dostert, "Analysis and modeling of impulsive noise in broadband powerline communications," *The Institute of Industrial Information Systems, University of Karlsruhe, 76187 Karlsruhe, Germany*, vol. 44, no. 1, pp. 249 – 258, September 28, 2001.

[42] O. N. M. for Power Line Communications, "Luca di bert, peter caldera, david schwingshackl and andrea m. tonello," *IEEE International Symposium on Power Line Communications and Its Applications*, pp. 283 – 288, 2011.

[43] B. E., "Coding and modulation for a horrible channel," *IEEE Communications Magazine*, vol. 41, no. 1, pp. 92 – 98, 2003.

[44] H. S. and V. H., "Coding for impulsive noise channels," *In IEEE International Symposium on Power Line Communications and Its Applications*, pp. 103 – 108, 2001.

[45] G. Laguna, "Wavelet transform application for the channels plc estimation," Ph.D. dissertation, Center for Computing Research (CIC),IPN, Zacatenco, Mexico, D.F., 2010.

[46] C. Lemieux, *Monte Carlo and Quasi-Monte Carlo Sampling*, 1st ed. Springer, 2009.

[47] P. B. y. K. S. S. Michel C. Jeruchim, *Simulation of Communications Systems. Modeling, methodology and techniques*, 2nd ed. Kluwer academic.Plenum publishers, 2000.

[48] G. E. P. Box and M. E. Müller, "A note on the generation of random normal deviates," *The Annals of Mathematical Statistics*, vol. 29, no. 2, pp. 610 – 611, 1958.

[49] R. Chanssaing, *Digital Signal Processing and Applications with the tms320c6713 and tms320c6416 dsk*, 1st ed. John Wiley & Sons, 2008.

[50] R. L. P. y. H. S. Rodger, E. Zieme, *Introduction to digital comunication*, 2nd ed., 1998.

[51] I. G. y Meter Grant, *Digital Communications*, 1st ed. Ed. Prentice Hall, 1998.

[52] I. Miñiz, *Voz y Datos a través de la Red Eléctrica*, 1st ed. Centro de Investigación e innovación en telecomunicaciones, 2005.

[53] P. Sweeney, *Error control coding. From theory to practice*, 1st ed. John Wiley, 2002.

[54] B. Sklair, *Digital communications. Fundamentals and Applications*, 1st ed. Prentice Hall, 2001.

[55] y. M. S. John G. Proakis, *Communication systems using MATLAB*, 2nd ed. PWS publishing company, 1998.

[56] Y. Jiang, *A Practical Guide to Error-Control Coding Using Matlab*, 1st ed. ARTECH HOUSE, 2010.

[57] C. H. y Stephen B. Wicker, *Turbo coding*, 1st ed. Kluwer Academic Publishers, 1999.

[58] P. J. L. y Klaus Dostert, "Ofdm system synchronization for powerline communications," *Institute of Industrial Information Systems, University of Karlsruhe, Germany.*

[59] K. Y. L. Haiyun Tang and R. W. Brodersen, "Synchronization schemes for packet ofdm system," *Barkeley Wireless Research Center*, pp. 1 – 5, 2000.

[60] J. I. G. N. y. I. U. A. Purroy, A. Sanz, "Research areas for efficient power line communication modems," *Department of Electronics and Communications Engineering, Centro Politécnico Superior, University of Zaragoza,Zaragoza, Spain.*

[61] R. Róka, "The design of a plc modem and its implementation using fpga circuits," *Department of Telecommunications, Faculty of Electrical Engineering and Information Technology, Slovak University of Technology, Ilkovicova 3, 812 19, Bratislava, Slovakia*, 2009.

[62] H. A. G.-B. y V. Alarcon-Aquino, "A power-line communication modem based ofdm," *Department of Computing, Electronics and Mechatrnocis, Universidad de las Americas*, pp. 208 – 213, Puebla, 2009.

[63] F. T. y. C. R. Souha Souissi, Arwa Ben Dhia, "Ofdm modem design and implementation for narrowband powerline communication," *International Conference on Design & Technology of Integrated System in Nanoscale Era*, pp. 1 – 4, 2010.

[64] K. D. Manfred Zimmermann, "An analysis of the broadband noise scenario in powerline networks," *IEEE, Institute of Industrial Information Systems University of Karlsruhe Hertzstrasse 16, D-76187 Karisruhe, Germany*, pp. 131 – 138, 2004.

[65] S. A. Abrantes, "From bcjr to turbo decoding map algorithms made easier," *Faculdad de Engenharia da Universidade do Porto (FEUP)*, pp. 1 – 30, April 2004.

[66] K. D. y. J. B. Marko Babic, Michael Hagenau, "Theoretical postulation of plc channel model," *IST Integrated Project No 507667, Open PLC European Research Alliance*, 30-03-2005.

[67] J. Tan and G. L. Stuber, "Analysis of iterative of turbo codes," *School of Electrical and Computer Engineering. IEEE*, 2001.

[68] J. L. C. y F. J. Cañete y Cortés y L. Díez, "A dmt módem prototype for broadband plc," *IEEE*, pp. 56 – 61, 2006.

[69] F. J. C. José Antonio Cortés, Luis Díez and J. J. Sánchez-Martínez, "Analysis of the indoor broadband power-line noise scenario," *IEEE Transactions on electromagnetic compatibility*, vol. 52, no. 4, pp. 849 – 858, 2010.

[70] A. J. H. V. Niovi Pavlidou and J. Yazdani, "Power line communications: State of the art and future trends," *IEEE Communicaions Magazine*, pp. 34 – 40, 2003.

[71] L. Simoes and J. A. B. Gerald, "A communication system for power lines," *IEEE*, pp. 1 – 4, 2003.

[72] C. ©2011 HomePlug Powerline Alliance. (2011, 3 de diciembre) Homeplug. [Online]. Available: http://www.homeplug.org/.

[73] plcforum. (2011, December) Power line communications. [Online]. Available: http://www.plcforum.com/

[74] C. s. IEEE. (2011, December) Ieee international symposium on power line communications and its aplications. [Online]. Available: http://www.ieee-isplc.org/

ANEXOS

Anexo A: Operaciones en formato Q15

A.1. Operaciones en formato Q15

EL formato Q15 es muy utilizado en procesadores de punto fijo con los cuales se realizan operaciones con números enteros. Las principales operaciones entre dos números en formato Q15 son las siguientes:

- **Suma:** $a + b = a + b$ dónde a y b son múmeros en formato Q15.
- **Multiplicación:** $(a \times b) >> q$
- **Resta:** $a - b = a - b$
- **División:** $\frac{a}{b} = \frac{(a<<q)}{b}$

Las operaciones realizadas entre un número en formato Q15 y un número entero decimal son:

- **Suma:** $a + b = a + (b << q)$
- **Resta:** $a - b = a - (b << q)$
- **Multiplicación:** $a \times b$
- **División:** $\frac{a}{b}$

La conversión de un número en formato q1 a q2 es:

- **Conversión:** Sí $q2 > q1$ entonces $a = q2 - q1$ sino $a = q1 - q2$.

Para convertir un número a y desde punto flotante son:

- **A formato:** $d \times (1 << q)$.
- **Desde formato Q15 a flotante:** $\frac{a}{(1<<(q))}$.

Anexo B: Función de transferencia para un IIR FSSMs

B.1. Función de transferencia para un IIR FSSMs con 4 estados

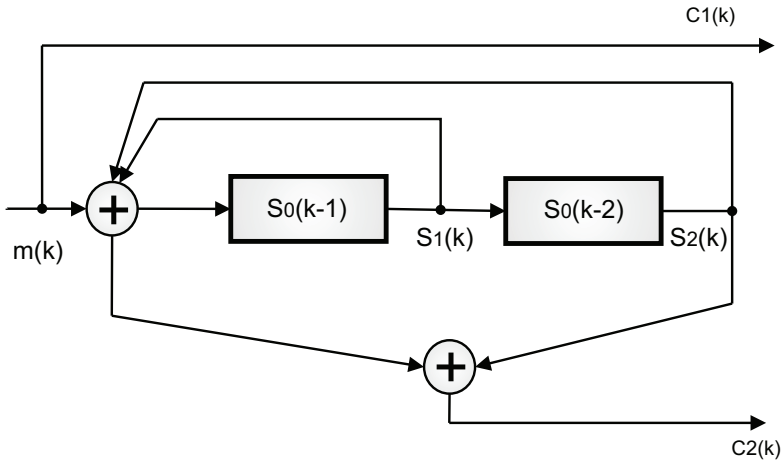

Figura B.1. Codificador IIR con 4 estados y dos registros de memoria.

$S_0(k) = m(k) + S_2(k)$
$S_1(k) = S_0(k-1)$
$S_2(k) = S_0(k-2)$
$c(k) = S_0(k) + S_0(k-2)$
En el dominio D:
$S_1(D) = D \cdot S_0(D)$
$S_2(D) = D^2 \cdot S_0(D)$

$S_0(D) = M(D) + S_1(D) + S_2(D) = M(D) + D \cdot S_0(D) + D^2 \cdot S_0(D)$
$S_0(D) + D \cdot S_0(D) + D^2 \cdot S_0(D) = M(D)$
$S_0(D) = \frac{M(D)}{1+D+D^2}$
$C(D) = S_D + D^2 \cdot S_0(D) = (1 + D^2) \cdot S_0(D) = (1 + D^2)\frac{M(D)}{1+D+D^2}$
La función de transferencia es:
$G(D) = \frac{C(D)}{M(D)} = \frac{1+D^2}{1+D+D^2}$
La función de transferencia de estados:
$S(D) = \left[\frac{S_0(D)}{M(D)} \frac{S_2(D)}{M(D)}\right] = [\frac{1}{1+D+D^2} \frac{D^2}{1+D+D^2}]$
Si la entrada $M(D) = 1 + D + D^2$ la secuencia de estados correspondientes es:
$[S_0(D)S_2(D)] = [\frac{1}{1+D+D^2} \frac{D^2}{1+D+D^2}] \cdot M(D)$
$= [\frac{1}{1+D+D^2} \frac{D^2}{1+D+D^2}] \cdot (1 + D + D^2)$
$= [1 \quad D^2]$

B.2. Función de transferencia para un IIR FSSMs con 8 estados

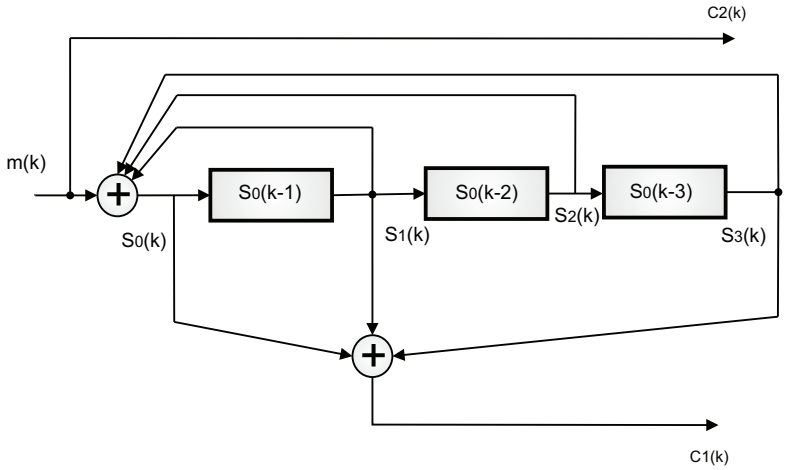

Figura B.2. Codificador IIR con 8 estados y 3 registros de memoria.

$S_0(k) = m(k) + S_1(k) + S_2(k) + S_3(k)$
$S_1(k) = S_0(k-1)$
$S_2(k) = S_0(k-2)$
$S_3(k) = S_0(k-3)$
$c(k) = S_0(k) + S_0(k-1) + S_0(k-3)$
En el dominio D:

$S_1(D) = D \cdot S_0(D)$
$S_2(D) = D^2 \cdot S_0(D)$
$S_3(D) = D^3 \cdot S_0(D)$
$S_0(D) = M(D) + S_1(D) + S_2(D) + S_3(D) = M(D) + D \cdot S_0(D) + D^2 \cdot S_0(D) + D^3 \cdot S_0(D)$
$S_0(D) + D \cdot S_0(D) + D^2 \cdot S_0(D) + D^3 \cdot S_0(D) = M(D)$
$S_0(D) = \frac{M(D)}{1+D+D^2+D^3}$
$C(D) = S_D + D \cdot S_0(D) + D^3 \cdot S_0(D) = (1 + D + D^3) \cdot S_0(D) = (1 + D + D^3) \frac{M(D)}{1+D+D^2+D^3}$
La función de transferencia es:
$G(D) = \frac{C(D)}{M(D)} = \frac{1+D+D^3}{1+D+D^2+D^3}$
La función de transferencia de estados:
$S(D) = \left[\frac{S_0(D)}{M(D)} \frac{S_1(D)}{M(D)} \frac{S_3(D)}{M(D)}\right] = [\frac{1}{1+D+D^2+D^3} \frac{D}{1+D+D^2+D^3} \frac{D^3}{1+D+D^2+D^3}]$
Si la entrada $M(D) = 1 + D + D^2 + D^3$ la secuencia de estados correspondientes es:
$[S_0(D)S_1(D)S_3(D)] = [\frac{1}{1+D+D^2+D^3} \frac{D}{1+D+D^2+D^3} \frac{D^3}{1+D+D^2+D^3}] \cdot M(D)$
$= [\frac{1}{1+D+D^2+D^3} \frac{D}{1+D+D^2+D^3} \frac{D^3}{1+D+D^2+D^3}] \cdot (1 + D + D^2 + D^3)$
$= [1 \quad D \quad D^3]$

B.3. Función de transferencia para un IIR FSSMs con 16 estados

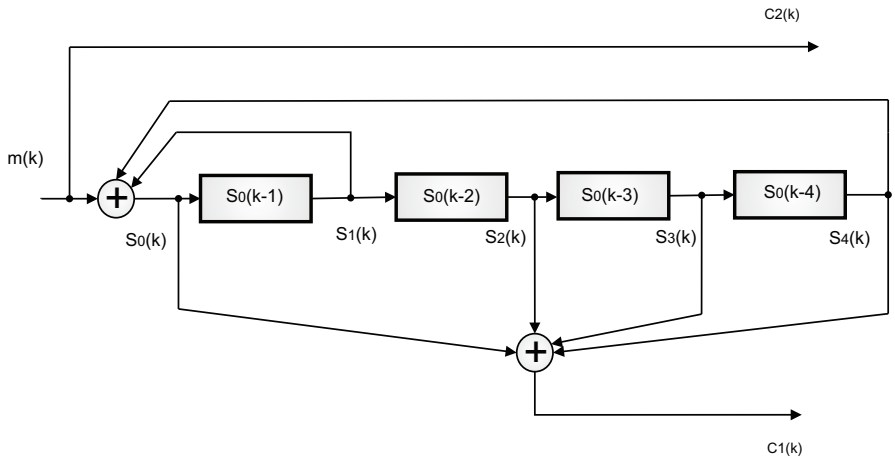

Figura B.3. Codificador IIR con 16 estados y 4 registros de memoria.

$S_0(k) = m(k) + S_1(k) + S_3(k)$
$S_1(k) = S_0(k-1)$
$S_2(k) = S_0(k-2)$

$S_3(k) = S_0(k-3)$

$S_4(k) = S_0(k-4)$

$c(k) = S_0(k) + S_0(k-2) + S_0(k-3) + S_0(k-4)$

En el dominio D:

$S_1(D) = D \cdot S_0(D)$

$S_2(D) = D^2 \cdot S_0(D)$

$S_3(D) = D^3 \cdot S_0(D)$

$S_4(D) = D^4 \cdot S_0(D)$

$S_0(D) = M(D) + S_1(D) + S_4(D) = M(D) + D \cdot S_0(D) + D^4 \cdot S_0(D)$

$S_0(D) + D \cdot S_0(D) + D^4 \cdot S_0(D) = M(D)$

$S_0(D) = \frac{M(D)}{1+D+D^4}$

$C(D) = S_D + D \cdot S_0(D) + D^2 \cdot S_0(D) + D^3 \cdot S_0(D) + D^4 \cdot S_0(D) = (1+D^2+D^3+D^4) \cdot S_0(D) = (1 + D^2 + D^3 + D^4)\frac{M(D)}{1+D+D^4}$

La función de transferencia es:

$G(D) = \frac{C(D)}{M(D)} = \frac{1+D^2+D^3+D^4}{1+D+D^4}$

La función de transferencia de estados:

$S(D) = \left[\frac{S_0(D)}{M(D)} \frac{S_2(D)}{M(D)} \frac{S_3(D)}{M(D)} \frac{S_4(D)}{M(D)}\right] = [\frac{1}{1+D+D^4} \frac{D^2}{1+D+D^4} \frac{D^3}{1+D+D^4} \frac{D^4}{1+D+D^4}]$

Si la entrada $M(D) = 1 + D + D^4$ la secuencia de estados correspondientes es:

$[S_0(D)S_2(D)S_3(D)S_4(D)] = \left[\frac{S_0(D)}{M(D)} \frac{S_2(D)}{M(D)} \frac{S_3(D)}{M(D)} \frac{S_4(D)}{M(D)}\right] \cdot M(D)$

$= [\frac{1}{1+D+D^4} \frac{D^2}{1+D+D^4} \frac{D^3}{1+D+D^4} \frac{D^4}{1+D+D^4}] \cdot (1 + D + D^4)$

$= [1 \quad D^2 \quad D^3 \quad D^4]$

Anexo C: Circuito implementado de la interfaz UART

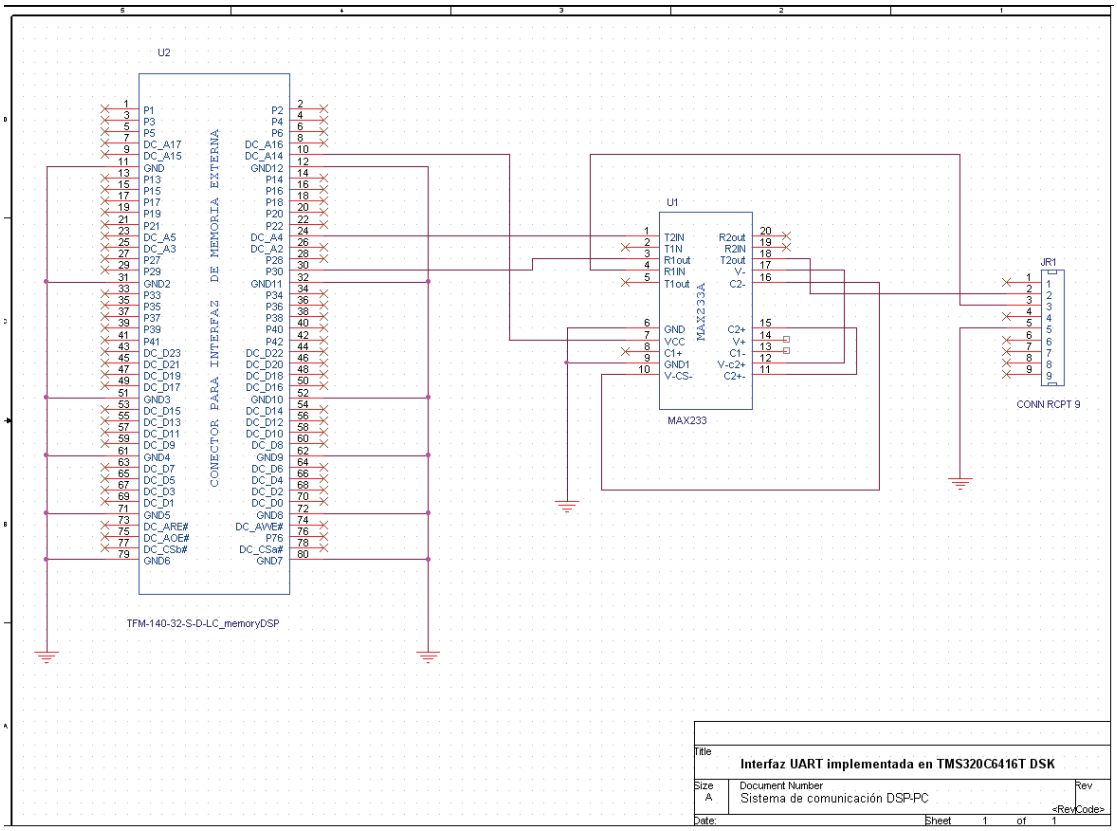

Figura C.1. Diagrama de conexiones de la tarjeta prototipo construida para la comunicación serial RS-232 entre la PC y el módulo TMS320C6416T DSK.

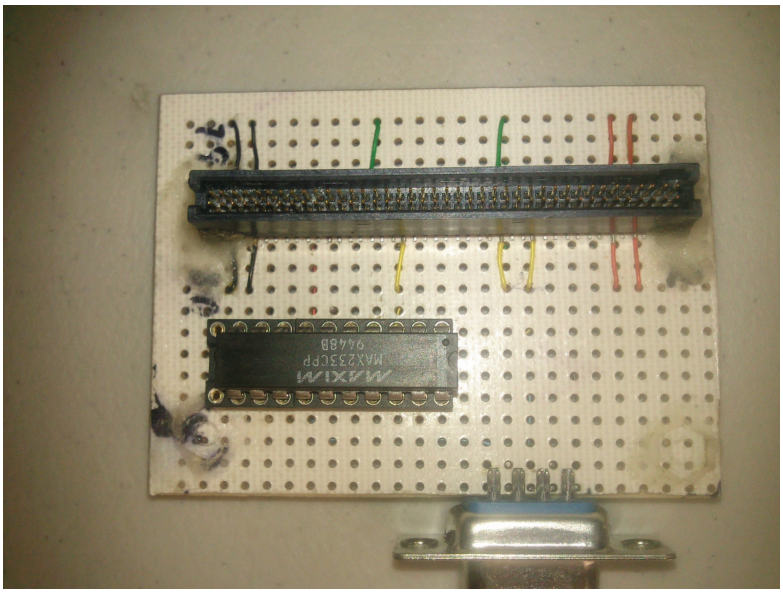

Figura C.2. Vista superior de la tarjeta construida para la comunicación serial.

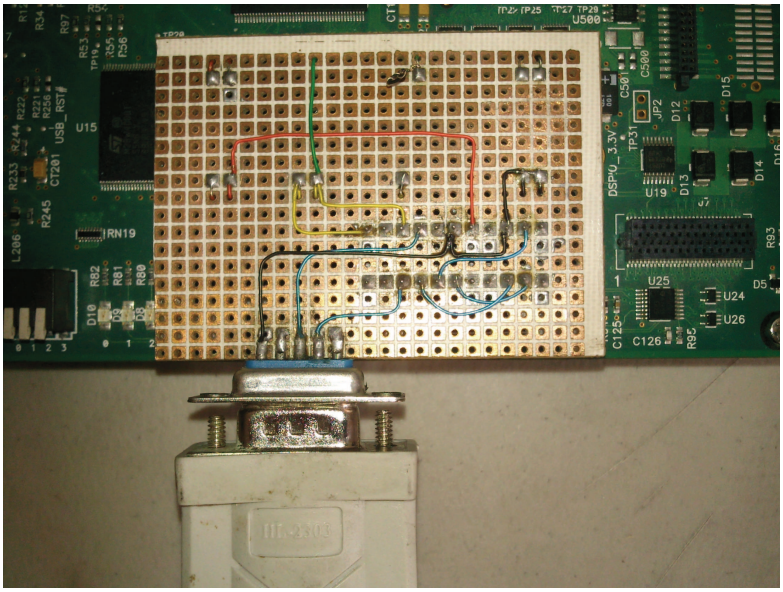

Figura C.3. Vista inferior de la tarjeta construida para la comunicación serial.

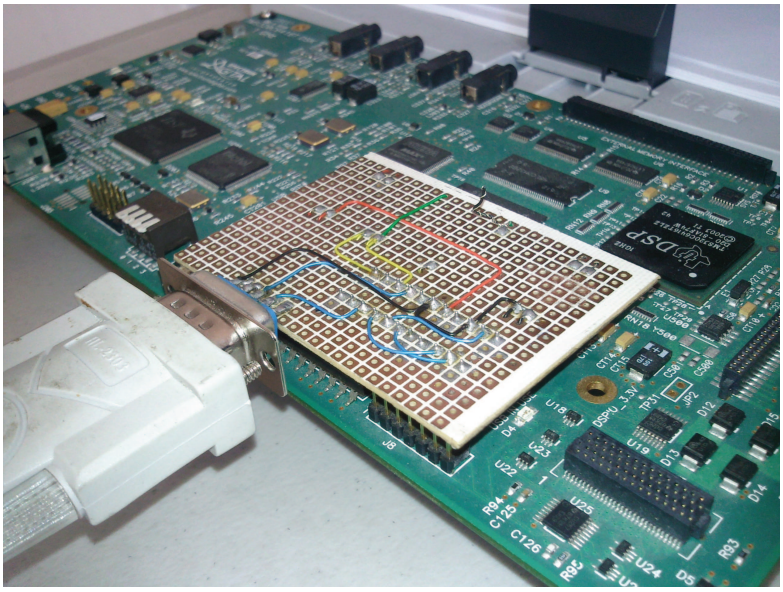

Figura C.4. Vista lateral derecha de la conexión entre la tarjeta construida y el módulo TMS320C6416T DSK.

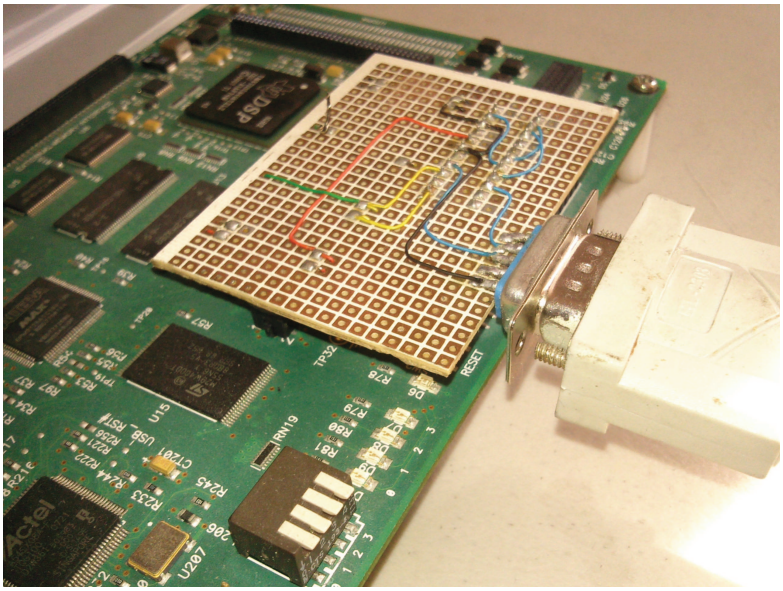

Figura C.5. Vista lateral izquierda de la conexión entre la tarjeta construida y el módulo TMS320C6416T DSK.