# INDICE ANEXOS

## INDICE DE FIGURAS

## ÍNDICE DE TABLAS

# 1. CONTROL DE ACCESO AL MEDIO (MAC - MEDIUM ACCESS CONTROL) PARA SISTEMAS POWERLINE

## 1.1 Realización de sistemas PLC( Power Line Communication)

### 1.1.1 Esquemas de Transmisión [1]

Existen varios esquemas de multiplexación que son investigados para su aplicación en los sistemas de transmisión PLC. De acuerdo a los esquemas de transmisión y al análisis del canal de transmisión PLC se pueden discutir los siguientes métodos de modulación y multiplexación:

- Multiplexación por División de Código (*CDM - Code Division Multiplexing*).
- Multiplexación por División de Frecuencias ortogonales (*OFDM - Ortogonal Frecuencies Division Multiplexing).*

CDM se caracteriza por ser poco sensible a las perturbaciones de banda angosta y a la atenuación selectiva y también trabaja con una señal de potencia baja, factor importante debido al problema de radiación de los sistemas PLC. Sin embargo, OFDM se perfila como el mejor candidato para la aplicación en sistemas de transmisión PLC con tasas de datos más altas y su excelente eficiencia de ancho de banda. OFDM suministra la transmisión de un número significativo de subportadoras de datos que presentan la característica que permite evitar el uso de frecuencias críticas usadas por otros sistemas de comunicación.

Los sistemas de transmisión OFDM usan un número de Sub-Portadoras (*SC - Sub-carriers*) distribuidas en un espectro de frecuencia (Figura 1.1). Cada subportadora tiene su propia capacidad de transmisión permitiendo crear grupos para formar los canales de transmisión (*CHs - Channels*) y aumentar la capacidad de estos canales.

Como se mencionó, existe una posibilidad de gestión de las subportadoras OFDM para evitar los espectros de frecuencia críticos utilizados por otros servicios de telecomunicaciones en la banda de operación de PLC. Esto da una oportunidad para la operación de los sistemas PLC sin interferir en el espectro de servicios importantes como son los servicios militares y de seguridad, control de vuelo, etc., y otros sistemas de radio. A pesar de esto, debido a la densa cobertura del espectro de frecuencias, la potencia de la señal en los sistemas PLC debe permanecer en un rango bajo.


*Figura 1.1. Estructura del canal OFDM*

La baja potencia del nivel de la señal en los sistemas PLC hace más difícil la transmisión a largas distancias. Esto podría solucionarse si se hace uso de los repetidores, cuya tarea es amplificar la potencia de la señal hasta un nivel suficiente y adecuado que posibilite la comunicación entre una sección de una red.

La longitud de una sección de red debe ser escogida de tal manera que se posibilite la transmisión usando un nivel de señal de acuerdo a los límites impuestos por los comités de regulación. El uso de repetidores aumenta el costo de la red.

Existen tres posibilidades para la gestión de la capacidad en los sistemas OFDM que pueden ser considerados en el desarrollo de la capa MAC [2]:

- A) un grupo de subportadoras (Figura 1.1) con una capacidad de transmisión fija para formar los CHs.

- B) un grupo de subportadoras con capacidad de transmisión variable.

- C) las subportadoras se agrupan para aumentar los canales de transmisión de acuerdo a la capacidad de transmisión disponible de las subportadoras y la capacidad de canal deseada.

En el caso A, la capa MAC negocia siempre con los canales de transmisión conservando una misma capacidad de transmisión (por ejemplo 64 kbps (*Kilo bits per second*)). Los canales de transmisión incluyen siempre las mismas subportadoras. Esto significa que si una de las subportadoras no está disponible

(una subportadora perturbada) el canal de transmisión no puede ser usado aunque las otras subportadoras estén disponibles.

Los sistemas OFDM pueden reaccionar reduciendo la capacidad de transmisión de las subportadoras de acuerdo a la situación de la perturbación (caso B). Debido a esto, es posible que la capacidad de un canal sea también reducida. En este caso, la capa MAC tiene que controlar los canales de transmisión de acuerdo a una capacidad de transmisión variable.

En el caso C, todas las subportadoras disponibles se sintetizan en un número de canales con una capacidad de transmisión. Esto significa, que un número de subportadoras se agrupa de acuerdo a su capacidad disponible para formar un canal de transmisión con una capacidad deseada. En este caso, los canales de transmisión no incluyen siempre las mismas subportadoras.

### 1.1.2 Estructura del canal [2]

Como se enunció anteriormente, los canales de transmisión pueden ser usados con una capacidad constante o variable. Si consideramos CDM como otro candidato para sistemas PLC podemos encontrar una estructura de canal similar como en los sistemas basados en OFDM. En este caso, el espectro completo disponible para transmisión es dividido en códigos ortogonales que pueden ser asignados a usuarios o servicios particulares. Esto se conoce como sistemas de Acceso Múltiple por División de Código (*CDMA - Code Division Multiplexing Access*). Con la posibilidad del uso de un código, un usuario tiene una oportunidad de enviar y recibir datos con una cierta velocidad de transmisión. Esto significa que se asigna una capacidad de transmisión al usuario como también un canal de transmisión, convirtiendo la estructura de canal en un sistema CDM.

En los sistemas con CDMA los usuarios pueden hacer uso de un espectro completo de tiempo y frecuencia para la transmisión. Si se tiene un sistema que use todas las subportadoras OFDM para cada transmisión, existe la posibilidad que se presenten colisiones. Para evitar colisiones entre diferentes usuarios, se deben definir algunas clases de organización de acceso. Entre las clases de organización está el Acceso Múltiple por División de Tiempo (*TDMA - Time Division Multiplexing Access*), en la cual al usuario se le asigna una parte de tiempo para que transmita sus datos. En los sistemas TDMA existe un número de slots de tiempo (que se repiten durante el tiempo en tramas), ofreciendo una capacidad de transmisión que puede ser comparada con canales de transmisión OFDM o CDMA.

Se puede concluir que los sistemas de transmisión PLC pueden tener una estructura de canal independiente de tecnología de transmisión usada, en el que el desarrollo de la capa MAC permitirá gestionar y negociar los canales lógicos para un protocolo MAC.

### 1.1.3 Manejo de errores [1]

Se espera que los sistemas de transmisión PLC trabajen con una señal de potencia por debajo del límite definido por las organizaciones de regulación, permitiendo la posibilidad de comunicación entre redes PLC. Esto significa que deberá existir un nivel de Relación Señal a Ruido (*SNR - Signal Noise Ratio*) en la red que haga posible la comunicación. Hasta el momento la SNR utilizada en los equipos actuales es suficiente para evitar las perturbaciones en la red sin que exista la necesidad de aplicación de otros métodos especiales contra las perturbaciones. Los valores de SNR deben ser capaces de evitar la influencia del ruido de trasfondo en una red PLC.

Dificultades adicionales para los sistemas de transmisión son causadas por el ruido impulsivo, cuyos niveles de señal son más altos que para el ruido de trasfondo. En este caso la SNR no es suficiente para evitar las perturbaciones y el daño de los datos (transmisión de error). Pero si la duración de una perturbación es suficientemente corta, de tal manera que sea más corta que una duración de un símbolo transmitido sobre la capa física de un sistema PLC, no existe influencia de la perturbación sobre la transmisión. Por ejemplo, si la duración de un símbolo OFDM es 300 us y un pulso de perturbación es más corto que 300 us no se afectarán los datos transmitidos.

En muchos sistemas de transmisión los mecanismos de Corrección de Errores hacia Adelante (*FEC - Forward Error Correction)* se aplican para hacer enfrentar las perturbaciones. En este caso, los sistemas de transmisión deben ser capaces de manejar una situación en las que se presenten un número de bits dañados, de tal manera que puedan ser corregidos y se haga posible la transmisión de los datos correctamente. Se espera que los mecanismos FEC sean implementados en sistemas PLC.

A pesar que los mecanismos FEC facilitan la corrección de errores, estos producen encabezamientos adicionales en las tramas de datos tomando una porción de la capacidad de transmisión de la red. No obstante el uso del 50 % del encabezado que usan los mecanismos FEC en Sistemas Globales para comunicaciones

Móviles (*GSM - Global System for Mobile communications)*, permiten mejorar la tasa de error de bit (*BER - Bit Error Rate*) a valores desde $10^{-3}$ (para canales de transmisión inalámbricos puros) hasta $10^{-6}$. Se debe tener en cuenta que el uso de mecanismos FEC no asegura una total corrección de errores en los datos.

Los datos transmitidos por la capa MAC son entregados a la siguiente subcapa de red: Control de Enlace lógico (*LLC- Logical Link Control*). En esta capa se puede reconocer si un segmento está libre de errores o no. En el caso que los datos presenten errores, los datos dañados deben ser retransmitidos por un mecanismo de Petición de Repetición Automático (*ARQ - Automatic Repeat Request*). El uso de mecanismos ARQ puede reducir la probabilidad de error a valores muy bajos y solamente está limitado por la probabilidad de error que se puede optimizar con el uso de mecanismos de Chequeo de Redundancia Cíclica *(CRC - Cyclic Redundancy Check)*, usados para reconocimiento de errores. Con el uso del código 240, 216 BCH-CRC[1] (24 bits de encabezado adicional a los bits de carga útil) se puede alcanzar una probabilidad de error de $10^{-8}$, que se considera suficiente adecuadamente para la transmisión de datos.

Debido al retardo de transmisión punto a punto ocasionado por la transmisión de reconocimientos *(ACKs - Acknowledgemens)* y la retransmisión de datos, ARQ no se considera adecuado para servicios de tiempo real como voz. Además, la retransmisión de datos y la transmisión de reconocimientos consumen una capacidad extra de la red, que disminuye su utilización. Por ello se espera el uso de mecanismo FEC y ARQ en los sistemas de transmisión PLC; pero para ciertos servicios (por ejemplo solamente FEC para conexiones de voz) y (por ejemplo para incrementar la utilización de la red en el caso de la transmisión de datos; la BER es reducida con el uso de FEC y de la misma manera la probabilidad de retransmisión de datos por ARQ).

Los mecanismos ARQ negocian de acuerdo a la relativa corta duración de las perturbaciones (algunos milisegundos) que muchas veces ocurren sobre una o varias unidades de datos. Si las perturbaciones son más largas harán que los canales de transmisión no estén disponibles por un tiempo más largo. En este caso, los mecanismos ARQ repetirían constantemente los datos haciendo que las transmisiones sean poco eficientes. Por esta razón, los canales perturbados no deberían ser usados para transmisiones hasta que las perturbaciones desaparezcan.

[1] BCH-CRC (*Bose-Chaudhuri-Hocquenhem-Codes CRC*) : Códigos de redundancia ciclica conocidos como BCH en honor a los investigadores que los propusieron *Bose-Chaudhuri-Hocquenhem*

Si los canales son afectados por las perturbaciones, se debe contar con un mecanismo que permita la reasignación de canales para hacer posible la comunicación. Los mecanismos de reasignación del canal deben ser también incluidos en las características de la capa MAC PLC.

## 1.2 Capa MAC en sistemas PLC

### 1.2.1 Tarea de la capa MAC [2]

El protocolo MAC debe gestionar la transferencia de datos sobre los canales de transmisión, es decir la asignación/reasignación del canal entre un número de subscriptores PLC y la transmisión de diferentes clases de servicios. La estructura lógica del canal puede tener varios esquemas de acceso: Acceso Multiple por División de Frecuencia (*FDMA - Frequency Division Multiple Access)*, TDMA, CDMA, TDMA/CDMA.

Un esquema de acceso múltiple de un protocolo MAC establece un método de división de recursos de transmisión en secciones accesibles, definidos de acuerdo a la estructura lógica de los canales utilizados.

En el sistema OFDM, los canales de transmisión se distribuyen en un espectro de frecuencia, que corresponde a una clase de FDMA. Debido a la estructura de OFDM y al número de subportadoras en cada uno de los canales de transmisión este método de acceso múltiple es llamado Acceso OFDM (*OFDMA - Access Orthogonal Frecuency Division Multiplexing)*.

Un protocolo MAC debe incluir una especificación del modo duplex, que se aplica a un sistema de transmisión. Cada uno de los métodos de acceso múltiple puede ser combinado con un esquema duplex. En sistemas de telecomunicación existen dos esquemas duplex:

- División de Frecuencia Duplex (*FDD - Frecuency Division Duplex*).
- División de Tiempo Duplex *(TDD – Time Division Duplex)*.

Una solución TDMA/FDD es aplicada en redes GSM, y la solución de Acceso por División de Tiempo *(TDA - Time Division Access*)/TDD es usada en sistemas de Telecomunicaciones Digitales Europeas inalámbricos (*DECT - Digital European Cordless Telecommunications).* Cualquiera de las soluciones duplex puede ser también aplicada a los sistemas PLC de acuerdo al esquema de acceso múltiple

usado y las características de los sistemas de transmisión PLC. Se presentan las siguientes soluciones para la realización del esquema duplex:

- Simétrico: La misma capacidad para tanto las direcciones de transmisión de enlace ascendente (uplink) y descendente (downlink).

- Asimétrico: Diferente, capacidad fija para cada dirección de transmisión.

- Modo duplex dinámico.

El modo duplex simétrico se puede encontrar frecuentemente en redes de comunicaciones. Sin embargo, no es adecuado para redes de acceso con tráfico de datos típico basado en Internet el cual se espera pueda ser prestado por redes PLC. En tales casos la carga de tráfico en la dirección de transmisión de enlace descendente (downlink) (Figura 1.2) es significativamente más alta que en la dirección de enlace ascendente (uplink) debido al hecho que el usuario/subscriptor transmite la mayoría de veces archivos más pequeños y descarga archivos más grandes de los servidores de Internet, ubicados fuera de la red de acceso.

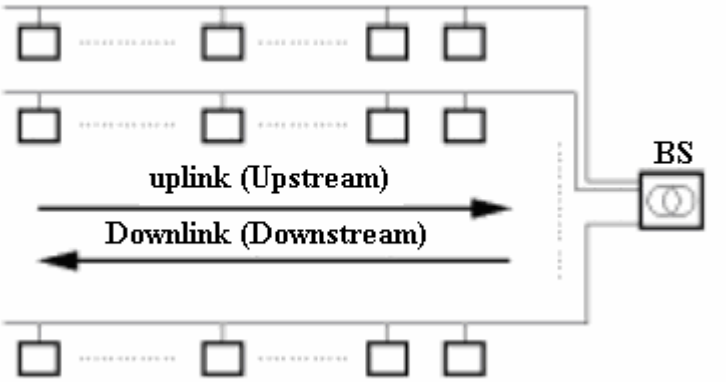



*Figura 1.2. Direcciones de Transmisión en una red de Acceso PLC*

Debido a eso, las soluciones con diferentes capacidades de transmisión en downlink y uplink son más adecuadas para la red de acceso como PLC. Como se mencionó, las capacidades de transmisión puede estar fijas (por ejemplo 10 % uplink, 90 % downlink) o la división de la capacidad de transmisión downlink y uplink gestionada dinámicamente dependiendo de la situación de tráfico en la red. El modo duplex dinámico ofrece una mejor utilización de la red pero se efectúa con una complejidad más alta en la realización del sistema.

### 1.2.2 Clasificación de protocolos MAC

Un protocolo MAC especifica una estrategia de recursos compartidos – acceso de múltiples usuarios a la capacidad de transmisión de la red – aplicada a un esquema de acceso múltiple. Generalmente, existen tres posibilidades para la organización del acceso.

- Acceso fijo
- Acceso dinámico
- Acceso basado en reserva.

Los sistemas de transmisión con esquema de acceso fijo, asignan a cada usuario una capacidad predeterminada y fija (canal o número de canales). Esto significa que un canal es asignado a un usuario (o varios usuarios) independiente de su necesidad para la transmisión de datos en ese momento (por ejemplo telefonía convencional). La estrategia de acceso fijo es adecuada para tráfico continuo, pero no para despliegue de tráfico violento (bursty) el cual es típico en la transferencia de datos.

Los esquemas de acceso dinámico son adecuados para la transmisión de datos en los que se debe asegurar una calidad de transmisión satisfactoria en servicios de tiempo real (tiempo y retardo críticos).

La reserva de la capacidad de transmisión para un usuario particular se realiza de acuerdo a la petición o demanda de transmisión, la cual se origina por el deseo de transmisión del usuario. La petición de transmisión se realiza desde el usuario hacia una unidad de red central (por ejemplo una estación base en la red) haciendo uso de cualquiera de los dos esquemas de acceso (fijos o dinámicos). Los sistemas de transmisión con esquema de reserva son adecuados para transportar tráfico híbrido (mezcla de tipos de tráfico originado por varios servicios) con una tasa de transmisión variable.

### 1.2.3 Esquemas de acceso dinámico (Señalización del canal)

En general, existen dos grupos de esquemas o protocolos de acceso dinámico (acceso al canal - señalización):

- Protocolos de contención, con colisiones.
- Protocolos de mediación, libre de colisiones.

Los protocolos de acceso de contención evitan colisiones entre las transmisiones de diferentes usuarios de red. Esto significa, que la transmisión podría no ser satisfactoria, porque otro usuario ha transmitido un dato al mismo tiempo, causando una colisión. En el caso de colisión, se realiza una retransmisión del dato originando un retardo de transmisión adicional. Por lo tanto, no existe una garantía de calidad de servicio (*QoS - Quality of Service*) para servicios de tiempo real y para cualquier clase de reserva de la capacidad de transmisión, como también para el establecimiento de las prioridades de transmisión. Una desventaja adicional es que se hace imposible alcanzar una utilización completa de la red.

Podemos encontrar los siguientes tipos de protocolos de contención:

- ALOHA, puro, ranurado.

- Acceso Múltiple por Detección de Portadora (*CSMA - Carrier Sense Multiple Access*) – 1 a n – persistente, con Detección de Colisión *(CD - Collision Detection*).

- Protocolos de resolución de colisiones.

Los protocolos libres de colisión aseguran que cada estación de red disponga de una parte adicional de la capacidad de transmisión (por ejemplo períodos de transmisión). La división de la capacidad de transmisión entre la estación de red puede ser realizada de acuerdo a los siguientes dos métodos:

- Token passing (Token-Ring, Token bus).
- Métodos polling.

Ambos métodos implementan un derecho de transmisión (por mensaje token o polling) el cual puede ser usado para una cantidad de datos particulares o un período de tiempo. Esto permite implementar una QoS garantizada en la red, que no es posible con los protocolos de contención. Sin embargo, con el incremento del número de estaciones de red el tiempo entre los derechos de transmisión para dos estaciones (tiempo de viaje redondo de mensajes token o polling) llega a ser más largo. Los protocolos Token Passing y Polling son capaces de asegurar una utilización de red casi completa, pero esto se logra sólo si un número grande de estaciones se encuentran activas (tienen datos para enviar).

Los sistemas con Token Passing tienen una organización descentralizada en la cual las estaciones de red intercambian un token proporcionando un derecho de

transmisión a otra estación de red. El método polling suministra una organización centralizada con una estación principal/central en la red. La estación principal envía un mensaje polling a las estaciones de red que quieran recibir el derecho de transmisión.

Las ventajas de la organización de acceso centralizado ofrecen garantías simples de calidad de servicio en la red. Aunque la estructura técnica de la estación de red puede ser más simple que en un sistema descentralizado, una falla de la estación de red causaría la ruptura completa de la red. Los sistemas centralizados pueden producir más encabezados de transmisión (transmisión de información de señalización).

### 1.2.3.1 Método de Acceso Aleatorio (ALOHA)

De acuerdo al protocolo ALOHA, una estación de red (Figura 1.3) trata de enviar peticiones a la estación base usando slots de petición aleatoria por el canal de señalización. En el caso de colisión con las peticiones de otras estaciones de red, las estaciones implicadas tratarán de retransmitir sus peticiones después de un tiempo aleatorio.

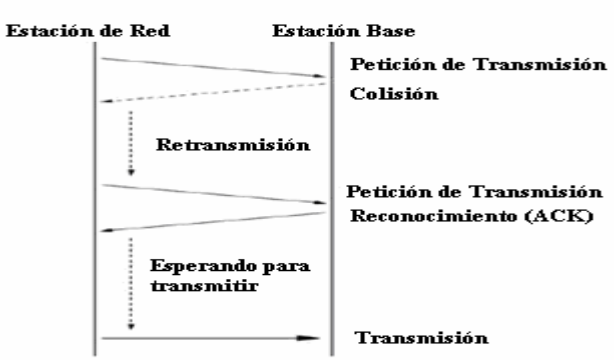



*Figura 1.3.Método de Acceso ALOHA*

### 1.2.3.2 Método de Acceso Dinámico (Polling)

El procedimiento Polling es realizado por la estación base, la cual se encarga de enviar los así llamados mensajes Polling a cada estación de red de acuerdo al procedimiento Round Robin (Figura 1.4). Solamente la estación que recibe un mensaje Polling tiene derecho de enviar una petición de transmisión. En este mecanismo no se presentan colisiones, pero las peticiones pueden ser perturbadas y en este caso deben ser retransmitidas en el próximo slot de petición dedicado.

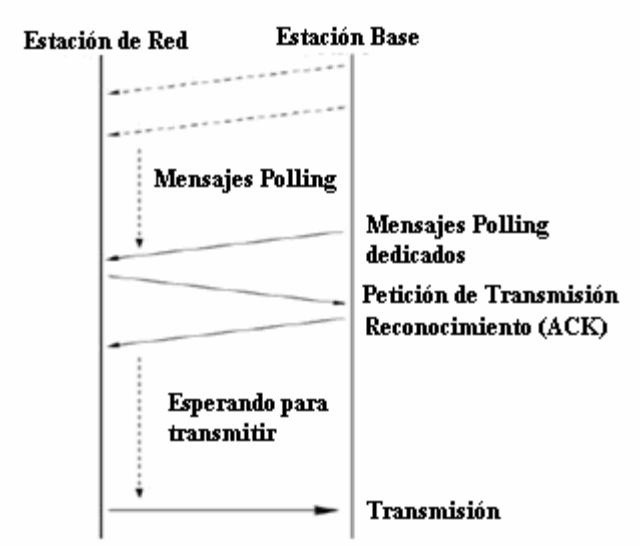



*Figura 1.4. Método de Acceso Polling*

El protocolo de reserva Polling muestra mejor funcionamiento según las pruebas obtenidas en [3], si las peticiones de transmisión son frecuentes y si la red está altamente cargada. Para una red de baja carga de tráfico, el protocolo ALOHA suministra retardos de señalización más cortos para peticiones no excepcionales y peticiones frecuentes. Si se considera la red con peticiones excepcionales, la ventaja de polling desaparece y los protocolos ALOHA suministran siempre retardos más bajos. Los protocolos basados en ALOHA son robustos contra perturbaciones debido a la naturaleza aleatoria que caracteriza este método.

#### 1.2.3.3 Protocolos de Reserva Extendidos

Algunas extensiones de los protocolos de acceso Polling y ALOHA, según las investigaciones en [3], permiten mejorar su funcionamiento. Entre las extensiones se encuentran los siguientes métodos:

- Piggybacking
- Uso de canales de datos para señalización
- Aplicación de mecanismos dinámicos backoff

Todos estos tres rasgos son adicionados a los protocolos de acceso básicos polling y ALOHA construyendo dos protocolos extendidos similares: ALOHA extendido y Híbrido –polling extendido. Ambos protocolos mantienen una organización diferente del canal de señalización. Los métodos de acceso basados en Polling se

convierten en un protocolo híbrido por a la aleatoriedad del uso de canales de datos para señalización.

#### 1.2.3.4 Piggybacking

Para disminuir la probabilidad de colisión sobre el canal de señalización con el protocolo ALOHA, en [3] se propone el uso del método de acceso Piggybacking. Una estación que transmite el último segmento de un paquete por ejemplo IP puede usar este segmento a la vez para solicitar una transmisión de un nuevo paquete, si existe uno en su cola. En este caso el canal de señalización no es usado para peticiones permitiendo disminuir la probabilidad de colisión.

De acuerdo a [3], el uso de piggybacking acorta el retardo de señalización, especialmente si la red está altamente cargada de tráfico. En el caso que una red presente peticiones frecuentes (el canal de señalización está más cargado), se alcanzaría una utilización de red con un rendimiento más alto.

### 1.2.4 Señalización sobre canales de datos.

Los retardos de señalización pueden ser también reducidos, si el número de canales de señalización se incrementa. Sin embargo, con la asignación o múltiple señalización de canales se puede perder capacidad de red valiosa y disminuir así la utilización de la red. Otra posibilidad es usar los canales de datos para el procedimiento de reserva sin asignar recursos de red adicionales para la señalización.

### 1.2.5 Mecanismos Adaptativos BackOff.

En este mecanismo, el tiempo de retransmisión después de una petición no está fijo, sino que cambia dinámicamente. El tiempo de retransmisión se puede calcular dependiendo del tráfico de la red, el número actual de estaciones activas, etc. Sin embargo, el cálculo del tiempo de retransmisión puede ser difícil por las condiciones dinámicas de la red.

El cálculo puede ser hecho por la estación base o por las estaciones de red distribuidas. En el primer caso, existe una necesidad de regeneración de la señal de información que transmite la estación base. Sin embargo, se incrementa la

dificultad y es una desventaja para las redes altamente perturbadas como. A pesar de esto en [3] se investiga varios mecanismos backoff variables sin necesidad de regenerar la señal. El algoritmo backoff que se presenta a continuación se comporta como un buen candidato para la aplicación en las redes como PLC (Figura 1.5). Este algoritmo se implementado en ambos protocolos extendidos y se usa para el acceso al canal de señalización de forma aleatoria en el caso del protocolo ALOHA extendido, como también para el acceso a los canales de datos con propósito de señalización.

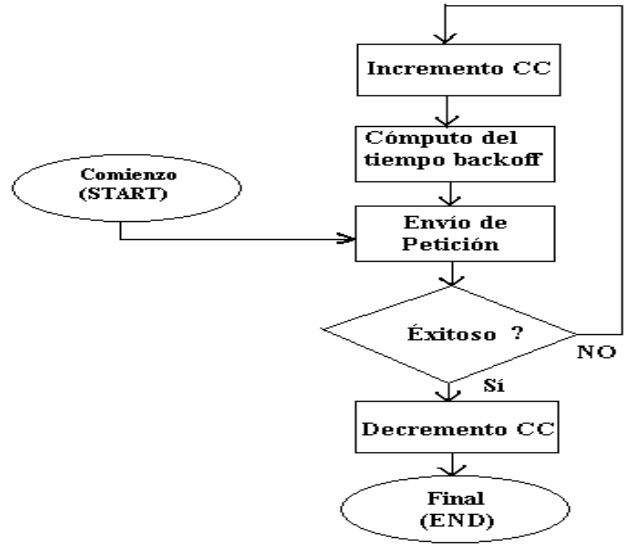



*Figura 1.5. Algoritmo Backoff*

Si un nuevo paquete llega a una estación, se realiza una petición de transmisión inmediatamente. En el caso que la petición no es satisfactoria, un contador de colisiónes (*CC - Collision Counter*) se incrementa en uno. Adicionalmente, se computa un tiempo para la petición de la retransmisión desde un Intervalo de Re - transmisión (*RI – Re-transmission Interval*) cuyo tamaño depende del valor del contador de colisión.

Así el intervalo de retransmisión se incrementa cada vez que una colisión ocurra en un tiempo predeterminado. De forma análoga, un tiempo de retransmisión se computa cuando las colisiones son frecuentes, lo que da a entender que existe más carga en la red. En este caso, un intervalo de retransmisión asegura una más baja probabilidad de colisión para las peticiones de transmisión.

Si una petición es satisfactoria, el contador de colisión se disminuye (if CC>0) y el algoritmo backoff es finalizado (Figura 1.5). De otra manera, el contador de colisión es incrementado de nuevo y la estación de red afectada computa un nuevo

tiempo de retransmisión. Así, el mecanismo backoff se adapta a la situación de carga en la red calculando siempre un nuevo intervalo de retransmisión. Se calcula un valor inicial durante un procedimiento de petición para iniciar el contador CC.

## 1.3 Organización de la capa MAC

### 1.3.1 Protocolo de acceso

La idea principal de un protocolo de acceso radica en la necesidad de transmisión de diferentes clases de servicio. En este caso, la mejor solución para la organización del acceso es ofrecida por los protocolos de reserva.
Para la aplicación de los protocolos de reserva es necesaria una capacidad adicional para transmitir la información de señalización. Esto significa que un número de canales lógicos definidos se usan para la señalización y el resto de la capacidad de transmisión está disponible para la transmisión de los datos del usuario.

### 1.3.2 Tratamiento de diferentes servicios.

El tratamiento separado de diferentes servicios de telecomunicaciones hace posible el cumplimiento de QoS particulares para cada uno de ellos. Por ejemplo considérese dos tipos de servicios; telefonía usando canales de transmisión de 64 kbps, y transmisión de datos no orientados a conexión, como dos tipos de servicio.

Con el uso de los protocolos de reserva de acceso se podría asegurar una transmisión libre de colisiones en los sistemas PLC. Los protocolos de transmisión libres de colisiones aseguran una buena utilización de la red, que es un factor muy importante en las redes PLC, debido a la baja capacidad de transmisión esperada. Si ocurre algún tipo de perturbación, es posible que la transmisión no sea satisfactoria.

Si se asume que la capacidad de transmisión de un canal lógico es 64 kbps, existe la posibilidad de asignar un canal para conexión de voz y los canales libres que no se asignan a conexiones de voz pueden ser usados para transmitir datos no orientados a conexión. Si se establece prioridad a las conexiones de voz, la llegada de una nueva conexión de voz llevaría a una reducción de la capacidad de transmisión para otro tipo de datos.

El protocolo MAC implementado en un sistema PLC debe suministrar características que permitan la realización de diferentes teleservicios que pueden ser agrupados de la siguiente manera [3]:

- Servicios orientados a conexión, como telefonía y otros servicios con Tasa de Bit Constante (*CBR - Constant Bit Rate*).
- Servicios no orientados a conexión sin garantizar QoS (por ejemplo servicios de mejor esfuerzo como Internet).
- Servicios específicos PLC.
- Transmisión de datos con QoS garantizada (servicios de Tasa de Bit Variable (*VBR - Variable Bit Rate*)).

Las redes PLC deben soportar el servicio de telefonía clásica, debido a su importancia y su gran penetración en el mundo de las comunicaciones. Otro importante servicio es la transmisión de datos, el cual permite el uso de Internet. También, una posibilidad para la transmisión de servicios más sofisticados, con requerimientos de QoS más altos (por ejemplo VBR, CBR con tasas de datos más altas) como también las características para servicios PLC específicos (automatización de casas, gestión de energía, seguridad) deberán ser incluidos en la capa MAC PLC.

### 1.3.3 Transmisión de Datos

Si existe prioridad para un tipo de servicio, se podría usar la capacidad restante para la transmisión de otro tipo de datos. Esto significa que un usuario con el derecho de transmisión puede enviar datos haciendo uso de todos los canales disponibles. La división de la capacidad de transmisión de acuerdo a los requerimientos de la QoS entre los usuarios, deberá ser suministrada por el protocolo de señalización.

La transmisión de datos podría estar organizada de la siguiente forma: la transmisión de varios usuarios se realiza en forma paralela (en el mismo tiempo) sobre diferentes canales. Por su puesto, el protocolo de señalización debe asegurar la transmisión libre de colisiones y una división aceptable de la capacidad de la red.

La decisión final de la organización del esquema de transmisión de los datos puede ser hecha luego de que sean conocidas las características de las perturbaciones. El conocimiento de la duración y los intervalos de llegada de las perturbaciones

afectarán algunos canales de transmisión, información importante para la realización del esquema de transmisión y la consideración del esquema de acceso múltiple que se usará en los sistemas PLC.

Debido a la influencia de las perturbaciones sobre las redes PLC se propone una segmentación de los datos de usuario en unidades de datos más pequeñas –segmentos PLC- con una longitud fija. Acordemente, si ocurre una perturbación es posible que sólo sean afectados un segmento PLC o un número de segmentos PLC. Los segmentos PLC afectados serán retransmitidos mediante el uso de mecanismo ARQ. En este caso las unidades de datos más pequeñas aseguran que una capacidad de red más pequeña sea usada para las retransmisiones. La longitud de un segmento PLC se debe escoger de acuerdo a las características de la perturbación.

Con el uso de pequeños segmentos de datos, existe una posibilidad de escalamiento aceptable para la capacidad de transmisión de la red. Esto hace posible una realización más simple de las garantías de QoS. Sin embargo no se debe olvidar el encabezamiento adicional a la carga útil en los segmentos PLC, en el que los procedimientos de ensamblado y re-ensamblado incrementan la complejidad del equipo de usuario (módem) y tiempo de reacción del dispositivo retardo adicional).

### 1.3.4 Impacto de las Perturbaciones [3]

Las redes PLC están caracterizadas por la gran influencia de ruido, que debe tenerse en cuenta dentro de las consideraciones de la capa MAC PLC. El ruido impulsivo asincrónico causa la mayoría de problemas afectando la correcta transmisión sobre redes PLC. Las perturbaciones ocurren sobre los canales de transmisión, causados por el ruido impulsivo que puede ser modelado por la cadena de Markovian con dos estados:

- $T_{OFF}$ representa la duración de un impulso el cual hace que un canal de transmisión no esté disponible en un instante cuando el canal está perturbado.

- $T_{ON}$ representa la duración de un período libre de perturbación.

Esos dos estados pueden ser representados por dos variables aleatorias distribuidas exponencialmente.

### 1.3.5 Protocolos de Señalización

Existe un número de canales en la red PLC usados para la organización de la señalización (canales de señalización). Debido a la capacidad de transmisión limitada en las redes PLC se propone que sólo un canal de transmisión sea usado para la señalización. Un protocolo de señalización aplicado debe asegurar:

- Eficiente transmisión de peticiones de conexión desde los usuarios a la estación base en la dirección de transmisión uplink (Figura 1.5).

- Utilización óptima de la capacidad de transmisión de señalización en la dirección downlink.

Existen una gran cantidad de protocolos que pueden ser aplicados al procedimiento de señalización en las redes PLC. Debido a la similitud en la organización de la señalización entre PLC y las redes inalámbricas, originadas por el uso de esquemas de transmisión (modulación y multiplexación) y un escenario de perturbación similar, se puede hacer uso de la experiencia de esas implementaciones e investigaciones con el fin de encontrar un protocolo adecuado para los sistemas PLC. En algunas investigaciones como en [4] se investiga el uso del protocolo ALOHA aplicado a la red móvil GSM.

# 2. MECANISMOS FEC PARA CORRECCIÓN DE ERRORES

De la teoría de comunicaciones se sabe que es posible, por codificación de k dígitos binarios en uno de M = 2k señales ortogonales, reducir la probabilidad de error de bit. Sin embargo esto se logra con un aumento del ancho de banda necesario para enviar un símbolo de información, y que aumenta proporcionalmente con M, o exponencialmente con k. Con al aumento del ancho de banda, se introduce mayor ruido al lado de receptor y la mejora en probabilidad de error es lenta comparada con el aumento de M. Los mecanismos FEC proporcionan formas de aumentar el ancho de banda mejorando la tasa de error (BER) [5].

Adicional a los códigos FEC, se usan los códigos Reed-Solomon, que son códigos correctores de error basados en bloques con un amplio rango de aplicaciones en comunicaciones digitales y almacenamiento. Los códigos Reed-Solomon se utilizan para corregir errores en varios sistemas incluyendo [5]:

- Dispositivos de Almacenamiento (Cintas, Discos Compactos, DVD, códigos de barras).
- Comunicaciones inalámbricas o móviles (Telefonía celular, enlaces de microondas, etc.)
- Comunicaciones satelitales.
- Televisión Digital/DVB[2] (*Digital Video Broadcasting*)
- Módem de alta velocidad como ADSL (*Asymetrical Digital Suscriber Line*), x DSL (*x Digital Suscriber Line*).

Dentro de las principales categorías de códigos FEC se tienen:

- Códigos en bloques
- códigos cíclicos
- códigos convolucionales
- Reed Solomon

[2] DVB: Estandar de Televisión Digital

A continuación se explicarán los códigos bloque, los convolucionales y los Reed Solomon como los esquemas de codificación más usados para construir sistemas PLC.

## 2.1 Código en bloques

En general los códigos en bloques separan el flujo de datos en bloques de k-bits, y (n-k) bits de chequeo son agregados en estos bloques. En la literatura, esto se referencia como un bloque de código (n,k). Por ejemplo un código (15,11) tiene 15 palabras de código, cuatro bits de paridad y el resto de bits de datos. Si se aumenta el número de bits de paridad es posible corregir una cantidad de errores mayor. Sin embargo, si se agregan más bits de paridad se disminuye el ancho de banda útil o se hace necesario un ancho de banda mayor para enviar los símbolos de información.

Si se aumenta el ancho de banda se introduce más ruido y la probabilidad de error también aumenta. Adicional a lo anterior, si se aumenta la longitud del código se requiere un tipo de codificador mejor y por lo general más costoso. Lo importante es escoger una longitud de código adecuada que posibilite una comunicación eficiente de acuerdo a las necesidades propias del canal de transmisión y del ancho de banda disponible.

Representando el dato k no codificado mediante un vector d:

$$d = (d1, d2, ..., dk)$$

Y la correspondiente palabra de código, representada por un vector de n-bits:

$$c = (c1, c2, \ldots, ck, ck+1, \ldots, cn-1, cn)$$

Cada paridad de bits consiste en una suma de módulo 2 ($\oplus$) de los bits de datos, por ejemplo:

$$C_{K+1} + 1 = h11\ d1 \oplus h12\ d2 \oplus \ldots \oplus h1k\ dk$$

Donde h11, h12, son los vectores binarios particulares de los bits de datos.

Por ejemplo un código (15,11), puede tener las siguientes ecuaciones de bits de paridad.

$c12 = d1 \oplus d2 \oplus d3 \oplus d5 \oplus d6 \oplus d8 \oplus d9.$
$c13 = d2 \oplus d3 \oplus d4 \oplus d6 \oplus d7 \oplus d9 \oplus d10.$
$c14 = d3 \oplus d4 \oplus d5 \oplus d7 \oplus d8 \oplus d10 \oplus d11.$
$c15 = d1 \oplus d2 \oplus d3 \oplus d5 \oplus d7 \oplus d9 \oplus d11.$

Cambiando ahora a notación matricial, podemos representar la palabra de código c como una matriz de operación sobre el dato decodificado (vector d):

$$c= d *G$$

G es la matriz generadora y debe tener dimensiones k y n, y se construye por concatenación de la matriz identidad I y la matriz de paridad P.

$$G= [I P]$$

El decodificador más simple debe repetir los cálculos de bits de paridad para determinar si cualquier error se ha producido. Matemáticamente se puede representar la operación de decodificación como:

$$d* P \oplus Cp =0$$

Donde Cp es el código de palabra que el decodificador cree que es correcta. Si ocurre un error, el decodificador puede escoger cualquier código de palabra cercano "posible" (mínima distancia Hamming hasta el vector recibido) teniendo alguna probabilidad de corrección o de lo contrario solicitar una retransmisión.

## 2.2 Código Reed Solomon [5]

Un sistema típico se muestra en la Figura 2.1:

El codificador Reed-Solomon toma un bloque de información digital y añade bits redundantes. Los errores ocurren durante la transmisión o almacenamiento de información por varios motivos (p. Ej. Ruido o interferencia, ralladuras en los discos compactos etc.). El decodificador Reed-Solomon procesa cada bloque e intenta corregir los errores y recuperar la información original. El número y tipo de errores que pueden ser corregidos depende de las características del código Reed-Solomon.

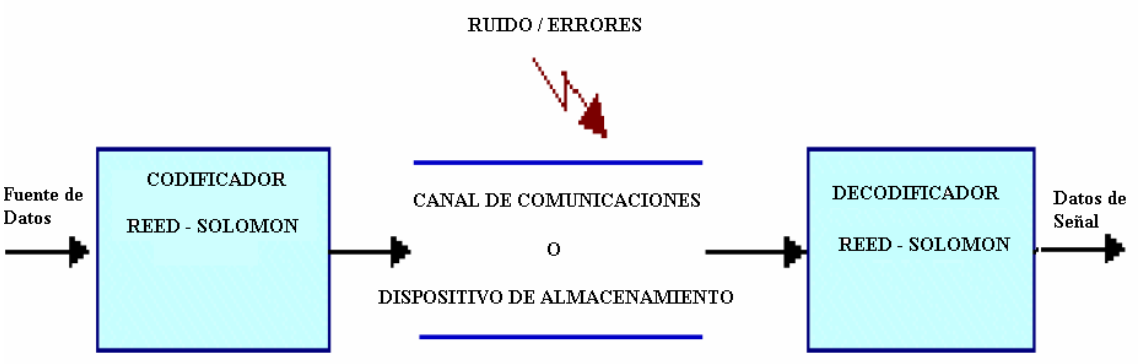



*Figura 2.1. Esquema típico de codificación Reed Solomon*

### 2.2.1 Propiedades de los códigos Reed-Solomon [5]

El código Reed-Solomon es un subconjunto de los códigos BCH y son de bloques lineales. Un código Reed-Solomon se especifica como RS[3](n,k) con símbolos de s bits. Lo anterior significa que el codificador toma k símbolos de los s bits y añade símbolos de paridad para hacer una palabra de código de n símbolos. Existen n-k símbolos de paridad de s bits cada uno. Un decodificador puede corregir hasta t símbolos que contienen errores en una palabra de código, donde 2t=n-k.

El siguiente diagrama muestra una típica palabra de código Reed-Solomon (este se conoce como un código sistemático puesto que los datos se dejan inalterados y los símbolos de paridad se anexan):

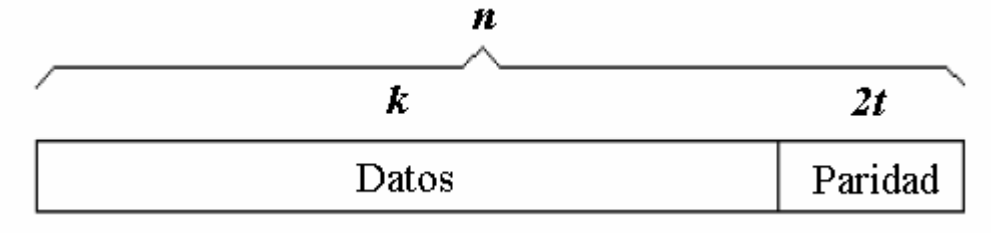



*Figura 2.2. Diagrama de una Palabra de Código Reed Solomon*

Por ejemplo un código popular Reed-Solomon es RS (255,223) con símbolos de 8 bits. Cada palabra de código contiene 255 bytes de palabra de código, de los cuales 223 bytes son datos y 32 bytes son paridad. Para este código se tiene:

N=255, k=223, s=8
2t=32, t=16

[3] RS: Codificador Reed Solomon

El decodificador puede corregir cualquier error de 16 símbolos en la palabra de código, es decir, errores de hasta 16 bytes en cualquier lugar de la palabra pueden ser automáticamente corregidos.

Dado un tamaño de símbolo s, la máxima longitud de la palabra de código (n) para un código Reed-Solomon es $n=2^S-1$.

Por ejemplo, la máxima longitud de un código con símbolos de 8 bits (s=8) es de 255 bytes. Los códigos Reed-Solomon pueden ser acortados haciendo un número de símbolos de datos igual a cero en el codificador, no transmitiendo estos, y reinsertando éstos en el decodificador.

Por ejemplo, el código (255,223) descrito anteriormente puede ser acortado a (200,168). El codificador toma un bloque de 168 bytes de datos añade 55 bytes cero, crea una palabra de código de (255,223) y transmite solo los 168 bytes de datos y 32 bytes de paridad. La cantidad de poder de procesamiento para codificar y decodificar códigos Reed-Solomon se relaciona con el número de símbolos de paridad por palabra de código. Un valor grande de t significa que un gran número de errores pueden ser corregidos pero requiere mayor poder computacional que un valor pequeño de t.

### 2.2.2 Errores de Símbolo [5]

Un error de símbolo ocurre cuando un bit en un símbolo es erróneo o cuando todos los bits en un símbolo se encuentran erróneos.
Por ejemplo el código RS (255,223) pude corregir 16 errores de símbolos. En el peor caso, errores de 16 bits pueden ocurrir, cada uno en un símbolo distinto (byte) de forma que el decodificador corrige errores de 16 bits. En el mejor caso, 16 errores de byte completos ocurren de tal forma que el decodificador corrige 16x8 errores de bit.
Los códigos Reed-Solomon son particularmente útiles para corregir burst error (cuando una serie de bits en el código de palabra se reciben con error).

*Decodificación*

Los procedimientos algebraicos de decodificación de Reed-Solomon pueden corregir errores y datos perdidos. Un "borrado" ocurre cuando la posición de un símbolo errado es conocida. Un decodificador puede corregir hasta t errores o hasta 2t "borrados".

La información sobre los "borrados" puede ser frecuentemente otorgada por el demodulador en un sistema de comunicación digital, es decir, el demodulador "marca" los símbolos recibidos que con probabilidad contienen errores.

Cuando una palabra de código es decodificada, existen tres posibilidades:

1. Si 2s+r < 2t (s errores, r "borrados") entonces la palabra de código original transmitida puede ser siempre recuperada.

   De otra forma:

2. El decodificador detectará que no puede recuperar la palabra de código original e indicará este hecho.

   O sino:

3. El decodificador decodificará erróneamente y recuperará una palabra de código incorrecta sin indicación.

La probabilidad de ocurrencia de cada una de las tres posibilidades anteriores depende del código Reed-Solomon en particular y en el número y la distribución de errores.

### 2.2.3 Ganancia de Codificación [5]

La ventaja de utilizar códigos Reed-Solomon es que la probabilidad de que quede un error en los datos decodificados es, usualmente, mucho menor que la probabilidad de ocurrencia de un error si Reed-Solomon no es utilizado. Esto se conoce usualmente como ganancia de codificación.

Por ejemplo un sistema digital de comunicaciones se diseña para operar a un BER de $10^{-9}$, es decir, no más de uno en $10^9$ bits se recibe con error. Esto puede ser obtenido aumentando la potencia del transmisor o añadiendo códigos correctores de errores. Reed- Solomon permite al sistema obtener este BER con una potencia de salida menor del transmisor. El "ahorro" de potencia dado por el código Reed-Solomon (en decibeles) es la ganancia de código.

### 2.2.4 Generador Polinomial [5]

Una palabra de código Reed-Solomon es generada usando un polinomio especial. Todas las palabras de código válidas son divisibles exactamente por el polinomio generador.

La forma general de este polinomio es:

$$g(x) = (x - a^{i})(x - a^{i+1})...(x - a^{i+2t})$$

y la palabra de código se construye utilizando:

$$c(x) = g(x)i(x)$$

Donde g(x) es el polinomio generador, $i(x)$ es el bloque de información, $c(x)$ es una palabra de código válida y "a" se conoce como un elemento primitivo del campo.

Por ejemplo el código generador para RS (255,249) es:

$$g(x) = (x - a^{0})(x - a^{1})(x - a^{2})(x - a^{3})(x - a^{4})(x - a^{5})$$
$$g(x) = x^{6} + g_5 x^{5} + g_4 x^{4} + g_3 x^{3} + g_2 x^{2} + g_1 x^{1} + g_0$$

## 2.3 Codificación convolucional [6]

La codificación convolucional es una codificación continua en la que la secuencia de bits codificada depende de los bits previos. El codificador consta de un registro de desplazamiento de K segmentos de longitud k (en total kK) que se desplaza k posiciones por ciclo y genera n funciones EXOR también por ciclo. La tasa de codificación es, entonces, R=k/n.

Un código convolucional queda especificado por tres parámetros (n, k, m):

- n es el número de bits de la palabra codificada.
- k es el número de bits de la palabra de datos.
- m es la memoria del código o longitud restringida

El número de bits por palabra de datos k, cumple:

$$k/n = R$$

A este cociente se le denomina ratio del codificador. Por ejemplo se define un codificador convolucional (2,1,3), es decir: un bit para representar la palabra de datos, dos bits de palabra de codificación por cada bit de palabra de datos y tres bits de longitud de registro de corrimiento. En nuestro caso el ratio es ½ (Figura 2.3).

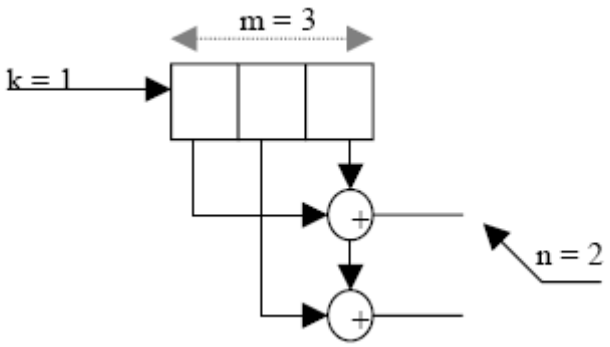



*Figura 2.3. Estructura simple de un Codificador Convolucional*

Cuanto mayor sea n, más segura será la transmisión y menor será la probabilidad de error, aunque del mismo modo, más complejo será el diagrama de estados del codificador y más difícil su posterior decodificación. Se debe encontrar, como siempre en la transmisión, el compromiso adecuado entre calidad y complejidad.
Un codificador convolucional puede ser fácilmente representado por un diagrama de estados de transición ya que tiene memoria finita. Para ello, se procede al desarrollo de los diferentes estados de la máquina. Se debe fijar que en el instante inicial el codificador está cargado con todo ceros; de acuerdo a esto se pueden ver los siguientes estados posibles:

Cargado los registros con cero, se introduce un cero y un uno (Figura 2.4):

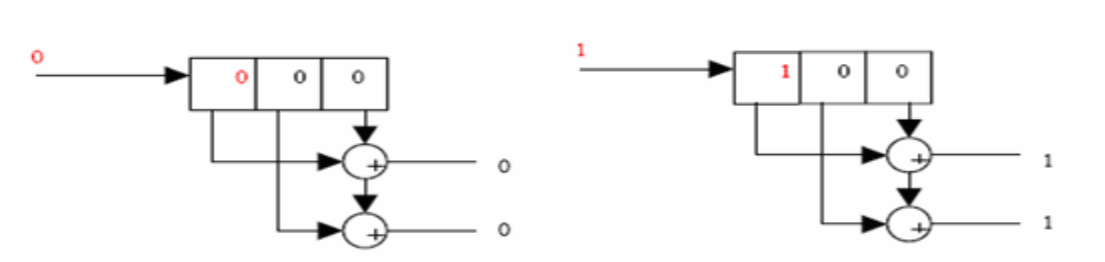



*Figura 2.4. Operación Codificador Convolucional inicializado y con entradas 0 y 1.*

Se sigue con la introducción de un uno (1) porque con el cero se puede observar la generación de un bucle. Ahora en la situación (1,0,0) se puede ver cómo codifica el sistema si se introduce un cero y cómo si es un uno el bit que entra al canal (Figura 2.5):

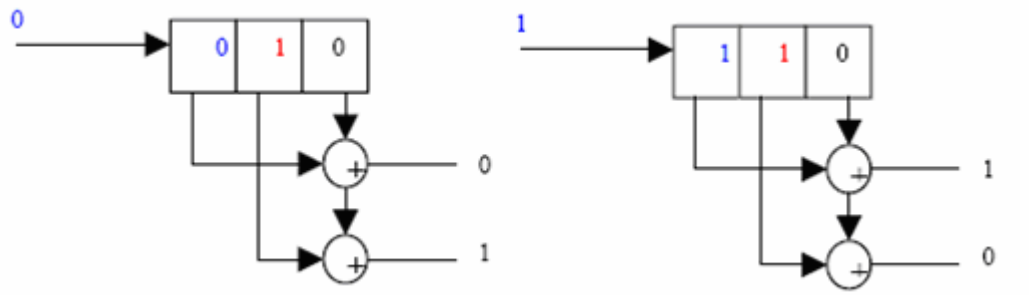



*Figura 2.5. Operación Codificador Convolucional (entrada 0 y 1)*

Y así sucesivamente hasta completar un ciclo y con él, la máquina de estados. En este diagrama, cada estado del codificador convolucional se representa mediante una caja y las transiciones entre los estados vienen dadas por líneas que conectan dichas cajas. Para saber de una manera rápida en que estado se encuentra el codificador, basta con observar los dos bits del registro de memoria más alejados de la entrada. Puede comprobarse interpretando la máquina de estados correspondiente. Para el codificador de la Figura 2.3, el diagrama de estados correspondiente es:

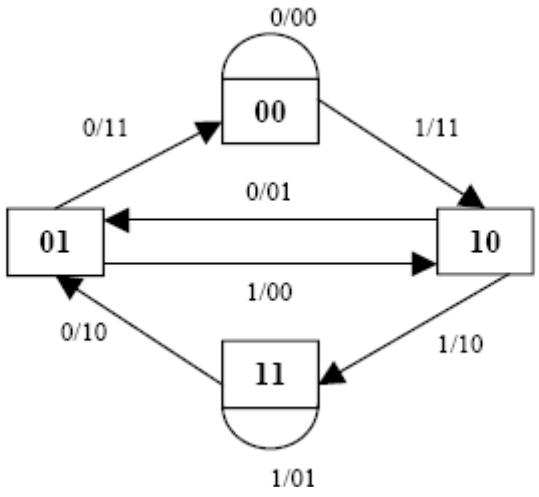



*Figura 2.6. Maquina de Estados para el Codificador Convolucional Analizado*

En cada línea viene representada la salida en función de la entrada. El número de líneas que salen de cada estado es, por lo tanto, igual al número de posibles entradas al codificador en ese estado, que es igual a $2^k$.

Una manera de representar las distintas transiciones y los caminos que éstas describen es mediante un *Diagrama de Trellis*. Una descripción de Trellis de un codificador convolucional muestra cómo cada posible entrada al codificador influye en ambas salidas y a la transición de estado del codificador. Un código de longitud restringida *m* tiene un Trellis con $2^{m-1}$ estados en cada intervalo $t_i$. Así que tendremos cuatro estados en $t_i$. De cada estado parten otros dos, uno si el bit enviado es un '1' y otro si es un '0'.
La Figura 2.7 muestra las transiciones de estados correspondientes al Diagrama de Trellis para el ejemplo visto anteriormente. Las flechas continuas muestran cómo el codificador cambia su estado si la entrada es un cero, y las flechas discontinuas cómo lo hace si la entrada es un uno:

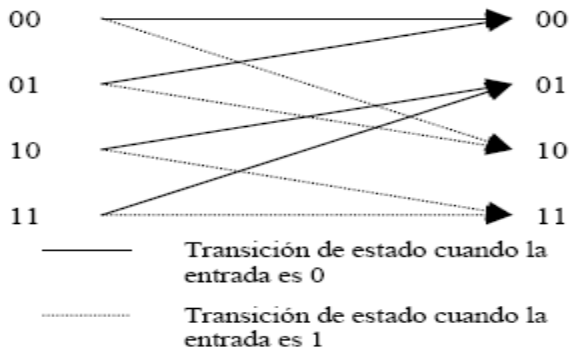



*Figura 2.7. Diagrama de Transiciones*

El sistema tiene memoria: la codificación actual depende de los datos que se envían ahora y que se enviaron en el pasado. Por lo tanto, el diagrama completo que obtenemos con nuestro codificador es (Figura 2.8):

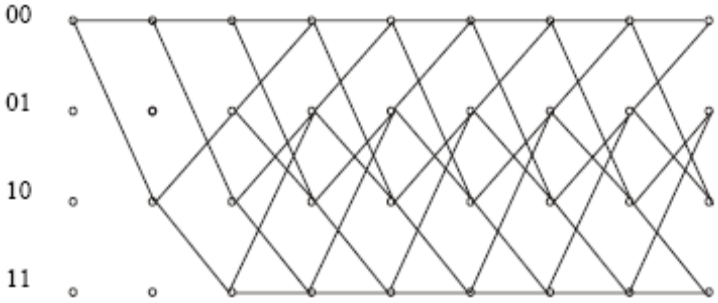



*Figura 2.8. Diagrama de Trellis*

El proceso de codificación evoluciona según el diagrama de estados. Entran al codificador k bits a la vez y las correspondientes *n* salidas se transmiten por el canal. Este procedimiento se repite hasta legar el último grupo de k bits que se codifican en una salida de *n* bits. Se acepta para una mayor sencillez a la hora de decodificar, que después del último envío de k bits, entran al codificador un grupo de k(m-1) ceros, y sus correspondientes salidas son enviadas.

En el caso (2,1,3), una vez enviado el último bit, se tendrá que enviar dos ceros para finalizar la secuencia en cero y de esta forma volver a poner el codificador a cero. Así, queda listo para otro envío futuro y se facilita por lo tanto el proceso.

Una vez explicado todo el proceso de codificación, se presenta un ejemplo a fin de ver la aplicación directa de estas bases teóricas y su funcionamiento. Para esto se divide el ejemplo en dos apartados:

- Codificación convolucional. Diagrama de Trellis.
- Decodificación haciendo uso del Algoritmo de Viterbi.

### 2.3.1 Ejemplo: Codificación y Trellis [6]

Sea una entrada de datos de 8 bits. Una vez elegido el codificador convolucional, se procede como se ha explicado en la teoría para obtener la secuencia de 16 bits correspondientes a la palabra codificada.

- Datos a la entrada: 1 1 0 0 1 0 1 1

Se ha dicho que el codificador parte del estado todo cero. De acuerdo a la máquina de estados, se parte del estado 00, si entra un 1, se ha explicado como ingresa al codificador, cuál es la salida codificada (11) y a qué estado pasa (10). A este nuevo se le llamará estado presente. En nuestro ejemplo:

- 1º Dato entrada: 1
- 1º Dato codificado: 11

- Estado presente: 10
- 2º Dato entrada: 1
- 2º Dato codificado: 10

- Estado presente: 11
- 3º Dato entrada: 0
- 3º Dato codificado: 10

- Estado presente: 01

Continuando el proceso se llega a un código de estados presentes y a la secuencia codificada que entra al canal:

- Estado presente: 10 11 01 00 10 01 10 11
- Entrada codificada: 11 10 10 11 11 01 00 10

Ahora, se representa cómo sería el camino de Trellis seguido por los datos correctos ya codificados. Para esto nos fijamos en la secuencia de estados presentes (Figura 2.9):

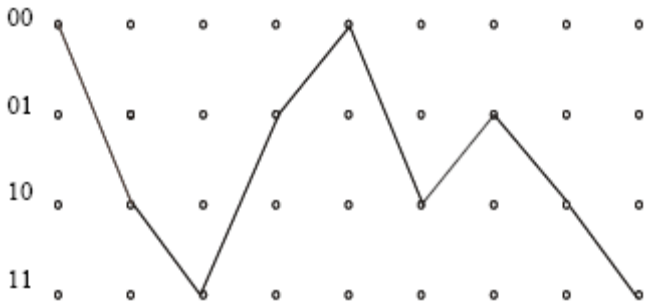



*Figura 2.9. Datos Correctos Codificados*

## 2.4 Decodificación [6]

Los mecanismos FEC detallados anteriormente, permiten codificar la señal de información de modo tal que puedan ser transmitidos por ambientes difíciles, en los que es posible encontrar una gran cantidad de errores, que afectan la correcta transmisión de los datos. Para que estos mecanismos puedan ser implementados, existe la necesidad de utilizar mecanismos de decodificación que permitan recuperar la información original codificada. Entre los mecanismos de decodificación el que sobresale es el algoritmo de codificación de Viterbi, usado cuando se codifica con códigos convolucionales, como también el algoritmo decodificador Reed Solomon usado cuando la codificación al lado del transmisor hace uso de este tipo de codificación RS.

En el caso de no poseer un mecanismo para recuperar la señal codificada, se hace necesario elegir la señal transmitida a partir de la recibida, muchas veces

contaminada con ruido y luego compararla directamente con todos los "posibles caminos" que se obtienen en el Diagrama de Trellis, además hallar la distancia Hamming a cada uno de estos (número de bits en que se diferencian) y elegir el camino que menos distanciado esté. Cuando se habla de "posibles", se refiere a que no todas las uniones que se pueden hacer en un Diagrama de Trellis son viables, ya que vienen determinados por las transiciones descritas en la máquina de estados que se han definido en el diseño del codificador. Con el fin de obtener las operaciones anteriormente enunciadas, se hace uso de la decodificación mediante una simplificación del Maximum Likelihood Decoder (ML), el cual selecciona, por definición, el valor estimado de la salida del decodificador (y' – Figura 2.10) que maximiza la probabilidad de haber transmitido una entrada concreta (r) condicionada a esa salida P(r|y').

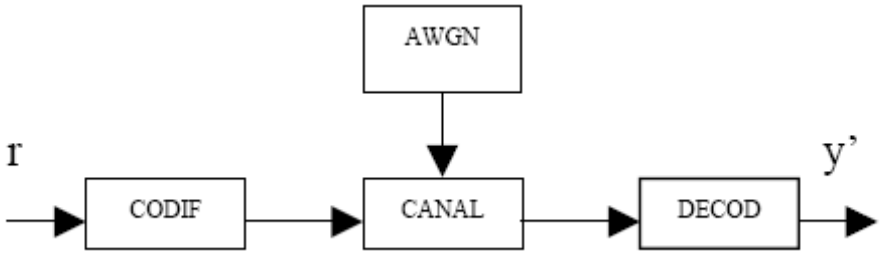



*Figura 2.10. Diagrama general de un Sistema de Transmisión*

Esta simplificación del decodificador ML es el Algoritmo de Decodificación de Viterbi. El proceso consiste en desechar algunos de todos los caminos posibles. Lo que se consigue aplicando este método es reducir el número de cálculos.

Según el algoritmo de Viterbi, para reducir el número de cálculos, cada vez que dos trayectos (también llamados ramas) se junten en un estado en el Diagrama de Trellis, el de mayor métrica acumulada se desecha en la búsqueda del trayecto óptimo. Esto se debe a que el estado actual resume la historia de todos los estados anteriores en cuanto a su influencia en los estados posteriores. Esto se hace en los $2^{k-1}$ estados, se pasa al intervalo $t_{i+1}$ y se repite el proceso.

Realmente, esta eliminación de posibles caminos no comienza hasta el tercer nivel de representación. Esto se debe a que hasta ese instante, no han podido converger dos ramas en un estado ya que por cada k bits de palabra de datos (k =1 en el ejemplo) surgen $2^{k}$ estados. Es decir, del primer estado (el 00 por definición) se tendrá dos estados en el siguiente nivel de representación, y por cada uno de estos, otros dos en el posterior nivel. Por tanto será en el tercer nivel cuando se empezará a tomar decisiones, porque ahora sí convergerán dos ramas en cada estado.

### 2.4.1 Ejemplo: Decodificador de Viterbi [6]

Una vez conocida la secuencia de 16 bits correspondientes a la palabra codificada en el ejemplo anterior, se supone una secuencia recibida con una serie de errores producidos por el ruido en el canal (en azul los bits erróneos en la recepción). Dado que también se conoce el Trellis que sigue la palabra enviada, se podrá observar gráficamente en el proceso de decodificación de Viterbi cómo se van obteniendo los caminos que conducen al resultado óptimo. De esta manera:

| | |
|---|---|
| Datos: | 1 1 0 0 1 0 1 1 |
| Estado presente: | 10 11 01 00 10 01 10 11 |
| Codificado: | 11 10 10 11 11 01 00 10 |
| Recibido: | 10 10 11 11 01 01 00 10 |

Las distintas representaciones gráficas que desarrollan el proceso según dicta la máquina de estados el ejemplo se presentarán a continuación.
Sobre cada línea de unión de estados se colocará en rojo el número de errores acumulados en relación con la señal recibida. En el momento de hacer la elección se coloreará en azul los caminos desechados:

- El primer y segundo nivel es fijo para cualquier entrada. Del estado 00 se tiene dos posibles ramas, hacia el estado 00 y hacia el 10.


*Figura 2.11. Secuencia 1 - Decodificador de Viterbi*

- Todavía no se debe realizar ninguna decisión ya que, como se ha dicho, estos caminos son los mismos independientemente de la entrada.

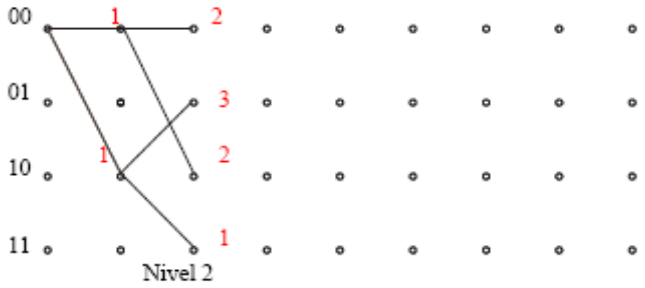



*Figura 2.12. Secuencia 2 - Decodificador de Viterbi*

- A partir del tercer nivel, en cada estado convergerán dos ramas, se analizan los errores en la métrica de cada camino posible en relación con la señal recibida.

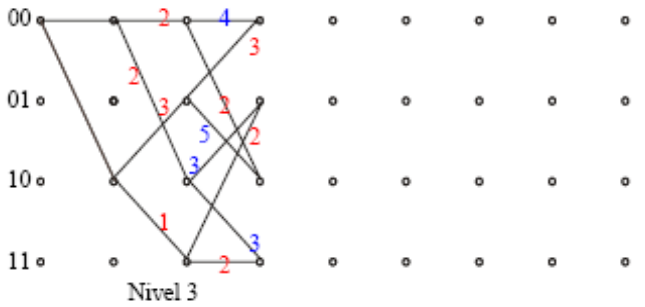



*Figura 2.13. Secuencia 3 - Decodificador de Viterbi*

- Según el número de errores acumulados en relación con la señal recibida, se elige un camino u otro. Se escoge el de menor error acumulado. Así pues, se inicia el análisis de menos ramas para el siguiente nivel del proceso:

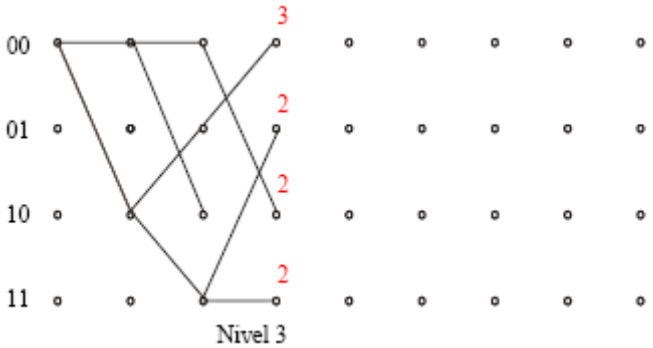



*Figura 2.14. Secuencia 4 - Decodificador de Viterbi*

- De cada nodo parten de nuevo otras dos ramas. Se colocan únicamente los errores correspondientes al último nivel, para una mejor visualización:

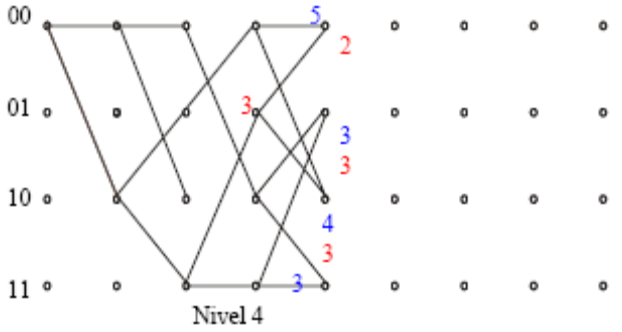



*Figura 2.15. Secuencia 5 - Decodificador de Viterbi*

- Se procede del mismo modo que para el tercer nivel y el análisis se continúa con las ramas de menor error acumulado:

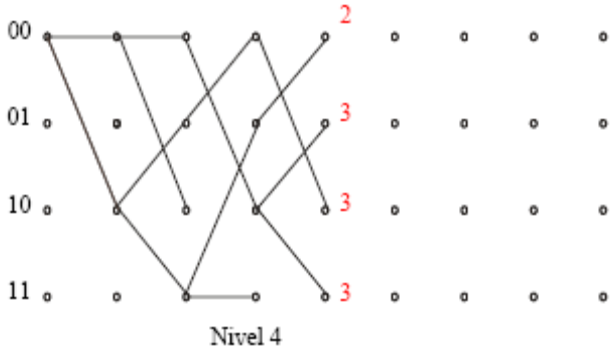



*Figura 2.16. Secuencia 6 - Decodificador de Viterbi*

- Con esta metodología aplicada al resto de niveles, las cuatro ramas seleccionadas que llegarían a los cuatro últimos estados dada su menor acumulación de error acumulada serán:

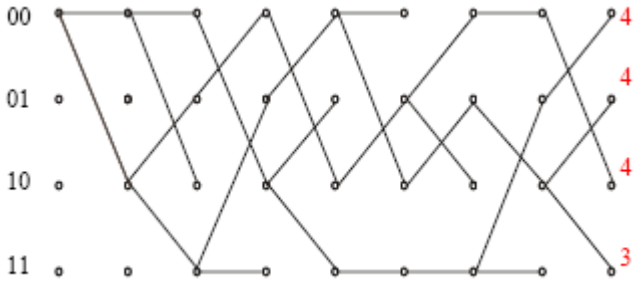



*Figura 2.17. Secuencia n - Decodificador de Viterbi*

- Por lo tanto, el trayecto óptimo es aquel que finaliza en el estado 11, ya que tiene tres errores acumulados frente a los cuatro de los otros estados. Así pues, el trayecto óptimo recorrido es:

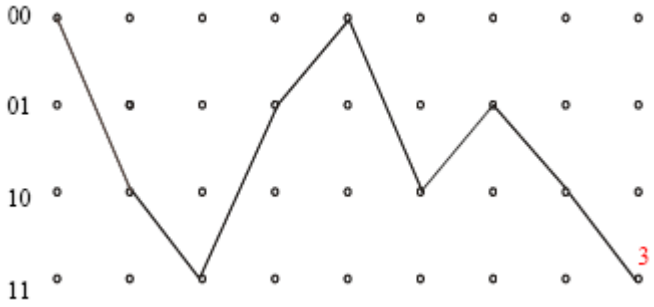



*Figura 2.18. Decodificación del los datos codificados por el Decodificador de Viterbi*

Se puede observar que coincide exactamente con la secuencia de datos enviada y codificada. Se puede concluir que mediante el Algoritmo de Viterbi es posible reconstruir perfectamente la secuencia de datos a partir de una secuencia codificada con errores introducidos por el canal.

# 3. ESPECIFICACIONES DE DISEÑO DE MODEMS OFDM y GMSK PARA EL CANAL ELECTRICO

En este apartado del documento se describe con precisión el código en Matlab de cada uno de los bloques del diseño de los modems OFDM y GMSK para el canal eléctrico. Además se da un soporte teórico de cada uno de estos bloques.

Retomando el digrama en bloques de la Figura 4.2 del documento "Solución de Acceso a una Intranet utilizando la Tecnología PLC", se realiza una descripción detallada de cada uno de los bloques que compone el sistema simulado. Es importante aclarar que el archivo de Matlab que llama todos los bloques del diseño es el archivo main.m. En este archivo se estructuran cada uno de los bloques de la Figura 4.2. Dicho archivo está compuesto por el siguiente código:

```
%******************archivo main.m************************
close all
K ;%corresponde a los datos de entrada al sistema PLC
modulador;%Corresponde al tipo de modulación
tipo_coder;
% si tipo_coder = 1 necesita un K igual a 96 datos o múltiplo pues
el %mapeo necesita
% 96 datos para un resultado de 48 símbolos que ingresan a formar el
% simbolo de 64.
% si el tipo_coder = 2 , codicador convolucional necesita un K = 48
%o
% múltiplo pues la tasa de codificación es 1/2 para obtener un total
%de 96
% datos antes de entrar al mapping
%si el tipo_coder = 3, Reed Solomon necesita un K = 64 o múltiplo de
%64
%pues como el codificador es rs(12,8) 8 símbolos de datos y 4 de
%paridad
%para un total de 12 símbolos cada uno de 4 bits, para un total de
48 %bits
%por palabra, entonces con un K = 64 se obtendrían 96 bits antes de
%entrar
%al mapping

xrand= rand(K,1); % matriz K filas y 1 columna que produce datos
%aleatorios mediante el comando rand
x= zeros(K,1); % se inicializa con ceros la matriz
for i=1:K;     % ciclo for para aproximar los valores por encima de
%0.5 a 1 y por debajo a 0
```

```
    x(i)=0;
    if xrand(i,1)>0.5
        x(i)=1;
    end
end

%*************************BLOQUE CODIFICADOR *********************
xcod= ofdm_coder(x, tipo_coder); % x es la entrada del codificador

%*********************************************************************

%************************BLOQUE INTERCALADOR************************

interleaving = inter(xcod);

%*********************************************************************
%*************************BLOQUE MODULADOR************************

if (modulador==1)%Modulador GMSK
    [T,sal]=gmskmod(interleaving);
    Period=T;
else %Modulador OFDM
    sal =ofdmmod(interleaving);
    end

%*********************************************************************
% ******************BLOQUES: CANAL, RUIDO Y ECUALIZADOR*************

snr;%Corresponde a la señal a ruido (dB)
yf = fncanal(sal,modulador,snr);

%*********************************************************************

%*************************BLOQUE DEMODULADOR************************

if(modulador==1)%Demodulador GMSK
    a=size(yf);
    datos=32;
    ltl=a(1);
    saldem = demgmsk(ltl,datos,Period,yf)';
else %Demodular OFDM
    saldem =demofdm(yf);
end

%*********************************************************************
```

```
%**********************BLOQUE DES-INTERCALADOR*********************

no_interleaving = deinter(saldem);

%*****************************************************************

%************************BLOQUE DECODIFICADOR***********************

dxcod= ofdm_decoder(no_interleaving, tipo_coder);

%*****************************************************************

if(x==dxcod)
     'no hay errores'
 else
     'hay errores'
 end

salida=x'-dxcod';
a=find(salida);
error=length(a);% # de errores

%*****************archivo main.m************************
```

### 3.1 Bloque Codificador - Codificación de la señal de información.

La codificación de la información se usa con el fin de que los datos se reciban correctamente al lado del receptor procurando que la información no se pierda por las perturbaciones del medio [7], especialmente en el canal powerline, que varía constantemente debido a los dispositivos conectados y las diferentes fuentes de ruido externas e internas al canal.

Los esquemas de codificación del canal incluyen códigos bloques, convolucionales, códigos Reed Solomon y turbo códigos.

Las líneas presentadas a continuación permiten seleccionar cualquiera de los dos tipos de codificación (ver archivo main.m en sección 3).

```
xcod= ofdm_coder(x, tipo_coder); % x es la entrada del codificador
```

La función *ofdm_coder* recibe los parámetros *x, y tipo_coder*, que corresponden a los datos a codificar (x) y el tipo de codificador a utilizar (tipo_coder).
La señal que se desea codificar será una señal binaria aleatoria cuyas líneas de código son las siguientes (ver archivo main.m en sección 3):

```
xrand= rand(K,1); % matriz N*1 que produce datos aleatorios mediante
el comando rand
x= zeros(K,1); % se inicializa con ceros la matriz
for i=1:K;     % ciclo for para aproximar los valores por encima de
0.5 a 1 y por debajo a 0
    x(i)=0;
    if xrand(i,1)>0.5
        x(i)=1;
    end
end
```

El comando *rand* permite generar los datos aleatorios de valores entre 0 y 1. Si los valores están por encima de 0.5, el valor de la señal será 1, de lo contrario será 0. Esta señal binaria es la señal que se desea codificar y la señal que va a ser transmitida por cualquiera de los dos transmisores.

Para el tipo de codificador (tipo_coder) se pueden escoger tres opciones: “1”, que corresponde a un esquema sin codificación, “2” que corresponde a un esquema de codificación convolucional, y “3” que corresponde a un esquema Reed Solomon.

```
function xcod= ofdm_coder(x, tipo_cod);

if (tipo_cod ==1)
    xcod= x;
end
if (tipo_cod ==2)
    codconvolu = xconvol(x);
    xcod = codconvolu;
end


if (tipo_cod==3)
    code_rs = rscoder(x);
    xcod = code_rs;
end
```

Si el esquema de codificación escogido es “1”, no se realiza codificación y la longitud de la señal deberá ser de 96 o múltiplo de 96 (96*n), donde n es un entero positivo. Si el esquema de codificación escogido es “2”, la función utilizada será

*xconvol(x)* y la longitud de la señal de entrada deberá tener mínimo 48 datos o ser un múltiplo de 48 (48*n), y si es Reed Solomon, la función será *rscoder(x)*, y la señal de entrada deberá tener mínimo 64 datos o ser múltiplo de 64 (64*n). A La salida del bloque codificador, la señal siempre será de longitud 96 o múltiplo de 96 (96*n).

### 3.1.1 Implementación del Codificador Reed Solomon

Como se explica en el apartado 2.2 del documento, el codificador Reed Solomon toma un bloque de información digital y añade bits redundantes.

Para la implementación de este codificador, se hace uso de las siguientes líneas de código propias de Matlab:

```
% codificador Reed Solomon

function rs_code = rscoder(x)
K = length(x);
m = 4; %bits por símbolo
n=12;
kk=8;

datodecimal = zeros (1,K/m);
dato = zeros(4,1);
i = 0;
for i=0:K/m-1
    tt = m*i+1;
    v= m*i+4;
    dato = x(tt:v,1);
    decimal= dato(1,1)* 2^3 + dato(2,1) * 2^2 +  dato(3,1) * 2^1
+dato(4,1) * 2^0;
    datodecimal(1,i+1) = decimal;
end

pcodrs = length(datodecimal)/kk;
for u=1:pcodrs

    datdecimal(1,1:8) = datodecimal(1,8*u-7:8*u);
    msg = gf(datdecimal,m);
    code = rsenc(msg,n,kk);
    binario = zeros (1,n*m);
```

```
    datbinario = [0 0 0 0 0 0 0 1 0 0 1 0 0 0 1 1 0 1 0 0 0 1 0 1 0
1 1 0 0 1 1 1 1 0 0 0 1 0 0 1 1 0 1 0 1 0 1 1 1 1 0 0 1 1 0 1 1 1 1
0 1 1 1 1];
    for b=1:n
        dat = code(1,b);
        if dat==0
           binario(1,4*b-3:4*b) = datbinario(1:4);
        end
       if dat == 1
           binario(1,4*b-3:4*b) = datbinario(5:8);
       end
       if dat ==2
           binario(1,4*b-3:4*b) = datbinario(9:12);
       end
       if dat ==3
          binario(1,4*b-3:4*b) = datbinario(13:16);
       end
       if dat ==4
          binario(1,4*b-3:4*b) = datbinario(17:20);
       end
       if dat ==5
          binario(1,4*b-3:4*b) = datbinario(21:24);
       end
       if dat ==6
          binario(1,4*b-3:4*b) = datbinario(25:28);
       end
       if dat ==7
          binario(1,4*b-3:4*b) = datbinario(29:32);
       end
       if dat ==8
          binario(1,4*b-3:4*b) = datbinario(33:36);
       end
       if dat ==9
          binario(1,4*b-3:4*b) = datbinario(37:40);
       end
       if dat ==10
          binario(1,4*b-3:4*b) = datbinario(41:44);
       end
       if dat ==11
          binario(1,4*b-3:4*b) = datbinario(45:48);
       end
       if dat ==12
          binario(1,4*b-3:4*b) = datbinario(49:52);
       end
       if dat ==13
          binario(1,4*b-3:4*b) = datbinario(53:56);
```

```
        end
        if dat ==14
           binario(1,4*b-3:4*b) = datbinario(57:60);
        end
        if dat ==15
           binario(1,4*b-3:4*b) = datbinario(61:64);
        end
     end
        pbinaria(1,48*u-47:48*u) = binario(1,1:48);%es la palabra

end
rs_code = pbinaria';
```

La función *rscoder* recibe un vector *x* con los datos provenientes de la señal de entrada (datos binarios). El codificador Reed Solomon implementado, utiliza una longitud de 4 bits por símbolo (m=4), con una longitud total de la palabra codificada de 12 bits (n=12), de las cuales 8 son datos o información. Es decir, que posee 4 símbolos de paridad (n-k = 4). Para poder codificar haciendo uso del algoritmo Reed Solomon, es necesario que los datos sean números decimales y que conformen un campo de Galois.

Las propiedades de un campo de Galois son las siguientes [8]:

- Los datos del campo de Galois, x, es una arreglo de MATLAB cuyos elementos son números enteros entre 0 y 2^m-1.

- (opcional) un número entero, m, que indica que x está en el campo GF (los 2^m). Los valores válidos de m están entre 1 y 16. El valor por defecto es m=1, que significa que el campo es GF (2).
- Un número entero positivo que indica qué polinomio primitivo para GF (2^m) se está utilizando en las representaciones de X. Si se omite, gf utiliza un polinomio primitivo por defecto para GF (2^m).

La función que permite crear el campo de Galois es *gf*. La salida de la función gf es una variable que MATLAB reconoce como un arreglo del campo de Galois, o como un arreglo de números enteros. Con lo anterior se sabe que un campo de Galois opera en arreglos de números decimales. Debido a que los datos llegan a la función *rscoder* en forma binaria (1s, 0s), se toman en grupos de 4 (m=4) para formar un dato decimal. Los datos decimales estarán entre 0 y 15 (2^m-1).

Para formar una palabra Reed Solomon (12,8), es necesario pasar 8 datos a la función gf con el fin de crear el campo de Galois, y luego mediante la función

rsenc(msg,n,k) crear el código bloque Reed Solomon, donde msg significa el mensaje que se desea codificar (arreglo de Galois), n es el longitud de la palabra que se desea, y k la cantidad de símbolos de información.

Luego de crear las palabras de código mediante la función *rsenc*, se hace necesario volver a convertir cada elemento de la palabra código en valores binarios, para enviarlos al siguiente bloque que corresponde al intercalador.

### 3.1.2 Implementación del Codificador Convolucional [9]

El codificador convolucional empleado, usa una tasa de codificación de ½. Esto significa que por cada bit de entrada, el codificador colocará en la salida dos bits, que representan la tasa de codificación.

El codificador convolucional utilizado, implementa el diagrama de estados o el diagrama de transiciones explicado en el apartado 2.3 del documento. Las líneas de código utilizado para dicha implementación en Matlab son las siguientes:

```
function codconvolu= xconvol(xsc);

msg=xsc;
trellis = struct('numInputSymbols',2,'numOutputSymbols',4,...
'numStates',4,'nextStates',[0 2;0 2;1 3;1 3],...
'outputs',[0 3;3 0;1 2;2 1]);

code = convenc(msg,trellis);
codconvolu = code;
```

La función *xconvol* recibe un vector *xsc* compuesto por datos binarios. Para codificar los datos de forma convolucional se hace uso de la función convenc(msg, trellis), donde msg es el dato que ingresa, y trellis es la estructura que define el diagrama de transiciones que se explicó en el apartado 2.3.1.

## 3.2 Bloque Intercalador o interleaving [10]

Puesto que los esquemas de codificación del canal están diseñados normalmente para tratar con errores independientes y no con errores ocurridos por interferencias o ruido, la técnica de interleaving es usada para garantizar esta independencia por errores que ocurren de forma aleatoria. Por esta razón, en el

transmisor y después de la codificación, los bits son permutados aleatoriamente de tal forma que los bits adyacentes son separados por varios números de bits [7].

Para la implementación del intercalamiento, la técnica usada consiste en organizar los datos provenientes del codificador en bloques de 16 datos, de tal forma que estos se organizan en 2 columnas. Los primeros 8 bits de dicho bloque se organizan en la primera columna y los 8 bits restantes se organizan en la segunda columna, cuando llega el segundo bloque de 16 datos los primeros 8 bits se organizan en la primera columna pero debajo de donde fueron organizados los bits del bloque anterior. Los 8 bits restantes del segundo bloque se organizarán en la segunda columna pero debajo de donde fueron organizados los ultimos 8 bits del primer bloque. Este mismo proceso se realiza para los bloques restantes. Note que se tiene un arreglo de 2 columnas por n filas, donde n depende de la cantidad de datos de entrada. Finalmente se leen los datos por filas y se organizan uno seguido de otro.

### 3.2.1 Implementación del intercalamiento [10]

La señal proveniente del esquema de codificación (convolucional, Reed Solomon) ingresará a la función de intercalamiento.

El bloque Intercalador es llamado a través de la siguiente línea de código (ver archivo main.m sección 3):

```
interleaving = inter(xcod);
```

Las líneas de código para la implementación de este bloque en Matlab son las siguientes:

```
function intercala = inter(f)
p=16;
y=8;
x=p/y;
lf=length(f);
na=ceil(length(f)/p);
lfc=na*p;
if lfc>lf,
    for n = (lf+1):lfc,
        f(n)=0;
    end
end
```

```
a=1;
for k = 0:na-1,
    % Escritura
    for i = 1:x,
        for j = 1:y,
            M(j,i)=f(a);
            a=a+1;
        end
    end
    % Lectura
    b=1;
    for j = 1:y,
        for i = 1:x,
            fi(k*p+b)=M(j,i);
            b=b+1;
        end
    end
end
intercala = fi';
```

## 3.3 Bloque Modulador

El bloque *Modulador* permitirá seleccionar entre las dos opciones de modulación o transmisión: GMSK u OFDM. Tanto el esquema GMSK como OFDM hacen uso de los tres bloques explicados anteriormente que son el codificador convolucional, el codificador Reed Solomon y el intercalador.

A continuación se explicará de forma resumida los componentes de cada uno de los moduladores.

### 3.3.1 OFDM

Retomando la Figura 4.3 del documento "Solución de Acceso a una Intranet utilizando la Tecnología PLC", se explicará el código de cada uno de los bloques del modulador OFDM

#### 3.3.1.1 Mapeo [11]

Para que una señal, ya sea analógica o digital, pueda transmitirse eficientemente a través de un medio o canal de comunicaciones, es necesario contar con algún

método de modulación. Esta modulación permitirá que la señal sea "mapeada" de forma conveniente, con el fin de que los bits sean transportados en un slot de tiempo *l*-th y sobre la subportadora *k*-th en OFDM. Aunque existen modulaciones analógicas, las modulaciones digitales han llevado a la popularidad los sistemas digitales, ya que ofrecen mayor capacidad para transportar de forma eficiente la información sobre diferentes canales de comunicación, alcanzando ventajas notables con respecto a los sistemas analógicos. Las principales diferencias radican en la mayor inmunidad al ruido, menor consumo de energía eléctrica y menor costo. Adicionalmente, en comparación con los sistemas analógicos, las técnicas de modulación digital proveen transmisiones de mejor calidad, permiten la compatibilidad con diferentes servicios digitales de datos y ofrecen mayor seguridad en la transmisión de la información [12].

En las técnicas de modulación digital, resulta práctico representar una fuente discreta de señales a partir de su "espacio de señal" o "constelación". Una constelación es una representación geométrica de un espacio de n dimensiones, en donde se visualizan todos los símbolos de salida posibles que puede generar un modulador. Gracias a que en una constelación cada símbolo tiene asociado un valor de magnitud y uno de fase (como sucede en representación polar), salvo en el caso de la modulación por Cambio de Frecuencia *(FSK - Frequency Shift Keying)*, todos los demás esquemas de modulación digital pueden representarse en un plano de dos dimensiones. La amplitud y la fase de una señal pueden modularse simultáneamente o por separado, aunque esto resulta difícil de generar y principalmente detectar. En vez de ello, resulta práctico separar la señal en dos componentes independientes conocidos como I (componente "en fase") y Q (componente "en cuadratura"), ambas ortogonales entre sí. En este caso se puede decir que la modulación digital está representada por una **Constelación bidimensional I-Q.** En una constelación I-Q, la componente "en fase" se proyecta en el eje de las abscisas (eje X) y la componente "en cuadratura" se proyecta en el eje de las ordenadas (eje y) de un plano cartesiano, como lo muestra la Figura 3.1. Una señal estará "en fase" cuando su ángulo de fase sea de cero grados (situado en el eje I) y que una señal estará "en cuadratura" cuando su ángulo de fase sea de cero grados (situado en el eje Q) [12].

En la Figura 3.1, es de notar que entre las señales existe un desfase de 90 º. También, en la representación I-Q, cada señal que se mapea en una constelación tendrá asociada una posición precisa (esto es, un punto de coordenadas (I,Q)). Es en base a esta posición que los equipos receptores pueden determinar qué señal se transmitió. Para ello, cada señal mapeada en la constelación tiene asociada una "región de decisión" (ver Figura 3.1). Sin embargo, conforme una señal se propaga

a través del canal de comunicación, ésta se verá afectada por ruido, provocando una modificación en la posición de los símbolos mapeados en la constelación. Cuando uno de los símbolos se ubica más allá de la "región de decisión" que le corresponde, éste se confundirá con alguno de los símbolos adyacentes y, en consecuencia, provocará un error de bits.

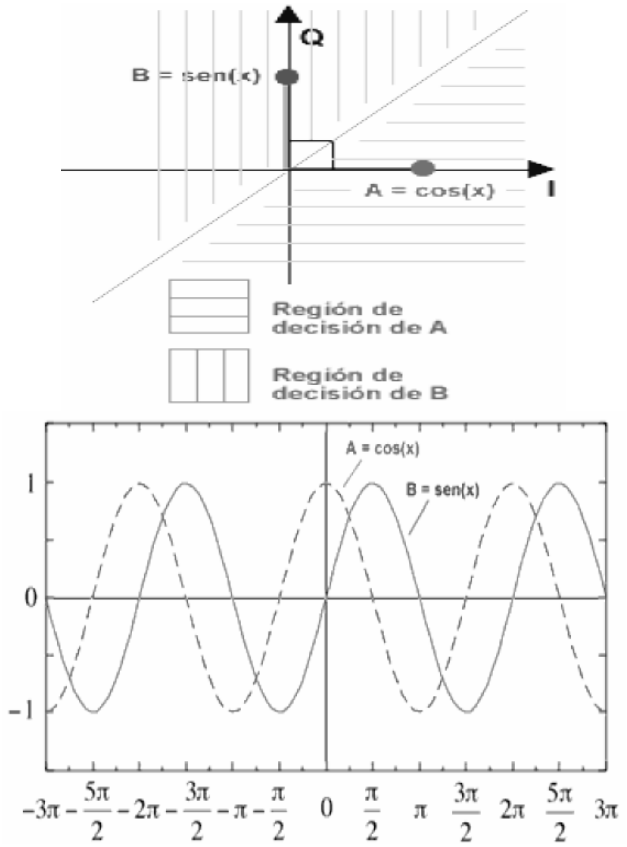



*Figura 3.1. Proyección de una señal "en fase" (coseno) y una señal "en cuadratura (seno)" en la "constelación I-Q"*

Cuando el mapeo se realiza mediante una modulación digital que utiliza una constelación bidimensional (símbolos complejos), se dice que el mapeo se logra por una codificación no diferencial, en el que los datos se mapean a símbolos de modulación compleja generalmente mediante modulaciones: por Cambio de Fase (M-PSK[4] - *M-ary Phase Shif Keying)*, por cambio de fase en cuadratura (*QPSK - Quadrature Phase Shift Keying )* o por por amplitud de cuadratura (M-QAM[5]- *M-ary Quadrature Amplitude)*. De lo contrario, se dice que la codificación es diferencial

[4] M-PSK: M hace referencia al número de combinaciones de salida
[5] M-QAM : M hace referencia al número de combinaciones de salida

donde los datos son mapeados al cociente de dos símbolos sucesivos de modulación compleja.

#### 3.3.1.1.1. Mapeo o Modulación Empleada

Para el diseño del transmisor OFDM implementado en Matlab, hacemos uso del esquema de modulación QPSK. Es un tipo de modulación MPSK que no es más que una extensión de la técnica de modulación digital por Detección de Fase (*PSK - Phase Shift keying)*. En PSK la señal modulada tiene M fases posibles para una misma frecuencia portadora. Para MPSK, al igual que PSK, las señales analógicas MPSK tienen una amplitud constante [11].

Matemáticamente se tiene:

$$M=2^K$$

Donde:

K= número de bits
M= número de combinaciones de salida posibles con k bits

Por lo tanto, lo que se hace es convertir grupos de k bits de información en una señal analógica de amplitud constante y con $2^K$ fases posibles.

En QPSK, M es igual a 4, es decir que la señal portara de frecuencia $\omega c$ puede tener 4 fases de salida diferentes, y por consiguiente k es igual a 2 ($4 = 2^2$). Entonces en QPSK los datos de entrada binarios están compuestos por grupos de 2 bits que reciben el nombre de dibits y que producen 4 posibles combinaciones: 00, 01, 10, y 11.

Para observar el funcionamiento de QPSK, el diagrama en bloques de la Figura 3.2, ilustra el diseño de un modulador QPSK

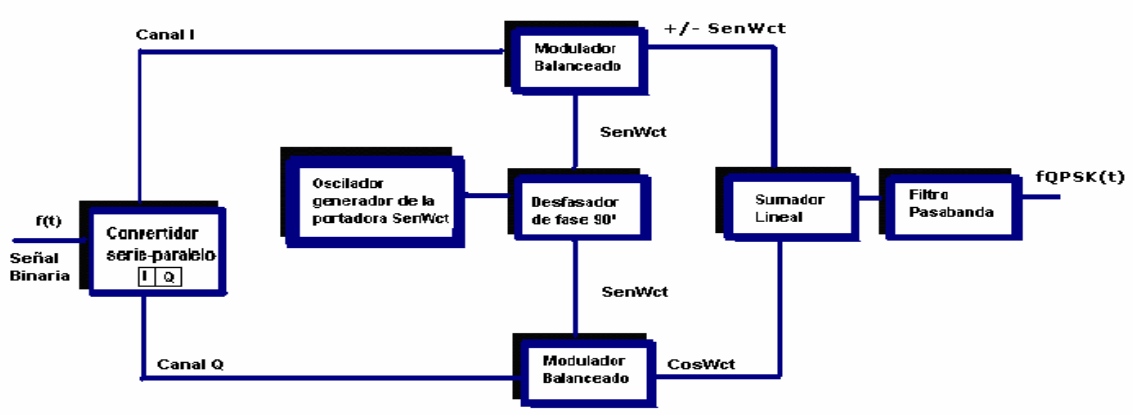



*Figura 3.2. Modulador QPSK [11]*

Una señal digital de información *f(t)* secuencial ingresa al modulador, con niveles de voltaje de $\pm 1$ voltio. En este modulador se distinguen dos canales, el I (componente en fase) y el Q (componente en cuadratura), cada uno de los cuales conducirá uno de los bits desde el convertidor serie-paralelo a su modulador balanceado correspondiente. El modulador balanceado opera de forma similar a un multiplicador analógico. El bit I, cuyo nivel de tensión puede ser -1 o +1 voltios, es multiplicado por la señal portadora ($sen\omega_c t$), mientras que el bit Q se multiplica por la portadora desplazada $90^0$ ($\cos\omega_c t$). La salida de ambos moduladores balanceados se suma linealmente para dar lugar a la señal QPSK [11].

El filtro pasabanda que se coloca a la salida del modulador QPSK tiene la función de eliminar los armónicos significativos de la señal modulada para no interferir con otras señales que pudieran transmitirse por ese mismo canal.

Los valores que puede tomar la señal de salida $f_{QPSK}(t)$ son los presentados en la Tabla 3.1.

| **Entrada binaria** | | **$F_{QPSK}(t)$** | **Fase de salida de la señal QPSK** |
|---|---|---|---|
| **Q** | **I** | | |
| 0 | 0 | $-\cos\omega_c t - sen\omega_c t$ | $-135^0$ |
| 0 | 1 | $-\cos\omega_c t + sen\omega_c t$ | $-45^0$ |
| 1 | 0 | $+\cos\omega_c t - sen\omega_c t$ | $+135^0$ |
| 1 | 1 | $+\cos\omega_c t + sen\omega_c t$ | $+45^0$ |

*Tabla 3.1. Tabla de verdad del modulador QPSK*

Su diagrama de Fasores y su constelación de puntos son los representados en la Figura 3.3.

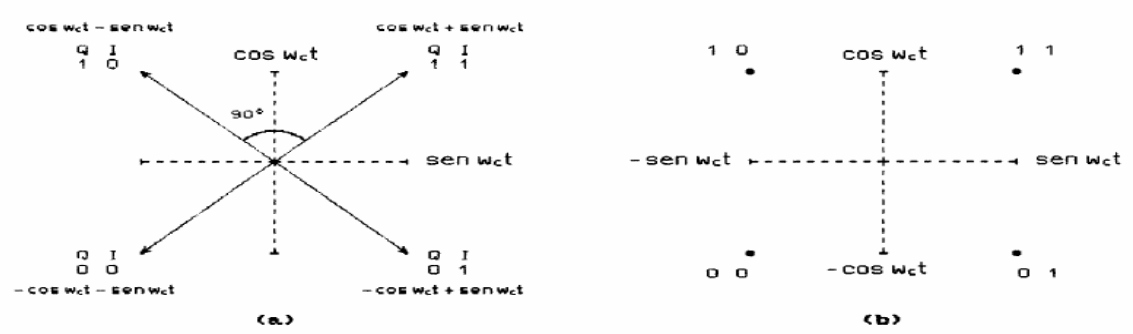



*Figura 3.3. Modulador QPSK: (a) Diagrama de Fasores. (b) Constelación de puntos*

En la modulación QPSK, como se puede observar en la figura anterior, la separación angular entre fases de salida adyacentes es de $90^0$. Además, para este modulador, cada dibit difiere del adyacente en un solo bit. Este sistema de codificación recibe el nombre de *Código Gray*, que permite disminuir el número de errores en la transmisión.

### 3.3.1.1.1.1. Implementación del bloque de mapeo QPSK para el transmisor OFDM

Para la implementación del bloque QPSK en Matlab utilizamos un esquema de codifiación Gray, tal y como lo muestra la Tabla 3.1. Las líneas de código para este bloque son:

```
function mapping= qpsk_mapping(xcod);
longPalabraCod = length(xcod);
xcomp= zeros((longPalabraCod/2),1) + j*ones((longPalabraCod/2),1);

for k =1:2:longPalabraCod;
     mm= xcod(k,1);
    nn= xcod(k+1,1);
    if  (mm==0 && nn==0)     % 00
        xcomp((k+1)/2,1)=  1+ j;
    end
    if  (mm==0 && nn==1)     % 01
        xcomp((k+1)/2,1)= -1+ j;
    end
    if  (mm==1 && nn==1)     % 11
        xcomp((k+1)/2,1)= -1- j;
    end
    if  (mm==1 && nn==0)     % 10
```

```
       xcomp((k+1)/2,1)=  1- j;
    end
end

mapping = xcomp;
plot(xcomp,'*'); % dibuja la constelación
axis ();
```

La función *qpsk_mapping* recibe un vector *xcod* en el que los datos de la componente I-Q vienen seguidos uno del otro (por ejemplo: I-Q-I-Q…I-Q). Luego, para el mapeo de los datos de acuerdo a la codificación gray, se compara grupos de dos bits, y se asigna un valor de $\pm 1$ dependiendo de los grupos que llegan (00, 01, 10, 11) y se forma un vector complejo que será pasado al siguiente bloque en el transmisor OFDM.

### 3.3.1.2 Modelo de Inserción/Estimación del Canal - Inserción de Pilotos

La función principal de un esquema de estimación del canal es identificar la amplitud y la fase de referencia de la constelación mapeada en cada subportadora a fin de que los símbolos de datos complejos puedan ser correctamente de-mapeados. La estimación del canal en sistemas OFDM requiere la inserción de símbolos conocidos o modelo de estructuras dentro de las señales OFDM. Esos símbolos conocidos producen puntos estimados de la frecuencia del canal y una operación de interpretación que indica los puntos permanentes de la respuesta de la frecuencia del canal desde puntos estimados. El funcionamiento del estimador depende fuertemente de cómo se transmite el modelo de información [13].

Según las especificaciones de los estándares HiperLAN/2 e IEEE 802.11ª, la longitud de un símbolo OFDM está formada por 64 subportadores, de las cuales 48 llevan información, 4 son los símbolos pilotos, y el resto son ceros, ubicados de manera tal que las bandas laterales sean 0 [14].

Para la distribución de los datos de acuerdo a lo enunciado anteriormente, se define la siguiente estructura de símbolo OFDM.

| Posición | 1 | 2-6 | 7 | 8-20 | 21 | 22-27 | 28 | 29-34 | 35 | 36-48 | 49 | 50-54 | 55-64 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Dato | 0 | Señal | 1 | señal | 1 | señal | 0 | señal | 1 | señal | 1 | señal | 0 |

*Tabla3.2. Estructura del símbolo OFDM*

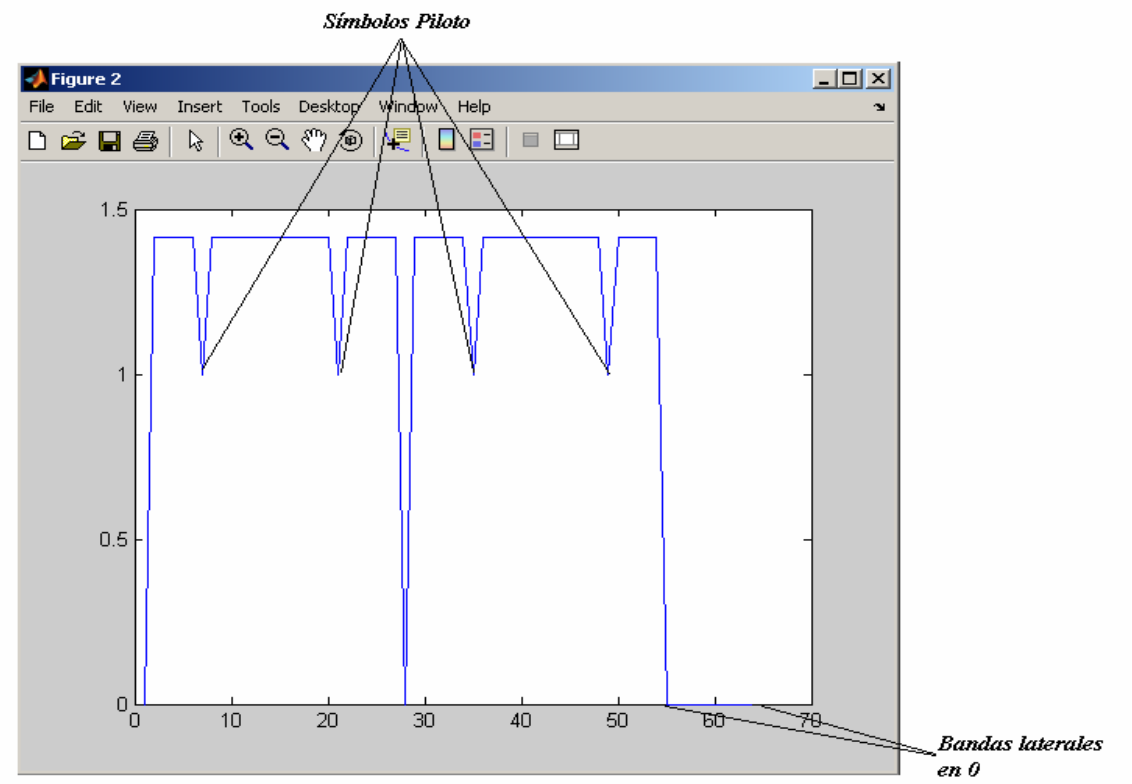



*Figura 3.4. Estructura OFDM*

Las posiciones 7, 21, 35, y 49 contienen los símbolos pilotos insertados, mientras que las posiciones 55-64 conforman los datos que permitirán hacer cero las bandas laterales de la señal OFDM. El primer dato es el cero correspondiente a la posición de continúa del espectro equivalente pasabajas.

#### 3.3.1.2.1. Implementación de la inserción de símbolos piloto y estructura final OFDM

De acuerdo a la distribución mostrada en la Tabla 3.2, las líneas de código en Matlab que permiten la creación del símbolo OFDM son las siguientes:

```
function simbolo = simbolo_ofdm(mapping);
xcompp= mapping;
longPalabraMapping = length(xcompp);
longPalabrapilot = longPalabraMapping+(longPalabraMapping*16/48);
xpilot = zeros(longPalabrapilot,1)+ j*ones(longPalabrapilot,1);

for q= 1:longPalabraMapping/48;
    xpilot(64*q-63,1)=0;                                % 1  (1)
```

```
    xpilot(64*q-62:64*q-58,1)=xcompp(48*q-47:48*q-43,1);% 5  (2-6)
    xpilot(64*q-57,1)=1;                                % 1  (7) uno
    xpilot(64*q-56:64*q-44,1)=xcompp(48*q-42:48*q-30,1);% 13 (8-20)
    xpilot(64*q-43,1)=1;                                % 1  (21)
uno
    xpilot(64*q-42:64*q-37,1)=xcompp(48*q-29:48*q-24,1);% 6 (22-27)
    xpilot(64*q-36,1)=0;                                % 1
(28)cero
    xpilot(64*q-35:64*q-30,1)=xcompp(48*q-23:48*q-18,1);% 6 (29-34)
    xpilot(64*q-29,1)=1;                                % 1  (35)
uno
    xpilot(64*q-28:64*q-16,1)=xcompp(48*q-17:48*q-5,1); % 13 (36-48)
    xpilot(64*q-15,1)=1;                                %1  (49) uno
    xpilot(64*q-14:64*q-10,1)=xcompp(48*q-4:48*q,1);    % 5 (50-54)
    xpilot(64*q-9:64*q,1)=0;                            %10(55-
64)cero

    simbolo(64*q-63:64*q,1)= xpilot(64*q-63:64*q,1);
end

plot (abs(xpilot));
figure
plot (angle(xpilot));
```

La función *símbolo_ofdm* recibe un vector *mapping* entregado por el bloque de mapeo. De este vector se toman muestras de longitud 48, con el fin de formar estructuras de 64 bits de cada una de las muestras formadas, para luego pasarlas concatenadas al siguiente bloque.

#### 3.3.1.3 Bloque Transformada Inversa de Fourier (*IFFT – Inverse Fast Fourier Transform).*

Este es el bloque principal de un transmisor OFDM. A la salida del bloque de mapeo se tienen símbolos ortogonales entre sí. A cada uno de ellos se le asignará una subportadora por medio de la transformada inversa de Fourier rápida IFFT. Sólo una pequeña cantidad de datos se transportan en cada subportadora y por esto la velocidad de bits por portadora disminuye, reduciendo significativamente la interferencia entre símbolos (*ISI - Inter Symbol interferente*).

La IFFT genera múltiples portadoras a diferentes frecuencias ortogonales, dividiendo el ancho de banda disponible entre el número N de subportadoras. La trama de datos en paralelo sujeta al proceso IFFT, constituye la modulación OFDM.

La IFFT convierte un número de puntos de datos complejos (amplitud y fase de cada componente) de longitud N la cual es una potencia de 2, en una señal en el dominio del tiempo de igual número de puntos. Después de la modulación OFDM, los datos son nuevamente convertidos a un formato serial [11].

#### 3.3.1.3.1. Implementación del Bloque IFFT

Las siguientes líneas de código, muestran la implementación en Matlab:

```
function inversetransform = ifft_ofdm(simbolo);

%IFFT TRANSFORMADA

NFFT =64;
longPalabraSimbolo = length(simbolo);
mm = longPalabraSimbolo/64;
for ii=1:mm
xsimbolo = simbolo(64*ii-63:64*ii,1);
Xifft= ifft(xsimbolo);
palabraIFFT(64*ii-63:64*ii,1) = Xifft;
end
inversetransform= palabraIFFT;
```

La función *ifft_ofdm* recibe un vector *símbolo* que contiene los datos del símbolo OFDM. Para lograr la modulación OFDM se hace uso de la transformada de fourier mediante la función *ifft(simbolo)* que realiza la transformada de acuerdo a la longitud de la señal *símbolo*.

### 3.3.1.4 Prefijo Cíclico o Intervalo de Guarda

Para mantener los efectos de la ortogonalidad de la modulación OFDM aún en los casos en que la señal pasa a través de canales dispersivos en el tiempo, existen sistemas que hacen uso de prefijos cíclicos (*CP - Ciclyc Prefix*) o intervalos de guarda. El CP no es otra cosa que la última parte del símbolo OFDM "prepegado" antes del símbolo activo, es decir que una copia de la última parte del símbolo OFDM se estaría transmitiendo antes de la transmisión del símbolo (Ver Figura 3.5). Esto introduce a la señal transmitida gran mejoría para evitar la interferencia ISI y la interferencia interportadora (*ICI - Inter-Carrier Interference*).

Cabe mencionar que el CP introduce una pérdida en la relación señal a ruido (SNR), pero esto es un precio aceptable a pagar para mitigar las interferencias ISI e ICI [14].

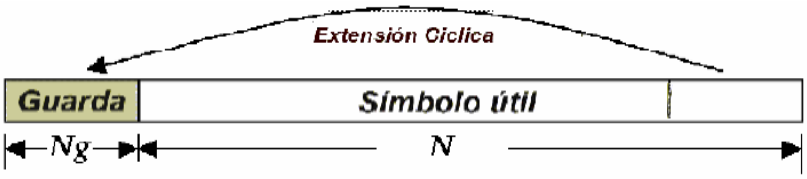



*Figura 3.5. Prefijo Cíclico*

#### 3.3.1.4.1. Implementación CP

Las siguientes líneas de código implementan la inserción del prefijo cíclico al símbolo OFDM.

```
function prefijo = prefix(palabra_transf);
palabra_transform=palabra_transf;
longPalabraifft = length(palabra_transform);
longPalabraPrefix = longPalabraifft+(longPalabraifft*16/64);
xprefix = zeros(longPalabraPrefix,1);
for i=1:longPalabraifft/64;
xprefix(80*i-79:80*i-64,1) = palabra_transform(64*i-15:64*i,1);
xprefix(80*i-63:80*i) = palabra_transform(64*i-63:64*i,1);
end
longitudp=length(xprefix)
prefijo = xprefix;
```

Para el prefijo cíclico, se realizó la copia de los últimos 16 bits de datos que conforman cada símbolo OFDM (64 datos en su estructura), y se adicionaron al principio de cada símbolo, formando un vector de 80 datos por símbolo OFDM.

### 3.3.2 GMSK

A continuación se explicara cada uno de los bloques del modulador GMSK ilustrados en la Figura 4.4 del documento "Solución de Acceso a una Intranet utilizando la Tecnología PLC".

Antes de mencionar cada bloque es importante tener en cuenta que cuando se ejecuta el programa principal main.m para operar con la modulación GMSK se debe

elegir modulador=1. El llamado de la modulación gmsk se hace a través de (ver archivo main.m sección 3):

```
% Fragmento 1

if (modulador==1)%Modulador GMSK
    [T,sal]=gmskmod(interleaving);
    Period=T;
else
    %ELSE (OFDM)
    sal =ofdmmod(interleaving);
end
```

Al elegir esta opción el programa principal llamará la función gmskmod.m:

```
%Función gmskmod.m
function [T,tf] =gmskmod(interleaving)
interleavingt=interleaving';
[T,tf]=modgmsk(interleavingt);
```

Esta función tiene como entrada la salida del bloque intercalador. Dicha entrada es interleaving, la cual se pasa en forma transpuesta a la función modgmsk. Interleaving es un vector columna con n filas, donde n es el numero de datos que salen del bloque Intercalador. Las salidas de la función gmskmod son las mismas de la función modgmsk. Estas salidas son T (Duración de cada bit de entrada) y tf (señal gmsk). Para obtener dichas salidas se llama la función modgmsk.m que internamente llama otras funciones.

```
% Función modgmsk.m
function [T,tf]= modgmsk(F)
[N_tramas,div_tramas,frame]=tramas(F);
[N_tramas,div_tramas,in]=precodif(N_tramas,div_tramas,frame);
[T,tf]=modulation(N_tramas,div_tramas,in);

La primera función que llama la función modgmsk es tramas.m

function [N_tramas,div_tramas,frame]=tramas(F);
div_tramas=32;
N_tramas=length(F)/div_tramas
for i=0:N_tramas-1% ciclo para 25 tramas
    for j=1:div_tramas
        frame(i+1,j)=F(div_tramas*i+j);
    end
end
```

tramas.m recibe la salida del bloque intercalador pero transpuesta, es decir F, que es un vector de 1 fila y n columnas. F se divide en tramas de 32 datos, porque como se explica más adelante en la función modulation, cada dato que entra a dicha función se repite 5 veces y a la salida se requieren 160 datos. La necesidad de este número de datos a la salida del modulador GMSK, se explica a continuación:

El modulador OFDM trabaja con tramas de 80 datos y a la salida de este las tramas también son de 80 datos, pero estos datos corresponden a una señal compleja en el tiempo. En los sistemas de transmisión se trabaja con señales reales en el tiempo, razón por la cual la señal que debe pasarse por el canal debe ser una señal real en el tiempo, es decir se debe convertir la señal compleja OFDM en el tiempo a una señal real en el tiempo. Este proceso de conversión se explica en 3.4.3. Una vez realizada la conversión se obtiene una señal OFDM real en el tiempo, la cual corresponde a tramas de 162 datos. Dichas tramas serán sometidas al comportamiento del canal PLC.

Con el fin de establecer una equivalencia del número de datos a enviar por el canal para ambas modulaciones y dado que las tramas que salen del modulador GMSK son señales reales en el tiempo (por lo que no requieren del proceso de conversión aplicado para una trama OFDM), a la salida del modulador GMSK deberían obtenerse tramas de 162 datos, pero este número no es múltiplo de 5, entonces el valor mas cercano es 160, lo que implica que el tamaño de una trama que entra al modulador GMSK es de 32 datos.

Las tramas que entran al modulador GMSK se guardaran en el vector frame. Note que frame guardara los datos en filas de 32 datos.

La función tramas devuelve el numero de filas (N_tramas) que corresponde al número de tramas, el tamaño de dichas tramas (div_tramas) y la trama (frame).

#### 3.3.2.1 Precodificador diferencial y NRZ

Estos bloques son bloques previos al sistema de modulación como tal. Como su nombre lo indica en el bloque precodificador diferencial se precodifican diferencialmente los bits provenientes de la función tramas.m antes de ser modulados. Este bloque se utiliza para evitar ambigüedades en la recepción. Seguido de dicho bloque se encuentra el bloque NRZ que tiene la función de

convertir los datos precodificados (cadena de 1's y 0's) en forma polar, es decir los 1's los convierte en 1 y los 0's en -1's.

#### 3.3.2.1.1. Implementación Precodificador diferencial Y NRZ

El proceso de precodificación diferencial lo hace la función *precodif*, que recibe las salidas de la función tramas.

```
%Función precodif.m

function[N_tramas,div_tramas,in] =
precodif(N_tramas,div_tramas,frame)
cod=[];
for k=0:1:N_tramas-1
    for i=1:1:div_tramas
      if i==1
            if(k==0)
                  cod(k+1,i)=xor((frame(k+1,i)),1);
            else
                  cod(k+1,i)=xor((frame(k+1,i)),cod(k,div_tramas));
            end
    else
        cod(k+1,i)=xor(frame(k+1,i),cod(k+1,i-1));
    end
  end
end
in=cod*2-1;
```

La función precodif opera de la siguiente forma:

Se realiza una x-or entre cada bit y el bit anterior. El resultado de esta operación será el bit precodificado. Esto se realiza para cada trama. Los bits precodificados se almacenan en el vector cod. Note que cuando entra el primer bit de la primera trama se asume que el bit anterior es 1, y cuando entra el primer bit de las siguientes tramas el bit anterior corresponde al último bit de la trama anterior.

Finalmente se convierten los datos precodificados en forma NRZ, los cuales se almacenan en el vector in.
La función precodif devuelve el número de filas (N_tramas) que corresponde al número de tramas, el tamaño de dichas tramas (div_tramas) y los datos precodificados en forma NRZ (in).

### 3.3.2.2 Conversor de bits a pulsos,Filtro Gaussiano,Integrador , SIN, COS, OSC.

El filtro de premodulación gausiano utilizado en GMSK- que es una clase de modulación de fase continua (*CPM – Continuos Phase Modulation*) - se caracteriza por tener una envolvente constante y ancho de banda estrecho. El prefiltrado aminora, la radiación en los canales adyacentes, pero a costa de cierta interferencia entre símbolos y una disminución del nivel de la señal, lo cual debe compensarse en la recepción. Este filtro se caracteriza por dos elementos $B_b$ que es el ancho de banda del filtro y $T$ que es el periodo de bit. La relación entre estas dos características es lo que se conoce como el ancho de banda normalizado $B_N = B_b T$. Entre menor sea $B_N$ el nivel de ISI incrementara. El filtro introduce interferencia intersimbólica en la señal transmitida, pero esta degradación no es grave si el parámetro $B_N$ del filtro es mayor de 0.5.

El parámetro $T$ (duración de bit) es el que permite convertir los bits que ingresan al sistema en pulsos. Es necesaria esta conversión pues el filtro Gaussiano opera con pulsos.

Las funciones de los bloques Integrador, SIN, COS y OSC se explicaron en la sección 4.3.1.3.2 del documento "Solución de Acceso a una Intranet utilizando la Tecnología PLC".

#### 3.3.2.2.1. Implementación Conversor de bits a pulsos, Filtro Gaussiano, Integrador, SIN, COS y OSC

Una vez se tienen los datos precodificador en forma bipolar (NRZ), se debe asignar una duración a cada bit de esta cadena. Dicha duración de bit es el periodo de la señal (T). Este parámetro se asigna a través de la variable T de la función modulation.m.

```
%Función modulation.m

function
[N_tramas,div_tramas,T,tf]=modulation(N_tramas,div_tramas,in)
T=0.5*10^(-6);
W=0.5;
W=W/T;
   for m=1:1:div_tramas*5
```

```
        tin1(1,m)=m*(T/5);%Vector tiempo se toman 5 muestras a cada
bit
    end
for k=0:1:(N_tramas -1)
   for m=1:1:div_tramas*5
        ting(k+1,m)=(k*div_tramas*5+m)*(T/5);%Se organiza el vector
%tiempo en filas.
    end
end
c=(2*pi*W)/sqrt(2*log(2));
t1=tin1+(T/2);
t2=tin1-(T/2);
x1=(c*t1);
x2=(c*t2);
conts=div_tramas*5*2;
for k=0:1:(N_tramas -1)
   for m=1:1:conts
        tin2(k+1,m)=((k*conts+m-1)*(T/10))+(T/5);
    end
end
lin=length(in(1,:));
salaux=zeros(N_tramas,lin);
aux=ones(N_tramas,5*lin);
for nt=1:1:N_tramas
y1=(sqrt(pi)/2)*erfc(x1);
y2=(sqrt(pi)/2)*erfc(x2);
g=sqrt(pi)*(1/(2*T))*(y2-y1);
tdib=-T:T/5:T
t1dib=tdib+(T/2);
t2dib=tdib-(T/2);
x1dib=(c*t1dib);
x2dib=(c*t2dib);
y1dib=(sqrt(pi)/2)*erfc(x1dib);
y2dib=(sqrt(pi)/2)*erfc(x2dib);
gdib=sqrt(pi)*(1/(2*T))*(y2dib-y1dib);
figure
plot(tdib,gdib)
title('Señal del filtro truncada y escalada')
xlabel('t')
ylabel('g(t)')
hold on
lg=length(g);
lg2=(length(g))/2;
aux=ones(N_tramas,5*lin);
for f=1:1:lin
```

```
 aux(nt,(f-1)*5+1:f*5)= in(nt,f)*aux(nt,(f-1)*5+1:f*5);%vector para
%guardar la entrada
end
z(nt,:)=conv(g,aux(nt,:));
figure
plot(tin1,z(1:lg))
title('Datos despues de pasar por el filtro')
xlabel('t')
ylabel('z(t)')
hold on
lz=length(z(nt,:));
h(nt,:)= cumtrapz(tin2(nt,1:2*div_tramas*5-1),z(nt,:));
figure
plot(tin1,h(1:lg))
title('Salida de la etapa de filtrado integrada')
xlabel('t')
ylabel('h(t)')
hold on
hi(nt,:)=cos(h(nt,:));
figure
plot(tin1,hi(1:lg))
hold on
title('Coseno de h(t)')
xlabel('t')
ylabel('I(t)')
hq(nt,:)=sin(h(nt,:));
figure
plot(tin1,hq(1:lg))
title('seno de h(t)')
xlabel('t')
ylabel('Q(t)')
hold on
auxj=zeros(nt,lin);

for l=1:1:lin
    auxj(nt,l)=hi(nt,l*5-2);
end
f=1/T
omega3=2*pi*f;
Icarrier(nt,:)=cos(omega3*tin1);
Qcarrier(nt,:)=sin(omega3*tin1);
f1(nt,:)=Icarrier(nt,:).*hi(nt,1:lg);
f2(nt,:)=Qcarrier(nt,:).*hq(nt,1:lg);
gmsk(nt,:)=f1(nt,:)+f2(nt,:);
figure
plot(ting(nt,:),gmsk(nt,(1:lg)))
```

```
title('Señal gmsk')
hold on
tf(nt,:)=gmsk(nt,1:lg);
end
```

Como se explico anteriormente, el filtro gaussiano tiene el parámetro $B_N = B_b T$ , el cual corresponde al ancho de banda del filtro. Este parámetro se asigna a través de la entrada W.

El vector tin1 es el vector de tiempo. Note que en el siguiente ciclo:

```
for m=1:1:div_tramas*5
        tin1(1,m)=m*(T/5);
end
```

se realizan divisiones de T/5 esto porque para cada bit que entra al modulador se toman 5 muestras.

A continuación se ilustran las señales g(t),z(t), h(t), I(t), Q(t) y gmsk(t). Estas señales se representan en el diagrama en bloques del modulador GMSK ilustrado en la Figura 4.4 del documento "Solución de Acceso a una Intranet utilizando la Tecnología PLC".

Estas señales son las más representativas para obtener una señal GMSK, para ello se realizo una prueba ingresando 96 datos al sistema PLC. Como se explico anteriormente si se escoge la modulación gmsk estos datos se dividen en tramas de 32 datos. A continuación se ilustraran las señales para la primera trama.

Consideremos que la primera trama que entra a la función modulation es:{1,-1, -1,-1,-1,-1, -1,1,-1,-1, -1,1,-1,1,1,1}. El periodo de cada bit es T= 0.5µseg y $B_N$ =0.5. Las señales g(t),z(t), h(t), I(t), Q(t) y gmsk(t) se ilustran en las siguientes Figuras:

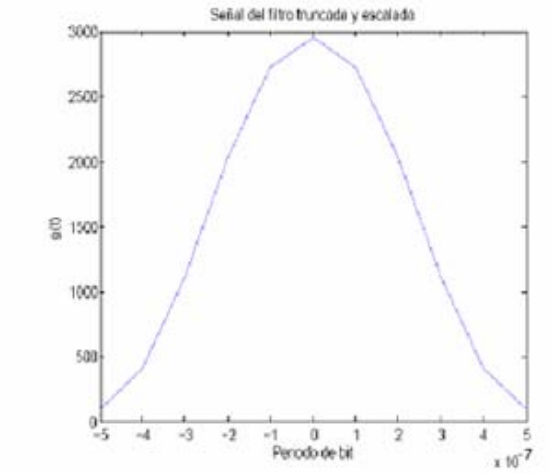



*Figura3.6 g(t): Respuesta del filtro entre –T y T*

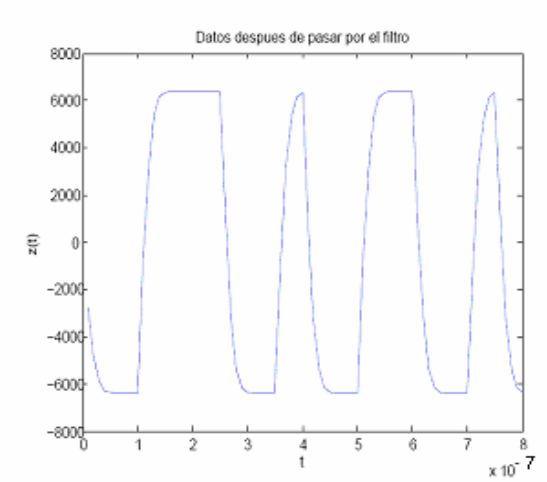



*Figura3.7. z(t) : Pulsos pasados a través del filtro*

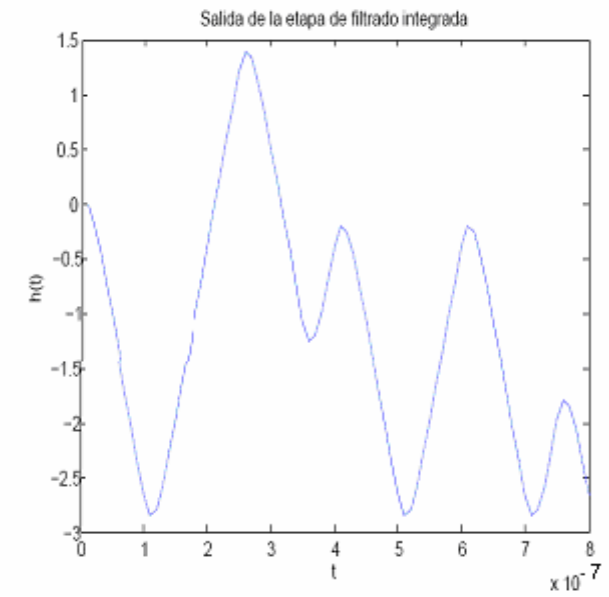



*Figura 3.8. Señal h(t)*

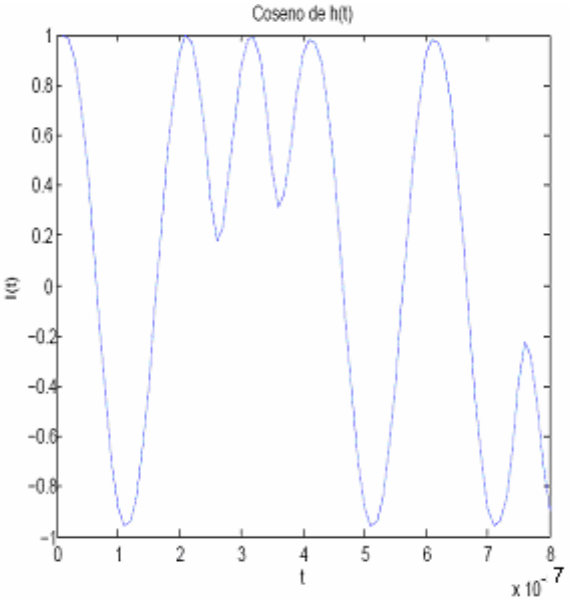



*Figura 3.9. Señal I(t)*

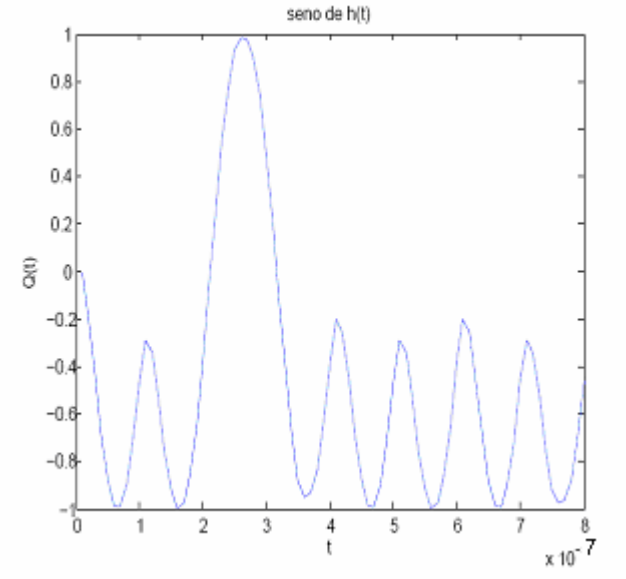



*Figura 3.10 Señal Q(t)*

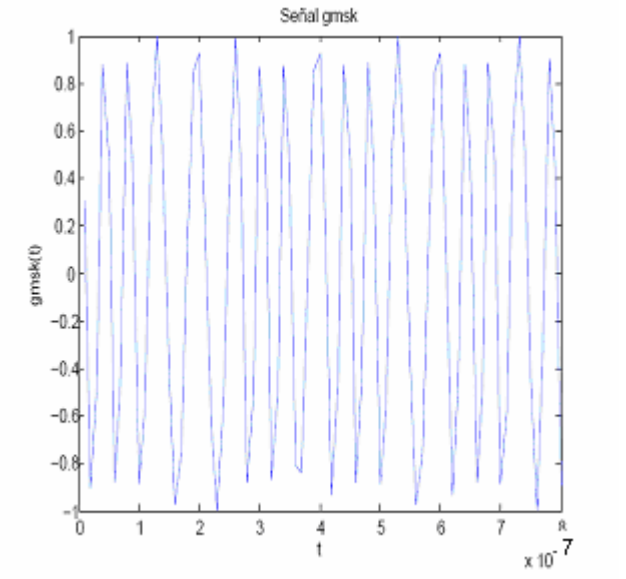



*Figura 3.11 Señal gmsk(t)*

Desde las figuras 3.7–3.11, el tiempo esta entre 0 y 16*(0.5 µseg)= 8 µseg, ya que cada bit dura 0.5 µseg.

La función modulation devuelve: la duración de cada bit (T) y la señal gmsk modulada (tf). Estas dos salidas son también las dos salidas de la función modgmsk, que a su vez es llamada por la función gmskmod. De esta forma gmskmod también devolverá estos dos parámetros.
Retornando al programa principal (Fragmento 1) las salidas de la función gmskmod se guardan en las variables T y sal.

## 3.4 Bloque Canal [15]

Existen dos formas de transmitir una señal digital. La primera, transmisión en *banda base*, permite transmitir la señal directamente a través del canal de comunicaciones sin efectuar ningún tipo de modulación. La segunda, transmisión en *banda ancha*, consiste en *modular* la señal con alguna técnica de modulación digital antes de ser transmitida por el canal de comunicaciones. *Modular una señal*, consiste en variar una o varias características (tales como amplitud, frecuencia o fase) de una señal (portadora), en función de las variaciones de la señal que contiene la información a transmitir. (Cabe aclarar que para el modelo diseñado, se ha hecho uso de modulaciones digitales como GMSK y QPSK para el modulador OFDM)

En un sistema real la señal proveniente de los dos moduladores se pasa a través de un conversor Digtal/Analogico *(D/A - Digtal/Analog*) para producir una señal análoga en banda base, la cual es modulada en Radio Frecuencia (RF - *Radio Frecuency)* para ser transmitida (banda de operación PLC). La señal se transmite a través de un canal que agrega diferentes clases de ruido, que perturban la señal.

En sistemas reales los convertidores D/A ni Analogico/Digital (A/D – *Analog /Digtal)* son necesarios para la correcta elaboración del transceptor, pero a manera de simulación no son necesarios, razón por la cual para los moduladores desarrollados en código se omitieron.

Las características del canal dependen del medio de transmisión por el cual se transmitirá la señal de información. Para canales inalámbricos el canal de transmisión es el aire; de forma análoga para sistemas alámbricos el medio será cables de cobre u otro tipo de cables adecuados para la transmisión.

El caso de interés es el medio de comunicaciones Powerline o línea eléctrica, que se analizará de acuerdo a sus condiciones.

El modelo del canal utilizado se describión en la sección 4.3.1.4 del documento "Solución de Acceso a una Intranet utilizando la Tecnología PLC".

### 3.4.1 Implementación de canal PLC

Es importante aclarar que en los sistemas de transmisión las señales que se tratan son reales. La señal que sale del modulador GMSK es una señal real en el tiempo, mientras que la señal que sale del modulador OFDM es una señal compleja en el tiempo, por lo que la señal OFDM debe convertirse en real. Adicional a esto para hacer una simulación más cercana a un sistema real, las señales deben operar en RF, es decir el espectro de la señal a pasarse por el canal debe corresponder con el espectro del canal. Para hacer coincidir el espectro de la señal con el del canal se debe trasladar la frecuencia de la señal al espectro del canal.

Una vez tenemos la señal modulada (sal), esta debe pasarse por el canal. Este proceso se hace a través del llamado de la función fncanal, así;

```
yf = fncanal(modulador,sal);
```

Note que una vez se tiene modulada la señal, ya sea OFDM o GMSK, dicha señal se pasara a través del canal. Primero explicaremos la función que realiza el espectro

del canal PLC, es decir rtacanal.m. Una vez se realiza el espectro del canal, la función fncanal llama a la función rtacanal.

De acuerdo a la función de transferencia del canal y sus parámetros asociados planteado en el documento [15], es posible construir un modelo en Matlab que permita representarlo. Esta representación se logra de acuerdo a las siguientes líneas de código y de acuerdo a la Tabla 3.4 (función canal):

```
% Modelo 4 rutas zimmermann.pdf

function Canal = canal(tipomod);
if (tipomod==1)
    lndato=0.5*160*5-1;
else
    if(tipomod==2)
    lndato=100*4;
    end
end
gi = zeros(4,1);
%gi = [0.029, 0.043, 0.103, -0.058, -0.045, -0.040, 0.038, -0.038,
0.071, -0.035, 0.065, -0.055, 0.042, -0.059, 0.049];
gi = [0.64, 0.38, -0.15, 0.05];

%di/m  = velocidad de fase
di = zeros(4,1);
di = [200, 222.1, 244.8, 267.5];

% i = numero de rutas
i = 4;

%k
epsilum =4;
k=1;
a0= 0;
a1= 7.8 * (10^(-10));

%f = [-pi:2*pi/512:pi-(2*pi/512)];
freq = [0.3*10^(6):20000000/lndato:20000000-20000000/lndato];
w= length(freq);
H = zeros(1,w);

for r=1:i
 H = H + ( gi(1,r) * exp(-(a0 + a1*freq)*di(1,r)) .* exp(-j *
2*pi*freq*di(1,r)*sqrt(epsilum)/(3 * 10^8)) );
end

if(tipomod==1)
freq1=[0 freq];
H1=[10^(-1.7) H];
%%%%%
```

```
figure
plot(freq,10*log(abs(H)))
title('Espectro del canal Modelo Zimmermann')
ylabel('Potencia (dB)')
xlabel('Frecuencia (hz)')
%%%%%%%%
figure
plot(freq1,10*log(abs(H1)))
title('Espectro del canal Modelo Zimmermann con nivel DC bajo')
ylabel('Potencia (dB)')
xlabel('Frecuencia (hz)')
%%%%%%%%
l=length(H1);
num=(2:l);
DD= [conj(H1(1)),conj(H1((l+2)-num)), H1(1:l)];

l1=(length(DD)+4)*0.5;
for u=0:(20000000/380):20000000-(20000000/380)
% f1GMSK y f2GMSK corresponde a las frecuencias donde se ubica un canal
%GMSK
% f3GMSK y f4GMSK corresponde a las frecuencias donde se ubica otro
canal GMSK



a=find(freq >=f1GMSK*10^(7) & freq <f2GMSK *10^(7) );
b=find(freq >=f3GMSK *10^(7) & freq <f4GMSK *10^(7) );
end
G1=DD(a-1+l1);
G2=DD(b-1+l1);
G=[G1 G2];

else
freq1=[0 freq];
H1=[10^(-1.7) H];
l=length(H1);
num=(2:l);
DD= [conj(H1(1)),conj(H1((l+2)-num)), H1(1:l)];

l1=(length(DD)+4)*0.5;
for u=0:(20000000/lndato):20000000-(20000000/lndato)
% f1OFDM y f2OFDM corresponde a las frecuencias donde se ubica un canal
%OFDM

a=find(freq >=f1OFDM*10^(7) & freq < f2OFDM *10^(7) );
end
G=DD(a-1+l1);
end
freq2 = [-20000000:20000000/(0.5*length(DD)):20000000-
(20000000/(0.5*length(DD)))];
%%%%%%%%%
figure
plot(freq2,10*log(abs(DD)))
title('Espectro par del Canal PLC')
```

```
xlabel('Frecuencia (Hz)')
ylabel('Potencia del espectro (dB)')
hold on
%%%%%%%%%
Canal=G;
save('Canal2.mat');
```

H corresponde al comportamiento del canal PLC del modelo de Zimmermann. Note que H esta en el dominio de la frecuencia y al graficar el espectro del canal se obtiene la figura 3.12.

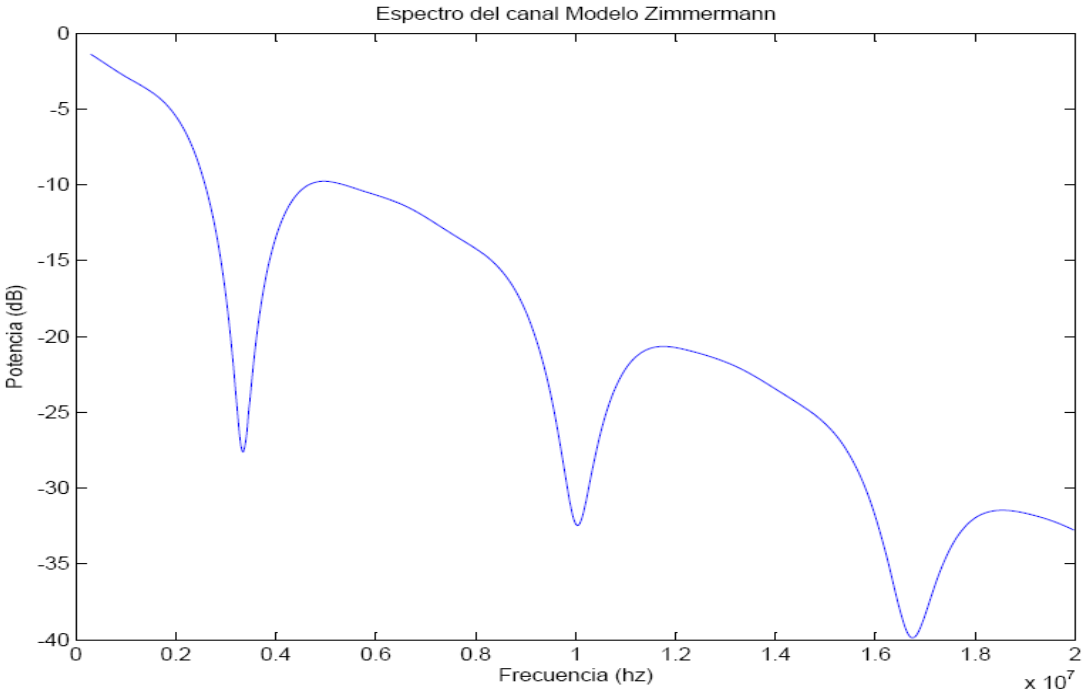



*Figura 3.12. Espectro canal PLC Modelo Zimmermann*

La Figura 3.12 corresponde a un espectro complejo y teniendo en cuenta que los sistemas de transmisión operan con señales reales en el dominio del tiempo, la señal del canal debe convertirse al dominio del tiempo y debe ser real. La manera para convertir el canal PLC al dominio del tiempo es calcular la ifft de H, pero esta señal seria compleja y no correspondería a los propósitos que tenemos de hacer una simulación teniendo en cuenta la operación de los sistemas de transmisión.

Note que en la Figura 4.5 (a) del documento "Solución de Acceso a una Intranet utilizando la Tecnología PLC", el modelo de Zimmerman no esta diseñado para la frecuencia de 0 Mhz, razón por la cual se estableció un nivel DC muy bajo de -70 dB. Dicho nivel DC se hizo a través del vector H1. Es decir el vector H (Figura 3.12) describe el modelo de Zimmerman, pero H1 tiene en cuenta la frecuencia de 0 Mhz

para el modelo de Zimmermann. La Figura 3.13 ilustra el vector H1, que tiene en cuenta la frecuencia de 0 Mhz para el modelo de Zimmermann.

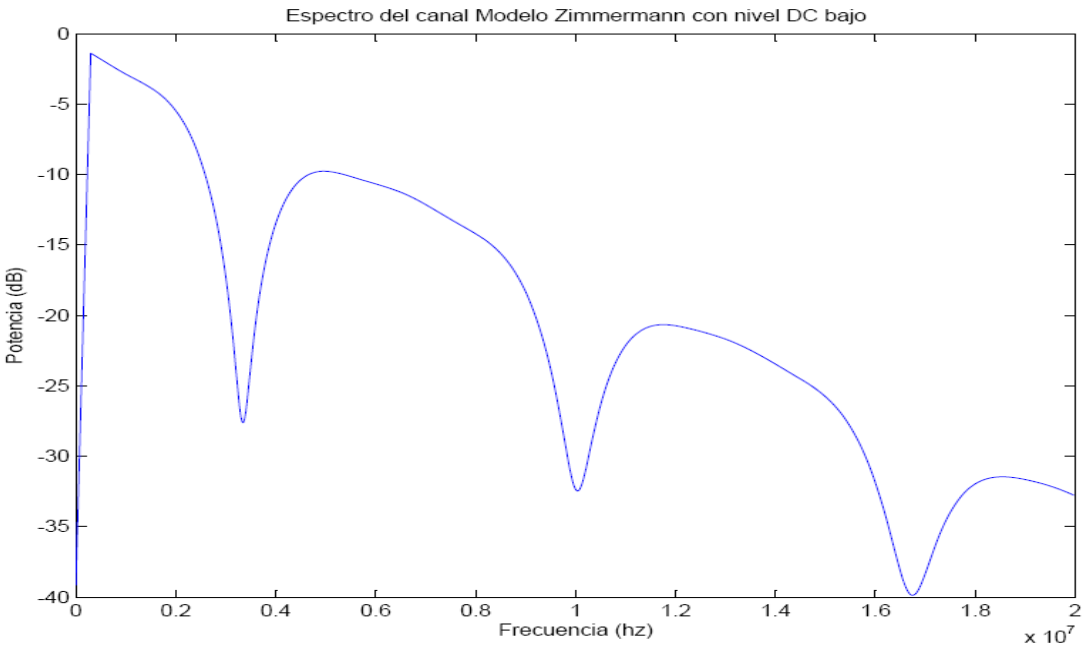



*Figura 3.13. Espectro canal PLC Modelo Zimmermann con nivel DC bajo*

El espectro de una señal real es par, es decir que si tenemos la mitad de dicho espectro podemos volver ese espectro simétrico, para así obtener la señal real nuevamente. De acuerdo a esto H1 correspondería a la mitad del espectro y al volverlo simétrico y convertirlo en el dominio del tiempo obtendríamos una señal real, que cumpliría con el propósito mencionado anteriormente.

Una señal real con longitud par *L* tiene un espectro simétrico alrededor de (*L*/2) +1. El punto de simetría es un valor real.

Es decir que el primer valor del espectro complejo, en este caso H1 debe ser un valor real. Efectivamente el primer valor de H1 es real, y el proceso para convertir el canal en real y en el dominio del tiempo es:

El punto de simetría de la nueva señal será el primer elemento de H1, de tal forma que H1 estará en la mitad derecha. La mitad izquierda se formara así: el ultimo elemento de la mitad izquierda será el conjugado del segundo elemento de la mitad derecha, el penúltimo elemento de la mitad izquierda será el conjugado del tercer elemento de la mitad derecha y así sucesivamente hasta obtener que el primer elemento de la mitad izquierda será el primer elemento de la mitad

derecha. Ya que el primer elemento de la mitad derecha es real no tiene sentido hablar del conjugado de este elemento. Una vez tenemos estas dos mitades, calculamos la ifft de esta nueva señal para obtener una señal real en el tiempo.
La parte de la función canal que hace el proceso anteriormente explicado es:

```
num=(2:lndato);
DD= [H1(1),conj(H1((lndato+2)-num)), H1(1:lndato)];
```

Dado que en Matlab los índices de los vectores son positivos se define un vector desde 2-lndato, donde lndato es la longitud del vector H, que varia si se trabaja con la modulación GMSK u OFDM. Si se escoge la primera modulación lndato será 160*5-1, si se escoge la segunda será 81*4.

DD es el espectro complejo par que contiene la mitad derecha e izquierda. El espectro de DD se ilustra en la Figura 3.14. Se verifico que la ifft de DD era una señal real.

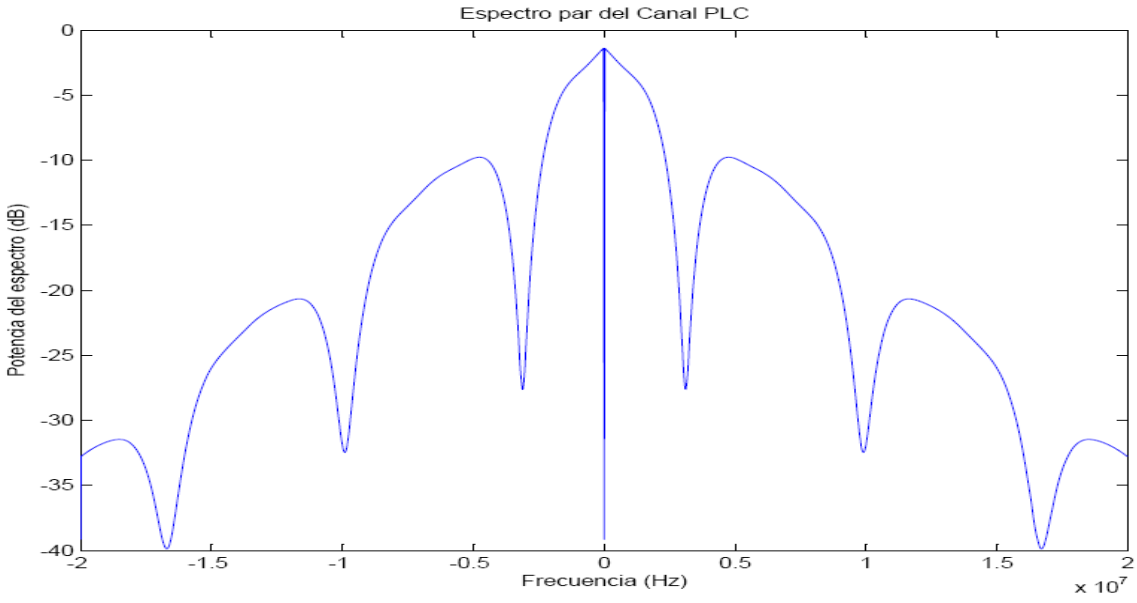



*Figura 3.14. Espectro par del canal PLC*

La función canal devuelve G, que corresponde a la parte del espectro del canal PLC donde se asigna el canal de la modulación elegida. Dicha porcion de canal esta entre: $f1_{GMSK}$ y $f2_{GMSK}$ para un canal GMSK y $f3_{GMSK}$ y $f4_{GMSK}$ para el otro canal GMSK ($f4_{GMSK}$ > $f3_{GMSK}$> $f2_{GMSK}$> $f1_{GMSK}$), donde $f1_{GMSK}$ y $f3_{GMSK}$ son las frecuencias inferiores de los canales GMSK y $f2_{GMSK}$ y $f4_{GMSK}$ son las frecuencia superiores. Este rango de frecuencias se almacena en las variables "a y b" de la funcion canal. Para OFDM el canal esta entre $f1_{OFDM}$ y $f2_{OFDM}$, ($f2_{OFDM}$> $f1_{OFDM}$) donde $f1_{OFDM}$ es la frecuencia

inferior del canal OFDM y $f2_{OFDM}$ es la frecuencia superior. Este rango de frecuencias se almacena en la variable "a" de la funcion canal.

La función canal es llamada por la función fncanal, la cual esta comprendida por el siguiente código:

```
%Función fncanal.m

function yf = fncanal(sal,modulador,snr)
frame=sal;
tipomod=modulador;
Canal= canal(tipomod);
save('modulador.mat')
if (tipomod==1)
    b=size(frame);
    ltl=b(1);

for r=1:1:ltl
    TL(r,:)= fft(frame(r,:));
    Y(r,:)= TL(r,:).*Canal;
    yn(r,:)= awgn(ifft(Y(r,:)),snr,'measured','linear');
    EQF(r,:)= Canal(1,:).^-1;
    YF(r,:)= EQF(r,:).*fft(yn(r,:));
    yf(r,:)= ifft(YF(r,:));
end

FRAME=fft(frame,160,2);
auxfr=[-2000000:(2000000/40):2000000-(2000000/40)]
freq=[-2000000:(2000000/80):2000000-(2000000/80)];
freq1=auxfr+3000000;
freq2=auxfr+7100000;
freq3=[freq1 freq2];
FRAME_MEAN=mean(TL);%TL es la señal GMSK
Y_MEAN=mean(Y);%Y es la señal despues del canal
yn_mean=mean(yn);%yn es la señal con ruido
YF_MEAN=mean(YF);%YF es la salida del ecualizador
save('fncanalmg.mat');
%%%%%%%%%%%%%%
figure
plot(freq,10*log(abs(FRAME_MEAN)))
title('Espectro GMSK en BANDA BASE')
xlabel('Frecuencia (Hz)')
ylabel('Potencia de la señal (dB)')
figure
plot(freq3,10*log(abs(Y_MEAN)))
title('Espectro GMSK despues del canal')
xlabel('Frecuencia (Hz)')
ylabel('Potencia de la señal (dB)')
figure
plot(freq3,10*log(abs(fft(yn_mean))))
title('Espectro Señal GMSK despues del canal con ruido')
xlabel('Frecuencia (Hz)')
```

```
ylabel('Potencia de la señal (dB)')
figure
plot(freq3,10*log(abs(YF_MEAN)))
title('Espectro Señal GMSK despues del Equalizador')
xlabel('Frecuencia (Hz)')
ylabel('Potencia de la señal (dB)')
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
else
    frame1=frame';
    a=size(frame1);
    ltlo=a(1);
    ltl1o=a(2);
    yaux=ones(ltl1o/80,(80+1)*2);
    yfs=ones(1,ltl1o);

    div_tramas=80;
    N_tramas=length(frame1)/div_tramas;%la longitud de F es 825,
entonces N_tramas=25
    for pe=0:N_tramas-1% ciclo para 25 tramas
    for je=1:div_tramas% se agrupan los bits 5 al 264 en cada trama
        frameofdm(pe+1,je)=frame1(div_tramas*pe+je);% se agrupan 260
bits en tramas, son 25 tramas
    end
    end

    TL=zeros(N_tramas,162);
    TL1=zeros(N_tramas,81);
     n=[1:div_tramas];
   shift=0.5*(exp(-j*(2/10)*pi.*n)+exp(j*(2/10)*pi.*n));
    for r=1:1:N_tramas
    xshift(r,:)=frameofdm(r,:).*shift;
    TL1(r,:)=[0 fft(xshift(r,:))];
    num=[2:length(TL1(1,:))];
    TL(r,:)=[TL1(r,1),conj(TL1(r,(length(TL1(1,:))+2-
num))),TL1(r,1:length(TL1(1,:)))];
    save('fncanal3.mat'
    Y(r,:)= TL(r,:).*Canal;
    yn(r,:)= awgn(ifft(Y(r,:)),snr,'measured','linear');
    EQF(r,:)= Canal.^-1;
    YF(r,:)= EQF(r,:).*fft(yn(r,:));
    yff(r,:)=
ifft(YF(r,0.5*(length(YF(r,:))+4):length(YF(r,:)))./shift(1,:);
    yf1(r,:)=yff(r,:);
   end

    for u=1:1:N_tramas

        yfs(1,80*u-79:80*u)=yf1(u,:);
    end
      yf=yfs';
TL1_MEAN=mean(TL1);
XSHIFT=fft(xshift,2);
XSHIFT_MEAN=mean(XSHIFT);
FRAME_MEAN=mean(TL);
```

```
Y_MEAN=mean(Y);
yn_mean=mean(yn);
YF_MEAN=mean(YF);
YF1=fft(yf1,2);
YF1_MEAN=mean(YF1);
freqb=[0:(4050000/80):4050000-(4050000/80)];
freqb1=[0:(4050000/80):4050000];
freq=[-4050000:(4050000/81):4050000-(4050000/81)];
freq1=freq+5000000;
%%%%%%%%%%% %%%%%%%%%%%%%%%%%%%%%%
figure
plot(freqb1,10*log(abs(TL1_MEAN)))
title('Espectro OFDM Banda base')
xlabel('Frecuencia (Hz)')
ylabel('Potencia de la señal (dB)')
hold on
figure
plot(freq,10*log(abs(FRAME_MEAN)))
title('Espectro PAR OFDM')
xlabel('Frecuencia (Hz)')
ylabel('Potencia de la señal (dB)')
hold on
figure
plot(freq1,10*log(abs(Y_MEAN)))
title('Espectro Señal OFDM despues del canal')
xlabel('Frecuencia (Hz)')
ylabel('Potencia de la señal (dB)')
hold on
figure
plot(freq1,10*log(abs(fft(yn_mean))))
title('Espectro Señal OFDM despues del canal con ruido')
xlabel('Frecuencia (Hz)')
ylabel('Potencia de la señal (dB)')
hold on
figure
plot(freq1,10*log(abs(YF_MEAN)))
title('Espectro Señal OFDM despues del ecualizador')
xlabel('Frecuencia (Hz)')
ylabel('Potencia de la señal (dB)')
hold on
figure
plot(freqb,10*log(abs(YF1_MEAN)))
title('Espectro Señal OFDM a enviar al demodulador OFDM')
xlabel('Frecuencia (Hz)')
ylabel('Potencia de la señal (dB)')
hold on
save('fncanalm2.mat');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
end
```

Cada una de las tramas de datos es pasada a través del canal

### 3.4.2 Trama GMSK por el canal PLC

Dado que la señal GMSK es una señal real en el tiempo, puede pasarse directamente hacia el canal. En el proceso de modulación GMSK, el bloque oscilador asigna una frecuencia a la señal modulada que corresponde a 1/T, donde T es el periodo de cada bit que entra al bloque modulador.

Para pasar la señal GMSK por el canal PLC se realiza una convolución entre la señal y la porción del canal PLC donde se ubica la trama GMSK lo que equivale en el dominio de la frecuencia a calcular la Transformada Rápida de Fourier (*FFT- Fast Fourier Transform*) de cada señal y multiplicar dichos resultados.

La señal que sale del canal es el vector yn, se somete a ruido y se pasa por el ecualizador. Finalmente la señal que sale hacia el receptor es el vector yf. Yf es un vector de n filas y m columnas, donde n corresponde al numero de tramas y m corresponde al tamaño de cada trama.

A continuación se ilustran en las Figuras 3.15, 3.16, 3.17 y 3,18 los espectros de las señales que se procesan en el bloque Canal, es decir la señal GMSK que entra al canal, la señal que sale del canal, la señal que sale del canal con ruido, y la señal que sale del ecualizador. Las gráficas obtenidas tienen los siguientes parámetros: sin codificación, SNR=10dB, 2 canales GMSK uno entre 1-5 Mhz y otro entre5.1-9.1 Mhz (Para el cálculo de los espectros se realizo el promedio de todas las tramas).

La Figura 3.15 ilustra el espectro una señal GMSK antes de someterse al canal. Este espectro. Se puede verificar que el espectro de la señal GMSK es par, pues GMSK es una señal real. Note que el ancho de banda de la señal esta entre -2 y 2Mhz, pues los módems PLC ASCOM utilizan un ancho de banda de 2Mhz por portadora.

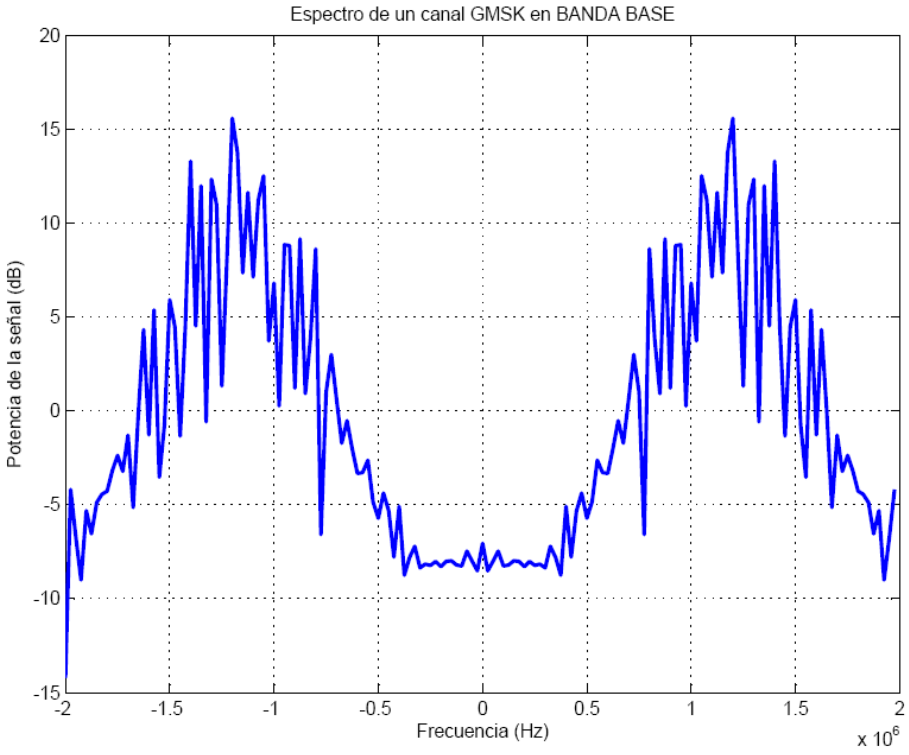



*Figura 3.15. Espectro de un canal GMSK en banda base.*

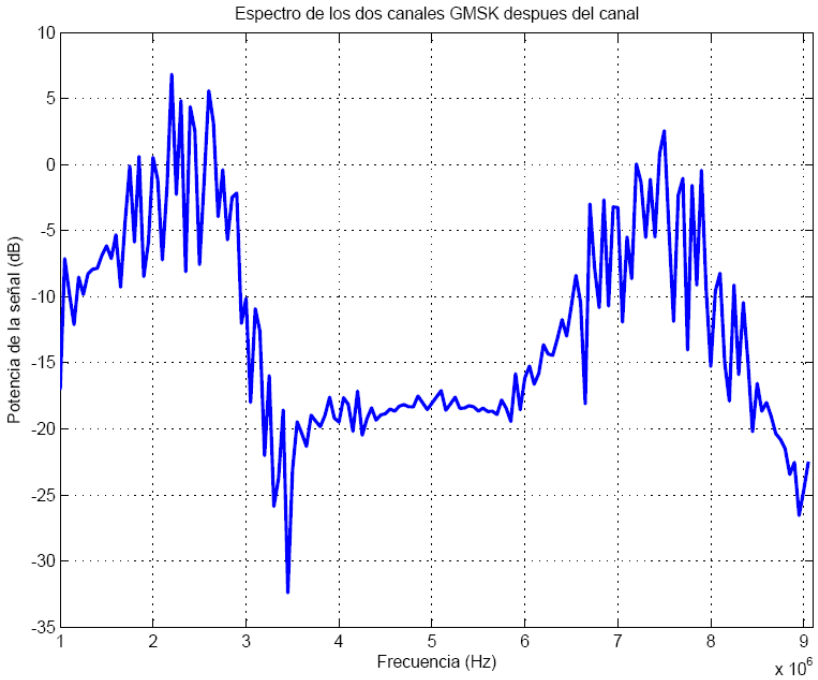



*Figura 3.16. Espectro de los dos canales GMSK después de pasar por el canal*

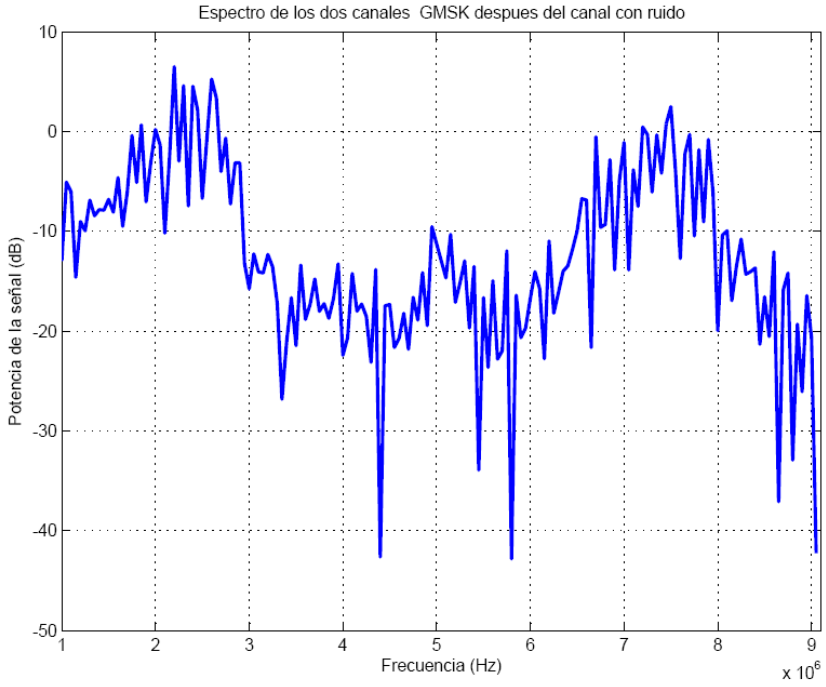



*Figura 3.17. Espectro de los dos canales GMSK después de pasar por el canal y adicionarle ruido (SNR=10dB)*

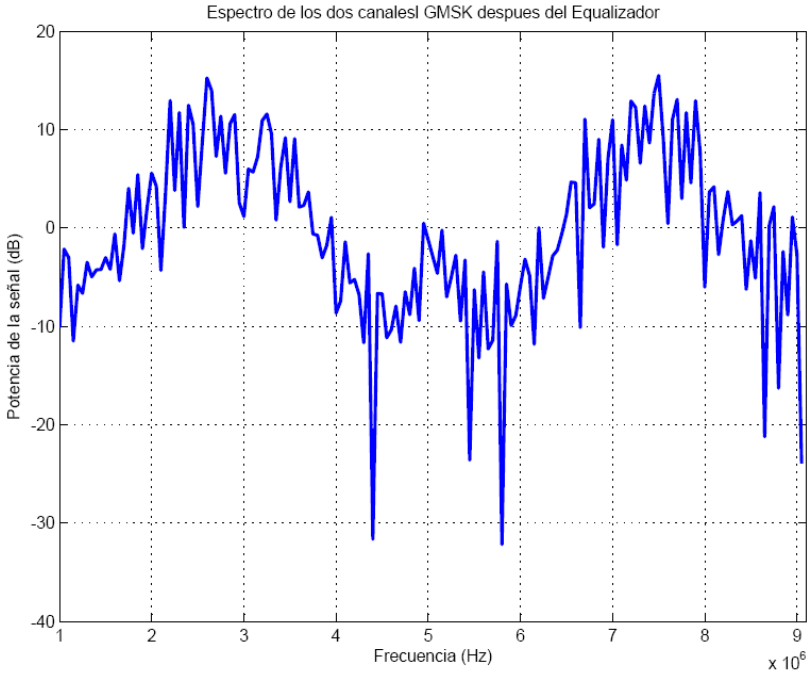



*Figura3.18. Espectro señal GMSK después de pasar por el ecualizador*

La señal que sale del ecualizador es la que se envía hacia el demodulador GMSK.

### 3.4.3 Trama OFDM por el canal PLC

Dado que la señal OFDM es una señal Compleja en el tiempo se requiere convertir esta señal a una señal real en el tiempo. Como se explico para el canal, se debe crear un espectro complejo par, por ello se debe pasar la señal OFDM compleja en el tiempo al dominio de la frecuencia para construir el espectro. Para trabajar en el dominio de la frecuencia se calcula la fft de la señal OFDM compleja.

Antes de realizar la conversión de la señal OFDM, se procederá a asignar una frecuencia portadora a la señal. Esto se obtiene multiplicando la señal por el factor shift = exp (-j*$\Pi$*x), donde x corresponde a la frecuencia en radianes donde se debe trasladar la señal en el espectro del canal.

Para calcular x se debe tomar la frecuencia máxima del canal PLC simétrico. Note que en la Figura 3.18 la frecuencia máxima es 20Mhz que corresponde a 2*$\Pi$ radianes. Entonces x se obtiene aplicando una regla de, así:

$$20\ \text{MHz} \longrightarrow 2*\Pi$$
$$\text{Portadora (MHz)} \longrightarrow x$$

De tal forma que la trama OFDM con su portadora será:

xshift=frame*shift

Ahora se procederá a calcular el espectro par. Para ello se debe calcular la fft de xshift. Ya que nadie nos garantiza que el primer elemento de la fft de xshift es real, -para construir el espectro par y garantizar que corresponda al espectro de una señal real-, se coloca un cero como primer elemento, así:

```
TL1=[0 fft(xshift()];
```

Note que TL1 tiene un elemento más que xshift.

A partir de TL1 se construye el espectro par, donde el punto de simetría es el primer elemento de TL1 es decir 0. TL corresponde al espectro par, asi:

```
num=[2:length(TL1(1,:))];
TL(r,:)=[TL1(1),conj(TL1((length(TL1(1,:))+2-
num))),TL1(1:length(TL1(1,:)))];
```

Donde Indato corresponde a la longitud de TL1, es decir 81 pues el tamaño de una trama OFDM es 80 mas un cero que se añade. Se verifico que al calcular la ifft de TL resultaba una señal real.

La señal OFDM real se pasa por el canal. Dado que trabajamos en el dominio de la frecuencia se multiplica TL por el canal, lo que equivale en el tiempo a una convolución entre las dos señales. Seguidamente se pasa la señal que sale del canal (fft(yn)) por el ecualizador. De yn debemos extraer la señal OFDM compleja que corresponde a la mitad derecha a partir de un elemento más del punto de simetría y se debe bajar a banda base dividiendo por el factor shift. La señal OFDM compleja se representa por yff. Finalmente la señal que sale hacia el receptor OFDM es yf. Yf es un vector de 1 fila y m columnas, donde m corresponde al tamaño de los datos que entran al sistema. Note que los datos que entran al canal se dividen en tramas de 80 datos que se guardan en el vector frame.

A continuación se ilustran los espectros de las señales que se procesan en el bloque Canal, es decir, la señal OFDM compleja con su frecuencia portadora ( El primer elemento de esta señal es un cero para garantizar que la señal que vaya hacia el canal sea real) para convertir la señal OFDM compleja en par, la señal OFDM que entra al canal, la señal que sale del canal, la señal que sale del canal con ruido, la señal que sale del ecualizador y finalmente la señal que se envia hacia el demodulador OFDM. Para el cálculo de los espectros se realizo el promedio de todas las tramas.

Las gráficas obtenidas tienen los siguientes parámetros: sin codificación, SNR=10dB, canal OFDM entre 1-9.1 Mhz, frecuencia portadora = 4Mhz.

La figura 3.19 ilustra el espectro de la señal OFDM en banda base (Note que el primer elemento es un cero para garantizar la conversión a una señal real)

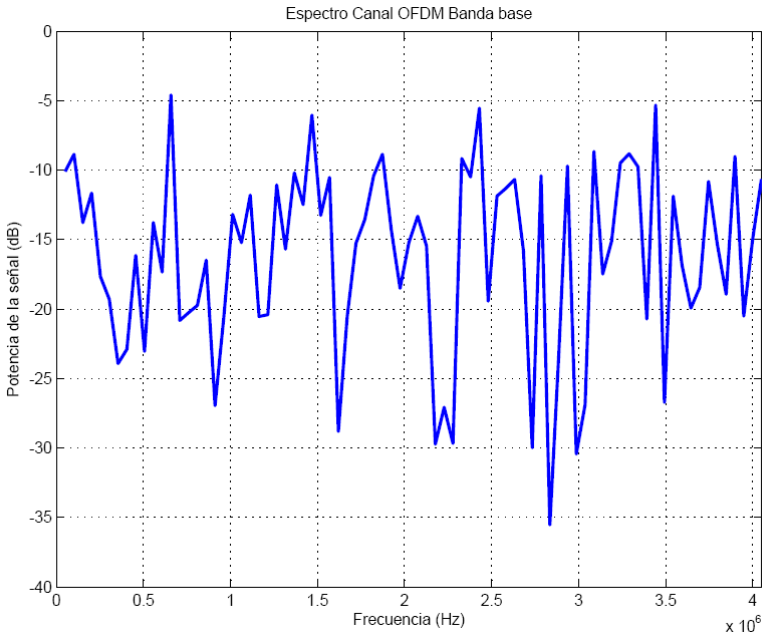



*Figura 3.19. Espectro Señal OFDM en Banda Base*

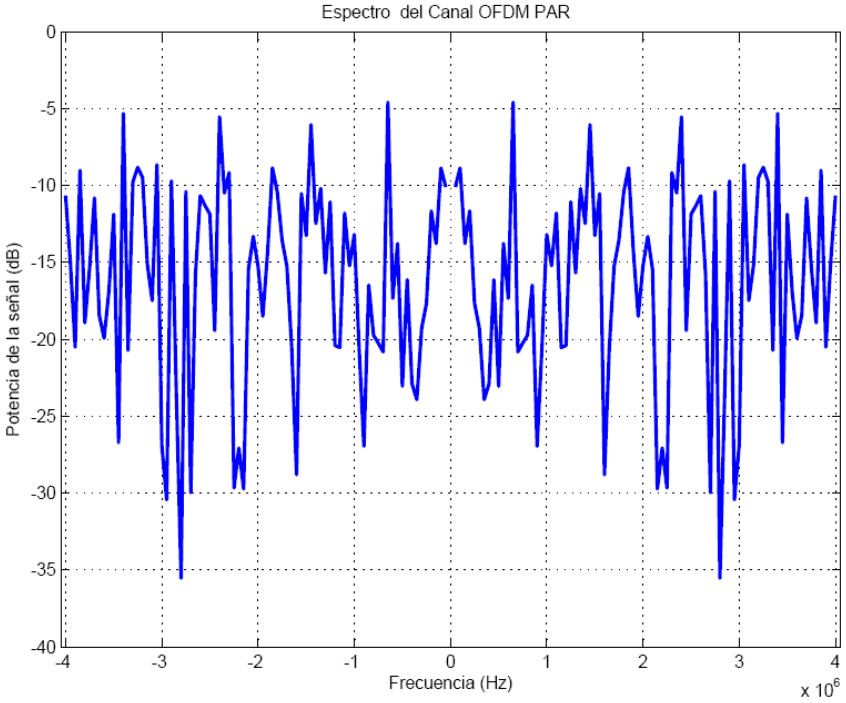



*Figura 3.20. Espectro decanal OFDM PAR*

Note que la figura 3.20 corresponde a un espectro par y el punto de simetria es un valor real. De esta manera se garantiza que al realizar la ifft de esta señal par la señal resultante será una señal real en el tiempo.

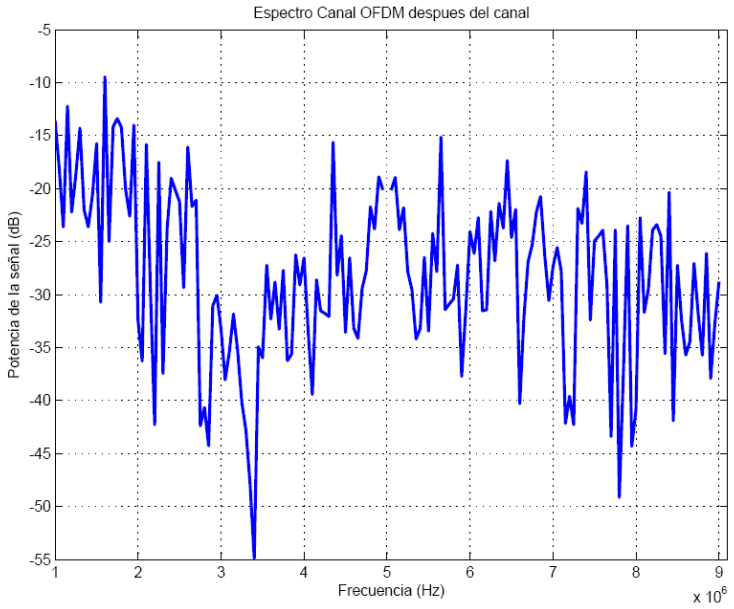



*Figura 3.21. Espectro señal OFDM después del canal*

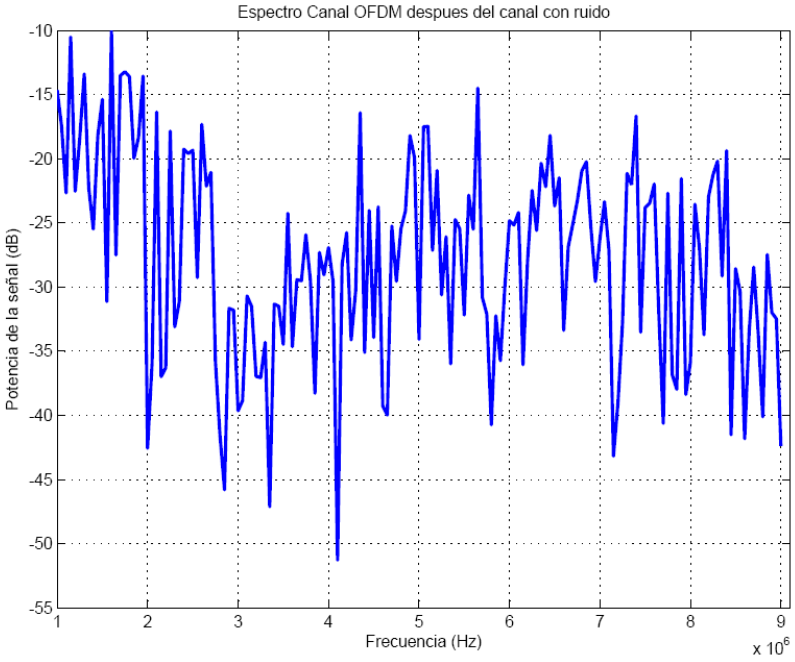



*Figura 3.22. Espectro Canal OFDM después del canal añadiendo ruido (SNR=10 dB)*

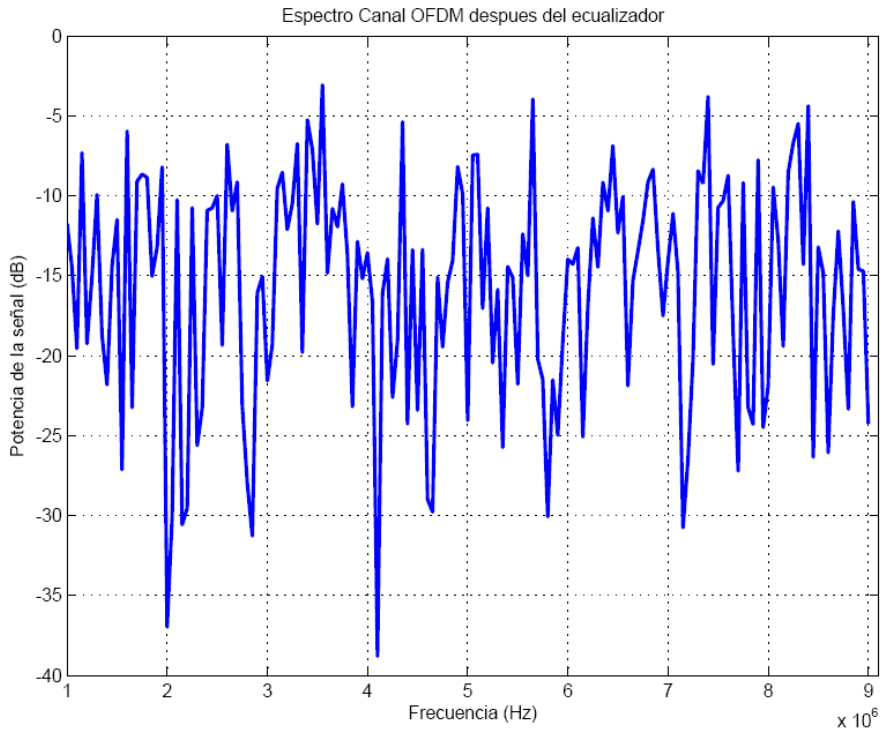



*Figura3.23. Espectro Canal OFDM después de pasar por el ecualizador*

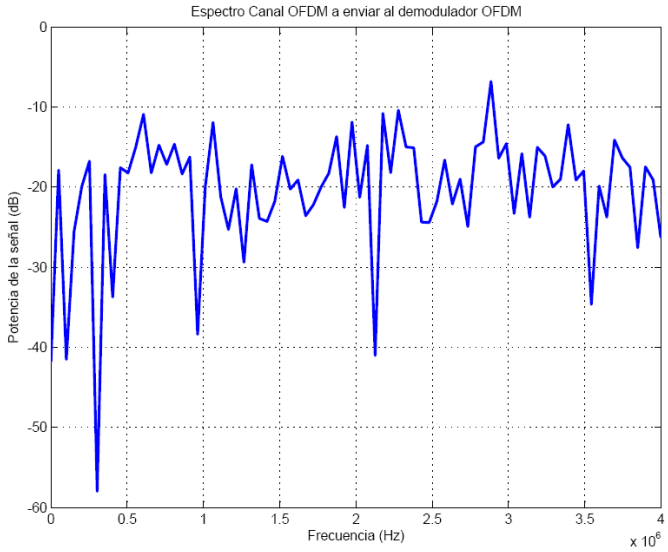



*Figura 3.24. Espectro canal OFDM a enviar al demodulador OFDM*

Note que la señal que sale del ecualizador no es la que se envia hacia el demodulador OFDM, pues la señal que sale del modulador OFDM es una señal compleja en el tiempo a la cual se le hizo el proceso de conversión para obtener

una señal real en el tiempo. Por esta razon se debe tomar la mitad derecha de la señal que sale del ecualizador con el proposito que al demodulador OFDM llegue una señal OFDM compleja, tal como en el modulador.

## 3.5 Bloque Demodulador

En el lado del receptor, cada transmisor utilizado (GSMK u OFDM) debe recuperar los datos que se enviaron. El diagrama en bloques del modelo de recepción se indica en la Figura 4.2 b) de la monografía "Solución de acceso a una Intranet utilizando la Tecnología PLC".

De acuerdo al modulador utilizado, al lado del receptor se debe escoger el tipo de modulador. Para esto, en el archivo main.m (ver sección 3) se utilizan las siguientes líneas de código:

```
if(modulador==1)
    %DEMODULADOR GMSK
    a=size(yf);
    ltl=a(1);
    saldem = demgmsk(ltl,96,Period,yf)';
else
    %DEMODULADOR OFDM
    saldem =demofdm(yf);
end
```

### 3.5.1 OFDM

Luego de realizar la comparación del tipo de modulador utilizado al lado del transmisor, se elige la opción para demodular la señal. Si el modulador utilizado fue OFDM, se llamará a la función demofdm(yf), donde yf es la señal que sale del ecualizador. El diagrama en bloques del demodulador OFDM se ilustra en la Figura 4.7 de la monografia "Solución de acceso a una Intranet utilizando la Tecnología PLC".

#### 3.5.1.1 Retiro del Prefijo Cíclico

Este bloque consiste en extraer los primeros 16 datos que se adicionaron a la estructura del símbolo OFDM al lado del transmisor.

#### 3.5.1.1.1. Implementación Retiro del Préfijo Cíclico

Las líneas de Código en Matlab que implementan este bloque son las siguientes:

```
function rem_prefijo = rem_prefix(palabra_tx);
palabra_transmitida = palabra_tx;
longPalabratransmitida = length(palabra_transmitida);
longPalabraSinPrefix = longPalabratransmitida-
(longPalabratransmitida*16/80);
for i=1:longPalabraSinPrefix/64
    sin_xprefix(64*i-63:64*i,1)=palabra_transmitida(80*i-63:80*i,1);
    rem_prefijo(64*i-63:64*i,1) = sin_xprefix(64*i-63:64*i,1);
end
```

### 3.5.1.2 Bloque FFT

Es el elemento básico en el receptor OFDM. La transformada rápida de Fourier transforma una señal periódica en el dominio del tiempo en su equivalente espectro de frecuencia. Esto es hecho para encontrar la forma de onda equivalente, generada por la suma de componentes sinusoidales ortogonales. Los bits de la FFT de la señal en el dominio del tiempo corresponden al peso de cada una de las portadoras, las cuales son subsecuentemente desintercaladas, decodificadas por el algoritmo apropiado y demapeadas para dar un estimativo de los datos originales [14].

#### 3.5.1.2.1. Implementación del Bloque FFT

Las siguientes son las líneas de código que implementan el bloque FFT

```
function directtransform = fft_ofdm(remov_prefix);
NFFT =64;
longremov_prefix =length(remov_prefix);
%palabraFFT = zeros(remov_prefix,1)+j*ones(remov_prefix,1);
nn=longremov_prefix/64;
for p=1:nn
    fft_xpilott = remov_prefix(64*p-63:64*p,1);
    Yfft= fft(fft_xpilott);
    directtransform(64*p-63:64*p,1) = Yfft;
end
```

La función fft_ofdm recibe un vector remov_prefix correspondiente a la estructura del símbolo OFDM sin el prefijo cíclico. La fft opera de acuerdo a la longitud de la estructura de símbolo que se pase como parámetro (palabras de 64 bits).

#### 3.5.1.3 Eliminación de Símbolos Piloto – Símbolo original / Estimación del canal

Debido a la inserción de los pilotos en la estructura del símbolo OFDM, la estimación de las posiciones por parte del receptor permitirá determinar si un símbolo fue afectado por las perturbaciones del canal. El receptor deberá determinar o saber de antemano las posiciones en las que se introdujeron los símbolos y deberá extraerlos con el fin de recuperar el símbolo original, para luego ser entregado al bloque de de-mapeo.

##### 3.5.1.3.1. Implementación del Bloque Eliminador de Símbolos Piloto

Las líneas de código en Matlab que implementan el bloque que extrae los pilotos de la estructura del símbolo OFDM recibido son las siguientes:

```
function simbolo_original = simb_original(DirectTransform)
longDirectTransform = length(DirectTransform);
longPalabraOriginal = longDirectTransform-
(longDirectTransform*16/64);
ycomp = zeros(longPalabraOriginal,1)+ j*ones(longPalabraOriginal,1);
for pp=1:longPalabraOriginal/48
ycomp(48*pp-47:48*pp-43,1) = DirectTransform(64*pp-62:64*pp-58,1);
ycomp(48*pp-42:48*pp-30,1) = DirectTransform(64*pp-56:64*pp-44,1);
ycomp(48*pp-29:48*pp-24,1) = DirectTransform(64*pp-42:64*pp-37,1);
ycomp(48*pp-23:48*pp-18,1) = DirectTransform(64*pp-35:64*pp-30,1);
ycomp(48*pp-17:48*pp-5,1) = DirectTransform(64*pp-28:64*pp-16,1);
ycomp(48*pp-4:48*pp,1) = DirectTransform(64*pp-14:64*pp-10,1);
simbolo_original(48*pp-47:48*pp,1)= ycomp(48*pp-47:48*pp,1);
end
```

#### 3.5.1.4 De-mapeo [11]

De acuerdo al mapeo o modulación utilizada en el transmisor, el receptor deberá implementar un esquema de de-mapeo que permita recuperar los datos transmitidos.

### 3.5.1.4.1. Demodulador Empleado

Teniendo en cuenta que para el transmisor se hizo uso de un esquema de modulación o mapeo QPSK, para el receptor se requiere un esquema de demodulación o demapeo mediante un demodulador QPSK.

El esquema del demodulador QPSK se muestra en la Figura 3.25.

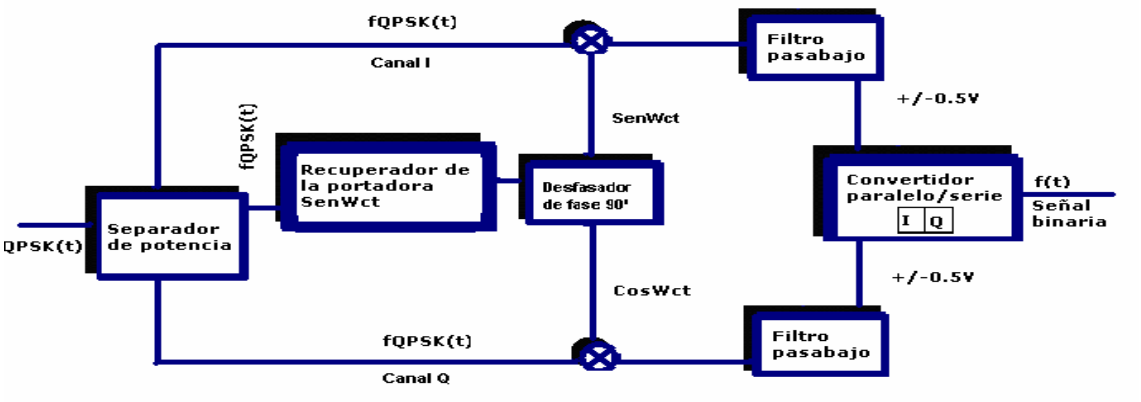



*Figura 3.25. Diagrama en Bloques de un Demodulador QPSK*

El separador de potencia triplica la señal recibida $f_{QPSK}(t)$ sin que ello conlleve una pérdida de potencia, y la conduce a través del canal I y el canal Q hasta los multiplicadores analógicos. En el circuito recuperador de la portadora (que opera de forma similar al modulador QPSK), se obtiene la portadora $sen\omega_c t$ a partir de la señal QPSK. Las salidas de los multiplicadores analógicos se hacen pasar por dos filtros pasabajos que tienen que tener una frecuencia de corte menor que $2\omega_c$.

Matemáticamente el proceso de demodulación es como sigue: suponemos que el dibit transmitido es QI=10. Entonces la señal analógica $f_{QPSK}(t) = +\cos\omega_c t - sen\omega_c t$.

Si analizamos primero el canal Q:

$$Q = \cos\omega_c t f_{QPSK}(t) = \cos\omega_c t(\cos\omega_c t - sen\omega_c t) = \cos^2\omega_c t - \cos\omega_c t sen\omega_c t)$$

$$Q = \frac{1+\cos 2\omega_c t}{2} - [\frac{sen(\omega_c t + \omega_c t)}{2} + \frac{sen(\omega_c t - \omega_c t)}{2}] =$$

$$Q = \frac{1}{2} + \underbrace{\frac{\cos 2\omega_c t}{2}}_{Filtrado} - \underbrace{\frac{sen2\omega_c t}{2}}_{Filtrado} - \underbrace{\frac{sen(0)}{2}}_{Igual-0}$$

$Q= +\frac{1}{2}$ *(1 lógico)*

Ahora para el canal I

$$I = sen\omega_c t f_{QPSK}(t) = sen\omega_c t(\cos\omega_c t - sen\omega_c t) = sen\omega_c t\cos\omega_c t - sen^2\omega_c t =$$

$$Q = [\frac{sen(\omega_c t + \omega_c t)}{2} + \frac{sen(\omega_c t - \omega_c t)}{2}] - \frac{1-\cos 2\omega_c t}{2} =$$

$$Q = \underbrace{\frac{sen2\omega_c t}{2}}_{Filtrado} - \underbrace{\frac{sen(0)}{2}}_{Igual-0} - \frac{1}{2} + \underbrace{\frac{\cos 2\omega_c t}{2}}_{Filtrado}$$

$I= -\frac{1}{2}$ *(0 lógico)*.

Así que al convertidor paralelo-serie llega QI=10 que efectivamente corresponde al valor del dibit transmitido. Para todos los demás casos posibles de $f_{QPSK}(t)$, se puede comprobar fácilmente la validez de este modulador.

#### 3.5.1.4.1.1. Implementación del bloque de de-mapeo QPSK para el receptor OFDM

La implementación de un demodulador QPSK en Matlab puede ser lograda mediante las siguientes líneas de código:

```
function demapping= qpsk_demapping(simbolo_original);
%datos originales
ycomp = simbolo_original;
ycod = zeros(96,1);
longSimboloOriginal = length(simbolo_original);
for i=1:longSimboloOriginal;
    if (real(ycomp(i,1))>0)
        if(imag(ycomp(i,1))>0)
        ycod(2*i-1,1)= 0;
        ycod(2*i,1)= 0;
        end
        if(imag(ycomp(i,1))<0)
        ycod(2*i-1,1)= 1;
        ycod(2*i,1)= 0;
        end
```

```
    end
    if (real(ycomp(i,1))<0)
        if(imag(ycomp(i,1))>0)
        ycod(2*i-1,1)= 0;
        ycod(2*i,1)= 1;
        end
        if(imag(ycomp(i,1))<0)
        ycod(2*i-1,1)= 1;
        ycod(2*i,1)= 1;
        end
    end
end
demapping = ycod;
figure;
plot(ycomp, '*')
```

La function *qpsk_demapping* recibe un vector *simbolo_original* compuesto por datos complejos. Para obtener los datos hacemos una estimación de acuerdo a los datos recibidos y con base al diagrama I-Q. Comparamos si los valores de I y Q están por encima de 0 o son menores. Si están por encima (componente I o Q mayor que cero) asignamos un 1, de lo contrario asignamos -1. De esta manera se logra una estimación de los datos que llegan al demodulador QPSK.

### 3.5.2 GMSK

El diagrama en bloques del demodulador GMSK se ilustra en la Figura 4.8 de la monografía "Solución de Acceso a una Intranet utilizando la tecnología PLC".
Si el modulador utilizado fue GMSK, se llamará a la función degmsk(ltl,datos,Period,yf),donde "ltl" es el numero de tramas, "datos" es el tamaño de cada trama, "Period" es la duración de cada bit en la modulación gmsk y "yf" es la salida del ecualizador. La función demgmsk esta formada por el siguiente código:

```
%Función demgmsk.m

function sal = demgmsk(N_tramas,div_tramas,T,yf)
 [N_tramas,lin,auxiq] =demodulation(N_tramas,div_tramas,T,yf);
 [filcol,salaux] = decod(N_tramas,lin,auxiq);
for j=0:1:(filcol(1))-1
    sal((filcol(2))*j+1:(filcol(2))*(j+1))=salaux(j+1,:);
end
```

Note que la función demgmsk hace un llamado a la función demodulation. Dicha función recibe los parámetros de demgmsk y esta compuesta por el siguiente código:

```
%Función demodulation.m

function [N_tramas,lin,auxiq]
=demodulation(N_tramas,div_tramas,T,yf)

lft=20;%para 16 datos que entran al modulador es 20
lin=div_tramas;
auxi=ones(N_tramas,lin);
auxq=ones(N_tramas,lin);
auxiq=ones(lin,2*N_tramas);
for m=1:1:div_tramas*5
        tin1(1,m)=m*(T/5);
end
lg=length(tin1);
for nt=1:1:N_tramas
f=1/T;
omega3=2*pi*f;
Icarrier(nt,:)=cos(omega3*tin1);
Qcarrier(nt,:)=sin(omega3*tin1);
Idemod(nt,:)=yf(nt,1:lg).*Icarrier(nt,:);
Qdemod(nt,:)=yf(nt,1:lg).*Qcarrier(nt,:);
ftI2(nt,:)=fft(Idemod(nt,:));
ftQ2(nt,:)=fft(Qdemod(nt,:));
ftI2(nt,lft:length(ftI2(nt,:))-lft)=0;
ftQ2(nt,lft:length(ftQ2(nt,:))-lft)=0;
Idemod2(nt,:)=ifft(ftI2(nt,:));
Idemod2(nt,:)=real(Idemod2(nt,:));
Qdemod2(nt,:)=ifft(ftQ2(nt,:));
Qdemod2(nt,:)=real(Qdemod2(nt,:));
for v = 1:1:lin ;
       auxi(nt,v)= Idemod2(nt,v*5-2);
       auxq(nt,v)= Qdemod2(nt,v*5-2);
end

for r=1:1:lin
auxiq(r,2*nt-1)=auxq(nt,r);
auxiq(r,2*nt)=auxi(nt,r);
end
end
```

### 3.5.2.1 Osc y Filtro Pasa Bajos (LPF – Low Pass Filter)

La señal gmsk que se pasó a través del canal llega a un oscilador (Osc), el cual se encarga de multiplicar dicha señal por 2 portadoras desfasadas en 90º (portadoras seno y coseno). El LPF es utilizado para obtener las componentes I y Q con sus respectivas portadoras, para ello se deben eliminar las frecuencias altas.

#### 3.5.2.1.1. Implementación OSC y LPF

Las líneas de código que implementan el oscilador local se encuentran dentro de la función demodulation y son:

```
f=1/T;
omega3=2*pi*f;
Icarrier(nt,:)=cos(omega3*tin1);
Qcarrier(nt,:)=sin(omega3*tin1);
Idemod(nt,:)=yf(nt,1:lg).*Icarrier(nt,:);
Qdemod(nt,:)=yf(nt,1:lg).*Qcarrier(nt,:);
```

Las siguientes líneas de código de la función demodulation implementan el filtro:

```
ftI2(nt,lft:length(ftI2(nt,:))-lft)=0;
ftQ2(nt,lft:length(ftQ2(nt,:))-lft)=0
```

### 3.5.2.2 Detector de nivel y Precodificador

El bloque detector de nivel permite establecer un umbral para las componentes I y Q que salen del bloque LPF, con el fin de obtener una cadena de 1's y 0's.

El bloque precodificador recibe dicha cadena de 1's y 0's. y realiza el proceso inverso del precodificador diferencial en el lado del transmisor.

#### 3.5.2.2.1. Implementación Detector de Nivel y Precodificador

Una vez se ha realizado el proceso de filtrado se tienen las señales I y Q con sus respectivas portadoras, las cuales están en el vector auxiq, note que en las columnas impares esta la componente I de cada trama y en las columnas pares esta la componente Q de cada trama.

Las salidas de la función demodulation son: el número de tramas (N_tramas), el tamaño de cada trama (lin) y las componentes I y Q (auxiq)

Una vez se ha llamado la función demodulation, se procede a llamar la función decod.m, la cual se encarga de realizar el proceso del detector de nivel y el proceso del precodificador. El código de la función decod es:

```
%Función decod.m
function [filcol,salaux] = decod(N_tramas,lin,auxiq)
filcol(1,1)=N_tramas;
filcol(1,2)=lin;
zona=ones(N_tramas,lin);
salaux=ones(N_tramas,lin);
for nt=1:1:N_tramas
for p=1:1:lin
%**********DETECTOR DE NIVEL*************************************
if((auxiq(p,2*nt-1)>0)&(auxiq(p,2*nt)>0))
    zona(nt,p)=1;
end
if((auxiq(p,2*nt-1)<0)&(auxiq(p,2*nt)>0))
    zona(nt,p)=2;
end
if((auxiq(p,2*nt-1)<0)&(auxiq(p,2*nt)<0))
    zona(nt,p)=3;
end
if((auxiq(p,2*nt-1)>0)&(auxiq(p,2*nt)<0))
    zona(nt,p)=4;
end
end
for x=1:1:lin-1
    if (x==1)
        if(or((zona(nt,x+1)==2),(zona(nt,x+1)==4)))
        cod_o(nt,x+1)=1;
        else
            if(or((zona(nt,x+1)==1),(zona(nt,x+1)==3)))
              cod_o(nt,x+1)=0;
            end
        end
        if(zona(nt,x)==zona(nt,x+1))
        cod_o(nt,x)=not(cod_o(nt,x+1));
        else
        cod_o(nt,x)=cod_o(nt,x+1);
        end
    else
```

```
        if(zona(nt,x)==zona(nt,x+1))
        cod_o(nt,x+1)=not(cod_o(nt,x));
        else
            cod_o(nt,x+1)=cod_o(nt,x);
        end
    end
end
%*****************************************************************
 for s=1:1:lin
    if(s==1)
        if(nt==1)
********************DECODIFICADOR DIFERENCIAL***********************
        salaux(nt,s)=xor(cod_o(nt,s),1);
        else
        salaux(nt,s)=xor(cod_o(nt,s),cod_o(nt-1,lin));
        end
    else
          salaux(nt,s)=xor(cod_o(nt,s),cod_o(nt,s-1));
    end
end
%*****************************************************************
end
```

La función decod.m recibe las componentes I y Q de cada trama ubicadas en el vector auxiq. Aquí se examina en que cuadrante están las componentes I y Q para asignar una zona a cada cuadrante (Figura 3.26). La asignación de zona es el detector de nivel. Entonces de acuerdo a la posición en la que se encuentren las componentes I y Q se asigna un valor de zona. Note que zona puede ser 1, 2,3 o 4. Una vez se tienen las zonas se continua con el proceso de precodificación diferencial. Si la zona correspondiente al primer elemento esta ubicada en la zona 2 o 4 el primer dato codificado será un 1. En caso que la zona correspondiente al primer elemento este ubicada en la zona 1 o 3 el primer dato codificado será un 0.

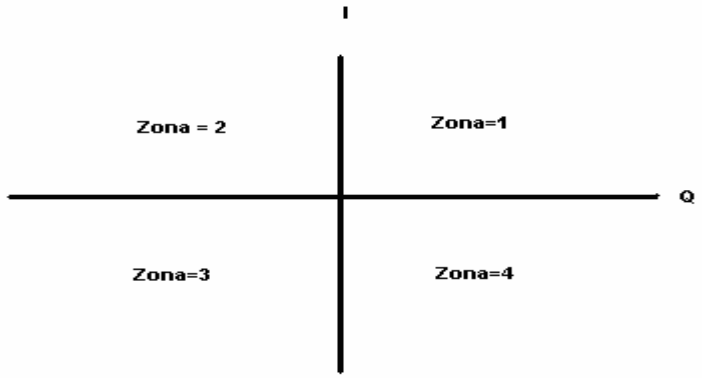



*Figura 3.26. Designación de cadena I y Q en recepción*

Para obtener el segundo dato codificado se examina que valor tiene la zona para el segundo elemento y si dicho valor es igual al de la zona anterior (el del primer elemento), el segundo dato codificado será la negación del primer dato codificado. Si valor que tiene la zona para el segundo elemento es diferente al de la zona anterior (el del primer elemento), el segundo dato codificado será el mismo del primer dato codificado. De la misma manera se procede para conseguir los datos codificados correspondientes a cada elemento de auxiq.

Los datos codificados en el receptor se almacenan en el vector cod_o. Una vez se tienen dichos datos estos deben corresponder a los datos codificados en el lado del transmisor.

La señal de salida del bloque codificador en el lado del receptor se almacena en el vector salaux y se obtiene a través del vector cod_o, así:
Para obtener el primer bit de salida se realiza una operación xor entre el primer elemento de cod_o y 1. Para obtener el segundo bit de salida la operación xor es entre el segundo elemento y primer elemento de cod_o. Para obtener el tercer bit de salida la operación xor es entre el tercer elemento y segundo elemento de cod_o. Es decir para obtener el bit de salida se realiza la operación xor entre el elemento actual y el elemento anterior de cod_o.

Dado que este proceso se realiza para cada trama es importante resaltar que cuando entra el primer elemento de la primera trama se realiza una xor entre ese primer elemento y el 1. Cuando ingresa el primer elemento de las tramas siguientes el xor es entre ese primer elemento y el último elemento de la trama anterior.

Las salidas de la función decod son: salaux que corresponde a las tramas que van hacia el desintercalador y filcol que corresponde al número de tramas. Ya que salaux corresponde a las tramas, estas se deben organizar en un vector fila antes de ir hacia el bloque des-intercalador. Para ello se utiliza el siguiente ciclo de la función demgmsk:

```
for j=0:1:(filcol(1))-1
    sal((filcol(2))*j+1:(filcol(2))*(j+1))=salaux(j+1,:);
end
```

Este ciclo organiza las tramas en un vector fila, pues la entrada al desintercalador es un vector fila.

Una vez se tiene la señal demodulada se envía hacia el desintercalador.

## 3.6 Bloque De-Interleaving o des-intercalador

Al lado del receptor, se realizó una operación de intercalamiento que consiste en permutar los bits con el fin de que los bits adyacentes sean separados, y así evitar que ráfagas de ruido afecten a palabras de bits continuas. La función del des-intercalamiento consiste en recuperar la señal intercalada de acuerdo al patrón definido al lado del transmisor, es decir, si en el transmisor se organizó la información por filas y por columnas en una matriz nxm (con n filas y m columnas), y se permutaron filas por columnas, al lado del receptor se deberá realizar la función contraria, es decir permutar columnas por filas.

### 3.6.1 Implementación del de-interleaving

Las siguientes son las líneas de código del desintercalador.

```
function nointer = deinter(isd)
p=16;
y=8;
x=p/y;
lisd=length(isd);
na=ceil(lisd/p);
lisdc=na*p;
if lisdc>lisd,
    for n = (lisd+1):lisdc,
        isd(n)=0;
    end
end
a=1;
for k = 0:na-1,
    % Escritura
    for j = 1:y,
        for i = 1:x,
            RM(j,i)=isd(a);
            a=a+1;
        end
    end
    % Lectura
    b=1;
    for i = 1:x,
        for j = 1:y,
            sd(k*p+b)=RM(j,i);
            b=b+1;
        end
```

```
    end
end
nointer = sd';
```

## 3.7 Bloque Decodificador - Decodificación de la señal de información original.

Debido a que en el receptor se utilizó los esquemas de codificación Reed Solomon y codificación convolución, en el receptor se debe implementar un algoritmo que permita recuperar los datos codificados. Para lograr este objetivo, para recuperar los datos que se codificaron mediante el Codificador Convolucional, se usa el algoritmo de Viterbi (sección 2.4.1), y para la decodificación Reed Solomon se usa el decodificador Reed Solomon.

### 3.7.1.1.1. Implementación Bloque Decodificador de Códigos Convolucionales – Algoritmo de Viterbi

Las siguientes líneas de código hacen uso de la función de viterbi propia Matlab:

```
function deccodconv= deconvol(xsc);
tblen = length(xsc);
%tblen=5; %número de rutas mayor o igual al numero de bits de
entrada
trellis = struct('numInputSymbols',2,'numOutputSymbols',4,...
'numStates',4,'nextStates',[0 2;0 2;1 3;1 3],...
'outputs',[0 3;3 0;1 2;2 1]);
code = xsc;
decoded = vitdec(code,trellis,5,'trunc','hard'); % hard decision
NSDEC =1;
decoded = vitdec(code,trellis,5,'trunc','soft',NSDEC); % soft
decision
deccodconv=decoded;
```

### 3.7.1.1.2. Implementación Decodificador Reed Solomon

El decodificador Reed Solomon permite decodificar la palabra de código cifrada al lado del transmisor, también mediante el algoritmo Reed Solomon.

La palabra de código recibida debe ser un arreglo de símbolos de Galois de *m* símbolos por bit. Los elementos de la palabra son valores entre 0 y $2^m$-1. Si no es así, se asume que la palabra de código recibido es una versión errónea de un código acortado.

Para el decodificador Reed Solomon, Matlab utiliza la función rsdec (n,k), donde n indica la longitud de cada palabra de código, y k los bits de información, es decir n-k bits de paridad se adicionaron por el transmisor.

Las líneas de código que implementan el codificador RS son las siguientes:

```
function decode_rs = rsdecoder(xcodif)
mdec = 4; %bits por símbolo
ndec=12;
kkdec=8;
Kdec = length(xcodif);
datodecimaldec = zeros (Kdec/mdec,1);
for i=0:Kdec/mdec-1
    tt = mdec*i+1;
    v= mdec*i+4;
    datodec = xcodif(tt:v,1);
    decimaldec= datodec(1,1)*2^3 + datodec(2,1) * 2^2 +
datodec(3,1) * 2^1 +datodec(4,1) * 2^0;
    datodecimaldec(i+1,1) = decimaldec;
end
decimaldec = datodecimaldec';
l_datdec = length(decimaldec)/ndec;
datbinario = [0 0 0 0 0 0 0 1 0 0 1 0 0 0 1 1 0 1 0 0 0 1 0 1 0 1 1
0 0 1 1 1 1 0 0 0 1 0 0 1 1 0 1 0 1 0 1 1 1 1 0 0 1 1 0 1 1 1 1 0 1
1 1 1];
for t=1:l_datdec
   datdecimaldec(1,1:12) = decimaldec(1,12*t-11:12*t);
   msgdec = gf(datdecimaldec,mdec);
   a = rsdec(msgdec,ndec,kkdec);
   [dec,cnumerr] = rsdec(msgdec,ndec,kkdec);
   for b=1:8
        dat = a(1,b);
        if dat==0
            Rx_binario(1,4*b-3:4*b) = datbinario(1:4);
        end
       if dat == 1
            Rx_binario(1,4*b-3:4*b) = datbinario(5:8);
       end
       if dat ==2
```

```
            Rx_binario(1,4*b-3:4*b) = datbinario(9:12);
        end
        if dat ==3
            Rx_binario(1,4*b-3:4*b) = datbinario(13:16);
        end
        if dat ==4
            Rx_binario(1,4*b-3:4*b) = datbinario(17:20);
        end
        if dat ==5
            Rx_binario(1,4*b-3:4*b) = datbinario(21:24);
        end
        if dat ==6
            Rx_binario(1,4*b-3:4*b) = datbinario(25:28);
        end
        if dat ==7
            Rx_binario(1,4*b-3:4*b) = datbinario(29:32);
        end
        if dat ==8
            Rx_binario(1,4*b-3:4*b) = datbinario(33:36);
        end
        if dat ==9
            Rx_binario(1,4*b-3:4*b) = datbinario(37:40);
        end
        if dat ==10
            Rx_binario(1,4*b-3:4*b) = datbinario(41:44);
        end
        if dat ==11
           Rx_binario(1,4*b-3:4*b) = datbinario(45:48);
        end
        if dat ==12
            Rx_binario(1,4*b-3:4*b) = datbinario(49:52);
        end
        if dat ==13
            Rx_binario(1,4*b-3:4*b) = datbinario(53:56);
        end
        if dat ==14
            Rx_binario(1,4*b-3:4*b) = datbinario(57:60);
        end
        if dat ==15
            Rx_binario(1,4*b-3:4*b) = datbinario(61:64);
        end
    end
       rs = Rx_binario;
      Rx_pbinaria(1,32*t-31:32*t) = Rx_binario(1,1:32);
end
decode_rs = Rx_pbinaria';
```

La función rsdecoder recibe un vector xcodif que contiene los elementos a decodificar en formato binario. Debido a que cada el algoritmo RS hace uso de campos de Galois para su operación, se agrupa la información en símbolos de cuatro bits y se convierten a su valor decimal correspondiente, para luego formar el campo de Galois mediante la función gf(msg,m), cuyo parámetro msg representa el mensaje que se desea convertir a campo de Galois, y m el número de bits por palabra mensaje. Luego mediante la función rsdec(gf(msg,m),n,k) se decodifica la palabra y de manera siguiente se convierte a símbolos binarios.

## 3.8 Especificaciones de la integración de los módulos del sistema.

El archivo main.m especificado en la sección 3 une los bloques del sistema diseñado. Las pruebas realizadas permitieron obtener el comportamiento de la probabilidad de error con respecto a la relación SNR en el sistema PLC utilizando las modulaciones GMSK y OFDM en conjunto con técnicas de codifcación e intercalamiento.

Para cada una de las pruebas se genero una señal de datos aleatoria, y se hicieron 10 medidas para cada valor de SNR con el fin de calcular un promedio del número de errores.

El archivo llamar.m permitió realizar las pruebas descritas en la sección 4.4 de la monografía "Solución de acceso a una Intranet utilizando la Tecnología PLC", y tiene los siguientes parámetros:

- h: Vector para guardar los errores. La longitud es 10 porque se realizaron 10 pruebas para cada SNR, con el fin de obtener un valor de error promedio.

- k: Datos de entrada al sistema PLC, estos datos varían de acuerdo a la modulación escogida si se elige OFDM corresponde a 1920 datos y si se elige GMSK corresponde a 768 datos.

- f= vector que permite almacenar el valor de la SNR y la respectiva probabilidad de error para cada SNR. En la fila 1 se almacena el valor de SNR y en la segunda fila la probabilidad de error obtenida para SNR.

- tipo_coder: Permite escoger el tipo de codificación. Este parámetro tiene tres posibles valores, así:

  1: El sistema PLC no utiliza codificación
  2: El sistema PLC utiliza codificación convolucional.
  3: El sistema PLC utiliza codificación Red Salomón

- modulador: Permite seleccionar el tipo de modulación a utilizar. Este parámetro tiene dos posibles valores:

  1: Modulación GMSK
  2: Modulación OFDM

- s: Corresponde al valor de SNR en dB. Note que es un parámetro que se va incrementando.

- a: Corresponde a la probabilidad de error promedio para una misma SNR. Para calcular el error promedio se realizan 10 mediciones y se suman los errores y se divide sobre el número de medidas. Finalmente para calcular la probabilidad de error se divide el error promedio sobre el número de datos que se ingreso al sistema PLC (K)

El comando save permite guardar en un archivo .mat todas las variables del archivo llamar.m

Note que el archivo llamar.m llama al archivo mainp como una función. El archivo mainp devuelve el número de errores para una señal de entrada al sistema PLC.

A continuación se ilustra el archivo llamar.m utilizando modulación GMSK sin codificación. Las variables para esta prueba se guardan en el archivo GMSKsincod.mat. Seguidamente se especifica el archivo mainp.m.

```
%Programa principal
h=zeros(1,10);
f=zeros(2,35);
K=1920;
tipo_coder=1;
modulador=1;

for z=1:5
    s=0.1*z
    for k=1:10
        h(1,k)=mainp(s,K,tipo_coder,modulador);
```

```
    end
    a= (sum(h(1,:))/10)/K;
%       a=(h(1,i))/K;
    f(1,z)=s;
    f(2,z)=a;
end
for p=1:1:10
    s=p
for k=1:10
        h(1,k)=mainp(s,K,tipo_coder,modulador);
    end
    a= (sum(h(1,:))/10)/K;
%       a=(h(1,i))/K;
    f(1,p+5)=s;
    f(2,p+5)=a;
end

for i=2:1:21
    s=10*(i-1)+1
    for k=1:10
        h(1,k)=mainp(s,K,tipo_coder,modulador);
    end
    a= (sum(h(1,:))/10)/K;
%       a=(h(1,i))/K;
    f(1,i+14)=s;
    f(2,i+14)=a;
end

save('GMSKsincod.mat');
figure
plot( f(1,:), f(2,:), 'x-');
hold on
y=zeros (35,2)
for t=1:1:35
    y(t,1)=f(1,t);
    y(t,2)=f(2,t);
end
xlswrite('datos', y)
```

Enseguida se especifica el código del archivo mainp.m, el cual es llamado por el archivo llamar.m. Note que el archivo mainp.m difiere del archivo main presentado en la sección 3, en que los parámetros de SNR , entrada de bits, tipo de codificador y tipo de modulador son pasados a través de las variables "snr", "K", "tipo_coder" y "modulador". La función mainp devuelve el número de errores

```
function error = mainp(snr,K,tipo_coder,modulador)
close all

xrand= rand(K,1); % matriz N*1 que produce datos aleatorios mediante
el comando rand
x= zeros(K,1); % se inicializa con ceros la matriz
for i=1:K;     % ciclo for para aproximar los valores por encima de
0.5 a 1 y por debajo a 0
    x(i)=0;
    if xrand(i,1)>0.5
        x(i)=1;
    end
end

%****Tipo de codificación 1 sinCodificación , 2 RS, 3
Convolucional********
xcod= ofdm_coder(x, tipo_coder); % x es la entrada del codificador
%*****************************************************************

%*********************************INTERCALAMIENTO*******************

interleaving = inter(xcod);
%interleaving = xcod;
%*****************************************************************

%***********************************MODULADORES*******************

if (modulador==1)%Modulador GMSK
    [T,sal]=gmskmod(interleaving);
    Period=T;
else
    %ELSE (OFDM)
    sal =ofdmmod(interleaving);
end

%*****************************************************************

% *********************************CANAL ************************
% yf=sal;
yf = fncanal(sal,modulador,snr);

%*****************************************************************

%*********************************DEMODULADOR***********************
if(modulador==1)
    %%%%DEMODULADOR GMSK
    a=size(yf);
    datos=32;
    ltl=a(1);
    saldem = demgmsk(ltl,datos,Period,yf)';
else
    %DEMODULADOR OFDM
```

```
    saldem =demofdm(yf);

end

%*********************************************************************

%*****************************DESINTERCALAMIENTO*********************
******
%saldem = interleaving;
no_interleaving = deinter(saldem);
%no_interleaving = saldem;
%*********************************************************************

%********************Decodificador RS/conv/**********************
dxcod= ofdm_decoder(no_interleaving, tipo_coder);
%*********************************************************************
salida=x'-dxcod';
a=find(salida);
error=length(a);
```

# 4. POSIBILIDAD DE IMPLEMENTACIÓN HARDWARE MEDIANTE FPGA`S

En años recientes se han popularizado en Microelectrónica los términos "design gap" (apertura de diseño) y "verification gap" (apertura de verificación) que hacen referencia a que el aumento de productividad de diseñadores, máquinas, y herramientas no consigue seguir el ritmo marcado por el aumento de puertas disponibles, y por tanto de la complejidad de los ASIC *(Aplication Specific Integrated Circuit)* y FPGAs (*Field-Programmable Gate Arrays*) fabricados.

A esto se añade el hecho de que el mercado está imponiendo tremendas presiones para reducir los tiempos y costos de desarrollo con el fin de mantener la competitividad. El resultado es que el porcentaje de ASICs que requieren reprocesado ha escalado hasta el 61%, siendo en el 71% de los casos debido a la existencia de errores funcionales. El impacto económico que esto supone se puede evaluar teniendo en cuenta que los costos de las máscaras para un ASIC fabricado en un proceso de 0.13 micras se aproximan al millón de dólares. A lo que se han de añadir los costes de ingeniería del rediseño [17].

Recientemente se ha puesto en el mercado un conjunto de herramientas y metodologías que intentan reducir los riesgos de fallo durante el proceso de diseño. Sin embargo, estas herramientas son tremendamente caras, superando en muchos casos el millón de dólares. Además de esto, algunas de estas tecnologías no han alcanzado el debido grado de madurez. Por tanto, al realizar este tipo de inversiones se corre un alto riesgo de seleccionar la tecnología equivocada. A pesar de esto, es posible, mediante el uso de estas herramientas, validar las tecnologías existentes y aportar a la futura tendencia de nuevos sistemas, en especial en el campo de la telecomunicaciones.

Entre las herramientas destacadas existen kits de desarrollo de empresas como ALTERA, XILINX, y ALCATEL Espacio, que permiten un desarrollo e implementación sobre sus productos licenciados. En este apartado del documento, se tratará un poco sobre el desarrollo de algunos bloques esenciales y una aproximación al modelo real de algunas modulaciones digitales, haciendo uso de la herramienta de XILINX.
Una de las principales ventajas que presenta el System Generator de Xilinx con sus librerías, consiste en la facilidad de integración con el software de Matlab. Estas librerías hacen parte del conjunto de bloques (blockset) de simulink y permite

integrar tanto elementos propios de Xilinx como elementos de simulink. La idea general de esta herramienta, es que las respuestas obtenidas por las librerías se comporten de forma idéntica. La ventaja de Xilinx con respecto a las de simulink es la facilidad de generación de código VHDL (*Very High Speed Integrated Circuit Hardware Description Language*) y de implementación sobre FPGA`s propias de esta compañía. De acuerdo a las características requeridas para el diseño que se desee realizar, existen diferentes clases de FPGA`s, avanzadas con componentes como conversores Digitales/Análogos y Análogos/Digitales, como dispositivos para extender capacidad e integrar con otros productos. Para los casos presentados, se explicará de manera resumida las implementaciones realizadas, como también una posible aproximación de elementos básicos como son los codificadores convolucionales y decodificación Viterbi.

## 4.1 Instalación del Software de Xilinx

La instalación es sencilla, pero se requiere saber algunos detalles importantes:

- La versión del software de Xilinx utilizada fue Xilinx ISE 8.1i, para lo que se requiere que el software de Matlab sea superior a 7.1.0.1.; en el caso nuestro, la versión utilizada fue la 7.1.0.246.

- Al instalar el software de Matlab es necesario que la ruta de instalación no tenga ningún espacio. Por lo general la ruta utilizada es c:/>\matlab71. Luego se procede a instalar el software de Xilinx ISE 8.1i.

- Se debe descargar el paquete System Generator (sistema generador) SysgenInstall_v8_1_01.exe, que permitirá que las librerías del ISE de xilinx, se carguen como librerías de Matlab. Para descargar este archivo, es necesario visitar la página http://www.xilinx.com/xlnx/xil_reg_profile.jsp, y crear una cuenta de usuario.

- Luego de crear la cuenta, se debe ir al link download, y seleccionar el software de System Generator. El siguiente paso es ejecutarlo para cargar las librerías de Xilinx.

Teniendo el software instalado, procedemos a trabajar en Matlab con la herramienta simulink. Para esto se desarrollaron algunos modelos que hacen uso de esta herramienta y se explicarán en las siguientes secciones.

## 4.2 Modulación por Cambio de Fase (*PSK* - *Phase Shift Keying*)

### 4.2.1 Definición

Es un tipo de modulación digital que también se conoce como Modulación digital por desplazamiento de fase binaria (BPSK - *Binary Phase Shift Keying*). En esta modulación la entrada digital se modula en fase, es decir dependiendo de los valores de dicha entrada, la señal analógica modulada va a tener una fase u otra fase de salida.

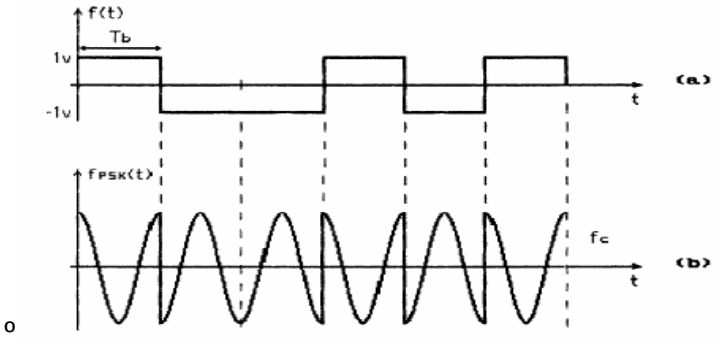
Sea $f(t)$ la señal de información digital, con unos niveles de tensión de $\pm$1 Voltios – para simplificar- y anchura de bit $T_b$ (ver Figura 5.1(a)) [12]. Consideremos una señal portadora de alta frecuencia ($CosW_c t$), donde $W_c = 2 * pi * f_c$ y $f_c$ es la frecuencia de dicha señal.

De esta manera, la función de la señal PSK va a ser:

$$f_{PSK}(t) = f(t) CosW_c t = +CosW_c t \text{ ; si } f(t) = +1V \text{ (1 lógico)}$$
$$= -CosW_c t = Cos(W_c t + \pi) \text{ ; si } f(t) = -1V \text{ (0 lógico)}$$

Ecuación 5.1

La señal PSK ( $f_{PSK}(t)$ ) se muestra en la Figura 5.1 (b)




o

*Figura 5.1. Señales de la modulación PSK: (a) Señal de entrada digital (información); (b) Señal modulada PSK [12]*

#### 4.2.1.1 Modulador y demodulador BPSK

En la Figura 5.2 Se ilustra el diagrama en bloques del modulador y demodulador BPSK.

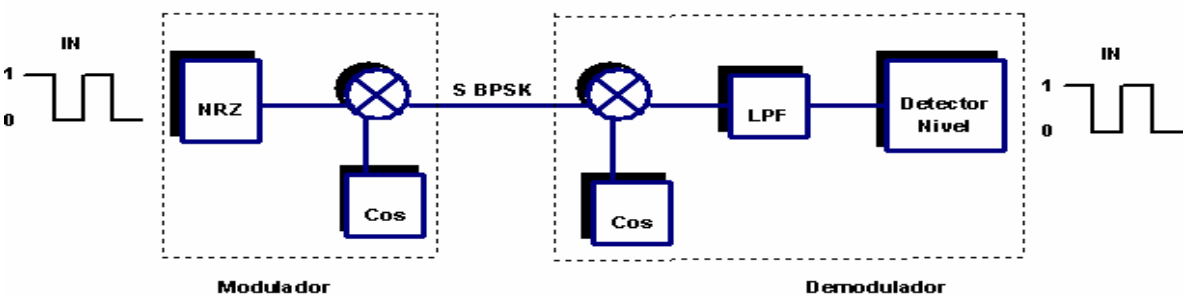



*Figura 5.2. Modulador y demodulador BPSK*

Modulador:

La señal de entrada se convierte en forma NRZ para ser multiplicada por la portadora $Cos(Wct)$. La salida del multiplicador corresponde a la señal BPSK (*S BPSK* de la Figura 5.2).

Demodulador:

La señal BPSK entra al modulador y es multiplicada nuevamente por la onda portadora $Cos(Wct)$. Con el propósito de recuperar la señal de entrada se deben remover las componentes de alta frecuencia a través de un filtro pasabajos (LPF de la Figura 5.2). El detector de nivel establece un umbral a la salida del filtro para obtener la señal de entrada.

### 4.2.2 Simulación de modulación y demodulación BPSK

La simulación BPSK se realizó utilizando Simulink y los componentes de la librería Xilinx, por ello es necesario conocer las siguientes herramientas necesarias para la simulación

Bloques de Simulink

- Signal From Space: simula un tren de pulsos que corresponde a la señal de entrada.
- Scope: osciloscopio utilizado para visualizar los resultados.
- AWGChanel: Añade ruido gausiano a la señal.

Herramientas de Xilinx

- Mcode: Hace una llamada a un archivo .m de Matlab y lo ejecuta dentro de la simulación [18].
- Gateway In: Hace una aproximación al comportamiento de una señal en hardware.
- Gateway Out: Regresa una aproximación del comportamiento de una señal en hardware al modo simulación.
- Mult: Realiza la multiplicación de una o más entradas.
- DAFIR v9_0: Simula un Filtro FIR, haciendo una llamada a la herramienta FDATool de Matlab.
- System Generator: Provee control del sistema y parámetros de simulación, y es usada para invocar al código generado. Es una herramienta de software que permite crear y verificar diseños de hardware para FPGAS de Xilinx, funciona en conjunto con Simulink y Matlab. Además permite la inclusión de herramientas DSP(*Digital Signal Processor*) para diseñar con FPGAs, generación automática de código HDL(*Hardware description language*) a partir de un modelo en Simulink y permite al usuario crear sus propias bibliotecas
- FDATool: Interfaz que permite configurar las características del filtro
- DDS v5_01: Permite generar ondas senoidales
- Convert: Convierte la entrada al tipo de dato que se necesita.
- Delay: permite añadir latencia al diseño.

El modelo BPSK que se diseño se basa en la Figura 5.2. Dicho modelo en Xilinx se muestra en la Figura 5.3.

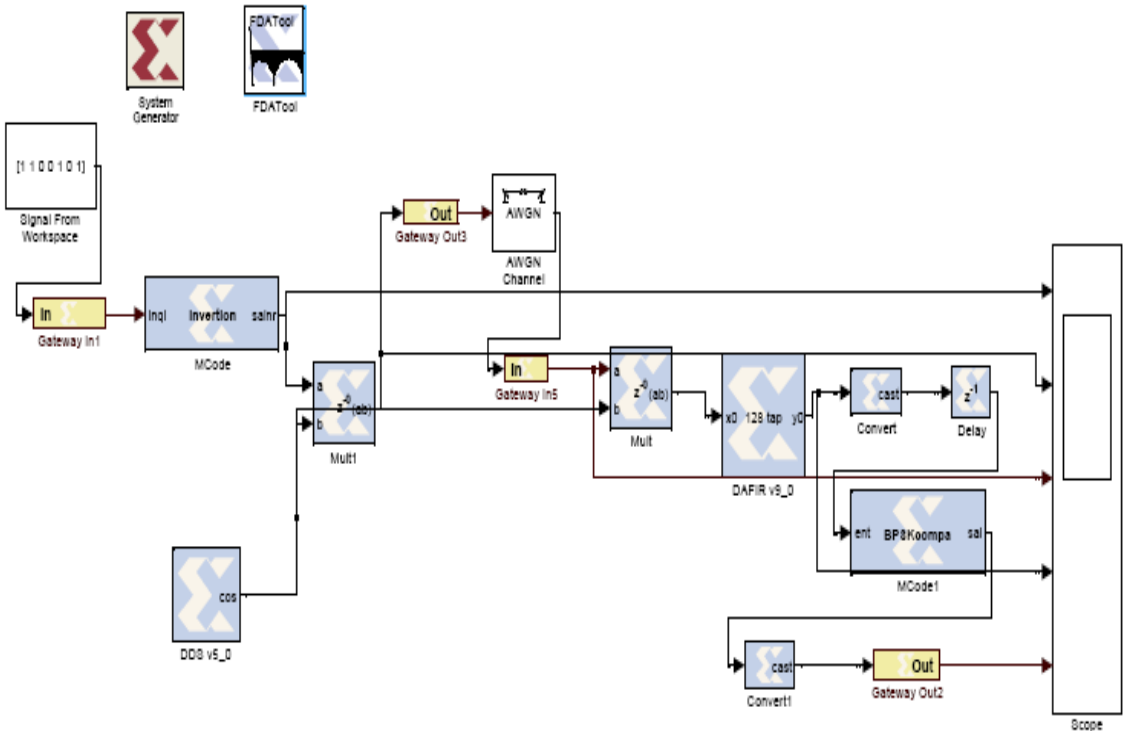



*Figura 5.3. Modulación y demodulación BPSK en Xilinx*

### 4.2.2.1 Explicación diseño

El bloque Signal from Workspace es la entrada al sistema BPSK. Es una secuencia de unos y ceros {1, 1, 0, 0, 1, 0,1}, donde la duración de cada bit es 500s. Esto quiere decir que la señal de entrada va de 0-3500seg.

Esta entrada se convierte en forma no retorno a cero a través del bloque Mcode que carga la función invertion.m. El código de dicha función es:

```
%**** Función invertion.m**************
function salnr = invertion(inqi)
if inqi==1
    salnr = 1;
else
    salnr = -1;
end
%******************
```

El código invertion.m convierte la señal de entrada que es una cadena de unos y ceros en forma NRZ, es decir los 1's los deja igual y los ceros los convierte en -1.

Una vez obtenida la entrada en forma NRZ, esta se multiplica por una portadora. El bloque DDS_V_5 genera la onda coseno, la cual se multiplica con la señal no retorno a cero y así obtenemos la señal BPSK.

Este sistema introduce ruido a la señal BPSK a través del bloque AWG Channel y esta señal con ruido es tratada en el demodulador.

La señal con ruido ingresa a un multiplicador que hace las veces de oscilador local. A continuación se deben remover las frecuencias altas a través de un filtro pasabajos. El bloque DAFIR_v9_0 nos permite diseñar filtros a través de la interfaz FDATool. Ya eliminadas dichas frecuencias, se debe crear un detector de nivel para recuperar la señal de entrada, el cual se hace en el bloque Mcode1 que carga la función BPSKcompa.m. El código de dicha función es:

```
%********Función BPSKcompa.m ***********
function sal = BPSKcompa (ent)
if ent > 0
    sal = 1;
else
    sal = 0;
end
%******************
```

Las principales señales del diseño ilustrado en la Figura 5.3, se pueden visualizar a través del bloque scope. Dichas señales se ilustran en la Figura 5.4.

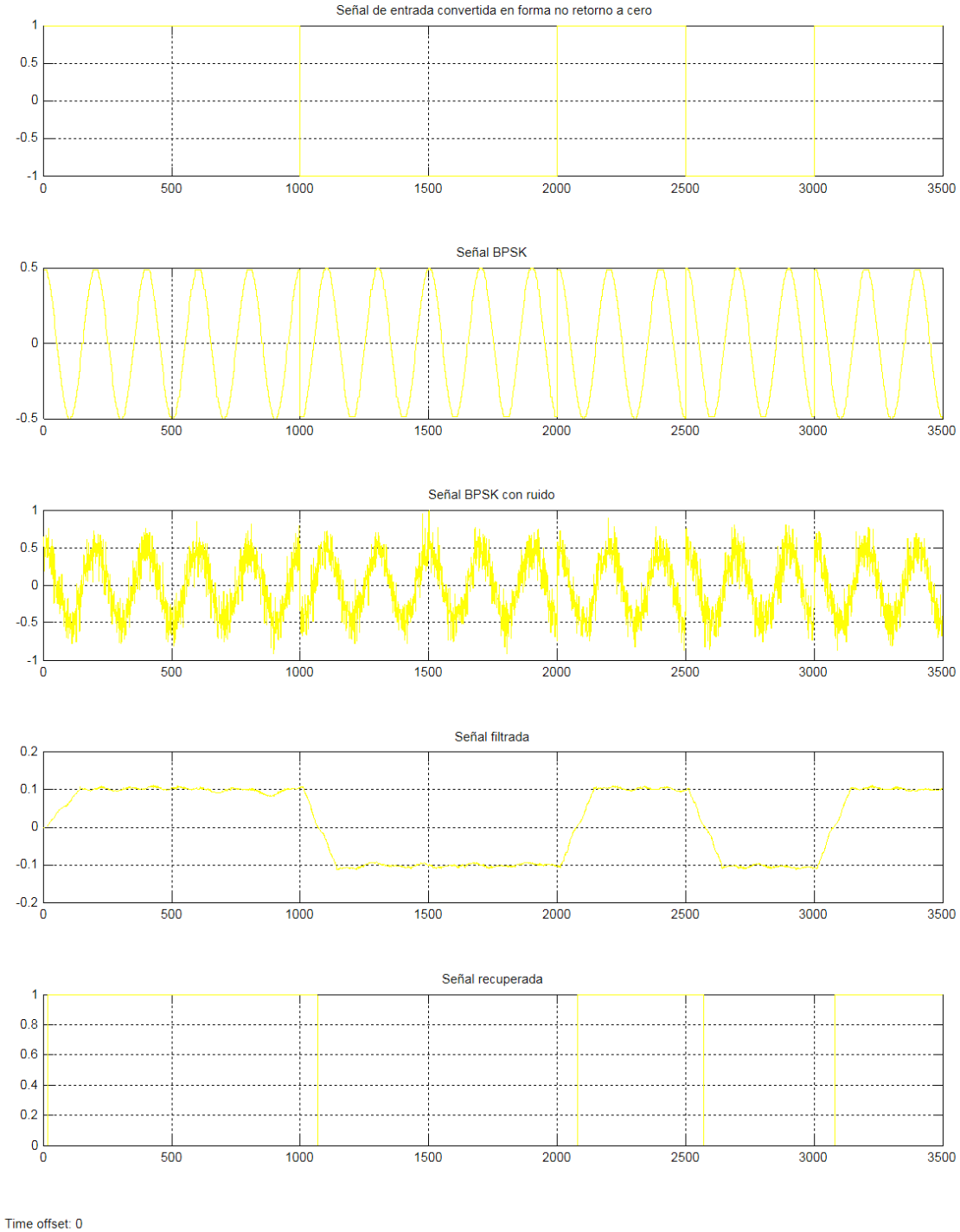



*Figura 5.4. Señales del modulador-demodulador BPSK.*

Note que la señal recuperada es la misma señal de entrada, pero con un pequeño retardo en el tiempo. Podemos ver que cuando hay un 1 lógico en la entrada la señal BPSK corresponde a una onda coseno y cuando hay un 0 lógico en la entrada la señal BPSK corresponde a una onda coseno desfasada 180º. Al comparar la señal BPSK de la Figura 5.4 con la de la Figura 5.1 la diferencia esta en la frecuencia de la señal.

La señal BPSK que se obtiene se le introduce ruido blanco con el bloque AWG chanel del Simulink de Matlab, es decir en esta modulación tenemos en cuenta la señal con ruido. De esta manera comprobamos que la modulación BPSK a manera de simulación funciona.

#### 4.2.2.2 Parámetros Bloques diseño BPSK

A continuación se ilustran los parámetros de cada uno de los bloques utilizados para el diseño en xilinx del modulador y demodulador BPSK.

Bloque: Signal from workspace
Dicho bloque se encuentra en Simulink -> Signal Processing Blockset -> Signal Processing Sources
Parámetros
Signal: [1 1 0 0 1 0 1]
Sample time: 500
Samples per frame=1
Form output after final data value by: setting to zero

Bloque: DDS V_5_0
Dicho bloque se encuentra en xilinx blockset -> Index - > DDS_V_5_0
Parámetros:
Function: cosine
Channels:1
Type: Fixed
Output Frecuency Array (MHz)=[1.0]
Type: None
DDS clok rate (MHz)=200.0
Spurious free Dynamic range (db)=36
Frecuency resolution(Hz)=0.4

Bloque Gateway In
Las gateway in, in1, e in 5 se encuentran en Xilinx Blockset -> Index - > Gateway in.
Parámetros:
Out type: signed
Number of bits: 10
Binary point: 8
Sample period: 1

Bloque Gateway out
Las Gateways out vienen con los parámetros por defecto.

Bloque AWG chanel
Awg Chanel : está en Communications Blockset -> Chanel -> AWGN Channel
Parámetros
Inicial speed:67
Mode: Eb/No
Eb/No=10
Number of bits per symbol 2
Input signal power: 1
Symbol period: 1

Bloque Mult
Esta en Xilinx Blockset -> Index - > Mult
Parámetros:
Precison: user defined
Out type: signed
Number of bits:10
Binary point: 8
Quantization: truncate
Overflow: wrap
Latency: 3

Bloque Mult1
Esta en Xilinx Blockset -> Index - > Mult
Parámetros:
Precison: user defined
Out type: signed
Number of bits:10
Binary point: 8

Quantization: truncate
Overflow: wrap
Latency: 0

DAFIR v9_0:
Esta en Xilinx Blockset -> Index - > DAFIR v9_0
Parámetros
Coefficients: xlfd_numerator ('FDATool'); de esta forma se invocan los parámetros diseñados en FDATool
Structure: inferred from Coefficients
Number of bits:10
Binary point: 8
Hardware over sampling: 1
Latency: 14

FDATool:
Esta en Xilinx Blockset -> Index - > FDATool.
Los parámetros de diseño se ilustran en la Figura 5.5.

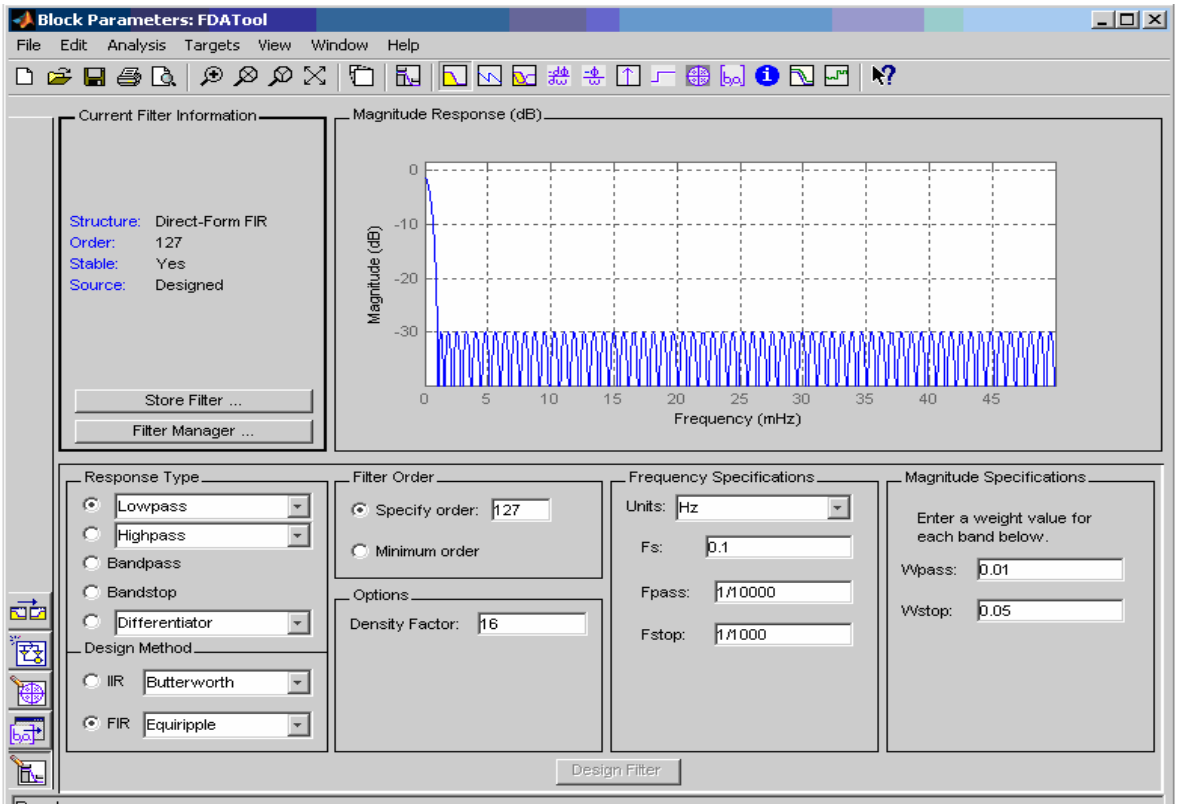



*Figura 5.5. Parámetros de diseño de Filtro*

El filtro pasa-bajas de Respuesta Finita Impulsiva (*FIR - Finite Impulse Response*) separa la señal continua de amplitud $\pm$ ½ recuperada de la señal demodulada compleja y permite seleccionar la señal de frecuencia cero (+1/2 o -1/2). Este filtro se obtiene haciendo una llamada a la herramienta de Matlab FDATool, interfaz que permite diseñar un filtro pasabajos. (Figura 5.5)

Bloque Convert
Se encuentran en Xilinx Blockset -> Index - > convert
Parámetros:
*Out precision*
Type: signed
Number of bits:10
Binary Point: 8
*Quantization*: truncate
*Overflow:* wrap
*Latency*: 0

Bloque Delay
Se encuentran en Xilinx Blockset -> Index - > Delay
Parámetros:
*Latency*: 1

## 4.3 QPSK

### 4.3.1 Definición

Es un tipo de modulación MPSK en la que M = 4 - es decir, la señal portadora de frecuencia $w_c$ puede tener 4 fases de salida diferentes - y por consiguiente k = 2 ($4=2^2$). Entonces en esta modulación los datos binarios están compuestos por grupos de dos bits (hay 2 bits por cada símbolo de la entrada) que reciben el nombre de dibits.

Para obtener 2 bits por cada símbolo de la señal entrada se utilizan 2 cadenas, la cadena I y la cadena Q.

La forma de obtener las cadenas I y Q a través de la señal de entrada se ilustra en la figura 5.6

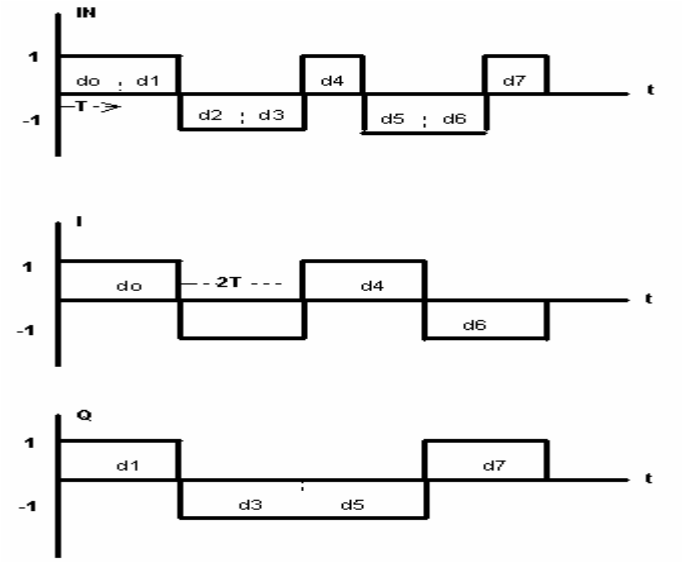



*Figura 5.6 (a) Señal de entrada (b) Señal I (c) Señal Q*

Note que para la cadena I se toman los valores de los bits pares de la señal de entrada (IN), es decir d0, d2, d4y d6. El mismo proceso se maneja para obtener la cadena Q pero se toman los bits impares.

Cada bit de la señal de entrada (IN) tiene una duración de T seg., mientras que en las cadenas I y Q cada símbolo tiene duración de 2T.
El bit I, cuyo nivel de tensión puede ser de -1 o +1 voltio, es multiplicado por la señal portadora ($Cosw_c t$). Mientras que el bit Q, se multiplica por una portadora desplazada en fase ($-Sinw_c t$). Una vez las cadenas I y Q se multiplican por sus portadoras se suman para obtener la señal QPSK. Este proceso se ilustra en la Figura 5.7

En la Figura 5.7 se puede apreciar que hay un cambio de fase cada 2T seg., si uno de los bits de la cadena I cambia con respecto al valor anterior de dicha cadena habrá un cambio de fase de 90º, similar para la cadena Q. Si ambos bits de las cadenas I y Q cambian con respecto a los valores anteriores de sus respectivas cadenas el cambio de fase será 180º.

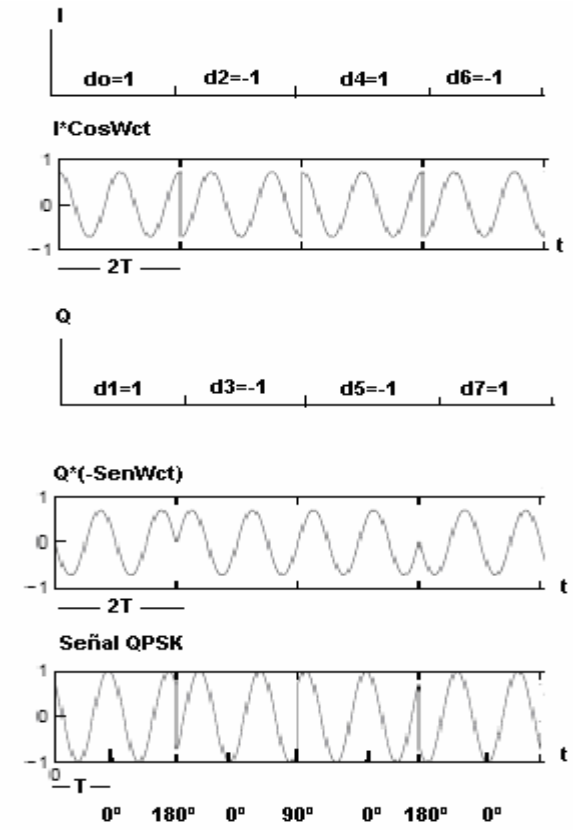



*Figura 5.7 Modulación QPSK [18]*

### 4.3.1.1 Modulador QPSK

El modulador QPSK se ilustra en la Figura 5.8.

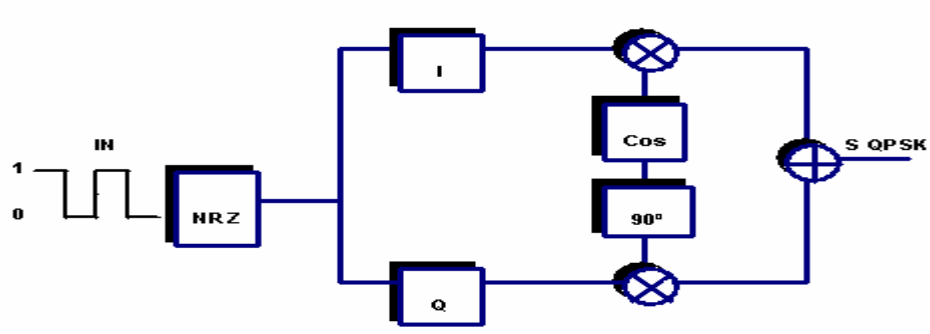



*Figura 5.8. Modulador QPSK*

La señal de entrada (IN) se convierte en forma NRZ. Los bits pares de esta señal de entrada se toman para formar la cadena I y los bits impares para formar la cadena

Q. La duración de los bits en las cadenas I y Q es el doble de lo que dura un bit de la señal de entrada. La cadena I es multiplicada por la portadora $CosW_1t$ (donde $W_1 = 2*\pi* f_1$ y $f_1$ es la frecuencia de la onda portadora); mientras que la cadena Q es multiplicada por dicha portadora desfasada 90º. La señal QPSK es la suma de estos dos productos.

### 4.3.2 Simulación de modulación QPSK

El proceso de modulación y demodulación de BPSK se pudo lograr utilizando la librería Xilinx. Para QPSK se inicio con el diseño del modulador, dicho diseño se ilustra en la Figura 5.9.

En esta modulación son necesarios 2 bits por cada símbolo de la entrada. Entonces se debía buscar la forma de obtener dos señales a partir de la señal de entrada, las cuales corresponden a las señales I y Q. Para la cadena I se tomaron los bits pares de la señal de entrada (Figura 5.6 (b)) y de acuerdo al valor que tuviera la señal de entrada en dichos bits pares se obtenía la cadena I, el mismo proceso se maneja para obtener la cadena Q pero se toman los bits impares. Retomando la Figura 5.6 se obtuvieron las señales I y Q a través del diseño en Xilinx de la Figura 5.9. Las principales señales de salida del modulador QPSK diseñado en xilinx se ilustran en la Figura 5.10.

Para crear el bloque MCode que carga el archivo generacioniq.m se tomo como referencia el modelo de la tesis "A Physical Layer Implementation of Reconfigurable Radio" [20].

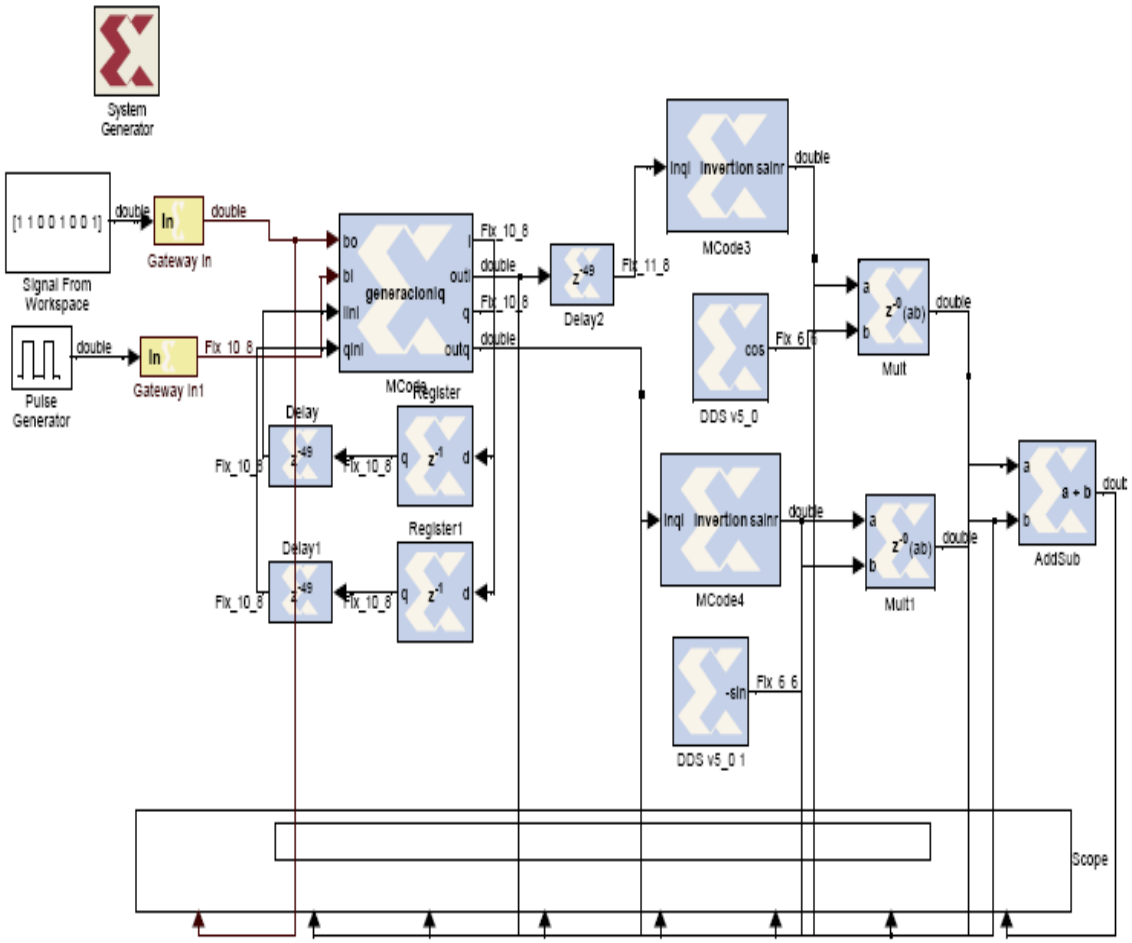



*Figura 5.9. Modulador QPSK*

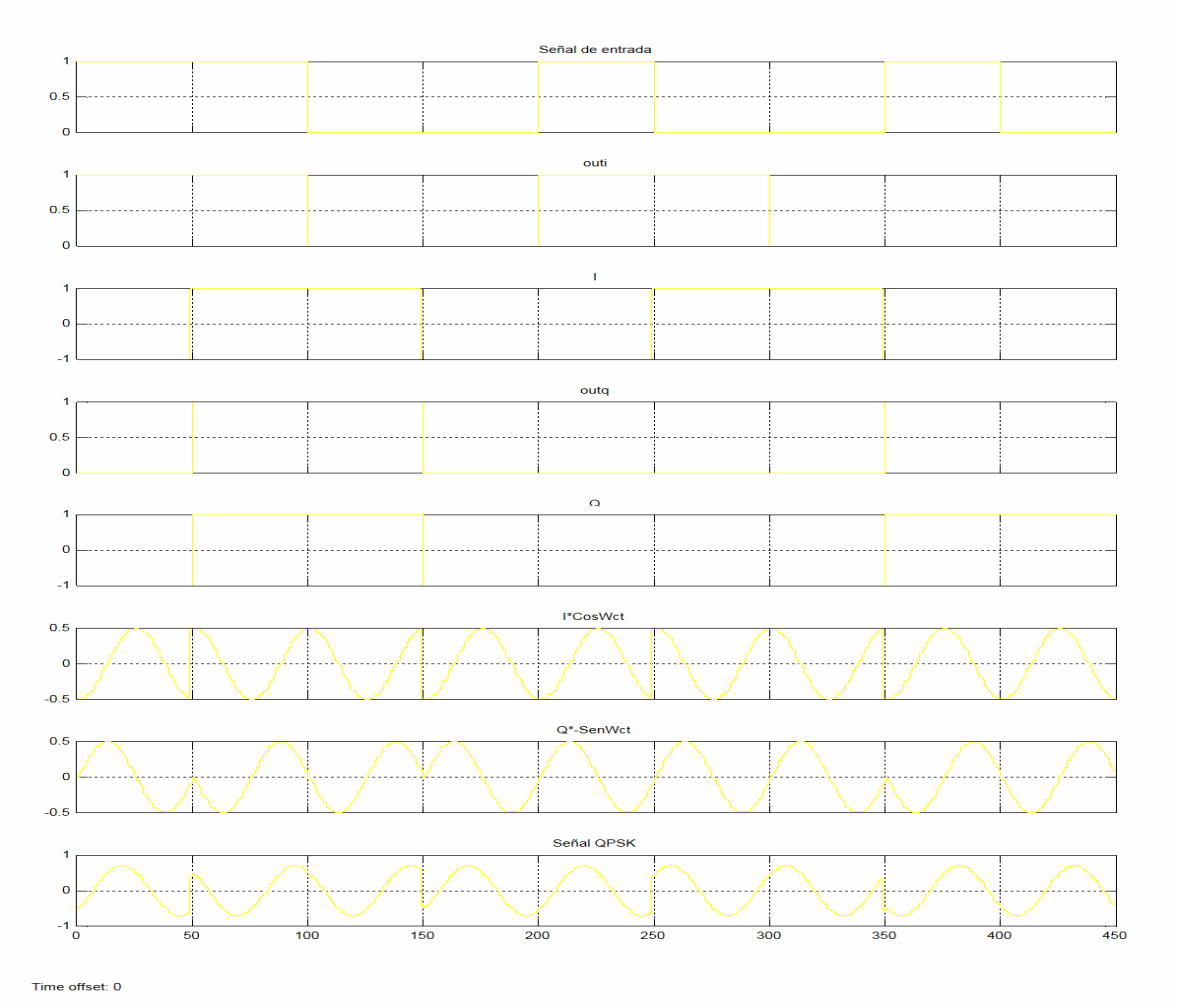



*Figura 5.10. Señales para obtener señal QPSK*

#### 4.3.2.1 Explicación diseño

La señal de entrada que se va a modular es una señal de unos y ceros con una duración de cada símbolo de 50s para efectos de simulación. Esta señal se genero a través del bloque Signal From Workspace ubicado en Signal Processing Blockset -> Signal Processing Sources -> Signal From Workspace.

Los parámetros de este bloque son:
Signal: [1 1 0 0 1 0 0 1]
Simple Time: 50; Duración de cada símbolo
Simples per frame: 1; muestras por trama

Esta señal corresponde a la señal de entrada que se ilustra en la Figura 5.6 (IN). Para obtener las 2 señales a partir de la señal de entrada se utilizo un tren de pulsos para muestrear la señal de entrada. Dicho tren se encuentra en Simulink -> Sources -> Pulse Generator. Los parámetros de dicho tren de pulsos son:
Amplitude: 1;
Period (sec): 100
Pulse Width (%): 50
Phase delay (sec): 0

Las Gateway in, Gateway in 1, tienen los siguientes parámetros:
Out type: signed
Number of bits: 10
Binary point: 8
Sample period: 1

El primer bloque MCode hace un llamado al archivo de matlab generacioniq.m, el cual tiene el siguiente código:

```
%Función generacioniq.m
&*****************************
function [i, outi, q, outq] = generacioniq (bo, bi, iini, qini)
  if bi==1
     i = bo;
     q=0;
  else
      i=0;
      q=bo;
  end
  outi=i+iini;
  outq=q+qini;
```

Note que las entradas iini y qini de este bloque se realimentan de las salidas i y q a través de dos registros. Luego de los dos registros se introducen dos bloques “Delay” con una latencia de 49. Este valor de latencia se debe a la duración del símbolo de la señal de entrada.

Las salidas del bloque MCode que invoca el archivo generacioniq.m son outi y outq. De acuerdo a la Figura 5.9 la señal outi es la misma señal de la Figura 5.6 (b) y la señal out q es la misma señal de la Figura 5.6(c) pero retrasada. Una opción para corregir la señal outq de tal forma que corresponda a la de la Figura 5.6(c) era adelantarla, es decir introducir un retardo negativo, pero Xilinx no permite manejar retardos negativos, razón por la cual es mas adecuado retrasar la señal

outi y de esta forma las cadenas I y Q estarían en el mismo instante de tiempo solo que retrasadas con respecto a la señal de entrada. El primer dato de ambas estas cadenas no se debe tener en cuenta para la modulación. La señal outi se retrasa un periodo de 50 seg. que es lo que dura cada bit de la entrada.

Una vez realizado este proceso las señales outi retrasada y outq se pasan por el bloque MCode que carga la función invertion. (Explicado en 5.1.2.1), con el fin de convertir las cadenas I y Q en forma NRZ. Si comparamos Las señales I y Q de la Figura 5.10 con las señales de la Figura 5.6 (b) y 5.6(c) se puede observar que las de la Figura 5.10 están retrasadas un periodo y el primer dato no tiene importancia.

La cadena I trabaja con la portadora $CosW_c t$ y la cadena Q con la portadora $-senW_c t$. Los bloques DDS_V_5 generan dichas portadoras para cada cadena y cada estas se multiplican con su respectiva portadora, así:

$$f_I = \text{I}^* CosW_c t \qquad \text{(Ecuación 5.2)}$$

$$f_Q = \text{Q}^*(-senW_c t) \qquad \text{(Ecuación 5.3)}$$

El bloques DDS_V_5 _0 tiene los siguientes parámetros:
Function: cosine
Channels:1
Type: Fixed
Output Frecuency Array (MHz)=[1.0]
Type: None
DDS clok rate (MHz)=50.0
Spurious free Dynamic range (db)=36
Frecuency resolution(Hz)=0.4

El bloque DDS_V_5_01 tiene los siguientes parámetros:
Function: Negative seno
Channels:1
Type: Fixed
Output Frecuency Array (MHz)=[1.0]
Type: None
DDS clok rate (MHz)=50.0
Spurious free Dynamic range (db)=36
Frecuency resolution(Hz)=0.4

Finalmente se obtiene la señal QPSK sumando las ecuaciones 5.2 y 5.3. Dicha suma se realiza a través del bloque AddSub de Xilinx.

$$f_{QPSK} = f_I + f_Q \qquad \text{(Ecuación 5.4)}$$

Las señales $f_I$, $f_Q$ y $f_{QPSK}$ se ilustran en la Figura 5.10 y corresponden a I*$CosW_c t$, Q*($-senW_c t$) y Señal QPSK.

Al comparar la señal QPSK de la Figura 5.10 con la de la Figura 5.7 note que la diferencia esta en que la señal de la Figura 5.10 esta retrasada 50 seg. con respecto a la señal de la Figura 5.7. El primer bit de la señal QPSK de la Figura 5.10 no tiene importancia para la modulación. Es decir que desde 50 seg.- 450 seg. la señal QPSK de la Figura 5.10 es la misma que la señal QPSK de la Figura 5.7.

## 4.4 Modulación Offset QPSK (OQPSK - Offset Quaternary Phase Shift Keying)

### 4.4.1 Definición.

Es un caso particular de la modulación QPSK, en la que uno de los bits – el del canal Q o el del canal I – es retrasado en medio periodo de bit ($T_b$/2) respecto del bit del otro canal. Como se explico anteriormente, para QPSK cada bit de la cadena I y Q tiene una duración de $T_b$ = 2T seg., mientras que la señal de entrada tiene una duración de T seg. para cada bit. De acuerdo a esto la cadena se retrasa T seg.

En la Figura 5.11 se muestra un ejemplo en el que el bit de información del canal Q es retrasado un tiempo $T_b$/2 respecto al bit del canal I. También se puede apreciar que si la cadena de entrada (IN) tiene un numero par de bits la cadena Q durara T segundos mas que la cadena de entrada, dado que se retrasa T seg. con respecto a la cadena I.

En la Figura 5.12 se puede apreciar que el desplazamiento por medio intervalo de bit, hace que las fluctuaciones de amplitud se minimicen puesto que la fase nunca cambia a 180º, es decir que se consigue que no se produzca el cambio de mas de un bit simultáneamente en código del dibit – nunca se dará una transición de 00 a 11 o de 01 a 10: Esto hará que tampoco haya un desplazamiento mayor de 90º en la fase de la señal OQPSK.

Nunca ocurrirá un cambio de 180º como ocurría en QPSK. Si una componente de la señal I o Q cambia con respecto al valor anterior de sus respectivas cadenas, ocurre un cambio de fase de 90º [20].

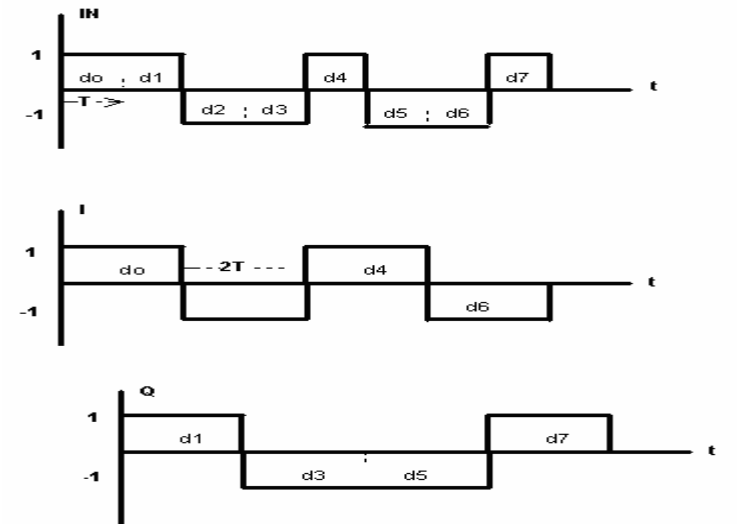



*Figura 5.11. Cadena I y Q para OQPSK*

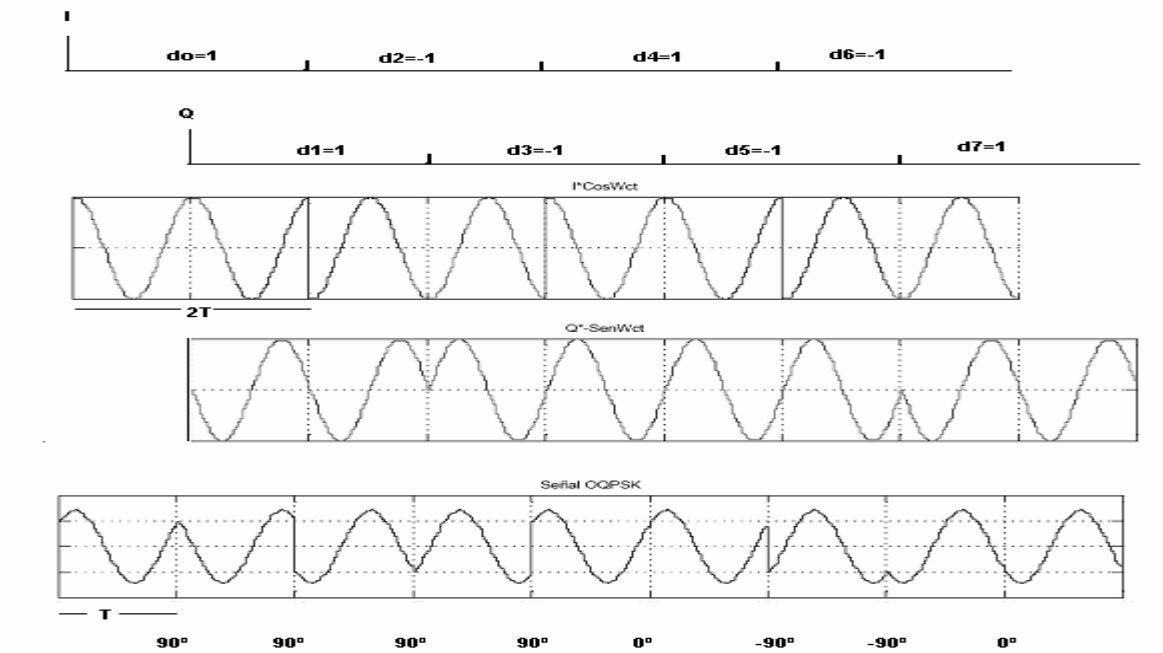



*Figura 5.12. Señal OQPSK*

OQPSK resulta ventajoso pues el desplazamiento de fase es limitado durante el proceso de modulación, pero presenta la desventaja que el ancho de banda mínimo necesario para transmitir una señal OQPSK es el doble del necesario de QPSK.

#### 4.4.1.1 Modulador OQPSK

En la figura 5.13 se ilustra el diagrama en bloque del modulador OQPSK

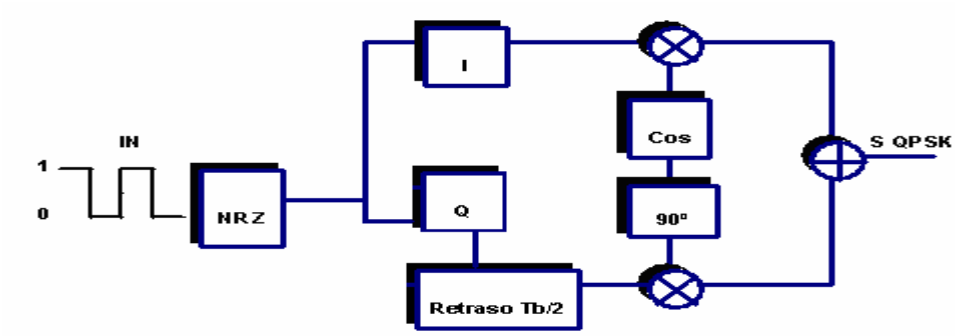



*Figura 5.13. Modulador OQPSK*

El proceso para el modulador OQPSK es el mismo explicado en 5.2.1.1, a excepción que la cadena Q se retrasa ($T_b$ /2)

### 4.4.2 Simulación de modulación OQPSK

El diseño del modulador OQPSK se ilustra en la Figura 5.14.

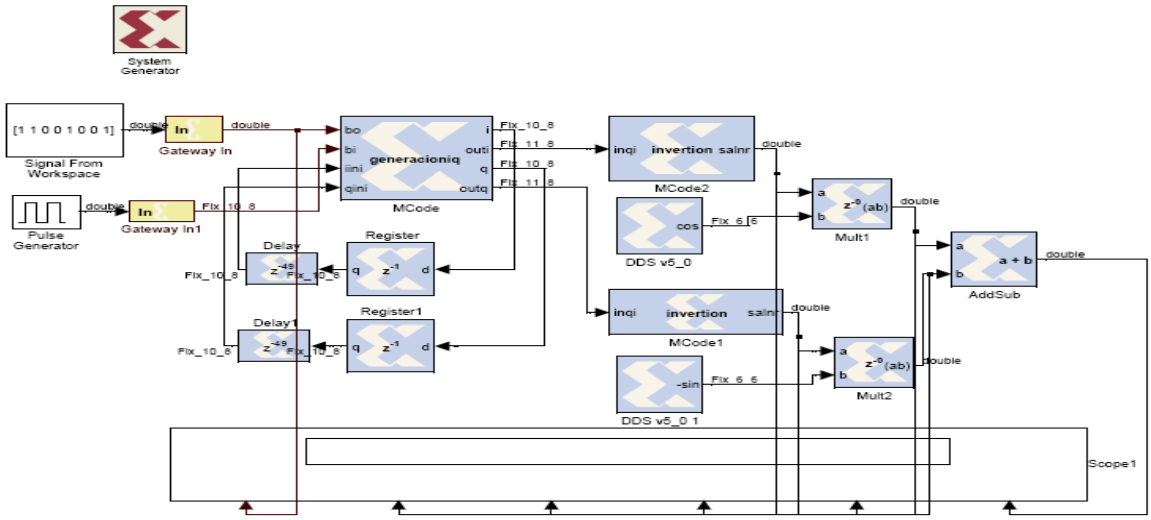



*Figura 5.14. Modulador OQPSK*

La Figura 5.15 presenta las señales para el diseño de la Figura 5.14

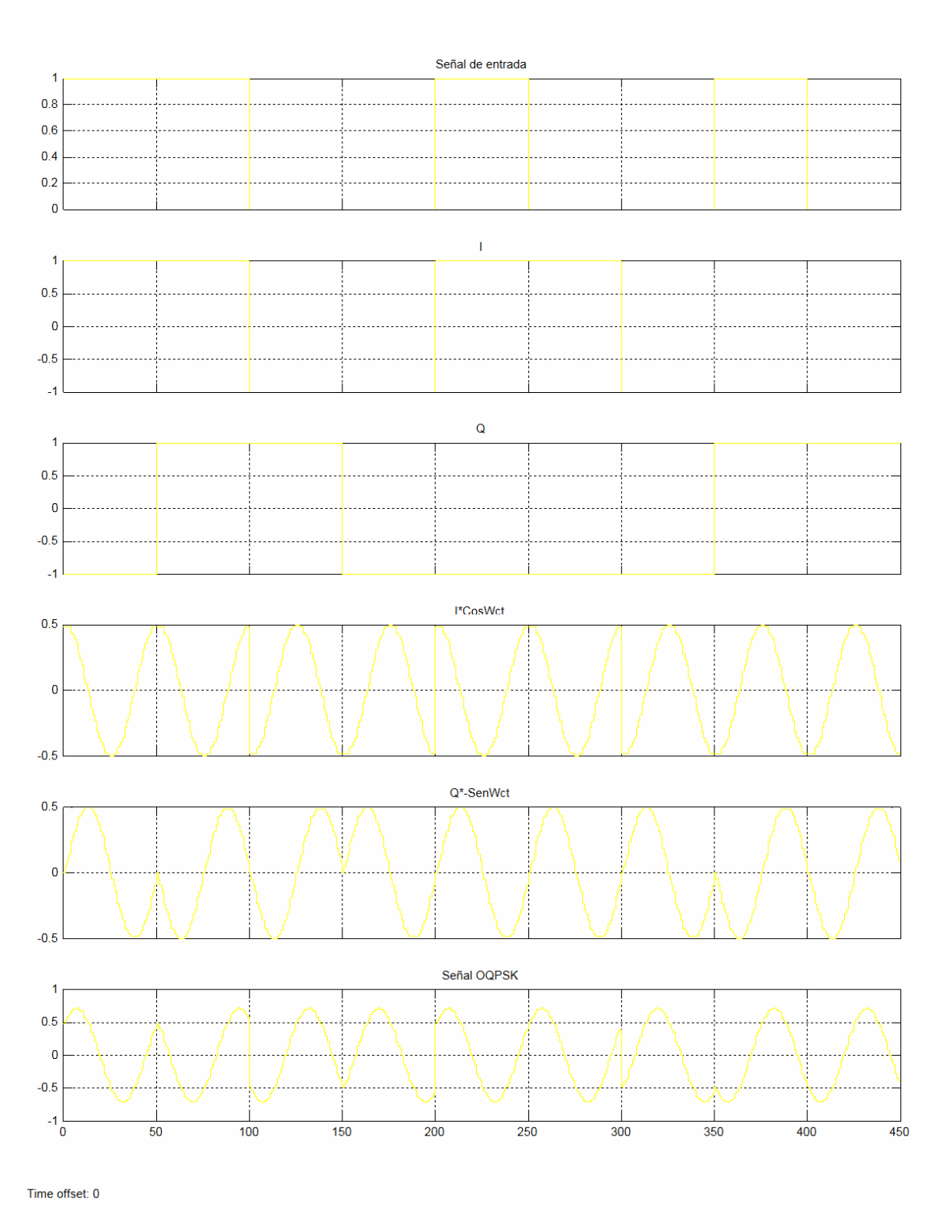



*Figura 5.15. Señales para modulador OQPSK*

#### 4.4.2.1 Explicación diseño

La modulación OPSK es la misma QPSK, pero se debe retrasar un periodo de bit una de las cadenas I o Q. En 5.2.2.1 se explico que la señal outq del bloque MCode salía retrasada T seg., es decir que no es necesario agregar un bloque de retardo a esta salida. La señal outi del bloque MCode no se requiere retrasar, pues outq ya esta retrasada. Por la tanto el diseño OQPSK solo tiene un bloque de diferencia (Delay2) con respecto al diseño que se realizo para QPSK (Figura 5.14)

Retomando la Figura 5.14, las salidas outi y outq están en forma retorno a cero, y se convierten en forma NRZ a través de los bloques Mcode2 y Mcode1, que cargan el archivo invertion.m. De esta forma las salidas de los bloques MCode son las cadenas I y Q. Dado que la cadena Q esta retrasada T seg., con respecto a la cadena I, en el tiempo comprendido entre 0 -T seg. se coloca un cero para la cadena Q. Una vez obtenidas las cadenas I y Q se multiplican por las portadoras $CosW_c t$ y $-senW_c t$ (como en QPSK). La señal OQPSK será la suma estos productos, es decir:

$$\text{QPSK= I*}CosW_c t \text{ + (Q*}-senW_c t\text{)} \qquad \text{(Ecuación 5.5)}$$

Si comparamos las señales QPSK y OQPSK de las Figuras 5.10 y 5.15 se puede notar que en QPSK se pueden presentar cambios de fase de 180º, mientras que en OQPSK nunca sucederá esto. De esta forma podemos comprobar la eficiencia de OQPSK con respecto a QPSK.

## 4.5 Modulación Desplazamiento de Frecuencia Mínimo (MSK – Minimum Shift keying)

### 4.5.1 Definición.

En 5.4.1 se menciono que OQPSK se obtiene de QPSK con el retardo de T seg. de la cadena Q con respecto a la cadena I. Este retardo no tiene efecto en el error o ancho de banda.

MSK se deriva de OQPSK reemplazando el pulso rectangular en amplitud con un pulso senoidal de medio ciclo. La señal MSK se define como:

$$f_{MSK}(t) = f_I(t)Sen(\frac{\pi t}{2T})Cos2\pi ft + f_Q(t)Sen(\frac{\pi t}{2T})Sen2\pi ft \qquad \text{(Ecuación 5.6)}$$

Donde $f_I(t)$ y $f_Q(t)$ son las cadenas rectangulares I y Q de OQPSK, $Sen(\frac{\pi t}{2T})$ es la onda senoidal por la que se multiplican dichos pulsos rectangulares para obtener los pulsos senoidales y $Cos2\pi ft$ , $Sen2\pi ft$ son las señales portadoras para los pulsos senoidales.

El proceso para obtener la señal MSK ( $f_{MSK}(t)$ ) se ilustra en la figura 5.16

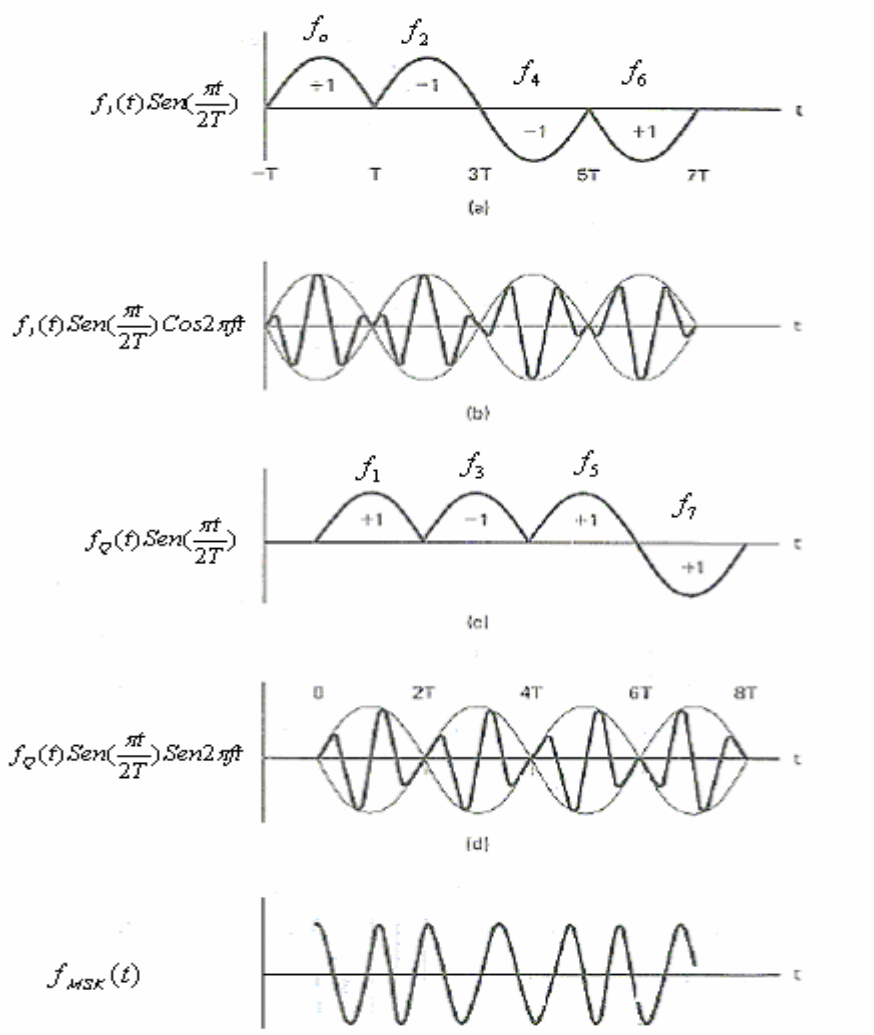



*Figura 5.16. Proceso para obtener Señal MSK [21]*

La modulación MSK mejora los cambios abruptos de fase como se puede observar en la Figura 5.16.

El proceso para la modulación MSK es el mismo que para QPSK a excepción que las cadenas I y Q rectangulares son reemplazadas por pulsos senoidales. Nótese que para obtener dichos pulsos senoidales se multiplican las cadenas rectangulares I y Q por una onda senoidal de periodo 4T. Donde T es la duración de cada bit de la entrada original de información. La duración de cada bit en la cadenas rectangulares I y Q y en los pulsos senoidales es 2T.

#### 4.5.1.1 Modulador MSK

EL diagrama en bloques del modulador MSK se ilustra en la Figura 5.17

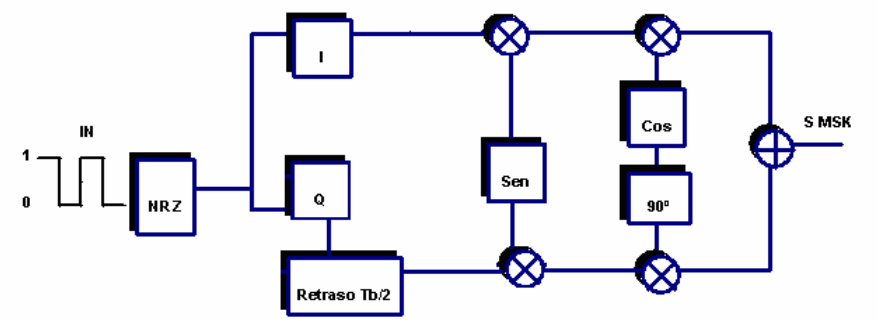



*Figura 5.17. Modulador MSK*

El proceso para el modulador MSK es similar que el modulador OQPSK (Figura 5.13). Note que la salida del bloque I se multiplica por un bloque seno que corresponde a la señal $Sen(\frac{\pi t}{2T})$ de la ecuación 5.6. El resultado de esta multiplicación produce los pulsos senoidales de la cadena I. Los pulsos Q retardados (salida bloque retraso Tb/2), también se multiplican por el bloque seno, para convertir los pulsos Q rectangulares en pulsos senoidales. Una vez obtenidos los pulsos I y Q senoidales se multiplican por sus respectivas portadoras. Finalmente se suman ambos productos para obtener la señal MSK.

### 4.5.2 Simulación modulación MSK

El diseño del modulador MSK se ilustra en la Figura 5.18.

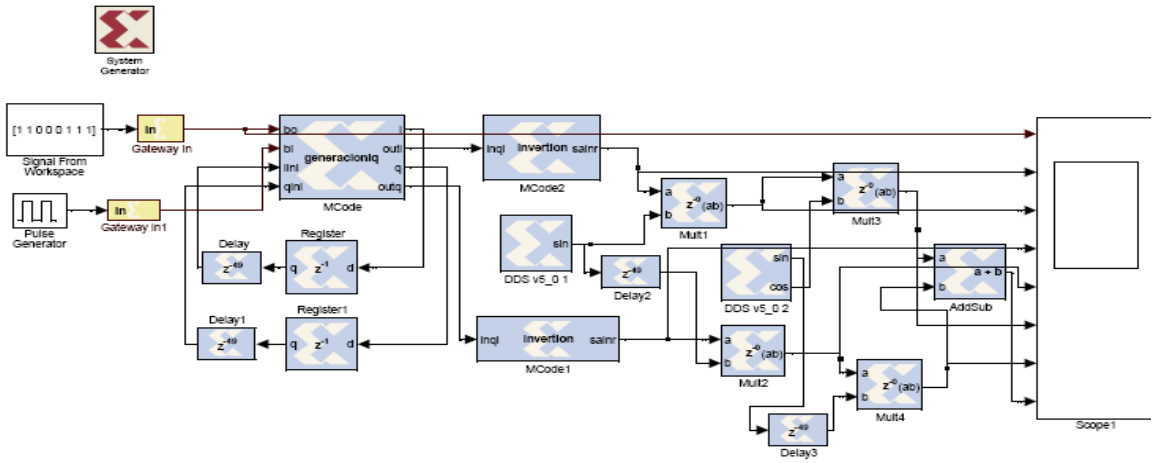



*Figura 5.18. Modulador MSK*

Las señales para obtener la señal MSK se ilustran en la Figura 5.19

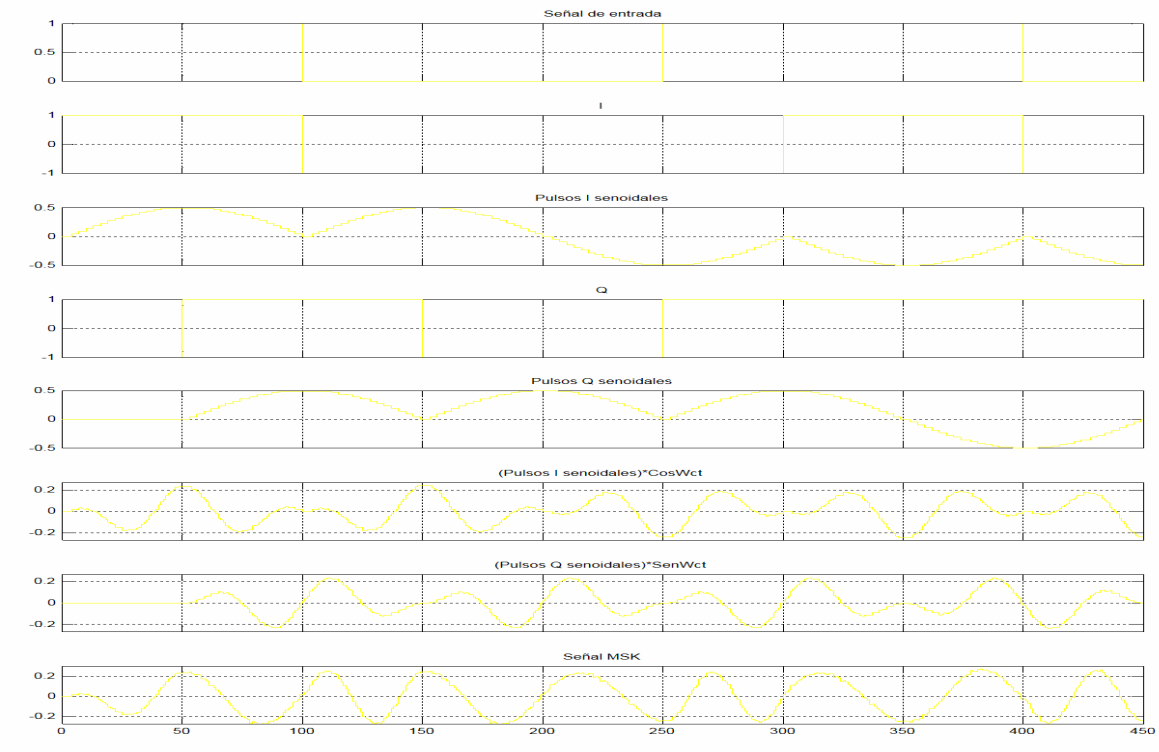



*Figura 5.19. Señales para obtener señal MSK*

#### 4.5.2.1 Explicación diseño

Dado que para MSK se deben reemplazar los pulsos rectangulares de las cadenas I y Q de OQPSK por pulsos senoidales, se mantiene el mismo diseño OQPSK hasta la parte de los bloques "MCode" que cargan la función invertion.m. Es decir la salida de estos bloques "MCode" corresponde a las cadenas I y Q rectangulares en forma NRZ.

Para convertir los pulsos rectangulares de la cadena I en pulsos senoidales, se multiplica dicha cadena por una onda seno con periodo de 4T. Para generar la onda seno se utiliza el bloque "DDS_V_5_01". El bloque "Mult1" realiza la multiplicación de la cadena I con la onda seno. La salida del bloque "Mult1" serán los pulsos I senoidales. Dichos pulsos se pueden apreciar en la Figura 5.19.

Para convertir los pulsos rectangulares de la cadena Q en pulsos senoidales, se multiplica dicha cadena por una onda seno con periodo de 4T. Dado que el primer bit de Q no tiene importancia, ya que Q esta retrasada T seg., la onda seno también debe retrasarse T seg. . Esto se hace a través del bloque "Delay2". Para generar la onda seno se utiliza el bloque "DDS_V_5_01". El bloque "Mult2" realiza la multiplicación de la cadena Q con la onda seno retrasada. La salida del bloque "Mult1" seran los pulsos Q senoidales. Dichos pulsos se pueden apreciar en la Figura 5.19. Note que los "pulsos Q senoidales" de esta figura empiezan desde 50 seg., pues el primer bit de Q no importa.

Los parámetros del bloque DDS_V_01 son:

Function: Sine
Channels:1
Type: Fixed
Output Frecuency Array (MHz)=[1.0]
Type: None
DDS clok rate (MHz)=200.0
Spurious free Dynamic range (db)=36
Frecuency resolution(Hz)=0.4

A continuación se deben multiplicar los pulsos I senoidales por $CosW_c t$. Los pulsos Q senoidales se deben multiplicar por $senW_c t$. La frecuencia de las ondas senoidales es 1/T. Para generar las ondas $CosW_c t$ y $senW_c t$ se utiliza el bloque "DDS_V_5_02" y la multiplicación de estas ondas senos con los pulsos senoidales se

hace a través de "Mult3" y "Mult 4". "El bloque Addsub" suma las salidas de los bloques de "Mult3" y "Mult 4" y la salida del bloque sumador es la señal MSK, que como puede observarse en la Figura 5.19 no presenta cambios abruptos de fase.

Los parámetros del bloque DDS_V_02 son:

Function: Sine and Cosine
Channels:1
Type: Fixed
Output Frecuency Array (MHz)=[1.0]
Type: None
DDS clok rate (MHz)=50.0
Spurious free Dynamic range (db)=36
Frecuency resolution(Hz)=0.4

## 4.6 Modulación GMSK

Es un esquema de modulación binaria simple que se puede ver como una derivación de MSK, en el que se reduce el espectro pasando la señal NRZ por un filtro Gaussiano antes del modulador MSK [22]. En GMSK, los lóbulos laterales del espectro de una señal MSK se reducen pasando los datos modulantes a través de un filtro Gaussiano de premodulación. El filtro gaussiano aplana la trayectoria de fase de la señal MSK y por lo tanto, estabiliza las variaciones de la frecuencia instantánea a través del tiempo. Esto tiene el efecto de reducir considerablemente los niveles de los lóbulos laterales en el espectro transmitido. En la Figura 5.20 se puede apreciar el diagrama en bloques de la modulación GMSK

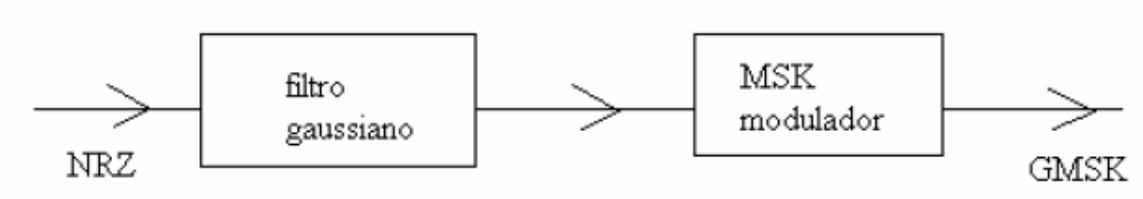



*Figura 5.20. Modulación GMSK*

El diagrama que se ilustra en la Figura 5.20 es la forma más sencilla de generar GMSK.

En la práctica, GMSK es muy atractiva por su excelente eficiencia de potencia y espectral. El filtro de premodulación introduce interferencia intersimbólica ISI en

la señal transmitida, pero esta degradación no es grave si el parámetro BT del filtro es mayor de 0.5.

La simulación de GMSK no se realizo en Xilinx, pues la herramienta FDATool no tiene esta clase de filtro; pero podría implementarse a través de un filtro FIR; con el cálculo de los coeficientes respectivos. Este aspecto no se alcanzo a estudiar, ya que se opto por realizar a manera de simulación en código en matlab la modulación y demodulación GMSK para un canal PLC. La modulación GMSK realizada en código Matlab es más eficiente que la del diagrama de la Figura 5.20, pero precisamente por ser más compleja, su implementación en Xilinx no es tan sencilla (ver sección 4.3.1.3.2 de la monografía "Solución de Acceso a una Intranet utilizando la Tecnología PLC").

## 4.7 Implementación de un Codificador Convolucional

De acuerdo al funcionamiento del codificador convolucional explicada en secciones anteriores, es posible implementar un modelo sencillo que permita verificar el funcionamiento de un codificador convolucional. En este ejemplo hacemos uso de los siguientes elementos:

- El bloque System Generator, que permitirá compilar las librerías de Xilinx utilizados para el diseño, Xilinx Blockset->Basic Elements-> System Generator.

- El bloque Convolutional Encoder, que se encuentra en la librería Xilinx Blockset->Communication->FEC->Convolutional Encoder v3.0.

- El elemento Gateway in, permite conectar los elementos de simulink con los elementos de xilinx. Gateway out, para conectar elementos de xilinx a elementos de simulink.

- Para la fuente de prueba se utilizó el elemento Bernoulli Binary Generator.

Si se desea comparar con el bloque codificador de simulink Communications Blockset->Error detection and correction->Convolucional->Convolutional Encoder. El elemento unbuffer es opcional, comúnmente utilizado cuando el flujo de datos es demasiado alto.

El diagrama implementado se muestra a continuación:

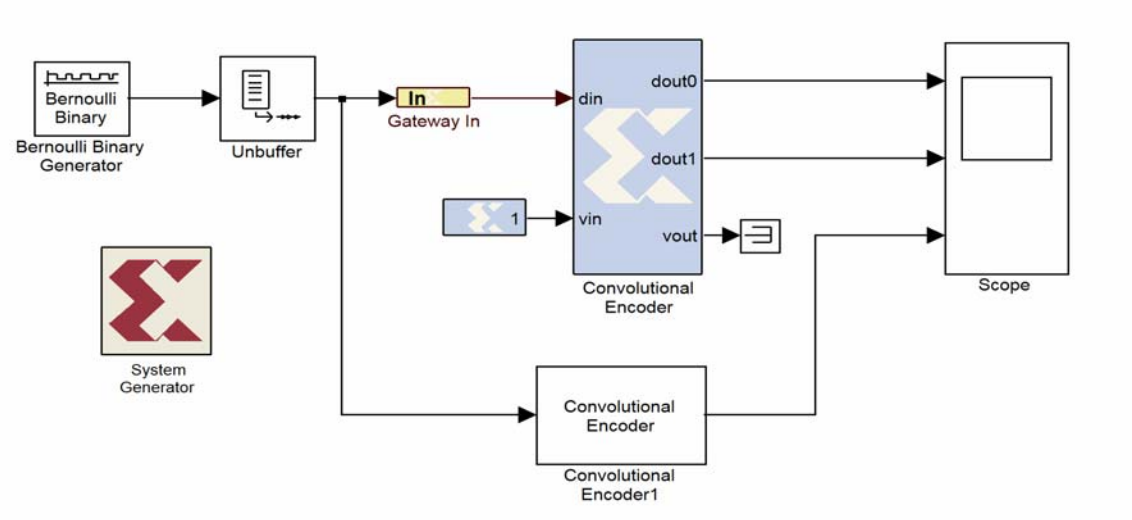



*Figura 5.21. Diagrama de Codificador convolucional implementado*

El elemento System Generator permite configurar las características de funcionamiento, en el cual se configurará el período de simulink utilizado tal como lo indica la Figura 5.22 (doble clic para abrir cada elemento del sistema):

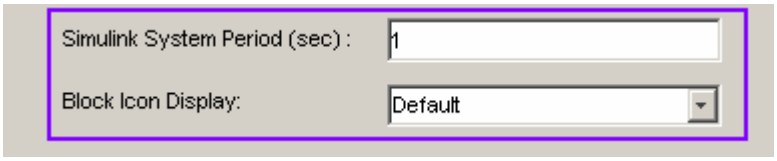



*Figura 5.22. Configuración del System Generator*

De acuerdo a esta configuración, la señal obtenida se muestra en la Figura 5.23

La salida 0 y 1 del codificador convolucional se encuentran retrasadas en un solo período de muestreo con respecto a la señal de codificador convolucional de simulink. Esto se debe a que los elementos reales introducen retardo de la señal debido a la inicialización de sus componentes.

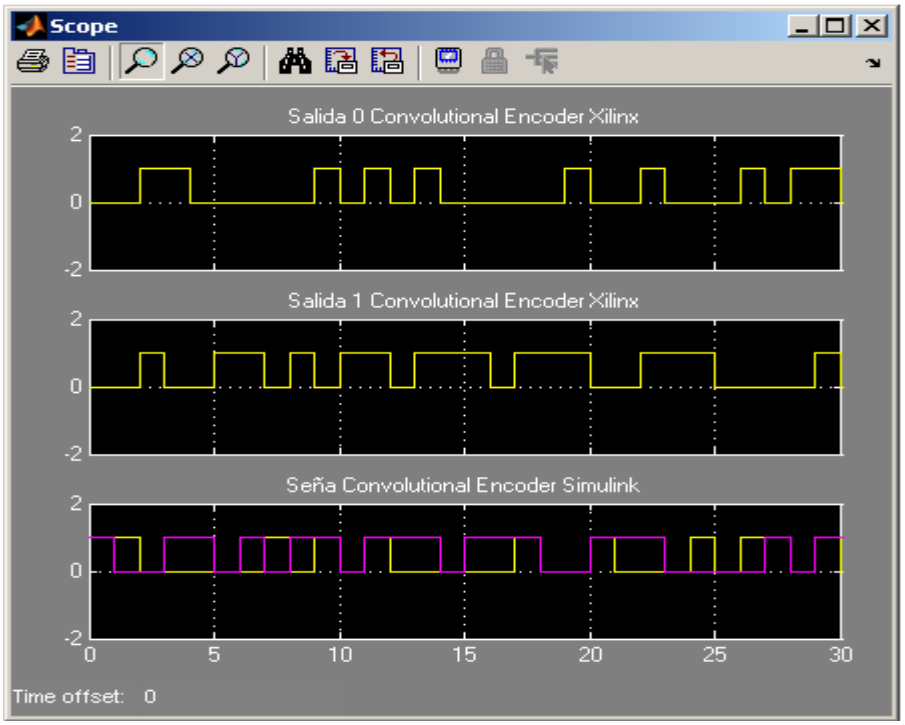



*Figura 5.23. Respuesta obtenida por el codificador convolucional*

## 4.8 Implementación de un Decodificador con Algoritmo de Viterbi

Como se ha visto en las secciones anteriores, la codificación convolucional puede ser decodificada haciendo uso del algoritmo de Viterbi. Tanto Xilinx como Simulink, hacen uso de bloques que implementan esta función. Las Figuras 5.24 y 5.25 ilustran la implementación realizada y la señal obtenida al utilizar tanto bloques de Simulink como bloques de Xilinx

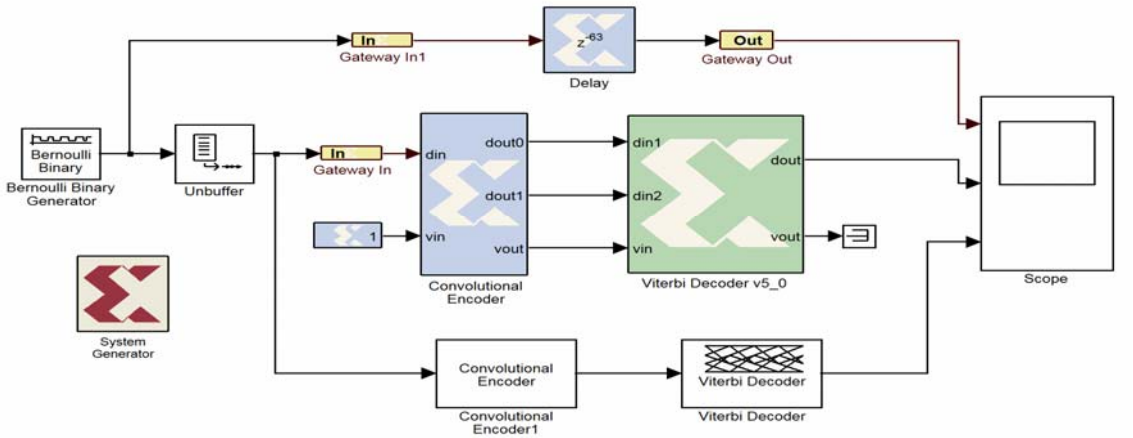



*Figura 5.24. Implementación de un codificador convolucional con decodificación de Viterbi*

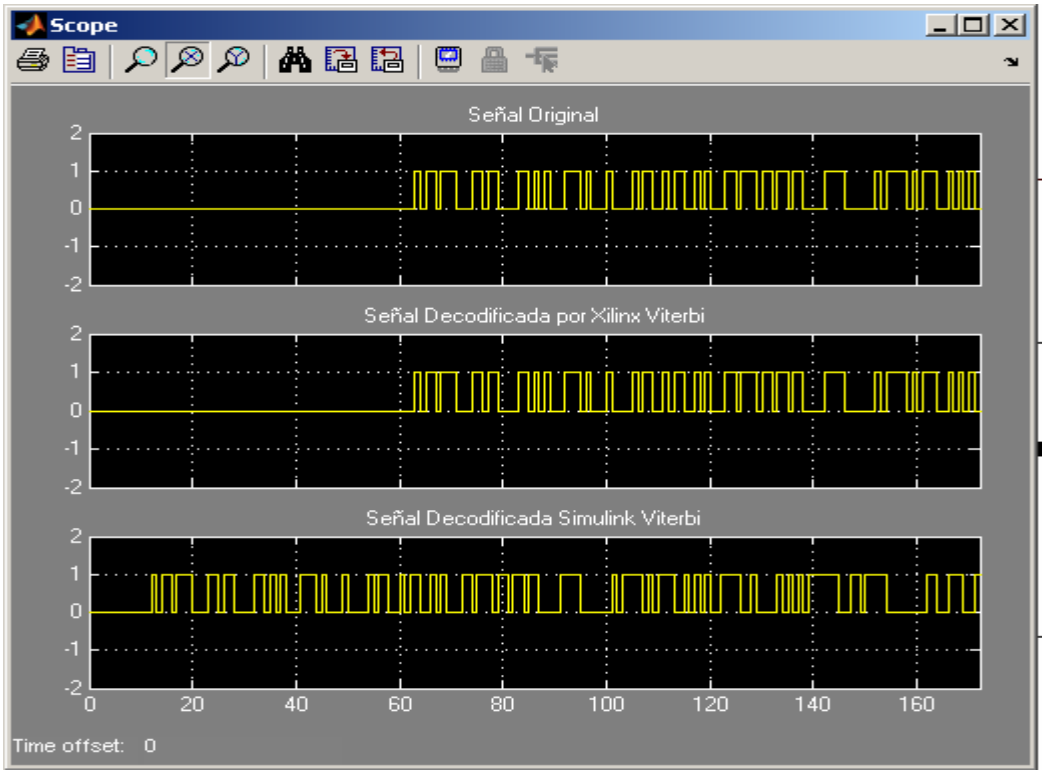



*Figura 5.25. Señales obtenidas en la implementación del decodificador Viterbi*

La configuración de los parámetros del codificador convolucional, es la misma utilizada para la Figura 5.21 y se muestra en el siguiente diagrama de configuración:

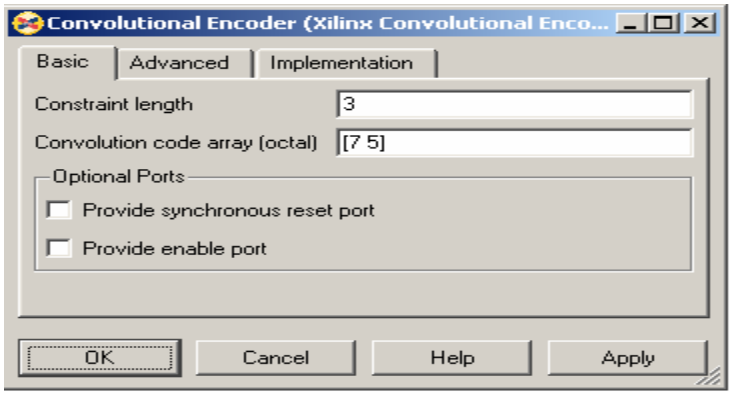



*Figura 5.26. Configuración del Codificador Convolucional en Xilinx*

La configuración de los parámetros del codificador convolucional se realizó de acuerdo al análisis de funcionamiento, y al ejemplo dado en la sección de códigos correctores de errores en el capítulo 2. El parámetro "Constraint length"

especifica la cantidad de registros de corrimiento utilizado, y el parámetro "convolution code array" especifica el arreglo de los sumadores (xor) utilizado para obtener la maquina de estados o secuencia del codificador.

Para el bloque Viterbi Decoder v5_0, la configuración es la siguiente:

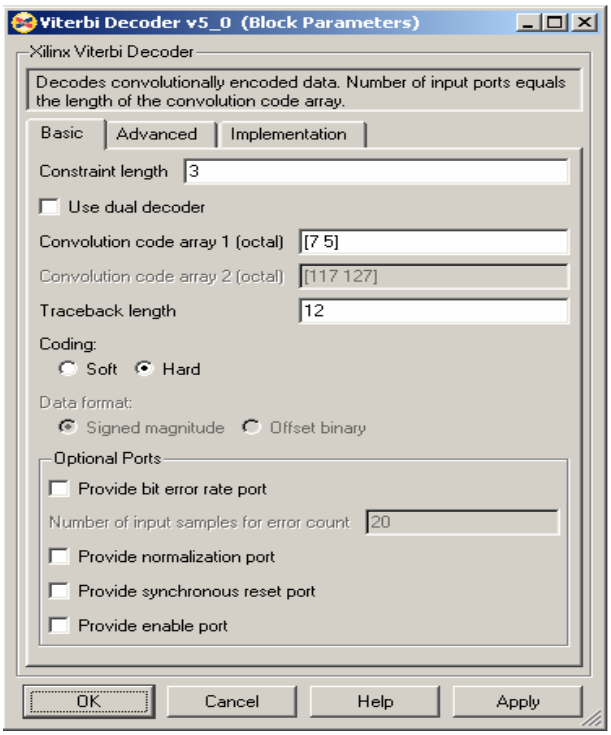



*Figura 5.27. Parámetros de Configuración del Decodificador de Viterbi*

El arreglo de convolución debe ser el mismo que el arreglo utilizado para el codificador convolucional. El parámetro "traceback length" especifica la longitud del número de rutas posibles que el codificador puede utilizar para decodificar la señal, teniendo en cuenta el diagrama de Trellis para su decodificación.

## 4.9 Implementación del bloque de Mapeo QPSK

El bloque de mapeo que se muestra a continuación se realiza mediante QPSK, aunque es posible realizar otro tipo de mapeo como Modulación por Amplitud de Quadratura (*QAM - Quadrature Amplitude Modulation)*.

Para la implementación del bloque de la Figura 5.28, se hizo uso de dos elementos adicionales:

- Serial to Parallel: Convierte el flujo serial entregado por cada salida del codificador convolucional a un flujo paralelo, para facilitar la obtención de las componentes en cuadratura que formarán la constelación deseada

- ROM MEMORY (Memoria de Solo Lectura (*ROM - Read Only Memory*)). Este elemento es el encargado de formar el mapeo asignando a cada dato de entrada su valor correspondiente en la constelación. Si se desea realizar un proceso de intercalamiento, los datos deben concatenarse para luego realizar la permutación de sus posiciones, y de forma siguiente pasar el flujo obtenido de serial a paralelo para su correspondiente mapeo.

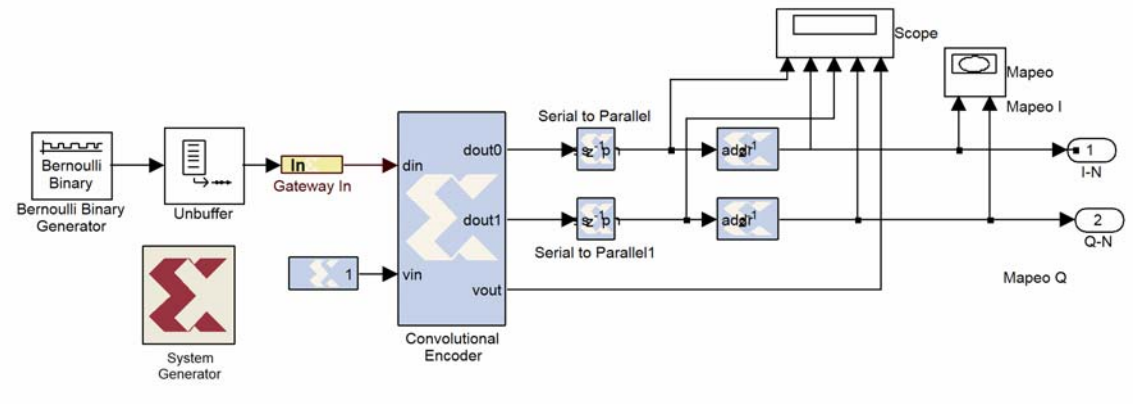



*Figura 5.28. Implementación del Mapeo QPSK*

La Figura 5.29 muestra la configuración básica de la ROM.

El parámetro depth indica la profundidad, es decir, el número de elementos que se tomarán de base para formar el símbolo mapeado (para QPSk se tomarán dos bits por dato mapeado). El vector inicial, son los posibles valores que puede tomar en la constelación, como se muestra en la Figura 5.30.

El valor del vector inicial de la ROM [-1 1], indica que si la señal de entrada es un 0, la señal de salida será un -1, y si la señal de entrada es 1 la señal de salida será 1. La ROM introduce un retardo de un ciclo de reloj respecto a la señal de entrada.

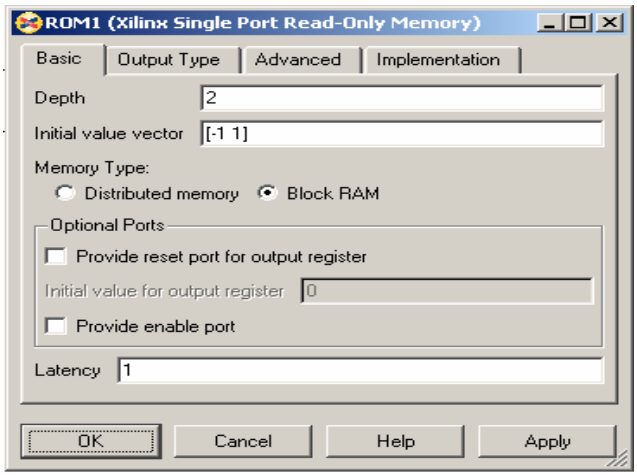



*Figura 5.29. Bloque slice Xilinx*

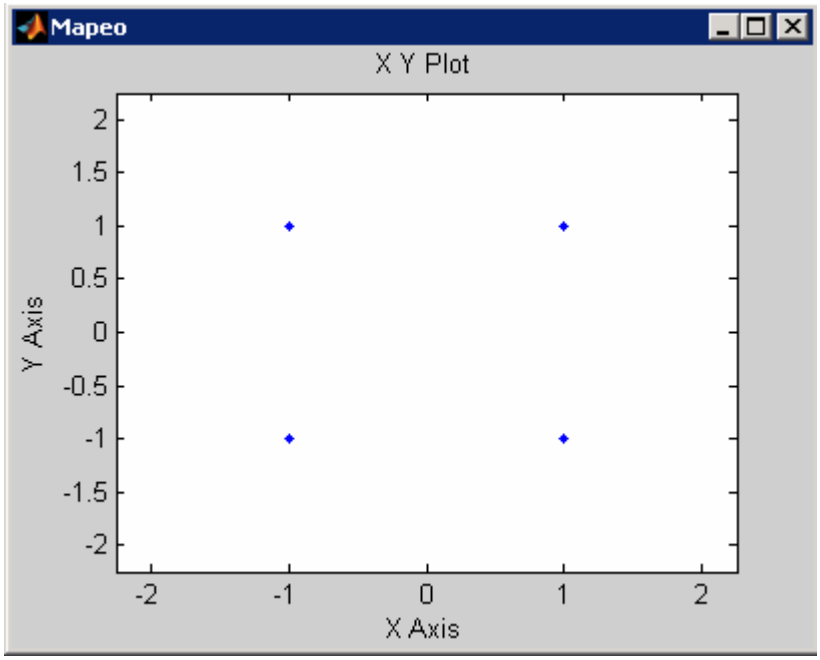



*Figura 5.30. Constelación QPSK obtenida*

Si se desea implementar un esquema QAM, es necesario aumentar el parámetro depth a 4, 16, 64, etc., de acuerdo al esquema que se quiera utilizar, y los posibles valores que tomará cada una de los símbolos en la constelación.

Dependiendo del tipo de Mapeo, el convertidor serial a paralelo debe ser configurado de modo tal que entregue los bits adecuados. Es decir, para QPSK debe entregar un número de bits igual a "1", pues QPSK utiliza dos bits por cada dato, obtenido como $2^n$, por lo cual los valores posibles en el vector inicial de la ROM son dos (-1 y 1).

La configuración del convertidor serial a paralelo se muestra en la Figura 5.31 para una configuración QPSK:

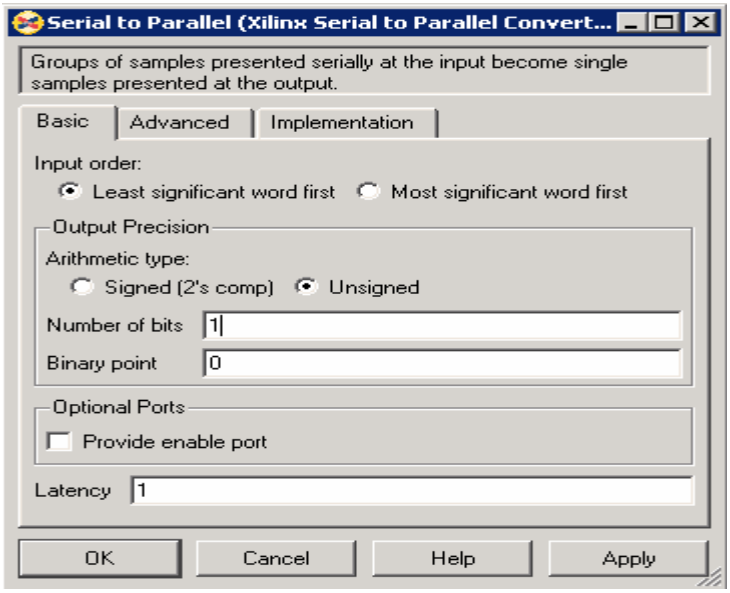



*Figura 5.31. Configuración del bloque Serial a Paralelo para un mapeo QPSK*

Para implementar un mapeo 16 QAM, se configura el bloque serial como lo indica la Figura 5.32 y la memoria ROM de la forma como lo indica la Figura 5.33.

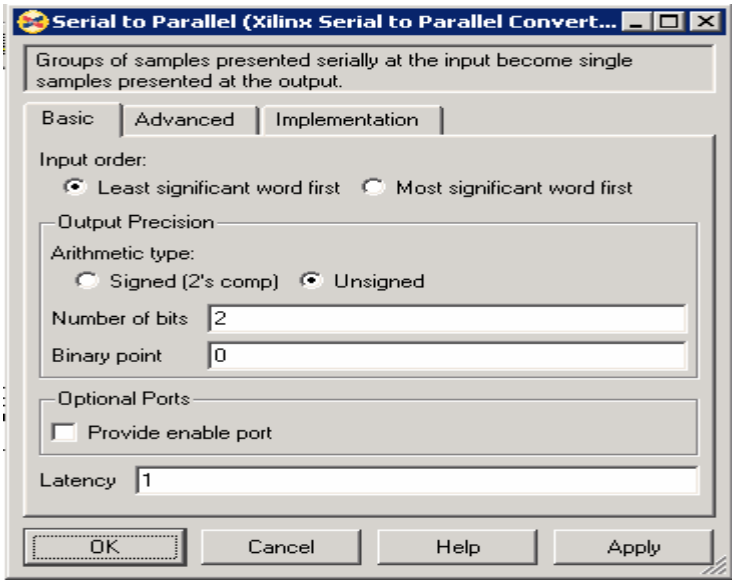



*Figura 5.32. Configuración bloque serial a paralelo para un mapeo 16QAM*

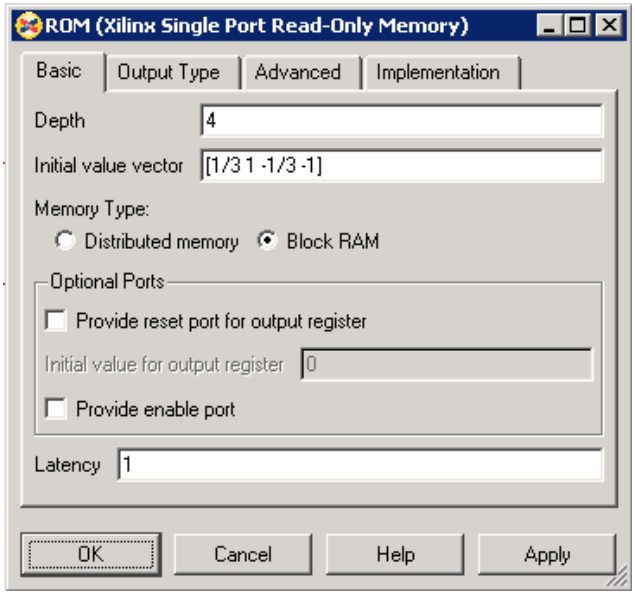



*Figura 5.33. Configuración de la ROM para un mapeo 16 QAM*

El diagrama del mapeo obtenido puede verse en la Figura 5.34.

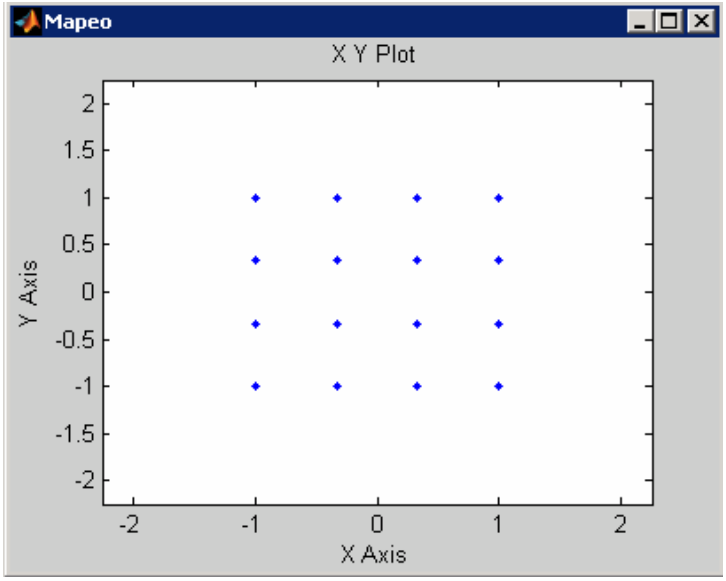



*Figura 5.34. Mapeo 16 QAM obtenido con los elementos de Xilinx.*

La latencia especificada por los bloques (igual a 1), es un parámetro por defecto que puede ser sólo mayor o igual a 1.

Para la implementación del bloque de demapeo, puede consultarse la ayuda de Matlab -> demos-> Xilinx, donde se da un modelo aproximado del demodulador QAM.

# 5. REFERENCIAS BIBLIOGRÁFICAS

[1] Hrasnica Halid, Lehnert Ralf, "Powerline Communications in Telecommunication Access Area", Cátedra de Telecomunicaciones, Universidad Tecnológica de Dresden, Alemania, 2003

[2] Hrasnica Halid, Lehnert Ralf, "Powerline Communications for Access Networks. Performance Study of the MAC Layer", Cátedra de Telecomunicaciones, Universidad Tecnológica de Dresden, Alemania 2003

[3] Hrasnica Halid, Lehnert Ralf, "Extended ALOHA and Hybrid-Polling Reservation MAC Protocols for Broadband Powerline Communications Access Networks", Cátedra de Telecomunicaciones, Universidad Tecnológica de Dresden, Alemania. 2003

[4] Stantcheva M, Begain K, Hrasnica H, Lehnert R, "Suitable MAC Protocols for an OFDM Based PLC Network – Proceedings of ISPLC2000 - 4th International Symposium on Power-Line Communications and its Applications", Limerick-Irlanda, Abril 5 – 7 de 2000.

[5] Arriagada Escudero Alvaro, Bustos Muñoz Jorge, Rojas Quintana Mauricio, "FEC (Forward Error Correction) y Código Reed-Solomon", Departamento de Ingeniería Eléctrica, Universidad de Concepción.

[6] Campuzano Zubeldia Álvaro, Varela Cuadrado Iker, "Reducción de la probabilidad de error en transmisión mediante la utilización de codificación convolucional y decodificación de Viterbi"
www.tecnun.es/Asignaturas/transdat/ficheros%5Cviterbi.pdf

[7] Hrasnica Halid, Haidine Abdelfatteh, Lenhnert Ralf, "Broadband Powerline Communications Networks", Universidad de Tecnología de Dresden, Alemania, 2003.

[8] Matlab Help – Reed Solomon. Ayuda proporcionada por el software de Matlab

[9] Matlab Help – Convolutional Encoder. Ayuda proporcionada por el software de Matlab

[10] Benigno Rodríguez Díaz, Estudio de la conveniencia del agregado de un bloque de precancelación de ruido, en un sistema basado en el estándar de WLAN´s IEEE 802.11ª. Monografía de Tratamiento Estadístico de Señales. Facultad de Ingeniería, Universidad de la República. Agosto de 2004.

[11] Joaquín Luque Rodríguez, Sebastián Clavijo Suero. Modulación de Señales Digitales. Departamento de Tecnología en Electrónica, Universidad de Sevilla. Sevilla España, 1995.

[12] Luis Gabriel Sienra, *QAM la guía completa.* Cinit Centro de investigación e innovación en telecomunicaciones. Disponible en <http://www.cinit.org.mx/content/pdfs/articulos/articulo10.pdf>

[13] Halid Hrasnica, Abdelfatteh Haidine, Ralf Lenhnert, *Broadband Powerline Communications Networks*. Universidad de Tecnología de Dresden, Alemania.

[14] Moisès Serra i Serra, Prototipatge Ràpid de la Capa Física d'OFDM: cas HIPERLAN/2. Universidad Autónoma de Barcelona, Bellaterra, Marzo de 2005

[15] Leonardo Jiménez, Joaquín Parrado, Carlos Quiza, Carlos Súarez. Modulación multiportadora OFDM. Revista Ingeniería - 2001-2. Universidad Distrital, 2001

[16] Manfred Zimmermann and Klaus Dostert, A Multipath Model for the Powerline Channel, IEEE Transactions on Communications, VOL. 50 – No 4, Abril de 2002.

[17] L. Berrojo, J. Prat, Diseño de ASIC Complejos: El reto AmerHis como caso práctico. Alcatel Espacio

[18] http://www.xilinx.com/products/software/sysgen/app_docs /user_guide_Chapter_7_Section_3.htm

[19] Daniel Warne. Software Radio Architectures - Part 2. University of Southern Queensland Faculty of Engineering & Surveying. Octubre 2004

[20] Nikhil S. Bhatia. A Physical Layer Implementation of Reconfigurable Radio. Faculty of the Virginia Polytechnic Institute and State University. December 2004.

[21] Dr. Víctor Hinostroza Scientific Atlanta. Convenio UACJ – SA .Comunicaciones análogas y digitales

[22] University Hull. Appendix D: Digital Modulation and GMSK

[23] Enrique Hernández. Ing.Informática 3. Sistemas de Telecomunicación. Tema 5: Modulación Digital