UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

IMPLEMENTACIÓN DE CODIFICACIÓN DE CANAL PARA SISTEMAS DE COMUNICACIONES DIGITALES

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL ELECTRICISTA

ALEX MAURICIO BECERRA SAAVEDRA

PROFESOR GUÍA:
SR. PATRICIO PARADA SALGADO.

MIEMBROS DE LA COMISIÓN:
SR. HÉCTOR AGUSTO ALEGRÍA.
SR. PABLO NAVARRETE MICHELINI.

SANTIAGO DE CHILE
MAYO 2010

RESUMEN DE LA MEMORIA
PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL ELECTRICISTA
POR: ALEX MAURICIO BECERRA SAAVEDRA
FECHA: 28 DE MAYO DE 2010
PROF. GUÍA: SR. PATRICIO PARADA SALGADO

IMPLEMENTACIÓN DE CODIFICACIÓN DE CANAL PARA SISTEMAS DE COMUNICACIONES DIGITALES

Los sistemas de comunicaciones digitales, y en particular, los moduladores y demoduladores han sido históricamente implementados mediante circuitos analógicos. El propósito de este trabajo es estudiar la factibilidad de realizar esta implementación en forma digital y resolver las posibles dificultades que se puedan presentar durante este proceso de migración.

El trabajo de memoria presenta los resultados de la implementación de un modulador y demodulador digital sobre plataformas FPGA (*Field Programmable Gate Array*) Spartan3E y Spartan3AN.

La investigación conecta los campos de telecomunicaciones y la electrónica. Mientras que el primero aporta la teoría de comunicaciones digitales, el segundo provee los fundamentos de la lógica digital a través de compuertas. Por ello, el reporte comienza por los fundamentos de modulación y demodulación digital, medidas de desempeño y el manejo de interferencia intersimbólica y continúa luego con una introducción al tratamiento digital de señales, la teoría y el diseño de filtros digitales, y la aritmética binaria empleada para efectuar ambas implementaciones.

Los resultados de este trabajo comprenden la programación, pruebas de funcionamiento y manejo de errores de un sistema de modulación PAM binario antipodal usando pulsos raíz cuadrada de coseno alzado como forma de onda principal. La primera etapa, programación, se realizó mediante lenguaje Verilog y esquemáticos a través del software ISE, de la empresa Xilinx. La segunda etapa, pruebas de desempeño, se concentró en la transmisión y recepción de secuencias binarias aleatorias de hasta 128 Kbits de longitud. El funcionamiento correcto del sistema da cuenta de una exitosa etapa de programación de los diversos bloques funcionales mientras que la ausencia de errores, en las pruebas realizadas, da cuenta de la minimización del efecto producido por la interferencia intersimbólica.

Algunos desafíos derivados de este trabajo corresponden a la optimización de recursos lógicos ocupados por los diferentes bloques del sistema, optimización del rendimiento de los bloques y, por ende, a la mejora de la tasa de transferencia de símbolos, la implementación de tipos de modulación de mayor complejidad y a la prueba de este sistema en ambientes de comunicación inalámbrica.

... para mi familia.

Agradecimientos

En primer lugar agradezco infinitamente todo el apoyo brindado por mi madre Nilda, mi padre Fernando y mis hermanos Felipe y Jaime. Siempre estuvieron conmigo (y lo estarán) en las situaciones buenas y en las malas que han ocurrido durante todo el proceso de obtención de este título y más aún, durante toda mi vida.

En segundo lugar agradezco todo el apoyo brindado por el profesor Patricio Parada, el cual siempre estuvo dispuesto a escuchar y resolver los diferentes problemas que fueron apareciendo en la elaboración de este trabajo. Un excelente profesor y una mejor persona. También agradezco el apoyo de los profesores Pablo Navarrete y Jorge Silva.

Finalmente quiero agradecer a mi gran amigo Roberto por estar siempre apoyándome, a mis amigos de la Universidad: Patricio, Marcela, Fabiola, a todos los Rodrigos, Sebastián, Andrés, Alonso, Alejandro, Cristina y muchos otros, los cuales hicieron muy agradable mi estadía durante estos años y dejaron gratos recuerdos que espero jamás olvidar.

A todos, muchas gracias.

Índice general

Índice de figuras

Índice de tablas

Lista de acrónimos

ADC Analog-to-Digital Converter

ASIC Application Specific Integrated Circuit

AWGN Additive White Gaussian Noise

BER Bit Error Rate

BFSK Binary Frequency Shift Keying

BPSK Binary Phase Shift Keying

BSP Bit Serial-Parallel

BUFGMUX Multiplexed Global Clock Buffer

CLA Carry Look-ahead Adder

CLB Configurable Logic Block

CORDIC Coordinate Rotation Digital Computer

CSA Carry Save Adder

CSD Canonic Signed Digit

DAC Digital-to-Analog Converter

DCM Digital Clock Manager

DDS Direct Digital Synthesizer

DSP Digital Signal Processing

FFT Fast Fourier Transform

FIR Finite Impulse Response

FM Frequency Modulation

FPGA Field Programable Gate Array

FSK Frequency Shift Keying

HDL Hardware Description Language

IDFT Inverse Discrete Fourier Transform

IIR Infinite Impulse Response

IOB Input/Output Block

ISI Intersymbolic Interference

JTAG Join Test Action Group

LSB Least Significant Bit

LUT Look-Up Table

MAP Maximum a Posteriori

MFSK M-ary Frequency Shift Keying

MLE Maximum Likelihood Estimator

MPSK M-ary Phase Shift Keying

MSB Most Significant Bit

NRZ Non-Return-to-Zero

NRZI Non-Return-to-Zero Inverse

OOK On-Off Keying

PAM Pulse Amplitude Modulation

PLL Phased Looked Loop

QAM Quadrature Amplitude Modulation

QPSK Quadrature Phase Shift Keying

RAM Random Access Memory

RC Raised Cosine

ROC Region of Convergence

ROM Read Only Memory

SCA Serial Column Adder

SNR Signal-to-Noise Ratio

SPI Serial Peripheral Interface

SRRC Square-root Raised Cosine

USB Universal Serial Bus

Capítulo 1

Introducción

Los sistemas de telecomunicaciones son un elemento primordial de la vida moderna pues permiten un rápido intercambio de información desde y hacia cualquier parte del globo en cualquier instante de tiempo. A medida que se trabaja con mayores volúmenes de datos y los aparatos agregan servicios adicionales a la simple comunicación por voz entre dos partes, se hace necesaria la continua evolución y mejora en el rendimiento general de los sistemas de telecomunicaciones, que permita satisfacer la creciente demanda de ubicuidad de las personas y el constante deseo en el aumento de las tasas de transferencia de datos.

En sus inicios, los sistemas de telecomunicaciones fueron desarrollados e implementados usando la electrónica analógica disponible en la época, como por ejemplo, transistores de gran tamañao (a tubos). Al mejorar las técnicas en la producción de semiconductores, los tamaños de estos dispositivos fueron disminuyendo, pero un problema constante de esta implementación es que la capacidad de evolución para alcanzar las especificaciones y constantes alzas de rendimientos es baja. Los aparatos analógicos suelen ser grandes, de alto consumo de potencia y limitado rendimiento; por otra parte, los componentes digitales ofrecen bondades en estos aspectos, pues su tamaño es menor, consumen menor potencia y sus rendimientos aumentan [1], situación que está ampliamente demostrada, muy particularmente, por la producción de la industria de los procesadores en la última década. Lo anterior evidenció claras señales de una inminente migración desde lo analógico hacia lo digital con el fin de aprovechar estas ventajas, pero esto tuvo un costo, y es que se ha tenido, hasta hoy en día, que investigar y desarrollar formas de traducir el comportamiento de los dispositivos electrónicos analógicos a algoritmos implementados en sistemas digitales que ejecuten la misma función de manera satisfactoria.

Otro ámbito que tuvo que evolucionar, en forma conjunta con lo anterior, fue el uso óptimo del espectro de frecuencias disponibles para hacer transmisiones a la mayor velocidad posible y, ojalá, con un gasto de potencia acotado. Esto dio lugar a variada investigación en el tema de las formas de modular las señales para lograr un óptimo uso de recursos, energía y ancho de banda, maximizando el rendimiento. El desarrollo anterior estuvo ligado directamente al éxito de la implementación digital de los antiguos componentes analógicos.

En el presente trabajo se desarrollará e implementará un sistema de modulación y demodulación por amplitud de pulsos binario ocupando filtros de raíz cuadrada de coseno alzado para obtener, en conjunto, un sistema que se comporte con las características de un *pulso de Nyquist* sobre plataformas FPGA con el fin de obtener una visión práctica y real sobre todos los procesos involucrados en el desarrollo de un sistema de comunicación, es decir, concentrar todo el conocimiento y desarrollo teórico en un aparato "tangible". La principal motivación de este desarrollo, en el cual se desea minimizar la interferencia intersimbólica, es que estas mismas clases de técnicas son las que se utilizan para aprovechar, cada vez de forma más eficiente, los reducidos anchos de banda destinados para los sistemas de telecomunicaciones.

La aplicación del trabajo desarrollado en esta memoria va desde la implementación de una experiencia demostrativa de la teoría enseñada en los cursos de pregrado (una experiencia de laboratorio, por ejemplo) hasta la confección, de forma rústica y simple, de pequeños transmisores para propósitos generales y cuya demanda de rendimiento no sea elevada.

Los objetivos específicos planteados en este trabajo son los siguientes:

- Estudiar los diferentes sistemas de modulación y demodulación ocupados en la actualidad.
- Estudiar los *pulsos de Nyquist* como señales capaces de mitigar la interferencia intersimbólica en el proceso de comunicación efectuada en canales sin ruido.
- Describir un sistema básico de comunicaciones detallando la funcionalidad de cada uno de los bloques que intervienen en la comunicación.
- Implementar a través de un lenguaje de descripción de hardware (HDL) bloques de aritmética binaria para la posterior construcción del sistema completo.
- Obtener conocimiento sobre el uso de plataformas FPGA y sus diferentes periféricos integrados.

El sistema de modulación y demodulación, no incluirá ni esquemas de corrección de errores, ni ecualización de canal; la solución será enfocada teniendo como único objetivo el correcto funcionamiento del sistema en un canal con rudio aditivo de tipo blanco y Gaussiano.

El presente documento se organiza de la siguiente manera: el capítulo 2 reune la investigación efectuada sobre los diferentes sistemas de modulación, interferencia intersimbólica y *pulsos de Nyquist*, medidas de desempeño de los sistemas de modulación descritos, demodulación a través de filtros adaptados, una pequeña introducción sobre filtros digitales, una breve descripción de la arquitectura de las FPGA y finalmente una descripción de bloques básicos de aritmética binaria. El capítulo 3 presenta los pasos realizados para la implementación y desarrollo del sistema, detalla la solución creada y las pruebas de funcionamiento realizadas. El capítulo 4 plasma las discusiones generadas durante el proceso de implementación así como también incluye observaciones generales respecto al trabajo desde un punto de vista más amplio. Finalmente, el capítulo 5 aglutina las conclusiones obtenidas a partir del desarrollo de este trabajo.

Capítulo 2

Información Teórica

2.1. Codificación de canal

Un sistema de comunicaciones se compone de tres elementos: un transmisor, un receptor y un canal a través del cual se intercambian señales. Tanto el transmisor como el receptor pueden, a su vez, estar compuestos por bloques cuyo principal objetivo es preparar el mensaje para ser transmitido en el canal y recuperarlo en su forma original, respectivamente. En el caso de que se utilice una sistema de comunicación digital, las funciones del transmisión y recepción pueden ser divididas en dos módulos: un codificador/decodificador de fuente, que toma el mensaje en su forma original y lo transforma en una secuencia binaria, y un codificador/decodificador de canal, que toma esta secuencia binaria y la prepara para ser transmitida a través del canal (figura 2.1).

El codificador de canal realiza dos funciones: la codificación para control de errores y la modulación del mensaje codificado en una señal de tipo analógica. En otras palabras, el codificador de canal agrega redundancia para controlar el número de errores que el canal puede introducir al mensaje, mientras que el modulador asocia a cada símbolo una señal analógica que es la que finalmente se transmite; en el lado del receptor, el demodulador se encarga de recuperar en forma binaria los símbolos analógicos recibidos, mientras que el decodificador intenta recuperar los posibles errores introducidos por el canal.

2.2. Esquemas de modulación y demodulación

Los moduladores y demoduladores cumplen funciones complementarias. Mientras uno transforma una serie de datos en formato digital para enviarlos a través de un canal analógico, el otro se encarga de tomar esta señal a la salida del canal y convertirla en la secuencia de símbolos digitales. A pesar de que estas funciones resultan complementarias, las unidades funcionales de estos bloques difieren considerablemente en su arquitectura.

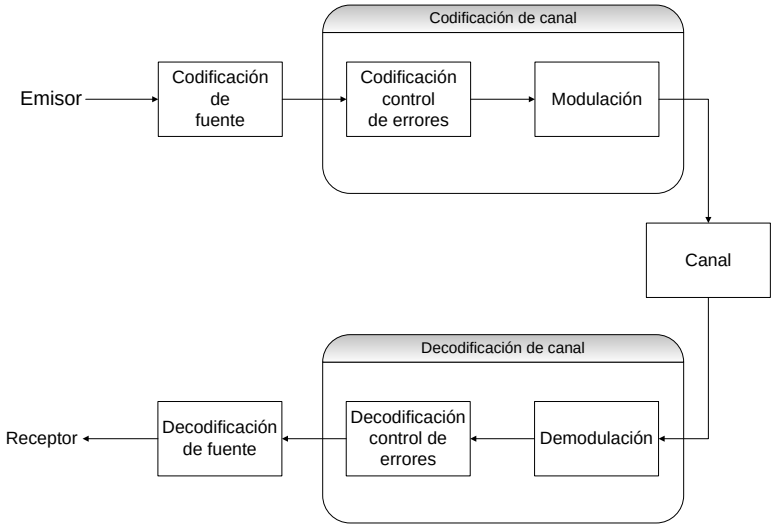


Figura 2.1: Diagrama de un sistema de telecomunicaciones básico.

El modulador transforma un símbolo o una secuencia de símbolos digitales, en una señal analógica que es la que se transmite en el canal. Las diferentes asignaciones para efectuar esta transformación dan origen a los diversos métodos de modulación utilizados en la actualidad.

La forma más general de modular es utilizando una familia de pulsos $s_m(t)$ donde la elección del índice m depende del mensaje original que se desea enviar. Este esquema permite enviar hasta 2^k mensajes distintos, donde cada mensaje consiste en una secuencia o palabra de k bits. La duración de un pulso se denomina *tiempo de símbolo* (T_s) y debe ser igual para todos los pulsos $s_m(t)$. La señal a la salida del modulador es igual a

$$c(t) = \sum_{l=-\infty}^{\infty} s_m(t - lT_s) \qquad m \in \left[1, \ldots, 2^k\right]. \tag{2.1}$$

El demodulador es un poco más complejo pues necesita la implementación de, a lo menos, dos bloques fundamentales: un bloque que filtre la señal de entrada y uno que decida si lo que se recibió constituye un determinado símbolo. Lo anterior asume que existe un mecanismo de sincronización con la señal proveniente del canal para la demodulación de símbolos dentro del demodulador. Adicionalmente se puede introducir un bloque que ecualice la señal para disminuir la interferencia intersimbólica introducida por la distorsión de un canal de comunicación con memoria [1].

[1] Un canal de comunicación con memoria es aquel cuya respuesta al impulso es distinta a una distribución Delta de Dirac, i.e., cuyo espectro de frecuencias $S(f) \neq 1$.

2.2.1. Fundamentos de comunicación digital

En esta sección se presentan los fundamentos de comunicación digital necesarios para entender de mejor forma el detalle de cada sistema de modulación.

Definición 2.1. Dos funciones complejas o reales, $f(t)$ y $g(t)$, ambas con soporte sobre un intervalo $[a, b]$, serán **ortogonales** si y sólo si

$$\int_a^b f(t) \cdot g^*(t)\, dt = 0. \tag{2.2}$$

donde * denota el valor conjugado de una función.

Definición 2.2. La **norma-2** o **norma Euclidiana** de una función en $L^2_{[a,b]}$ (Funciones integrables - Riemann o Lebesgue - con soporte en un intervalo $[a, b]$) se calcula de la siguiente forma

$$\| f(t) \|_2 = \sqrt{\int_a^b |f(t)|^2\, dt}. \tag{2.3}$$

Definición 2.3. Dos funciones complejas o reales, $f(t)$ y $g(t)$ serán **ortonormales** si además de ser ortogonales entre sí, la norma de cada una de ellas es igual a uno

$$\int_a^b f(t) \cdot g^*(t)\, dt = 0 \quad \wedge \quad \|f(t)\| = \|g(t)\| = 1. \tag{2.4}$$

Definición 2.4. La **distancia euclidiana** entre un par de funciones $f(t)$ y $g(t)$ se define como la norma de la diferencia entre ellas, es decir,

$$d(f, g) = \|f(t) - g(t)\|. \tag{2.5}$$

Definición 2.5. La **energía** de una función definida en un intervalo $[a, b]$ se define como la norma al cuadrado de la función. La energía se denota mediante el símbolo ε.

$$\varepsilon = \|f(t)\|^2 = \int_a^b |f(t)|^2\, dt. \tag{2.6}$$

Definición 2.6. La **correlación cruzada** entre dos funciones complejas con soporte en un intervalo $[a, b]$, denotada por el símbolo $\star$, se define como:

$$f \star g = \int_a^b f^*(\tau) \cdot g(t + \tau)\, d\tau. \tag{2.7}$$

donde * denota el valor conjugado de una función.

2.2.2. Modulación en banda base

Se denomina modulación en banda base o pasabajos a aquella que utiliza pulsos de modulación $s(t)$ cuyo espectro $S(f)$ se encuentra contenido en un intervalo de frecuencias cercanas a cero.

2.2.2.1. Modulación PAM

La *modulación por amplitud de pulsos* agrupa k bits de la secuencia de entrada, y a cada uno de aquellos grupos les asigna uno de los $m = 2^k$ posibles valores de amplitud de una función base $s(t)$.

$$s_m(t) = a_m \cdot s(t) \tag{2.8}$$

El valor de a_m dependerá del l-ésimo símbolo de la secuencia a modular.

Un ejemplo de asignación posible para PAM binario (k=1) es el siguiente:

$$a_m = \begin{cases} -A & \text{si el } l\text{-ésimo símbolo es 0} \\ A & \text{si el } l\text{-ésimo símbolo es 1.} \end{cases} \tag{2.9}$$

Si se escoje una función triangular (figura 2.2) definida por:

$$\Lambda(t) = \begin{cases} \frac{2t}{T_s} & \text{si } 0 < \text{t} < T_s/2 \\ \frac{-2t}{T_s} + 2 & \text{si } T_s/2 < \text{t} < T_s \\ 0 & \text{cualquier otro caso} \end{cases} \tag{2.10}$$

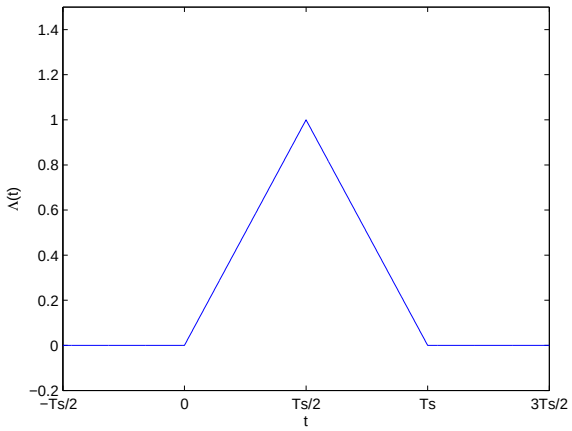


Figura 2.2: Pulso triangular $\Lambda(t)$.

como función $s(t)$, se obtiene la siguiente representación para este tipo de modulación (figura 2.3).

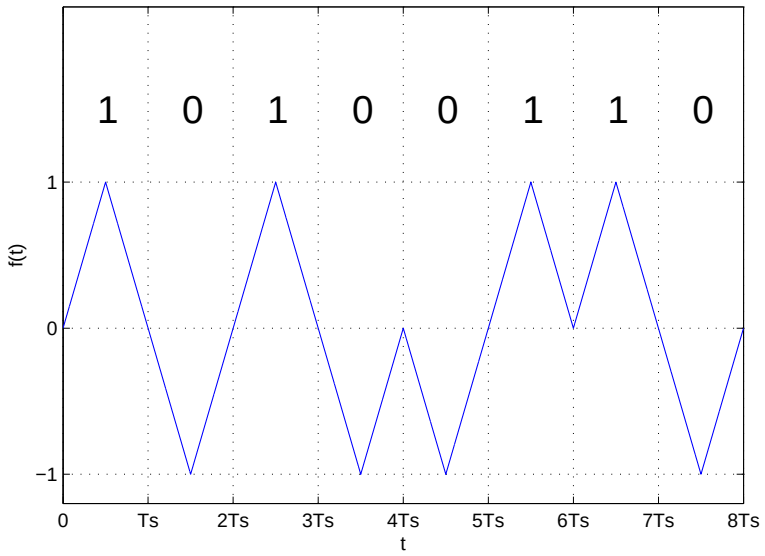


Figura 2.3: Modulación PAM binario usando un pulso triangular.

En este caso particular de PAM binario se utilizó un par de funciones que difieren sólo en el signo de la amplitud. En tal caso, el conjunto de señales $(s_m(t), m \in [0, 1])$ se denomina *antipodal*. Formalmente, las señales que poseen un índice de correlación cruzada igual a -1 son llamadas *antipodales*.

Modulación OOK Un caso particular de PAM y, a la vez, una forma más eficiente de modular en términos de energía, se obtiene usando la siguiente asignación:

$$a_m = \begin{cases} 0 & \text{si el } l\text{-ésimo símbolo es 0} \\ A & \text{si el } l\text{-ésimo símbolo es 1.} \end{cases} \tag{2.11}$$

El caso particular en que la modulación OOK utilice una señal cuadrada (figura 2.4), $s(t) = rect(\cdot)$, el esquema recibe el nombre de NZR (figura 2.5).

$$rect(t) = \begin{cases} 1 & \text{si } 0 < \text{t} < T_s \\ 0 & \text{cualquier otro caso.} \end{cases} \tag{2.12}$$

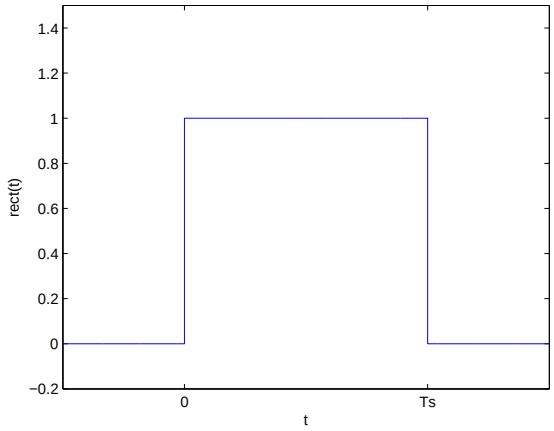


Figura 2.4: Pulso rectangular $rect(t)$.

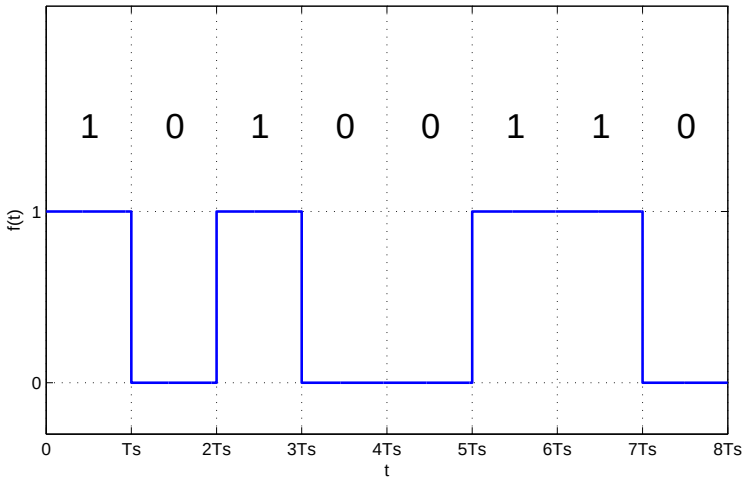


Figura 2.5: Modulación NRZ utilizando pulsos rectangulares.

Si se usa una señal cuadrada, pero en vez de modular un símbolo, se modula la transición de un símbolo a otro, el nuevo esquema recibe el nombre de NZRI (figura 2.6). Formalmente NZRI queda definido de la siguiente forma.

$$a_m = \begin{cases} A & \text{si el } l\text{-ésimo símbolo es diferente al } (l-1)\text{-ésimo símbolo} \\ 0 & \text{si el } l\text{-ésimo símbolo es igual al } (l-1)\text{-ésimo símbolo.} \end{cases} \tag{2.13}$$

En el caso anterior se asume que $a_{(l=-1)} = 0$.

Esta última forma de modulación recibe el nombre de *modulación diferencial*.

La ventaja de usar un tipo de modulación u otro (NRZ o NRZI) depende exclusivamente de la naturaleza de la secuencia particular a codificar. Dependiendo de las transiciones presentes en la secuencia, una forma de modular será más efectiva que otra considerando como principal meta la minimización de la energía utilizada para efectuar la modulación.

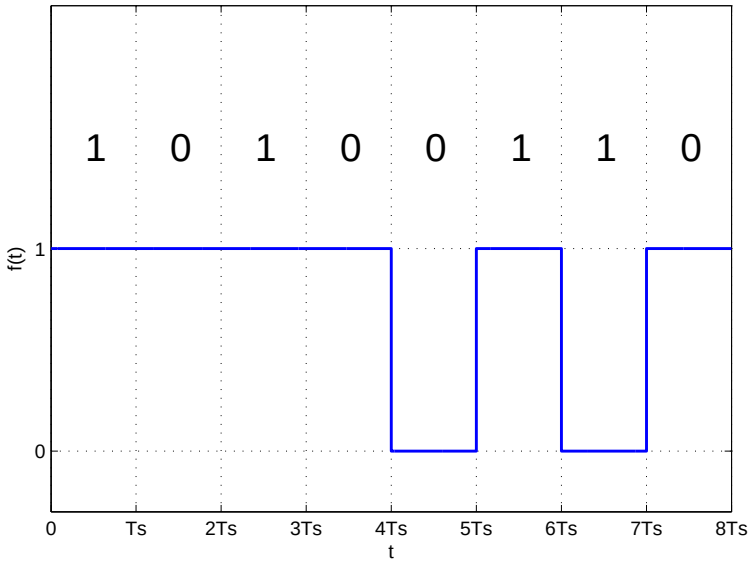


Figura 2.6: Modulación NRZI utilizando pulsos rectangulares.

2.2.2.2. Modulación PSK

Otra forma de modulación consiste en elegir un conjunto de funciones $s_m(t)$ que difieran en la fase respecto de una función base $s(t)$. El valor de la fase θ_m, dependerá del l-ésimo símbolo a modular. Lo anterior da lugar a la *modulación por desplazamiento de fase*. La expresión que representa de forma general al conjunto de funciones $s_m(t)$ es

$$s_m(t) = \mathrm{Re}\left[s(t) \cdot e^{j2\pi ft} \cdot e^{j\theta_m}\right]. \tag{2.14}$$

El caso en el cual los símbolos contienen solo un bit de información ($k = 1$) recibe el nombre de *codificación por desplazamiento de fase binario* (BPSK). Una posible asignación es la siguiente.

$$\theta_m = \begin{cases} 0 & \text{si el } l\text{-ésimo bit es 0} \\ \pi & \text{si el } l\text{-ésimo bit es 1.} \end{cases} \tag{2.15}$$

Si se utiliza una función $s(t)$ cuadrada, tal como la presentada en la ecuación (2.12) se obtiene la representación de la modulación BPSK (figura 2.7).

En el caso en que los símbolos agrupen más de un bit, la modulación toma el nombre de *codificación por desplazamiento de fase M-aria* (MPSK)

Cabe destacar que la asignación de fases puede hacerse de cualquier forma siempre y cuando $\theta_m \neq \theta_n$, $\forall\, m, n \in [1, \ldots, 2^k]$, $m \neq n$. Más adelante se observará que para lograr la menor tasa de errores en la demodulación, los valores se eligen de tal manera que la distancia entre todas las fases sea igual (fases equiespaceadas).

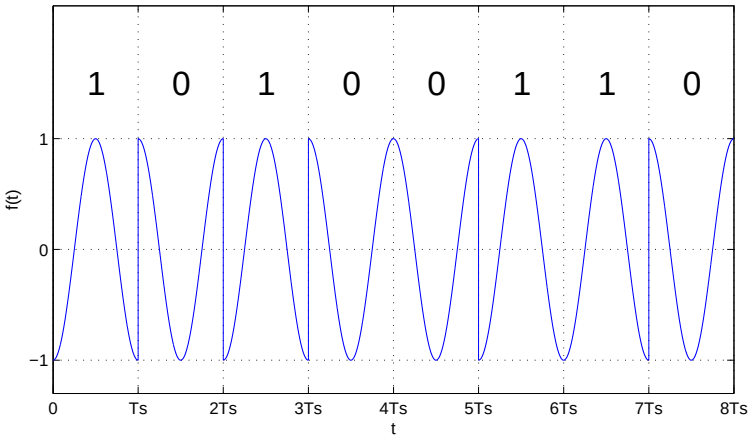


Figura 2.7: Modulación BPSK.

2.2.2.3. Modulación FSK

La *modulación por desplazamiento de frecuencia* se basa en la misma idea de la modulación PSK, sólo que en este caso, la familia de funciones $s_m(t)$ difieren en la frecuencia respecto de una función base $s(t)$ y además son ortogonales entre sí.

En este caso, la familia de funciones $s_m(t)$ tiene la forma

$$s_m(t) = \mathrm{Re}\left[s(t) \cdot e^{j2\pi f_m t} \cdot e^{j\theta}\right]. \tag{2.16}$$

Para el caso binario, denominado *codificación por desplazamiento de frecuencia binario* (BFSK) se puede realizar la siguiente asignación.

$$f_m = \begin{cases} f_0 & \text{si el } l\text{-ésimo símbolo es 0} \\ f_1 & \text{si el } l\text{-ésimo símbolo es 1} \end{cases}. \tag{2.17}$$

Utilizando una función $s(t)$ similar descrita en (2.12), $f_0 = nf$ y $f_1 = mf$, con $m > n$ y f una frecuencia cualquiera, se obtiene la representación temporal del esquema BFSK (figura 2.8).

El caso en el cual se ocupan más de dos frecuencias se denomina *codificación por desplazamiento de frecuencia M-ario* (MFSK).

La frecuencia f_m se expresa de la siguiente forma:

$$f_m = f + m\Delta f. \tag{2.18}$$

Luego, para que las funciones sean ortogonales entre sí, la mínima separación [2, pág. 177] corresponde a

$$\Delta f = \frac{1}{2T_s}. \tag{2.19}$$

donde T_s es el tiempo de símbolo.

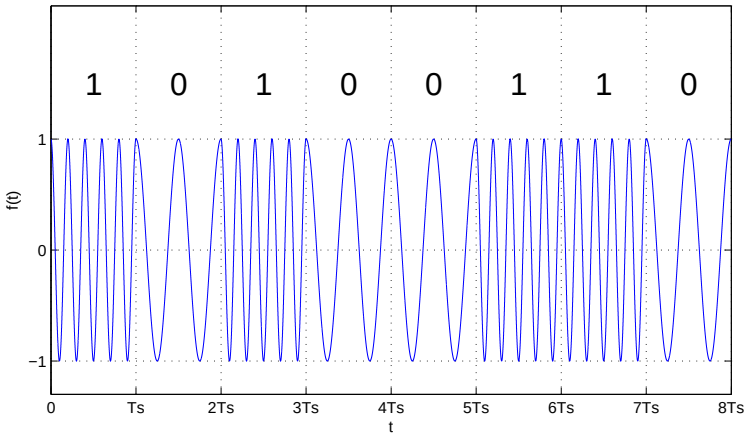


Figura 2.8: Modulación BFSK.

Esta asignación mínima asegura la ortogonalidad entre las funciones y además permite usar el mínimo ancho de banda posible.

2.2.2.4. Modulación QAM

La eficiencia de uso espectral y la velocidad de datos transmitidos se puede mejorar combinando las reglas de dos sistemas de modulación para formar uno nuevo, por ejemplo, si además de modular una secuencia de bits asignándole una fase determinada se le asigna también una amplitud dada. En este caso se obtiene el sistema de *modulación por amplitud en cuadratura* (QAM). La representación general de las formas de onda del sistema de modulación QAM viene dado por:

$$s_m(t) = \mathrm{Re}((A_{mc} + jA_{ms}) \cdot s(t) \cdot e^{j2\pi ft}), \tag{2.20}$$

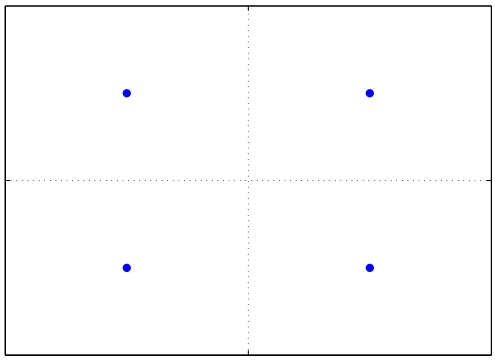
lo cual se puede reescribir como

$$s_m(t) = \mathrm{Re}((A_m e^{j\theta_m}) s(t) \cdot e^{j2\pi ft}), \tag{2.21}$$

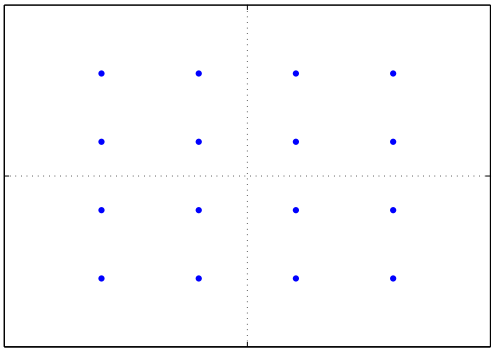
con $A_m = \sqrt{A_{mc}^2 + A_{ms}^2}$ y $\theta_m = \arctan\left(\dfrac{A_{ms}}{A_{mc}}\right)$.

El diagrama de todas las posibles asignaciones para los símbolos se denomina *constelación*[2]. La constelación no es única debido a que puede variarse la amplitud o la fase o ambas a la vez. Cada constelación posee ventajas y desventajas, por ejemplo, en el ámbito de la energía usada y de la facilidad para demodularlas. En la figura 2.9 se muestran las *constelaciones rectangulares* para 4 QAM y 16 QAM.

[2]El término constelación se puede expandir a los métodos de modulación vistos anteriormente. Esto se tratará en la sección de errores en la demodulación junto al concepto de *distancia*.



(a) 4 QAM.



(b) 16 QAM.

Figura 2.9: Constelaciones rectangulares para la modulación QAM.

2.2.3. Demodulación en banda base

El demodulador es el elemento que se encarga de recibir la señal analógica proveniente del canal y transformarla en una secuencia de símbolos usando un alfabeto predefinido de valores discretos. Dependiendo de la naturaleza del canal, la señal puede llegar con distintos niveles de distorsión, interferencia y ruido, lo cual dificulta la correcta demodulación y posterior detección de la secuencia. Se necesita entonces, diseñar sistemas o métodos que permitan recuperar la secuencia original de símbolos con la mayor integridad posible.

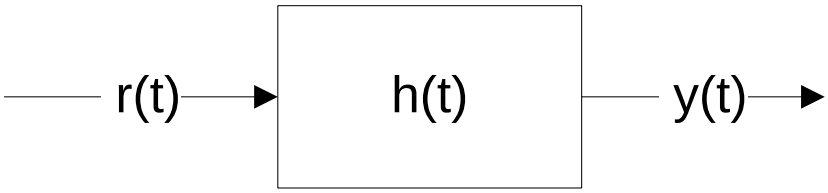


Figura 2.10: Filtrado de la señal $r(t)$ a través del filtro con respuesta al impulso $h(t)$.

2.2.3.1. Filtro adaptado

Consideraremos en primer lugar un canal perfecto, esto es que la señal que emite el modulador es exactamente igual a la que recibe el demodulador. Bajo este supuesto se desea encontrar la forma óptima de recuperar la secuencia original de bits modulada en el transmisor. Para este objetivo se necesita filtrar la señal de entrada, de tal forma que al muestrear la señal de salida del filtro se puedan obtener los valores de la secuencia enviada por el transmisor.

Si la señal de salida del modulador está dada por la ecuación (2.1), donde $s(t)$ es un pulso de energía igual a 1, y el filtro es lineal e invariante en el tiempo con respuesta al

impulso igual a $h(t)$, su salida viene dada por la expresión:

$$y(t) = \int_{-\infty}^{\infty} c(\tau) \cdot h(t - \tau)\, d\tau. \tag{2.22}$$

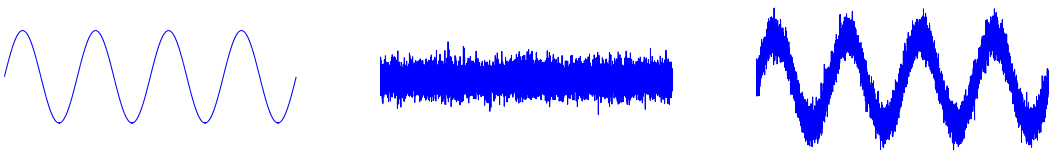
Se desea que al muestrear la señal $y(t)$ cada T_s segundos se pueda recuperar la secuencia original de datos de forma íntegra y sin errores. Para lograr esto, la elección de $h(t)$ se efectúa de forma que

$$h(t) = s(-t). \tag{2.23}$$

Este filtro se conoce como filtro adaptado (*matched filter*). La demostración de la igualdad (2.23) se incluye en el anexo (A.1).

Consideraremos ahora el caso de un canal de comunicaciones real. El efecto más común de encontrar un canal de comunicaciones es el ruido termal propio de la electrónica del receptor. Este ruido se puede modelar como una señal que se superpone en forma aditiva a la señal o mensaje y por ello recibe el nombre de canal aditivo. La señal recibida en el demodulador, $r(t)$ se puede modelar como la suma de dos señales: $c(t)$, que proviene del modulador y la señal $n(t)$ que corresponde al ruido introducido por el canal imperfecto (figura 2.11).

$$r(t) = c(t) + n(t). \tag{2.24}$$



(a) Señal limpia. (b) Ruido. (c) Señal con ruido.

Figura 2.11: Efecto aditivo del ruido sobre una señal en un canal de comunicaciones real.

La presencia inevitable del ruido exige que los demoduladores basados en la implementación de un filtro adaptado tengan la capacidad, idealmente, de filtrar la señal $c(t)$ para obtener la secuencia de símbolos originales y de rechazar la señal $n(t)$. Alcanzar por completo estos dos objetivos en canales imperfectos es imposible[3]. Lo mejor que se puede hacer en la práctica es diseñar una solución de compromiso, que maximice un indicador de desempeño que ligue ambas cantidades.

Para determinar la función $h(t)$ que define el comportamiento del filtro en este ambiente ruidoso, se suele poner el problema en el siguiente contexto. El ruido $n(t)$ es de tipo Gaussiano aditivo blanco (AWGN), su esperanza ($E[n(t)]$) es cero y su varianza (var$[n(t)]$,

[3]Esto pues el espectro del ruido se extiende en todo el eje de las frecuencias; luego, aunque se filtre la señal, una pequeña parte del ruido pasará con ella.

σ_n^2) corresponde al valor de su densidad espectral de potencia $N_0/2$ $[W/Hz]$. La medida de desempeño es la razón señal a ruido (SNR) la cual está definida, de forma general, como el cuociente entre la densidad espectral de potencia de una señal y la densidad espectral de potencia del ruido en un determinado intervalo de tiempo (ecuación 2.25).

$$\mathrm{SNR}_{t=t_0} = \frac{E[x^2(t=t_0)]}{E[n^2(t=t_0)]}. \tag{2.25}$$

El objetivo es determinar la función $h(t)$ que maximiza la razón señal a ruido de las señales de salida del filtro durante el intervalo en el cual $s(t)$ tenga soporte, es decir,

$$\underset{h\in\mathbb{F}}{máx}\ \frac{E[y_c^2(t=T)]}{E[y_n^2(t=T)]}, \tag{2.26}$$

donde $\mathbb{F}$ es el espacio de funciones L_2 (también conocidas como *"de energía finita"*) y las funciones y_s e y_n son las señales de salida del filtro correspondientes a la señal modulada y al ruido, respectivamente, es decir:

$$y_c = \int_0^T c(\tau)\cdot h(T-\tau)\,d\tau. \tag{2.27}$$

$$y_n = \int_0^T n(\tau)\cdot h(T-\tau)\,d\tau. \tag{2.28}$$

Este problema de optimización arroja, para la naturaleza del filtro $h(t)$ el mismo resultado que muestra la ecuación (2.23) y el máximo valor alcanzado del SNR es

$$SNR_{máx} = \frac{2\varepsilon_s}{N_0} \tag{2.29}$$

donde ε_s corresponde a la energía del pulso $s(t)$. La demostración de este resultado (2.29) se encuentra en [2, pp. 237-239].

El filtro adaptado es un filtro no causal, pero puede transformarse en uno causal en el caso de que exista un tiempo t_0 para el cual el filtro tiene valor cero. Luego, desplazando la respuesta al impulso en un tiempo t_0 se obtiene la causalidad deseada. Si por el contrario, no existe un tiempo para el cual el filtro se haga indefinidamente cero, se debe elegir un instante t_0 muy grande tal que el filtro adaptado tenga una forma cercana a la requerida.

En el caso de que el filtro $h(t)$ no corresponda al filtro adaptado ideal se produce un empeoramiento en la razón señal a ruido. Para ser más precisos, consideremos que el nuevo filtro $g(t)$ se escribe como

$$g(t) = h(t) + \varepsilon f(t), \tag{2.30}$$

donde ε es un número pequeño, $h(t)$ es el filtro adaptado ideal y $f(t)$ es una señal de energía finita.

Se puede demostrar que el nuevo SNR está definido por

$$\frac{S}{N} = \left(\frac{S}{N}\right)_{\varepsilon=0} - \varepsilon^2 A + ... \tag{2.31}$$

donde A es una constante positiva. La demostración de este resultado se incluye en el anexo (A.2).

Este resultado indica que pequeñas variaciones en la construcción del filtro adaptado afectarán el valor de la razón señal a ruido.

2.2.4. Interferencia intersimbólica

La interferencia intersimbólica (ISI) ocurre cuando las formas de onda que representan a los símbolos son enviados a través de un canal de comunicación con memoria, dando lugar a una interferencia mutua creando símbolos que originalmente no estaban en la secuencia enviada, o destruyendo los que originalmente fueron enviados. La interferencia intersimbólica puede ser producida por el comportamiento temporal del pulso usado para la modulación (lo cual se traduce en un problema de una correcta elección del pulso $s(t)$) o por la imperfección de los canales de comunicación usados en la realidad. Nyquist desarrolló criterios con los cuales se puede suprimir el efecto de esta interferencia en el proceso de comunicación.

2.2.4.1. Pulsos de Nyquist

Hasta ahora se ha hablado sobre el funcionamiento de los distintos métodos de modulación y el principio de la demodulación pero, tan importante como lo anterior, es definir la señal base o pulso de forma $s(t)$ que transportará los símbolos través del canal de propagación.

En los ejemplos de NRZ y NRZI, se menciona el uso de una señal cuadrada (2.12) como señal base $s(t)$. Esta elección tiene el beneficio de que la señal es acotada en el tiempo, pero que necesita un ancho de banda infinito (señal $S(f) = sinc(f)$).

Tomando el ejemplo anterior se podría usar, de forma inversa, la señal $sinc(t)$ en el dominio del tiempo (figura 2.12) teniendo un intervalo infinito en esta dimensión dado su lento decaimiento (proporcional a $1/t$) pero un ancho de banda limitado (señal $rect(f)$).

$$sinc(t) = \frac{\sin(\pi t)}{\pi t}. \tag{2.32}$$

Las formas de onda descritas anteriormente constituyen la representación de las cualidades extremas que pueden tener los denominados *pulsos de Nyquist*. Un *pulso de Nyquist* es una señal que, al usarla como filtro en un sistema de comunicaciones, permite la eliminación teórica de la interferencia intersimbólica siempre que el canal de comunicación sea perfecto. Matemáticamente, ello se define como

$$s(lT) = \begin{cases} 1 & l = 0 \\ 0 & l \neq 0 \end{cases}, \tag{2.33}$$

dado un período de muestreo T y $l \in \mathbb{Z}$.

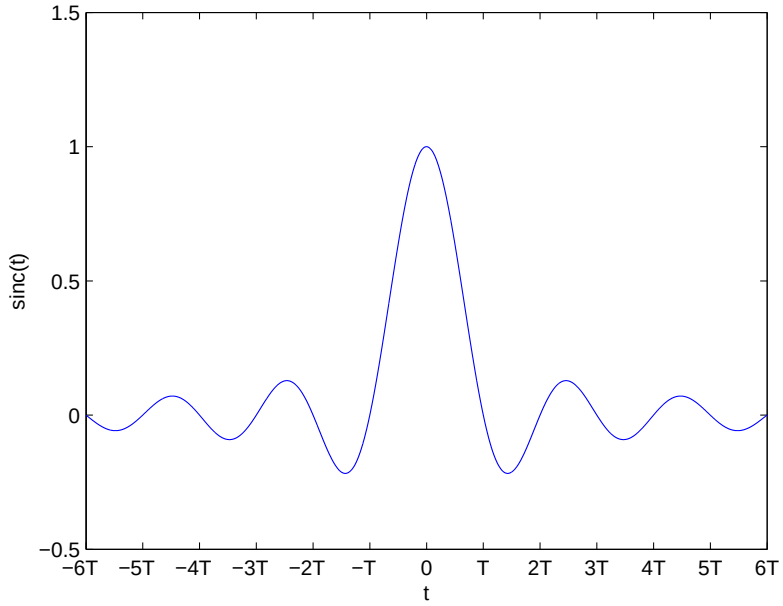


Figura 2.12: Señal $sinc(t)$.

Otra forma de definir, ahora en el dominio de la frecuencia, un *pulso de Nyquist* es mediante el siguiente resultado.

Teorema 2.1. *Un pulso $s(t)$ será de Nyquist si y sólo si su transformada de Fourier, $S(f)$, satisface*

$$\frac{1}{T}\sum_{-\infty}^{\infty} S\left(f+\frac{n}{T}\right)=1 \qquad |f| \leq \frac{1}{2T}. \tag{2.34}$$

Demostración. La demostración de este teorema se encuentra en [2, pp. 557-558]. □

La representación gráfica del teorema se muestra en la figura 2.13.

Se desea encontrar funciones que posean un equilibrio entre la ocupación de recursos en el tiempo, en la frecuencia y que además sean *pulsos de Nyquist*. La familia de funciones denominada de coseno alzado (*raised cosine*) cumple con estas tres exigencias.

Definición 2.7. La ecuación paramétrica de la familia de funciones coseno alzado es la siguiente:

$$ca(t) = \operatorname{sinc}\left(\frac{\pi t}{T}\right) \cdot \frac{\cos\left(\frac{\alpha\pi t}{T}\right)}{1-\frac{4\alpha^2 t^2}{T^2}} \qquad \alpha \in [0,1]. \tag{2.35}$$

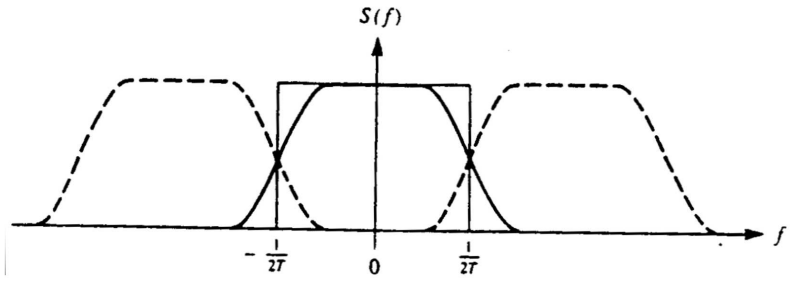
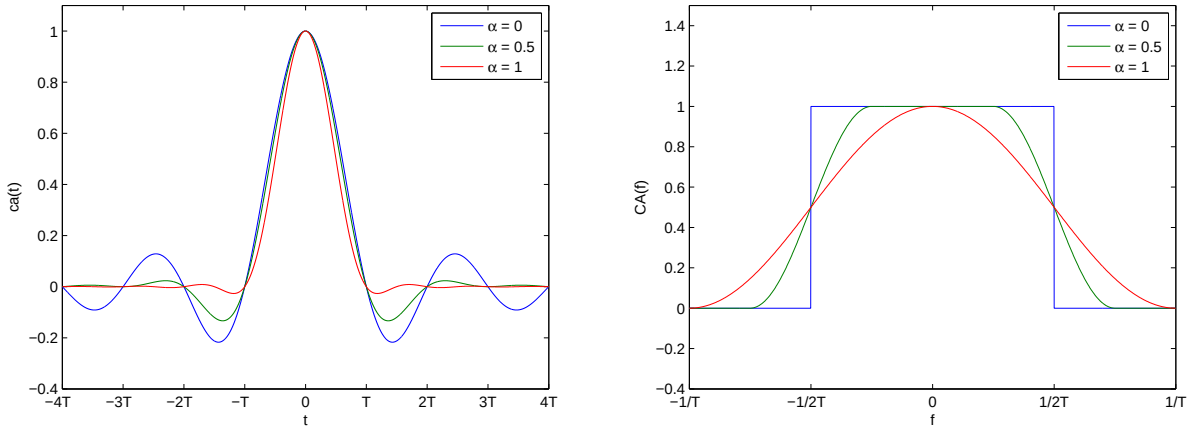


Figura 2.13: Representación gráfica del teorema (2.1).

y su espectro de frecuencias está dado por la siguiente expresión

$$CA(f) = \begin{cases} T & 0 \leq |f| \leq \frac{1-\alpha}{2T} \\ \frac{T}{2}\left(1 + cos\left[\frac{\pi T}{\alpha}\left(|f| - \frac{1-\alpha}{2T}\right)\right]\right) & \frac{1-\alpha}{2T} \leq |f| \leq \frac{1+\alpha}{2T} \\ 0 & |f| \geq \frac{1+\alpha}{2T} \end{cases} \tag{2.36}$$

Para diversos valores de α, tanto su distribución temporal como su espectro en el dominio de las frecuencias, se muestran en la figura (2.14).



(a) Señal temporal.

(b) Señal espectral.

Figura 2.14: Función coseno alzado para diferentes valores de α.

Se puede apreciar que el decaimiento en el espacio del tiempo es más rápido y pronunciado que el de la señal $sinc(t)$, por lo que un truncamiento "temprano" de esta señal es menos perjudicial comparado con esta última. El decaimiento temporal de los pulsos coseno alzado es del orden de $1/t^3$.

2.2.4.2. Diseño de sistemas de comunicaciones libres de interferencia intersimbólica

En la práctica se desea que la respuesta en frecuencia del sistema de comunicación completo (transmisor, canal y receptor) tenga la característica de cumplir las condiciones de Nyquist para así mitigar la interferencia intersimbólica. Matemáticamente, se desea algo como

$$CA(f) = T(f) \cdot C(f) \cdot R(f). \tag{2.37}$$

donde $CA(f)$ corresponde a la respuesta en frecuencia de un pulso coseno alzado, $T(f)$ la respuesta en frecuencia del filtro transmisor, $C(f)$ la respuesta en frecuencia del canal y $R(f)$ la respuesta en frecuencia del filtro receptor. Asumiendo un canal de transmisión ideal, es decir $C(f) = 1$, y que el filtro receptor corresponde a un filtro adaptado (2.23), se obtiene que

$$T(f) = R(f) = \sqrt{CA(f)}. \tag{2.38}$$

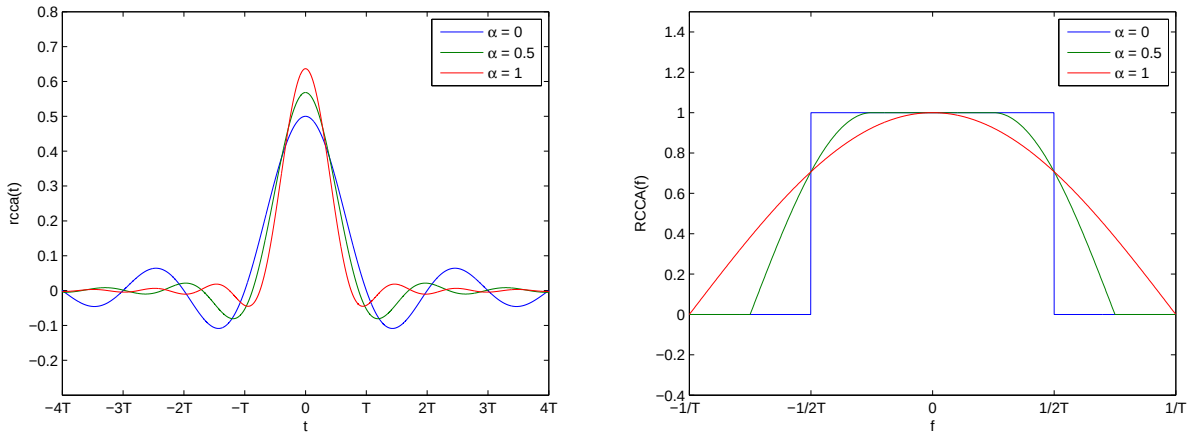
La formulación anterior da lugar a un nuevo tipo de señal denominada raíz cuadrada de coseno alzado (SRRC) cuyo espectro de frecuencias esta dado por

$$RCCA(f) = \sqrt{|CA(f)|}. \tag{2.39}$$

mientras que su distribución temporal está dado por la ecuacion (2.40)

$$rcca(t) = \frac{4\alpha}{\pi\sqrt{T}} \frac{cos\left(\frac{(1+\alpha)\pi t}{T}\right) + \frac{T}{4\alpha t} sin\left(\frac{(1-\alpha)\pi t}{T}\right)}{1 - \left(\frac{4\alpha t}{T}\right)^2}. \tag{2.40}$$

En la figura (2.15) se puede apreciar tanto su distribución temporal como su espectro en el dominio de las frecuencias.



(a) Señal temporal.

(b) Señal espectral.

Figura 2.15: Función raíz cuadrada coseno alzado para diferentes valores de α.

Cabe mencionar que esta nueva señal, a diferencia del pulso coseno alzado, en los instantes de tiempo nT ($n \in \mathbb{R}$, $n \neq 0$) la señal no cruza por cero salvo que el factor α sea igual a cero.

2.2.5. Medidas de desempeño

En la sección anterior se nombró la razón señal a ruido (SNR) como una medida de desempeño a maximizar para permitir una correcta demodulación de la señal proveniente del canal de comunicación entre el transmisor y el receptor. Aquella medida se basa en el cálculo de las potencias de las señales. Un indicador distinto de desempeño se puede obtener considerando los errores cometidos en la demodulación de la secuencia binaria.

La secuencia demodulada puede contener errores del tipo "inversión de un bit", esto es cuando dentro de toda la cadena binaria existe detectado un "1" cuando en realidad se transmitió un "0" y viceversa. La medida de desempeño *tasa de error por bit* (BER) se basa en la probabilidad de que estos eventos ocurran.

Para determinar estas probabilidades se considerarán las mismas condiciones del canal asumidas en la obtención del filtro adaptado, es decir, un canal con ruido blanco aditivo Gaussiano y ausencia de interferencia intersimbólica.

Definición 2.8. La **función Q** indica la probabilidad de que una variable aleatoria X sea mayor que un valor a bajo una distribución Gaussiana normal.

$$Pr\{X > a\} = Q(a) = \frac{1}{\sqrt{2\pi}} \int_a^{\infty} \exp\left(\frac{-x^2}{2}\right) dx. \tag{2.41}$$

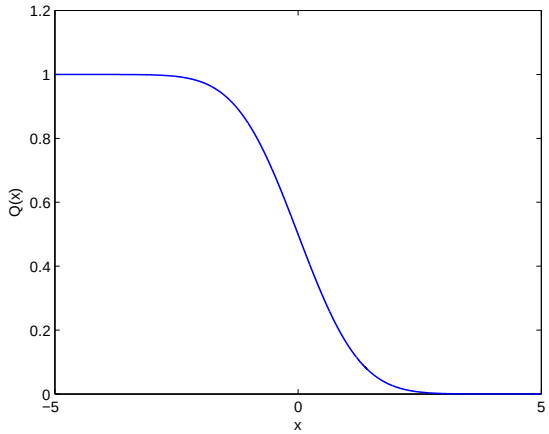


Figura 2.16: Función Q.

La función $Q(\cdot)$ es utilizada frecuentemente en el contexto de las comunicaciones y se relaciona con la función $erf(x)$,

$$erf(x) = \frac{2}{\sqrt{\pi}} \int_0^{x} e^{-z^2} dz, \tag{2.42}$$

de la siguiente forma:

$$Q(x) = \frac{1}{2}\left(1 - erf\left(\frac{x}{\sqrt{2}}\right)\right). \tag{2.43}$$

2.2.5.1. Modulación binaria

En el caso de la codificación binaria, el filtro adaptado entrega como salida la correlación entre la señal recibida y la señal transmitida (ver ecuaciones (2.24), (2.27) y (2.28)). En base a este dato estadístico y a una cantidad conocida como umbral, se toma la decisión de cuando una señal transmitida corresponde a un "0" ó "1". El valor del umbral se elige de tal manera que las probabilidades de ocurrencia de los dos errores posibles, detección errónea de un bit, sean iguales. Lo anterior es válido sólo en el caso de que el ruido tenga una distribución Gaussiana y que los símbolos sean equiprobables, de no cumplirse estas condiciones, se suele utilizar otros criterios de estimación tales como máxima probabilidad a posteriori (MAP) o máxima verosimilitud (MLE)

Para desarrollar los resultados siguientes se debe tener en cuenta que al introducir ruido blanco Gaussiano a un filtro lineal e invariante en el tiempo, la salida de éste será también ruido Gaussiano y, por ende, cualquier otra señal de salida del filtro será una variable aleatoria con distribución Gaussiana, es decir, con una función densidad de probabilidad

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right), \tag{2.44}$$

cuya esperanza está dada por

$$E[x] = \mu \tag{2.45}$$

y su varianza por

$$Var[x] = \sigma^2. \tag{2.46}$$

Considerando el caso PAM binario mostrado en (2.9) se tiene que las distribuciones de los dos tipos de señales están dados por la ecuación (2.47)

$$\begin{aligned} p_{simb=0}(x) &= \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x+A)^2}{2\sigma^2}\right) \\ p_{simb=1}(x) &= \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-A)^2}{2\sigma^2}\right) \end{aligned} \tag{2.47}$$

A partir de estas distribuciones aleatorias se puede obtener la distribución de probabilidad de ambos eventos (figura 2.17).

Para definir la probabilidad de error en la detección de un símbolo se debe fijar un umbral, en este caso se puede apreciar que para igualar las probabilidades de error en los dos eventos posibles el umbral debe ser cero. Con ello las probabilidades de error se pueden escribir de la siguiente manera:

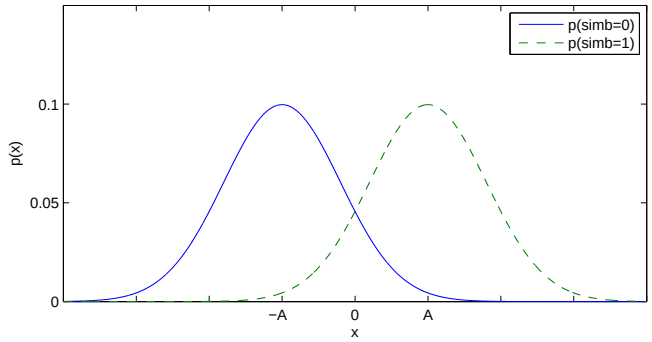


Figura 2.17: Distribución de probabilidad para PAM binario antipodal.

$$\begin{aligned} p_{e|simb=0}(x) &= \int_{0}^{\infty} \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x+A)^2}{2\sigma^2}\right) dx \\ p_{e|simb=1}(x) &= \int_{-\infty}^{0} \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-A)^2}{2\sigma^2}\right) dx \end{aligned} \tag{2.48}$$

Asumiendo que ambos errores tienen la misma probabilidad de ocurrir, la probabilidad total de error de PAM binario está dada por la siguiente expresión.

$$p_e = (1/2)p_{e|simb=0} + (1/2)p_{e|simb=1}, \tag{2.49}$$

efectuando un cambio de variable,

$$y = \frac{x + E[x]}{\sqrt{Var[x]}}, \tag{2.50}$$

y considerando que el cálculo de ambas integrales es el mismo, la probabilidad de error usando la función $Q(\cdot)$ queda expresada como

$$p_e = Q\left(\frac{A}{\sigma}\right). \tag{2.51}$$

La razón A/σ se puede escribir en función de la energía por bit (E_b) y la densidad espectral ($N_0/2$) a partir de la relación mostrada en [3, pág. 58] y los parámetros estadísticos del ruido blanco Gaussiano. Finalmente, la expresión para la probabilidad de error se escribe como

$$p_e = Q\left(\sqrt{\frac{2E_b}{N_0}}\right). \tag{2.52}$$

Para el caso de modulación OOK se tiene que una de las funciones de densidad de probabilidad tiene media cero, por lo que la nueva figura de las distribuciones se muestra en 2.18.

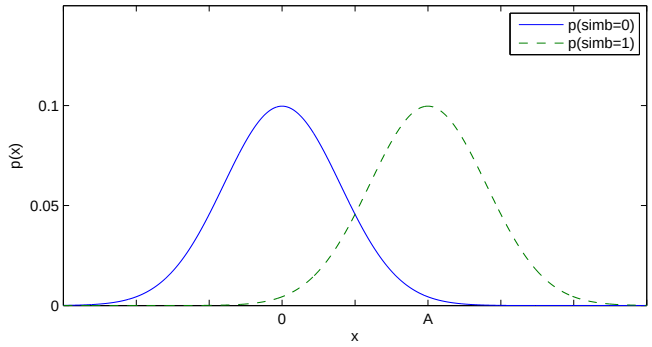


Figura 2.18: Distribución de probabilidad para OOK.

Siguiendo el mismo procedimiento que el caso PAM binario antipodal se llega a la ecuación (2.51) pero en este caso, para reemplazar A se ocupa la relación mostrada en [3, pág. 59]. Asumiendo igual probabilidad de aparición de símbolos, la probabilidad de error del caso OOK está dado por

$$p_e = Q\left(\sqrt{\frac{E_b}{N_0}}\right). \tag{2.53}$$

Se puede notar que a medida que la energía por bit del sistema disminuye la probabilidad de error aumenta. Otra forma de ver este aumento es observando que la distancia entre las dos señales disminuye. Esta noción de distancia permite caracterizar cada método de modulación mediante una constelación.

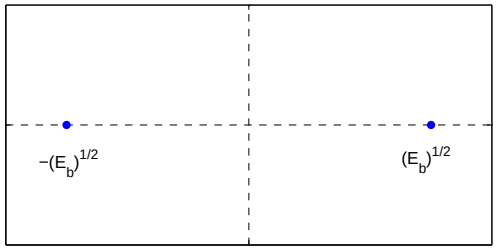


Figura 2.19: Distancia entre dos señales en modulación PAM binaria antipodal.

Con lo anterior, la probabilidad de error se puede expresar en función de la distancia entre dos señales. Luego, este valor para cualquier señalización binaria está dada por

$$p_e = Q\left(\sqrt{\frac{d^2}{2N_0}}\right) \tag{2.54}$$

donde d es la distancia mínima en la constelación.

En el caso de ocupar dos señales ortogonales (BFSK), la tasa de error por bit se puede calcular usando la ecuación (2.54). La distancia entre las dos señales ortogonales se puede ver en la figura 2.20.

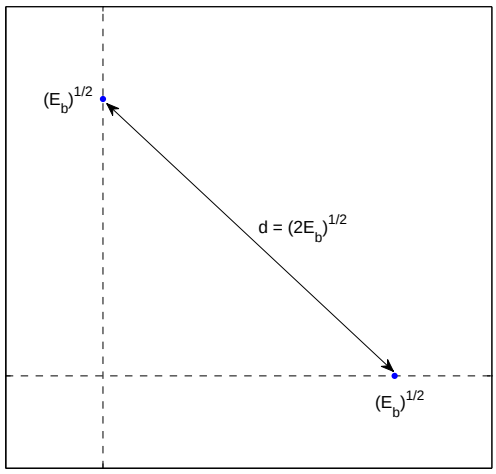


Figura 2.20: Distancia de señales ortogonales utilizadas para modulación binaria.

Luego, reemplazando este nuevo valor, se obtiene que la probabilidad de error del caso binario con señales ortogonales está dado por

$$p_e = Q\left(\sqrt{\frac{E_b}{N_0}}\right). \tag{2.55}$$

El caso de BPSK es similar al de PAM binario, por lo que la probabilidad de error está directamente expresada en la ecuación (2.52).

La comparación de las probabilidades de error de los distintos tipos de modulación nombrados, en función de la razón E_b/N_0, se muestra en la figura (2.21).

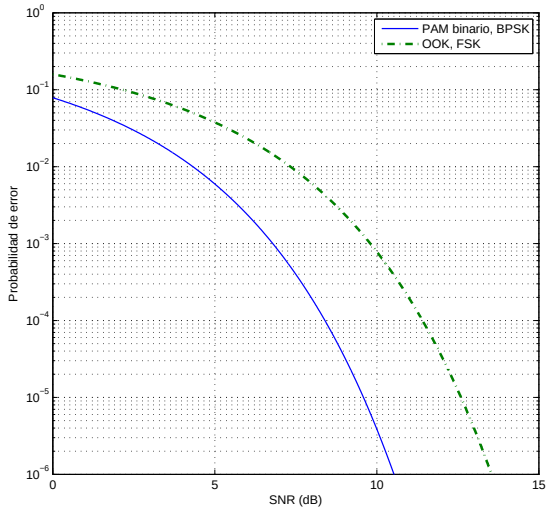


Figura 2.21: Probabilidad de error versus SNR para diferentes tipos de modulación.

2.2.5.2. Modulación M-aria

Calcular la probabilidad de error en un tipo de modulación que ocupa más de dos valores en la variable correspondiente, significa un cálculo que escapa del tema de esta memoria, por lo que sólo se presentarán resultados. Los desarrollos completos se pueden obtener en [2, cap. 5], [4, cap. 5].

MFSK El resultado mostrado para este tipo de modulación es el más general pues sólo requiere como condición que las señales usadas para la modulación sean ortogonales, por lo que, en particular, es válido en el caso de FSK M-ario.

$$p_e = \left(\frac{2^{(\log_2 M)-1}}{2^{\log_2 M}-1}\right) \cdot \frac{1}{2\pi} \int_{-\infty}^{\infty} \left[1 - \left(\frac{1}{2\pi} \int_{-\infty}^{y} \exp\left(-\frac{x^2}{2}\right) dx\right)^{M-1}\right] \exp\left[-\frac{1}{2}\left(y - \sqrt{\frac{2\varepsilon_s}{N_0}}\right)^2\right] dy \tag{2.56}$$

donde M es el número de frecuencias usadas y ε_s es la energía por símbolo.

PAM multinivel El caso en que PAM utilice más de dos valores para la amplitud en la modulación y además estén equiespaceadas, la probabilidad de error es:

$$p_e = \frac{2(M-1)}{M} Q\left(\sqrt{\frac{(6\log_2 M)\overline{\varepsilon}_b}{(M^2-1)N_0}}\right) \tag{2.57}$$

donde $\overline{\varepsilon}_b$ es la energía promedio por bit.

MPSK Para el caso en que se utilice una modulación por fase con más de dos valores y que además estén equiespaceados, la probabilidad de error se aproxima mediante la siguiente expresión:

$$p_e \approx \frac{2}{\log_2 M} Q\left(\sqrt{\frac{2\log_2 M \varepsilon_b}{N_0}} \sin\left(\frac{\pi}{M}\right)\right). \tag{2.58}$$

QAM La probabilidad de error por bit de la modulación QAM va a depender de la forma de la constelación que se ocupe. Por un compromiso entre la facilidad para demodular y una utilización media de potencia se ocupa QAM con constelaciones rectangulares como las presentadas en la figura (2.9). La probabilidad de error se aproxima como

$$p_e \approx 4\left(1 - \frac{1}{\sqrt{M}}\right) Q\left(\sqrt{\frac{3\overline{\varepsilon}_b}{(M-1)N_0}}\right). \tag{2.59}$$

donde M corresponde al número de símbolos modulados.

2.3. Filtros digitales

Al trabajar con secuencias finitas de bits se fuerza a que todos los elementos que interfieren en el proceso de modulación y demodulación sean digitales, lo que conlleva a un tratamiento discreto de las señales. Uno de los elementos trascendentales es el filtro adaptado, que en este trabajo lo implementaremos en forma digital.

Existen dos tipos de filtros discretos: los filtros de respuesta finita al impulso (FIR) y los de respuesta infinita al impulso (IIR). La diferencia entre ambos radica en la duración de su respuesta al impulso. El primero sólo depende de un número finito de entradas (filtro no recursivo) por lo que, pasado ese determinado número, su respuesta será nula. El segundo, además, depende de un número finito de salidas previas del filtro (filtro recursivo) por lo que está en constante retroalimentación, provocando que el filtro responda infinitamente.

La ecuación que relaciona la secuencia de entrada $x[n]$ y la de salida $y[n]$ para un filtro FIR es:

$$y[n] = \sum_{k=0}^{N} h[k]x[n-k]. \tag{2.60}$$

donde $h[k]$ corresponde al coeficiente k-ésimo del filtro y N es el orden o memoria del filtro.

La ecuación que caracteriza el comportamiento de un filtro IIR es:

$$y[n] = \frac{1}{g[0]} \left(\sum_{m=0}^{M} h[m]x[n-m] - \sum_{l=1}^{L} g[l]y[n-l] \right) \tag{2.61}$$

donde $h[m]$ corresponde al coeficiente m-ésimo del filtro causal y N a su orden, $g[l]$ corresponde al coeficiente l-ésimo del filtro de retroalimentación y L a su orden.

Aplicando transformada Z a las expresiones anteriores y asumiendo que los coeficientes son constantes se obtiene, respectivamente,

$$Y(z) = \sum_{k=0}^{N} h[k]z^{-n} \qquad , z \in ROC. \tag{2.62}$$

$$Y(z) = \frac{\sum_{k=0}^{N} h[k]z^{-k}}{\sum_{l=0}^{L} g[l]z^{-l}} \qquad , z \in ROC. \tag{2.63}$$

Se puede apreciar que los filtros de tipo FIR serán estables siempre pues sólo poseen ceros. El caso del tipo IIR es un poco más complicado pues intervienen un determinado número de polos, lo cual puede introducir inestabilidad al sistema dependiendo de su ubicación en el plano complejo. Esta característica, sumado al hecho de que los filtros FIR son más

fáciles de implementar, son relativamente insensibles al ruido de cuantización y, bajo ciertas condiciones, poseen fase lineal [5], harán que sean el objeto de estudio en lo que resta de la sección.

La respuesta en frecuencia de un filtro FIR se obtiene calculando su transformada de Fourier de tiempo discreto o lo que es equivalente, reemplazando $z = e^{jw}$ en la ecuación (2.62)

$$Y(\omega) = \sum_{k=0}^{N} h[k] e^{-j\omega k}. \tag{2.64}$$

Expandiendo la exponencial compleja en senos y cosenos a partir de la fórmula de Euler, se llega a la siguiente expresión:

$$Y(\omega) = \sum_{k=0}^{N} h[k] \cos(\omega k) - j \sum_{k=0}^{N} h[k] \sin(\omega k). \tag{2.65}$$

Lo anterior se puede escribir en la forma polar:

$$Y(\omega) = M(w) e^{j\theta(\omega)} \tag{2.66}$$

con

$$M(\omega) = \sqrt{\left(\sum_{k=0}^{N} h[k] \cos(\omega k) \right)^2 + \left(\sum_{k=0}^{N} h[k] \cos(\omega k) \right)^2}. \tag{2.67}$$

y

$$\theta(w) = \tan^{-1} \left(\frac{-\sum_{k=0}^{N} h[k] \sin(wk)}{\sum_{k=0}^{N} h[k] \cos(wk)} \right) \tag{2.68}$$

Se define el retraso de fase (*phase delay*) como

$$\tau_p = -\frac{\theta(w)}{w}. \tag{2.69}$$

Luego, para que el filtro tenga fase lineal se necesita que $\theta(w) = -\alpha w$ con $\alpha > 0$. Imponiendo esto en la ecuación (2.69) se obtiene

$$\tan(-\alpha w) = \frac{-\sum_{k=0}^{N} h[k] \sin(wk)}{\sum_{k=0}^{N} h[k] \cos(wk)} \tag{2.70}$$

$$\sum_{k=0}^{N} h[k]\left(\cos(wk)\sin(w\alpha) - \sin(wk)\cos(w\alpha)\right) = 0 \tag{2.71}$$

$$\sum_{k=0}^{N} h[k]\sin(w\alpha - wn) = 0 \tag{2.72}$$

La solución [6] para la ecuación anterior es

$$\alpha = \frac{N}{2}. \tag{2.73}$$

$$h[k] = h[N-k]. \tag{2.74}$$

Esto último implica que para la obtención de una respuesta en frecuencia del filtro con fase lineal se debe imponer que los coeficientes sean simétricos.

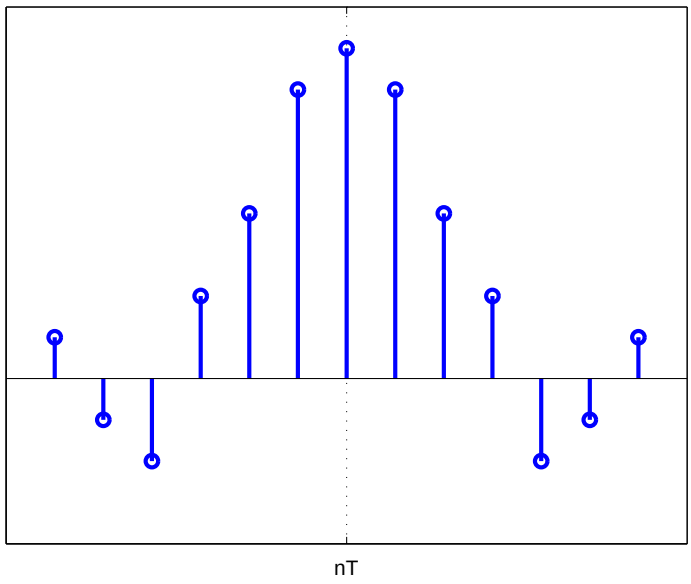


Figura 2.22: Coeficientes simétricos del filtro (N impar).

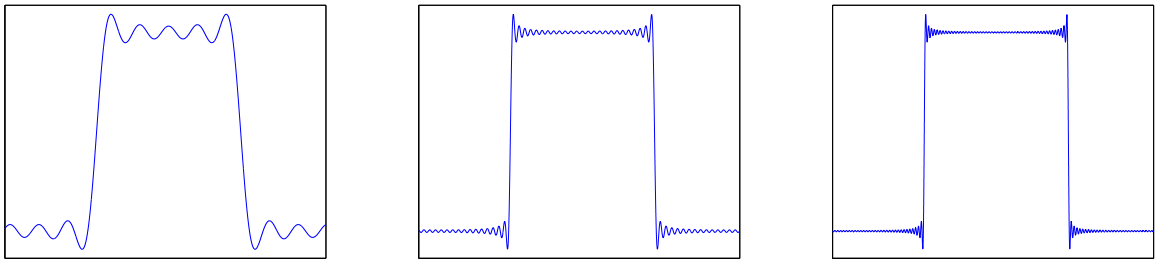
2.3.1. Diseño del filtro

En esta sección se tratarán, de forma introductoria, los diferentes métodos que existen para obtener el valor de los coeficientes del filtro que se quiera implementar de acuerdo a las necesidades y a los requerimientos del problema.

El primer método y a la vez el más simple, consiste en aplicar la transformada discreta de Fourier inversa (IDFT). Esto es, teniendo la representación en frecuencia del filtro objetivo, se obtiene la representación temporal de la respuesta al impulso y con ello, el valor de los

coeficientes. Cabe destacar que esta simple técnica tiene algunos inconvenientes. Uno de ellos consiste en que la señal obtenida es no causal y además, infinita en el dominio del tiempo. La solución para los problemas anteriores es, truncar la señal temporal, es decir, reducir el número de coeficientes anulándolos a partir de cierto instante de tiempo y, para la no causalidad agregar un determinado retraso luego de haber truncado la señal.

Este método también presenta otro problema conocido como el *fenómeno de Gibbs* (figura 2.23). Este fenómeno consiste en la aparición de oscilaciones en los puntos de discontinuidad del filtro, de amplitud constante y menor duración a medida que aumenta el número de coeficientes del filtro.



(a) 5 coeficientes. (b) 25 coeficientes. (c) 50 coeficientes.

Figura 2.23: Ilustración del fenómeno de Gibbs.

La solución para el *fenómeno de Gibbs*, y que a la vez se traduce en un nuevo método, consiste en hacer un truncamiento gradual de los coeficientes, método que es conocido como **diseño por ventanas**. El truncamiento descrito en líneas anteriores, es un caso particular de diseño por ventanas en el cual se ha utilizado una ventana rectangular discreta para obtener el valor de los coeficientes. El uso de diferentes ventanas permite atenuar los efectos producidos por el *fenómeno de Gibbs* al introducir bandas de transición en puntos de discontinuidad de la respuesta en frecuencia. Sin embargo, como desventaja se obtiene la introducción de ruido en la banda de paso de diferente amplitud dependiendo de la ventana usada. Diversos autores han propuesto fórmulas [6] para la señal que se usará como ventana, a continuación se muestran las ecuaciones de algunas de ellas.

$$
\begin{aligned}
&\text{von Hann} && w(nT) = \begin{cases} 0,5 + 0,5 \cdot cos\left(\frac{2\pi n}{N-1}\right) & \text{para } |n| \leq \frac{N-1}{2} \\ 0 & \text{otro caso} \end{cases} \\
&\text{Hamming} && w(nT) = \begin{cases} 0,54 + 0,46 \cdot cos\left(\frac{2\pi n}{N-1}\right) & \text{para } |n| \leq \frac{N-1}{2} \\ 0 & \text{otro caso} \end{cases} \\
&\text{Blackman} && w(nT) = \begin{cases} 0,42 + 0,5 \cdot cos\left(\frac{2\pi n}{N-1}\right) + 0,08 \cdot cos\left(\frac{4\pi n}{N-1}\right) & \text{para } |n| \leq \frac{N-1}{2} \\ 0 & \text{otro caso} \end{cases}
\end{aligned}
\tag{2.75}
$$

En la sección anterior se determinó que para que un filtro tenga fase lineal es necesario que sus coeficientes sean simétricos. Sin embargo, alcanzar esta meta mediante el uso de los métodos nombrados resulta muy difícil. La naturaleza del pulso a ocupar en la modulación, una función simétrica y par, hace que mediante el primer método, el más simple de todos,

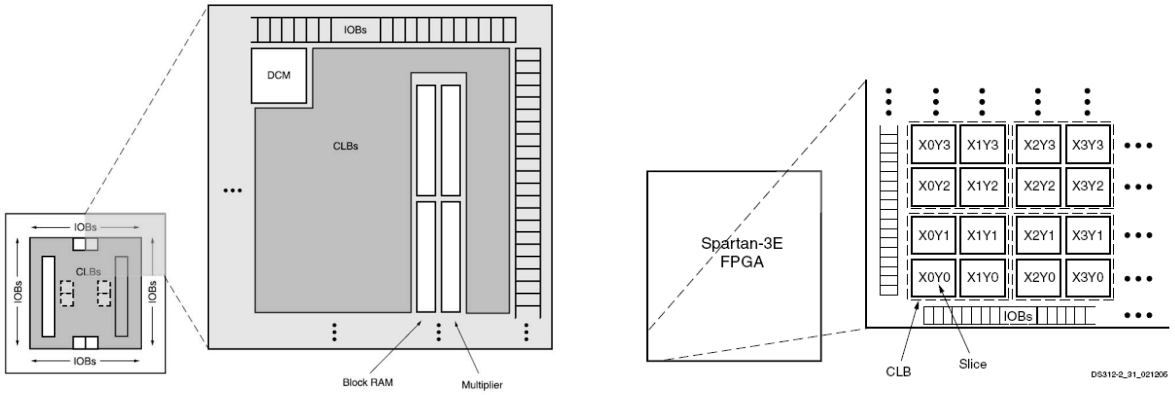
la condición de los coeficientes se cumpla. Por lo anterior se utilizará éste para determinar el valor de los coeficientes del filtro.

2.4. Técnicas de diseño de hardware digital

2.4.1. Field Programmable Array Gates

Un arreglo de campo de compuertas programables o FPGA, consiste en un arreglo bidimensional de compuertas lógicas cuya interconexión es programable mediante software.

La unidad funcional de las FPGA es la celda lógica (*logic cell*), la cual está constituida por una *Look-up Table* (LUT) y un elemento de memoria. A partir de este nivel, la conformación estructural de estas celdas, su organización y su distribución espacial dependen de cada fabricante de plataformas.



(a) Bloques dentro de una FPGA. (b) Ordenamiento de slices dentro de un CLB.

Figura 2.24: Arquitectura de una FPGA Spartan3E.

En la figura 2.24 se muestra la arquitectura de una plataforma Spartan3E del fabricante Xilinx. Se puede apreciar que esta empresa en particular usa el concepto de *slice* para referirse a la unidad funcional de su plataforma. Una *slice* equivale a 2,25 celdas lógicas simples (medición en base a pruebas de rendimiento). Cuatro *slices* conforman un bloque de lógica configurable (CLB) y éste, finalmente, viene a ser la unidad básica que se replica a través de todo el chip de la FPGA. Adicionalmente, a estos bloques de lógica configurable se les suman bloques de entrada y salida (IOB), bloques de memoria RAM, bloques de multiplicadores y bloques que efectúan un control del reloj digital (DCM).

Las *slices* se agrupan en pares y por columna. Las *slices* de la columna derecha (conocidas como SLICEM) implementan funciones de memoria y de lógica, mientras que las de la izquierda (SLICEL) sólo lógica (figura 2.25).

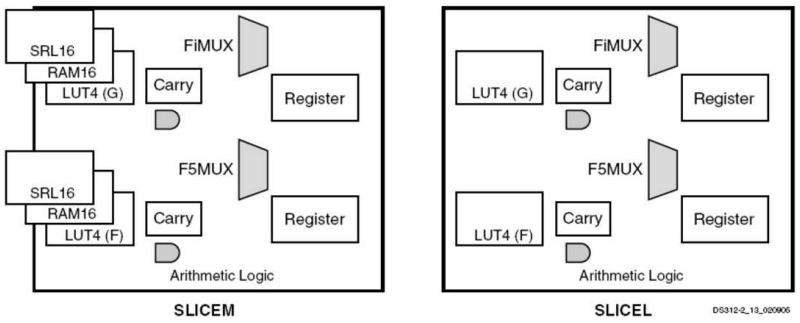


Figura 2.25: Tipos de slice: slicem (izquierda), slicel (derecha).

El uso de las FPGA ha ido en constante aumento debido a su gran flexibilidad y versatilidad frente a otras alternativas como son los chips DSP y los ASIC.

En el caso particular de este trabajo, donde se busca implementar un modulador y demodulador digital, existe una característica especial de estos dispositivos que los hacen muy atractivos respecto de otros tipos de hardware, como lo son los microcontroladores y los DPS. Esta característica corresponde a la posibilidad de paralelizar y/o usar pipeline en la ejecución de ciertas tareas con el objetivo de hacerlas más rápidas y eficientes. Como gran desventaja, se puede nombrar la reducida capacidad de cálculo, lo cual hace que muchas veces una FPGA se acompañe de un DSP externo que compense esta desventaja.

Si bien se puede adquirir el chip de una FPGA directamente, lo común es que se compre una plataforma donde este chip venga con productos que le dan valor agregado y amplían su espectro de uso. Esto se denomina una *plataforma FPGA*. Actualmente, ellas pueden incluir salidas de diversa índole, con el objetivo de conectar variados periféricos a ésta. Típicamente las plataformas FPGA incluyen puertos Ethernet, puertos USB, salidas de video (VGA), etc.

La realización de un proyecto, que en términos generales consiste en la programación de las LUT, memorias, pines de entrada y salida y las conexiones entre los bloques de lógica básicos, se puede realizar a través de diversos métodos, aunque los más usados son dos. El primero consiste en la programación de las funciones por medio de un lenguaje de descripción de hardware (HDL) tal como VHDL o Verilog, mientras que el segundo consiste en la realización de un esquemático (*Schematic*) a nivel de compuertas lógicas. Existen otros métodos tales como la creación de máquinas de estado y la edición directa de las conexiones de los bloques internos de la FPGA, pero estos no son usados comúnmente. Una vez realizada la etapa de programación, definición de entradas y salidas, etc., el software de la plataforma FPGA correspondiente se encarga de crear un archivo que es transferido al dispositivo para efectuar las interconexiones necesarias y lograr el comportamiento deseado. Esta transferencia se puede hacer mediante un dispositivo llamado JTAG (transmisión serial) o directamente mediante la grabación de una memoria EEPROM.

Las empresas más conocidas en el ámbito del desarrollo de plataformas FPGA son Xilinx y Altera. La primera es conocida por sus series *Spartan* (figura 2.26) y *Virtex*, mientras que la

segunda lo es por sus series *Flex10K* y *Cyclone*. Cada una de estas empresas posee software para efectuar los desarrollos sobre los diferentes dispositivos, *Quartus* y *MaxPlus* por parte de Altera e *ISE* por parte de Xilinx. Otras empresas del rubro son Lattice, Actel y QuickLogic.

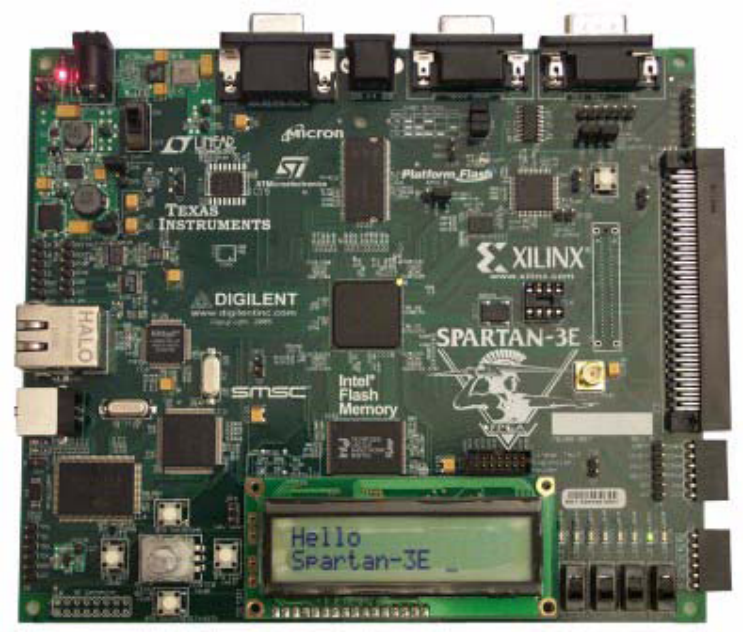


Figura 2.26: Plataforma Spartan3E.

Las operaciones, dentro de una FPGA se manejan a nivel de bits, por lo que efectuar una multiplicación, división, extracción de raíz cuadrada, etc; no es una tarea tan trivial como en otros lenguajes de alto nivel. Para efectuar operaciones más complejas se necesita conocer como se efectúa la aritmética binaria básica. A continuación se presentan los elementos mínimos necesarios que servirán para el desarrollo de bloques mas complejos.

2.4.2. Aritmética binaria básica

2.4.2.1. Sumadores

El sumador aritmético más básico que se puede implementar es el medio sumador (*Half adder*) el cual se define mediante las siguientes ecuaciones lógicas:

$$\begin{aligned} suma &= a \oplus b \\ carry_{out} &= a \wedge b. \end{aligned} \tag{2.76}$$

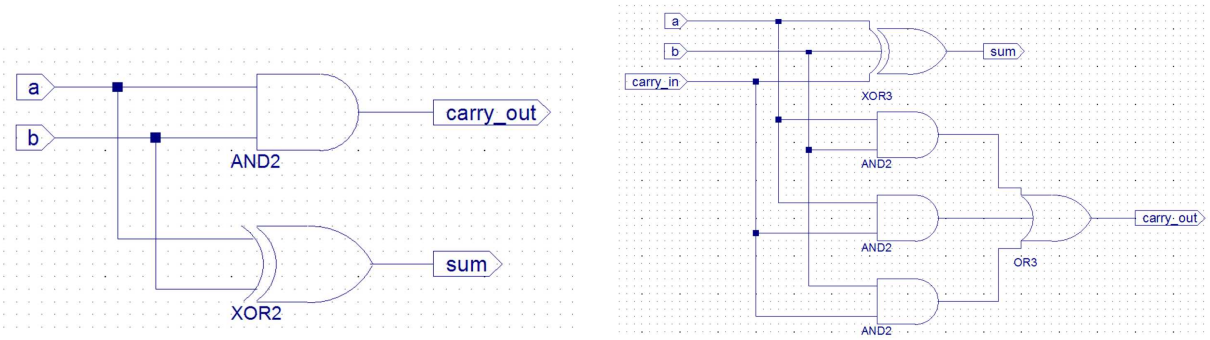
El operador $\wedge$ corresponde al *Y lógico* (and), mientras que el operador $\oplus$ corresponde al operador *O exclusivo* (xor), a y b son palabras de 1 bit.

Otro tipo de sumador corresponde al sumador completo (*Full adder*) el cual, a diferencia del anterior, tiene una entrada adicional, carry_{in}, que corresponde al resto de una suma anterior. Las ecuaciones que lo definen son las siguientes.

$$\begin{aligned} suma &= a \oplus b \oplus carry_{in} \\ carry_{out} &= (a \wedge b) \vee (carry_{in} \wedge a) \vee (carry_{in} \wedge b). \end{aligned} \tag{2.77}$$

donde el operador $\vee$ corresponde a la operación *O lógica* (or).

El esquématico de los sumadores nombrados anteriormente, se muestra a continuación.



(a) Medio sumador.

(b) Sumador completo.

Figura 2.27: Sumadores de 1 bit.

Los sumadores mostrados efectúan el proceso de suma de dos palabras de 1 bit cada una. Para realizar una suma de palabras con un largo igual a n bits basta conectar n sumadores completos en cascada. Ello tiene la desventaja de que a medida que el largo de la palabra crece, el tiempo que demora el sumador en realizar el cálculo (latencia) aumenta, y en consecuencia, su rendimiento decae. Para evitar que el tiempo de cálculo se eleve se utiliza una arquitectura de sumador llamada sumador con predicción de *carry* (CLA). El principio de funcionamiento de este tipo de sumadores se basa en la incorporación de dos nuevas salidas en el sumador completo con la finalidad de que puedan predecir los *carry* por cada par de bits sumados, eliminando la espera forzosa de los resultados de las etapas anteriores. Las dos nuevas salidas de este nuevo sumador completo CLA se denominan generadora (g) y propagadora (p).

$$\begin{aligned} g &= a \wedge b \\ p &= a \oplus b. \end{aligned} \tag{2.78}$$

La función generadora vale "1" sólo cuando se va a generar un *carry* no considerando el *carry* entrante al sumador, es decir, sólo intervienen los valores de a y b. La función

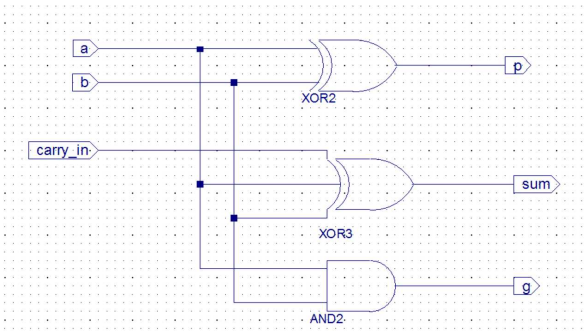


Figura 2.28: Sumador completo CLA 1 bit.

propagadora será "1" sólo cuando se propaga un *carry* para la etapa siguiente. El esquemático del sumador CLA de 1 bit se muestra en la figura 2.28.

Los *carry* sucesivos se pueden generar a partir de estas salidas g y p mediante la siguiente ecuación:

$$c_{i+1} = g_i \vee (p_i \wedge c_i). \tag{2.79}$$

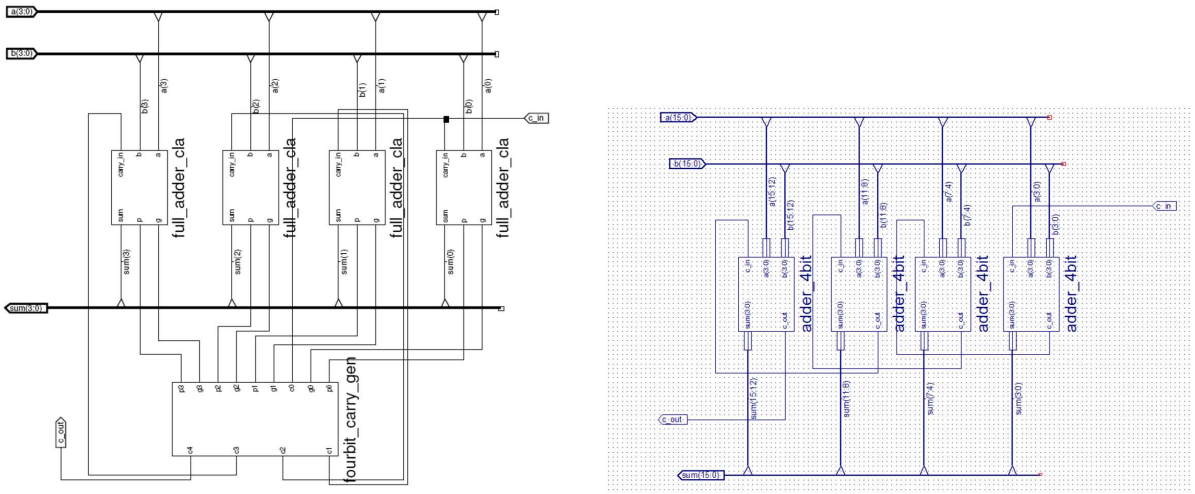
Teniendo todos estos datos se puede crear un sumador de tamaño n. La dificultad del esquema anterior es que a medida que n crece, la lógica necesaria para obtener todas las predicciones de *carry* también aumenta. Para solucionar esto se construyen sumadores CLA de grupos reducidos de bits (4 u 8) y luego se conectan en cascada.

Las ecuaciones de los 4 carry en un sumador CLA de 4 bits se pueden obtener a partir de la ecuación (2.79) mediante reemplazos sucesivos:

$$\begin{aligned}
c_0 =& c_0 \\
c_1 =& g_1 \vee (p_1 \wedge g_0) \vee (p_1 \wedge p_0 \wedge c_0) \\
c_2 =& g_2 \vee (p_2 \wedge g_1) \vee (p_2 \wedge p_1 \wedge g_0) \vee (p_2 \wedge p_1 \wedge p_0 \wedge c_0) \\
c_3 =& g_3 \vee (p_3 \wedge g_2) \vee (p_3 \wedge p_2 \wedge g_1) \vee (p_3 \wedge p_2 \wedge p_1 \wedge g_0) \vee (p_3 \wedge p_2 \wedge p_1 \wedge p_0 \wedge c_0) \\
c_4 =& g_4 \vee (p_4 \wedge g_3) \vee (p_4 \wedge p_3 \wedge g_2) \vee (p_4 \wedge p_3 \wedge p_2 \wedge g_1) \vee (p_4 \wedge p_3 \wedge p_2 \wedge p_1 \wedge g_0) \vee \\
& (p_4 \wedge p_3 \wedge p_2 \wedge p_1 \wedge p_0 \wedge c_0)
\end{aligned} \tag{2.80}$$

Con ello se puede armar un sumador CLA de 4 bits (figura 2.29a) y con cuatro de estas unidades descritas anteriormente, un sumador de palabras de largo igual a 16 bits (figura 2.29b).

Hasta el momento se han descrito sumadores en los cuales interfieren sólo dos operandos, pero a veces es necesario sumar más de dos términos a la vez. La situación anterior se resuelve usando el sumador serial por columnas (SCA). La unidad funcional de este nuevo sumador es el sumador guarda *carry* (CSA) (figura 2.30), el cual tiene una sola salida, el valor de la suma, ya que el *carry* se retroalimenta para incidir en la siguiente suma. Para lograr este comportamiento necesariamente debe introducirse un elemento que guarde este valor para usarlo en la siguiente operación; esta función es realizada por un *flip-flop*.



(a) 4 bits. (b) 16 bits.

Figura 2.29: Sumadores CLA.

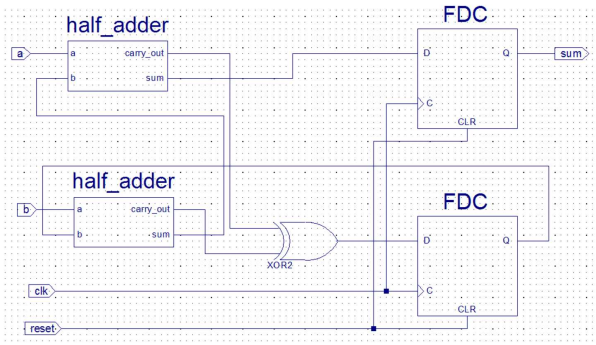


Figura 2.30: Sumador CSA 2 bits.

Para armar el SCA se implementa un árbol de sumadores CSA tal como muestra la figura (2.31).

Es importante que al terminar una operación se reinicien los *flip-flops* que guardan el valor del resto de la suma de todos los sumadores involucrados en el proceso, para evitar que la operación siguiente se altere con resultados de la operación anterior.

2.4.2.2. Complemento dos

Este bloque calcula el complemento dos de un número binario. Este bloque se emplea para corregir el signo de las operaciones y por lo tanto, tener la capacidad de efectuar operaciones correctas independiente del signo de los operandos y sin tener que estar corrigiendo alguno de ellos. El bloque funciona copiando los valores de entrada hasta que encuentra el primer uno de la secuencia, ese es el momento en el cual comienza a invertir los valores en-

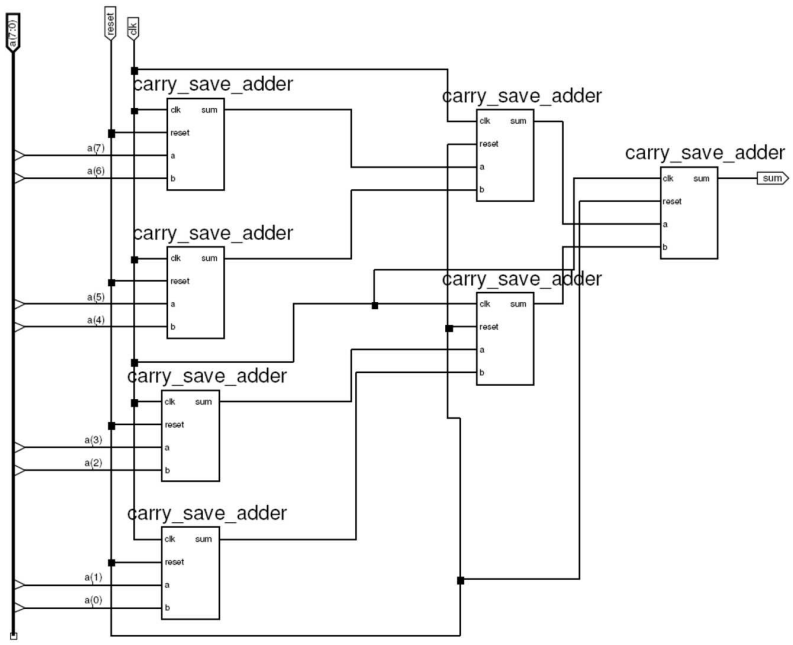


Figura 2.31: Sumador CSA 8 bits.

trantes. Al igual que otros bloques, este debe limpiarse antes de cada operación con tal de no generar resultados erróneos en operaciones posteriores (figura 2.32).

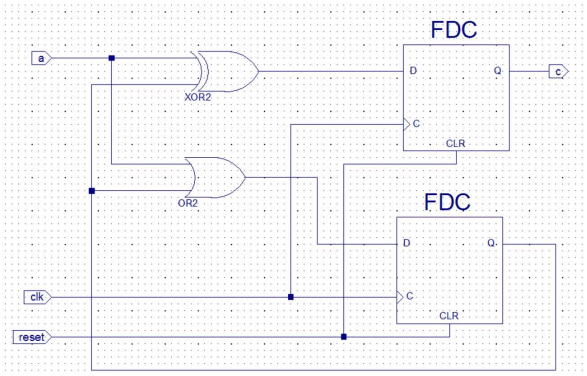


Figura 2.32: Complemento dos.

2.4.2.3. Multiplicadores

El proceso de multiplicación de dos números binarios corresponde a una suma sucesiva de palabras de largo n, una que es multiplicada por un bit del multiplicando y la otra que es el resultado anterior acortanda en un bit mediante la operación de *shift* hacia la derecha, es decir, se suma una palabra de largo n con otra de igual largo debido a que sólo

se han considerado los bits más significativos. El bit menos significativo no será alterado por ninguna operación posterior por lo que constituye, inmediatamente, parte del resultado final de la operación. Este algoritmo de multiplicación se conoce como el método de la suma y el acumulador (figura 2.33).

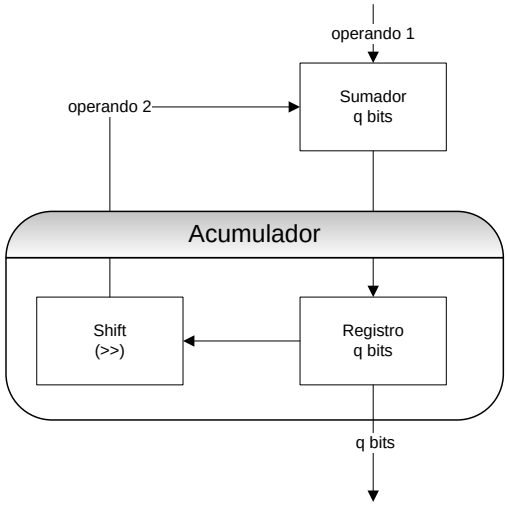


Figura 2.33: Multiplicación a través del método de la suma y el acumulador.

Otro método para efectuar la multiplicación, consiste en ingresar el multiplicando de forma paralela mientras que el número multiplicador se ingresa de forma serial obteniendo el resultado también de forma serial. Este multiplicador se denominda *Bit Serial-Parallel - BSP*. La operación se realiza ingresando primero, el bit menos significativo del multiplicador. El resultado de la operación se obtiene al efectuar un *Y lógico* entre el bit del multiplicador ingresado y cada uno de los bits del multiplicando. Este resultado debe seguir la misma metodología descrita en el método de la suma y el acumulador, esto es, efectuar una suma y luego un *shift* hacia la derecha al término de cada multiplicación. Lo anterior se efectúa utilizando un sumador SCA para guardar el resultado de cada operacion lógica, guardar el resto de la suma para la operación siguiente y, mediante un *flip-flop* interno, establecer el valor del resultado actual como entrada para que el sumador SCA ubicado a la derecha pueda utilizarlo en el siguiente ciclo de reloj, conformando de esta manera el proceso de *shift* requerido. El multplicador debe ser capaz de obtener los resultados correctos sea cual sea la combinación de signos de las entradas; para lograr lo anterior se utiliza un bloque de complemento dos, que efectúa la corrección del signo para los números entrantes.

El diagrama del multiplicador BSP se muestra en la figura 2.34.

2.4.2.4. Delays

Un delay es una función elemental que retrasa el flujo de bits en un tiempo de bit. Luego, una unidad compuesta de *n delays* generará un retraso de n tiempos de bit al flujo de bits entrantes. Lo anterior será util para alinear conjuntos de bits dentro de las operaciones aritméticas (figura 2.35).

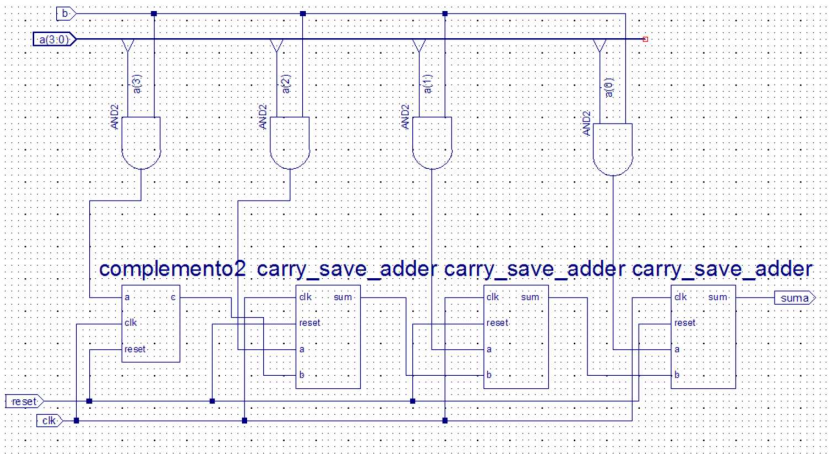


Figura 2.34: Multiplicador BSP 4 bits.

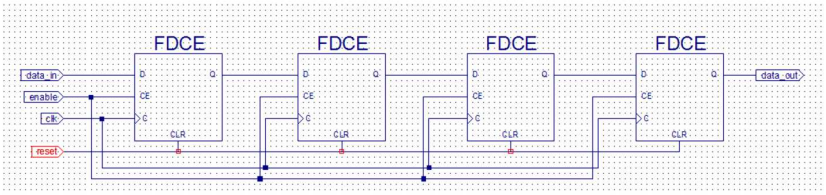


Figura 2.35: Delay 4 bits.

2.5. Implementación de modems con FPGA

La implementación de diversos métodos de modulación y demodulación ha sido, desde hace varios años, un tema impulsor de un gran número de trabajos respecto a los diferentes aspectos que abarca la digitalización de los componentes típicamente analógicos que se utilizan en un módem.

En la actualidad existen problemas de investigación y desarrollo de filtros FIR sobre plataformas FPGA, realización de operaciones aritméticas binarias de forma eficiente (en términos de hardware y tiempo de procesamiento), implementación de funciones trigonométricas, codificación de las secuencias binarias, por nombrar sólo algunos de los más destacados.

La investigación de desarrollo de técnicas en el ámbito digital se justifica ya que esta última ofrece mayor precisión en la respuesta en frecuencia, es fácil de modificar y permite la duplicación exacta de, por ejemplo, filtros tanto en el transmisor como receptor de un sistema de comunicaciones [7].

En los últimos años la densidad de celdas lógicas en los chips ha crecido considerablemente, lo cual junto a la disminución de los retrasos en las compuertas (*delays*), y por ende, el aumento de las máximas frecuencias de reloj a las cuales pueden operar, la disminución de la potencia que consumen [8], bajo costo con respecto a los ASIC [9], reducción

del *time-to-market* [10], flexibilidad ilimitada de cambiar la configuración para probar diferentes alternativas y probar modificaciones en los diseños [11], ahorro de componentes [12] y en algunos modelos de plataformas, la posibilidad de reconfiguración *online* [13], [14]; han hecho de las FPGA una alternativa ampliamente preferida en el desarrollo digital de grandes proyectos.

La optimización de operaciones aritméticas binarias básicas se centra principalmente en los multiplicadores, pues estos son los elementos que hacen mayor uso de hardware y de tiempo de procesamiento con respecto al resto de las operaciones. Diferentes alternativas aparecen en la literatura especializada, dependiendo de los objetivos del diseño. Con bajo costo en recursos lógicos pero con limitada frecuencia de operación apareceren los multiplicadores paralelos [11], con una alta frecuencia de operación pero elevado uso de recursos aparecen multiplicadores que ocupan *pipeline* (seriales) [11], [9], [15], [16] o arreglos sistólicos [11]. También es posible encontrar implementaciones que se ubican en un punto intermedio de las dos metas mencionadas, como por ejemplo, los multiplicadores de dígitos (conjuntos de n bits de una palabra de mayor tamaño) seriales [12]. Los resultados prácticos de la implementación de estos multiplicadores dependen exclusivamente del número de bits que tengan los operandos y, en menor medida, de la plataforma FPGA usada. Por esto, los resultados en cuanto a área y tiempo de procesamiento son muy variados, permitiendo así, una elección muy acorde a los requerimientos del problema.

Otros autores van mas allá de sólo mejorar la implementación del proceso de multiplicación en sí y proponen nuevas alternativas de codificación de los operandos para disminuir la cantidad de operaciones involucradas en el cálculo. Se destaca el uso de codificación de Booth [17] y el uso de CSD [18], [19].

La implementación de un filtro FIR reune las optimizaciones mostradas en el ámbito de aritmética computacional así como en su arquitectura, diferentes tipos: directa, transpuesta y transpuesta sistólica [7], [9], [19] son propuestas para su realización. Además, se agregan implementaciones con aritmética distribuida simple [10], [20], [21] y optimizada [13], métodos que si bien reducen considerablemente el uso de recursos de la plataforma FPGA, disminuyen el rendimiento general [22]. Adicionalmente, se puede usar la optimización de coeficientes [23] o coeficientes codificados mediante CSD en un espacio de potencias de dos [18]. Todo lo anterior, nuevamente, se mueve entre la meta de lograr un diseño veloz o un diseño que ocupe mínimos recursos. La soluciones, al igual que en el caso dé los multiplicadores, son muy variadas y permiten una elección acorde a la situación.

La generación de funciones trigonométricas, fundamentales en los esquemas de modulación y demodulación, se desarrollan a través de implementación de CORDIC, algoritmo que permite la obtención de senos y cosenos mediante un algortimo de aproximaciones sucesivas [24]. Otro método, de generación de senos y cosenos es DDS [25].

Existen también desarrollos para casos particulares, como por ejemplo, la implementación de un método de calculo de la función arco tangente para un demodulador FM/FSK [14].

Un área que muestra el nivel alcanzado en la digitalización de los elementos típicamente analógicos es el desarrollo de algoritmos que permiten implementar PLL de forma digital [26],

[27], [28].

Finalmente, el desarrollo de sistemas completos de transmisión y recepción encapsulan todos los desarrollos descritos anteriormente. Es posible encontrar implementaciones de moduladores BPSK, QPSK y FSK [29], de demoduladores FM /FSK [14], FSK [30] y QAM [31], de módems QPSK [32], [25] y BPSK [33], incluso, un sistema que podría considerarse bastante complejo, como un radar, también es objeto de investigación y desarrollo [34].

Capítulo 3

Diseño e implementación de la solución en la plataforma FPGA

En este capítulo nos concentraremos en el desarrollo y posterior implementación de un sistema de modulación y demodulación en una plataforma FPGA. Para resolver este problema nos enfocaremos primero en los bloques básicos que componen el modulador y demodulador, y que luego se integran para formar el sistema completo. Para cumplir con lo anterior, la primera tarea a realizar es la definición de forma detallada de los diagramas de bloque de cada solución y el detalle de cada unidad funcional presente en ellos. Una consideración respecto a la implementación y programación de los bloques, es que ésta se desarrolló de la forma más general posible con el fin de que el diseño pueda ser trasladado a cualquier otra plataforma FPGA cuya capacidad, en cuanto a celdas lógicas se refiere, sea suficiente y éste siga funcionando, es decir, que exista independencia con respecto a la arquitectura.

El diseño del sistema no considera alguna meta en especial salvo la de funcionamiento correcto de éste, es decir, las implementaciones no buscan ser óptimas en cuanto uso de celdas lógicas, así como tampoco se desea que operen a la máxima frecuencia de operación posible. El diseño tampoco considera algún esquema de corrección de errores dentro del demodulador.

3.1. Parámetros de la solución

Consideraremos primero la modulación PAM antipodal con una onda base del tipo raiz cuadrada coseno alzado con un factor α igual a 0.4 (2.40). Debido a que la señal es infinita en el espacio temporal, se decidió utilizar el intervalo comprendido entre los instantes $-4T$ y $4T$ segundos. La duración del pulso, más exactamente el valor de T, depende exclusivamente del valor de la frecuencia del reloj utilizado en el modulador y a su vez, esta última, está ligada con la frecuencia del reloj del demodulador. Las consideraciones para determinar los valores de estos relojes se explican más adelante.

En el modulador, un aspecto clave es la generación del pulso para modular los datos binarios. Se consideraron tres alternativas. La primera consiste en calcular, a través de módulos de aritmética binaria, la forma de la señal dada su fórmula algebraica, la segunda consiste en usar memorias que contengan las muestras calculadas por un programa externo y la tercera consiste en implementar un filtro que genere la señal. Dada la complejidad matemática de generar una forma de onda raíz cuadrada de coseno alzado y de la imposibilidad de utilizar memorias con lecturas multiples, se decidió implementar la tercera opción con un filtro FIR de 33 *taps*. Los valores de los coeficientes de este filtro son valores reales codificados en trece bits en formato Q12.0, con el fin de que su manipulación a través de los distintos bloques del modulador diera a lugar números con el formato requerido por el conversor digital a análogo integrado en la plataforma.

Definición 3.1. El **formato Q** expresa, para números de punto fijo con signo, la cantidad de bits ocupados para codificar la parte fraccionaria y la cantidad usada para la parte entera. La notación para este formato es del tipo $Qx.y$ donde x corresponde a la cantidad de bits usados para representar en complemento dos la parte entera e y los bits usados para representar en complemento dos la parte fraccional. El número total de bits usados por un número con formato $Qx.y$ son $x + y + 1$ bits.

Para sumar un número en formato $Qx.y$ con otro en formato $Qw.z$ estos deben ser alineados con respecto al punto decimal, lo cual se logra con operaciones de *shift* y extensión de signo. El resultado de esta operación será un número con formato $Qa.b$ donde $a = max(x, w)$ y $b = min(y, z)$. Este resultado puede tener un error de *overflow* que se evita añadiendo un bit mas a la codificación de la parte entera. La multiplicación de un número en formato $Qx.y$ con otro en formato $Qw.z$ da lugar a un número en formato $Qa.b$ donde $a = x+w$ y $b = y+z$

En el demodulador, dado que se debe implementar un filtro de características similares, se usó un filtro FIR con igual número de *taps* sólo que en este caso los coeficientes, reales positivos y negativos, fueron codificados en formato Q0.13. A medida que crece el número de *taps* de un filtro se necesitan más bits para guardar con precisión el valor correcto al término de cada operación. Dado que en este caso no se necesita gran precisión, los datos provenientes del conversor análogo a digital fueron convertidos de formato Q0.13 a Q0.11.

Los coeficientes, para ambos filtros, fueron obtenidos utilizando la función `rcosine` de Matlab. En esta última, se ha ingresado como parámetro, que entre puntos de decisión sucesivos existan cuatro muestras (razón entre los parámetros F_s y F_d del comando `rcosine` igual a 4).

Condiciones generales impuestas para la aritmética, sumas y multiplicaciones, en cualquiera de los bloques fueron las siguientes: procedimientos que necesiten sumar dos términos con gran cantidad de bits ocupan arreglos de sumadores CLA (2.29) de acuerdo a la cantidad de bits que posea el término más largo. La suma de más de dos términos ocupa sumadores SCA (2.31). La multiplicación ocupa multiplicadores BSP de tamaño relativo a la situación(2.34).

La plataforma FPGA utilizada para armar el modulador corresponde a una unidad Spartan XC3S700AN de la empresa Xilinx, mientras que la plataforma ocupada para la

implementación del demodulador es una Spartan XC3S500E, de la misma empresa. La programamción de estas plataformas se realizó a través del software ISE mediante lenguaje Verilog y esquemáticos.

3.2. Diagramas de bloques

Los diagramas de bloque dan la orientación necesaria para el desarrollo de las unidades funcionales. Tal como se ha planteado en el capítulo 2, el problema está dividido, a grandes rasgos, entre la implementación del modulador y la del demodulador.

3.2.1. Modulador

El diagrama de bloques propuesto para el modulador se muestra en la figura (3.1).

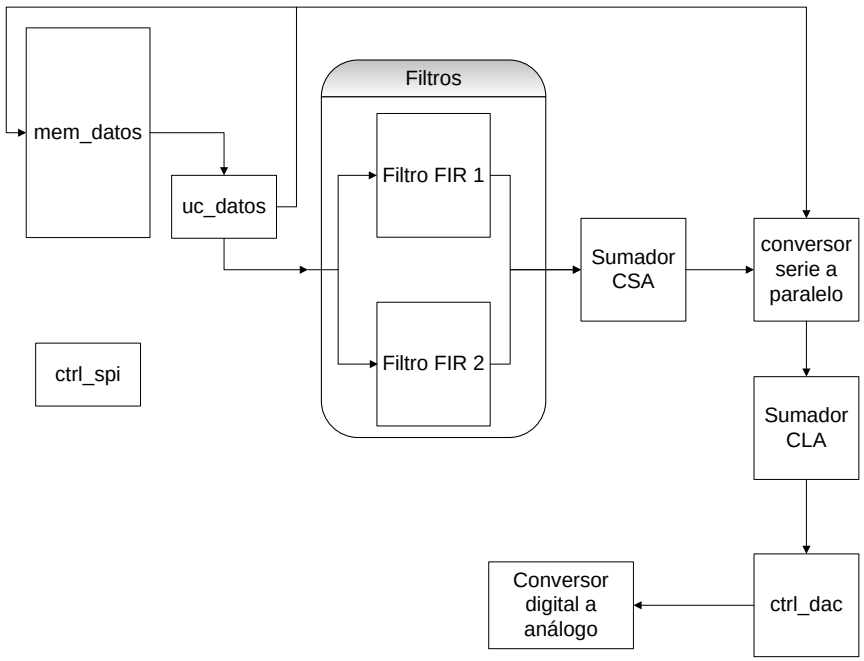


Figura 3.1: Diagrama de bloques modulador.

En el diagrama se muestra, en primer lugar, la unidad principal de control denominada *uc_datos* la cual se encarga de leer los datos almacenados en la memoria de nombre *mem_datos*. Este bloque de memoria es de un bit de ancho y altura variable y contiene la secuencia de unos y ceros que se envia a través del modulador. La altura de esta memoria depende del largo de la secuencia a enviar y además está limitada por el número de recursos disponibles en la plataforma FPGA luego de la implementación de todos los bloques esen-

ciales restantes. El tipo de esta memoria es de sólo lectura (ROM) por lo que no se puede modificar durante la ejecución del modulador.

El dato obtenido desde la memoria es ingresado a la unidad central *uc_datos* la cual decide hacia que filtro debe enviar un impulso (una serie de bits de valor uno decimal). Estos filtros contienen las formas de onda a utilizar en el sistema de modulación escogido.

Dado que el envío de los pulsos raiz cuadrada de coseno alzado se hace traslapando unos con otros, un sumador CSA efectúa la suma de los resultados provenientes de los dos filtros y los entrega hacia un conversor serie a paralelo el cual es sincronizado, en la adquisición de datos, por la unidad *uc_datos*. Los datos obtenidos por el conversor serie a paralelo no concuerdan con el formato con el que se deben enviar hacia el conversor digital a análogo, para solucionar este problema se agregó un sumador CLA que corrige esta situación.

Los datos entregados por el sumador CLA son capturados por una unidad, *ctrl_dac*, que se encarga de enviarlos hacia el conversor digital a análogo. Este controlador implementa una comunicación mediante un bus SPI. En el diagrama tambien existe un bloque denominado *ctrl_spi* que se encarga de deshabilitar otros dispositivos que ocupen bus SPI con el fin de no generar conflictos en la comunicación hacia el integrado.

La secuencia que sigue el modulador para modular un bit es la siguiente: se da la orden de partida para que el bloque *uc_datos* lea un bit de la memoria *mem_datos*, una vez leído el dato esta unidad decide hacia que filtro se envía el "impulso" para generar la forma de onda deseada. El dato corregido proveniente del filtro FIR es capturado por la unidad *ctrl_dac* e inmediatamente es enviado hacia el conversor. Una vez que el envío de la información termina, se activa una señal para que la unidad central continúe con su funcionamiento. Esta última ahora envía un cero decimal hacia los filtros y espera nuevamente la señal de término dada por la unidad *ctrl_dac*. Este último proceso se repite 3 veces y luego, se vuelve a leer un valor de la memoria *mem_datos*. Cuando la unidad central termina de leer la última dirección, comienza indefinidamente a enviar flujos de bits con valor cero decimal hacia ambos filtros.

3.2.2. Demodulador

El diagrama de bloques para el demodulador se presenta en la figura (3.2).

La primera unidad que se observa en el diagrama de modulador es una unidad denominada *ctrl_spi* que se encarga de deshabilitar otros dispositivos que ocupen el bus SPI para que éstos no interfieran con el funcionamiento del amplificador y del conversor análogo a digital. Cabe mencionar que estas dos unidades no pueden ser configuradas al mismo tiempo, por lo que las señales de control para la habilitación o deshabilitación de las unidades se encuentran implementadas dentro de cada módulo controlador. El bloque siguiente, *ctrl_amp*, se encarga de efectuar la activación del amplificador, para lo cual, envía mediante bus SPI el valor de la ganancia de cada amplificador. Este bloque se activa usando un botón adecuado para tal propósito.

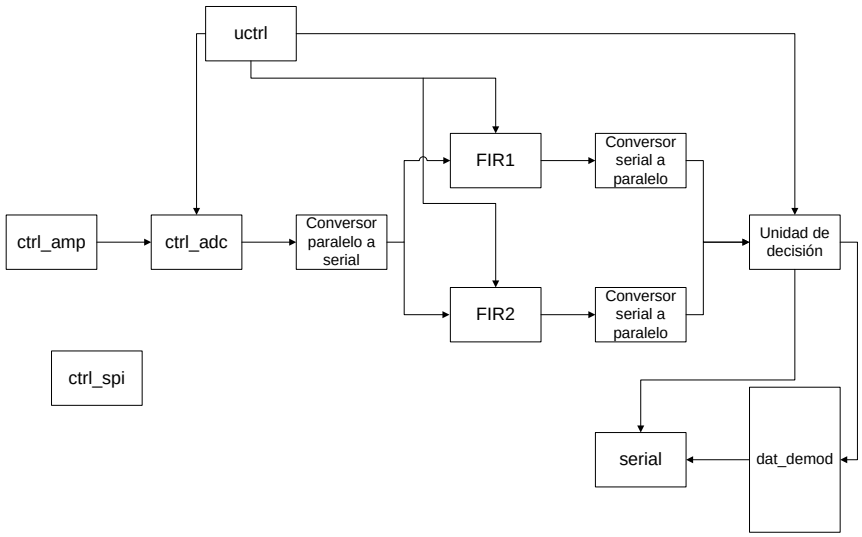


Figura 3.2: Diagrama de bloques demodulador.

A continuación se encuentra el módulo que controla la adquisición de datos por parte del conversor análogo a digital denomindado *ctrl_ adc*. La comunicación se produce a través de un bus SPI, sin embargo, esta unidad entrega el resultado de forma paralela. El bloque siguiente, un conversor paralelo a serial, se encarga de recibir el dato adquirido y traspasarlos hacia el filtro FIR de tal manera que el primer bit saliente sea el bit menos significativo.

Los bloques siguientes corresponden a filtros FIR. Existen tantos filtros como formas de ondas usadas para efectuar la modulación. La salida de cada filtro FIR es de tipo serial, por lo que un conversor serial a paralelo se ubica entre ellos y la unidad de decisión.

La unidad de decisión se encarga de decidir, a través de los datos entregados por el filtro, qué símbolo fue transmitido por el modulador y además se encarga de guardarlo en una memoria *dat_ demod*. Una vez que la unidad de decisión ha llenado la memoria con datos demodulados envía una señal de inicio para que la siguiente unidad, *serial*, lea la memoria y envié los datos a través de un puerto RS-232 con el objetivo de visualizar el dato en el computador mediante un programa de escaneo de puerto serial.

Un último bloque, *uctrl* es implementado para controlar y sincronizar todas las señales de los diversos bloques para el correcto funcionamiento del demodulador. Las tareas de esta unidad son: controlar la adquisición de datos por parte del conversor análogo a digital, reiniciar los multiplicadores y sumadores e indicarle a los conversores serial a paralelo cuando tomar un dato proveniente del filtro FIR.

3.3. Implementación de bloques principales

3.3.1. Conversores (DAC, ADC) y amplificadores

Para la utilización del conversor digital a análogo quádruple LTC2624, del amplificador dual LTC6912-1 y del conversor análogo a digital dual LTC1407A-1, se han implementado unidades que permitan la configuración, el envío y la recepción de datos a través de bus SPI siguiendo los diagramas de tiempo mostrados en [35]. También se ha implementado, dentro de cada módulo, y cuando se requiera, de forma externa, el arbitraje del bus SPI, puesto que éste es compartido por los tres integrados además de otros dispositivos que no serán ocupados en este proyecto.

3.3.2. Filtro FIR

El filtro FIR tiene 33 coeficientes de 13 bits en formato Q0.12 para el modulador y 14 bits en formato Q0.13 para el demodulador. Para este último, y considerando la supresión de los dos últimos bits del valor entregado por el ADC, el resultado esperado a la salida del filtro tendrá formato Q7.24, para lograr obtener resultados correctos de 32 bits, uno de los términos debe tener el formato Q7.24, esto se logrará ejecutando una extensión de signo de los datos provenientes del conversor análogo a digital en los conversores paralelos a serie que preceden a los filtros FIR para obtener palabras del largo adecuado.

La composición del filtro considera 3 sub-bloques. El primero de ellos corresponde a un arreglo de delays de largo 13 bits en el caso del modulador y 32 bits en el caso del demodulador. La cantidad de *delays* presentes (13 ó 32 bits según sea el caso) es igual al número de coeficientes del filtro. Las entradas de este sub-bloque son las señales de reloj, *reset*, datos y un *enable*, señal de control proveniente del conversor parelelo a serial que indica en qué momento comenzar a guardar y desplazar las palabras. El segundo sub-bloque corresponde a un arreglo de multiplicadores BSP junto a un bloque que entrega el valor del coeficiente para cada *tap* del filtro; sus entradas son una señal de reloj, la entrada de datos para cada multiplicador proveniente de cada *delay* de largo 32 bits o 13 bits, según sea el caso, y la señal de *reset*. El tercer sub-bloque corresponde a un sumador SCA de 33 términos compuesto por bloques sumadores SCA mas pequeños (2, 4, 8 ó 16 bits). Las entradas de este último bloque son un reloj, los resultados provenientes de cada multiplicador y un *reset*.

La tabla (3.1) muestra la cantidad de recursos ocupados por un filtro FIR en una plataforma Spartan XC3S700AN en el modulador, mientras que la tabla (3.2) muestra la cantidad de recursos ocupados por un filtro FIR en una plataforma Spartan XC3S500E en el demodulador.

	Ocupadas	Disponibles	Porcentaje de ocupación
Utilización de lógica			
Slices de flip-flops	1.349	11.776	11 %
LUTs de 4 entradas	906	11.776	7 %
Distribución de lógica			
Slices	804	5.888	13 %
LUTs de 4 entradas	906	11.776	7 %

Tabla 3.1: Recursos ocupados en la implementación de un filtro FIR de 33 coeficientes de 13 bits sobre una plataforma FPGA Spartan3AN en el modulador.

	Ocupadas	Disponibles	Porcentaje de ocupación
Utilización de lógica			
Slices de flip-flops	2.023	9.312	21 %
LUTs de 4 entradas	965	9.312	10 %
Distribución de lógica			
Slices	1.270	4.656	27 %
LUTs de 4 entradas	965	9.312	10 %

Tabla 3.2: Recursos ocupados en la implementación de un filtro FIR de 33 coeficientes de 14 bits sobre una plataforma FPGA Spartan3E en el demodulador.

3.3.3. Unidad de decisión

La unidad de decisión es el elemento encargado de determinar qué símbolo fue enviado, en este caso particular donde los símbolos son agrupaciones de un bit, se debe determinar si se recibe un "0" o un "1". Esta unidad implementa el algoritmo de detección detallado en el apartado referente al filtro adaptado, es decir, cada T segundos debe tomar los resultados provenientes de los distintos filtros FIR y en base a los valores de las muestras decidir el símbolo demodulado. Este método fue explicado y desarrollado para un sistema de demodulación analógico, sin embargo, como la implementación de esta unidad está inserta dentro de un sistema totalmente digital, el problema ya no involucra intervalos de tiempos, sino que la variable a considerar corresponde al número de muestras. El correcto funcionamiento de esta unidad tiene como elemento fundamental la correcta sincronización entre el flujo de datos recibidos y el instante en que la decisión es tomada, esta última debe hacerse en el momento en que el filtro entregue el mayor valor posible de la convolución. Lo anterior se logró adaptando el modulador y la parte de captura de datos analógicos del demodulador para que la sincronización se logre en el primer símbolo transmitido.

El modulador en su estado inicial envía una señal de valor cero hacia el conversor análogo a digital, éste último, según lo descrito en [35], entrega un valor que corresponde al máximo valor positivo posible (0x1FFF) el cual produce en la salida del filtro FIR el valor máximo posible de la convolución. El valor del primer símbolo enviado se fija en "1". Con lo anterior se logra que al empezar a enviar la secuencia de símbolos, el valor de la convolución sea

siempre menor que el máximo. En el momento en que la muestra proveniente del FIR difiere del valor máximo, la unidad de decisión pasa a un estado en el cual comienza a funcionar un contador. Al llegar a la muestra 33 (este número depende de los coeficientes del filtro), esta contendrá el mayor valor de la convolución entre la señal recibida y los coeficientes del filtro y significará la detección del primer símbolo. Las decisiones sucesivas se efectuan cada 4 muestras tal como fue estipulado en los parámetros del sistema.

3.4. Construcción del sistema de modulación

Para construir el sistema de modulación se siguió la configuración mostrada por la figura (3.1). La figura (3.3) muestra el sistema armado completamente en el software ISE mediante esquemáticos.

El número total de recursos usados por el modulador completo se muestran en la tabla (3.3).

	Ocupadas	Disponibles	Porcentaje de ocupación
Utilización de lógica			
Slices de flip-flops	2.828	11.776	24 %
LUTs de 4 entradas	2.051	11.776	17 %
Distribución de lógica			
Slices	1.766	5.888	29 %
LUTs de 4 entradas	2.051	11.776	17 %
IOBs	13	372	3 %
RAMB16BWEs	8	20	40 %
BUFGMUXs	7	24	29 %
DCMs	3	8	37 %

Tabla 3.3: Recursos ocupados en la implementación del modulador sobre una plataforma FPGA Spartan3AN.

El sistema de demodulación armado se muestra en la figura (3.4). Un resumen con los recursos ocupados por el demodulador se muestra en la tabla (3.4).

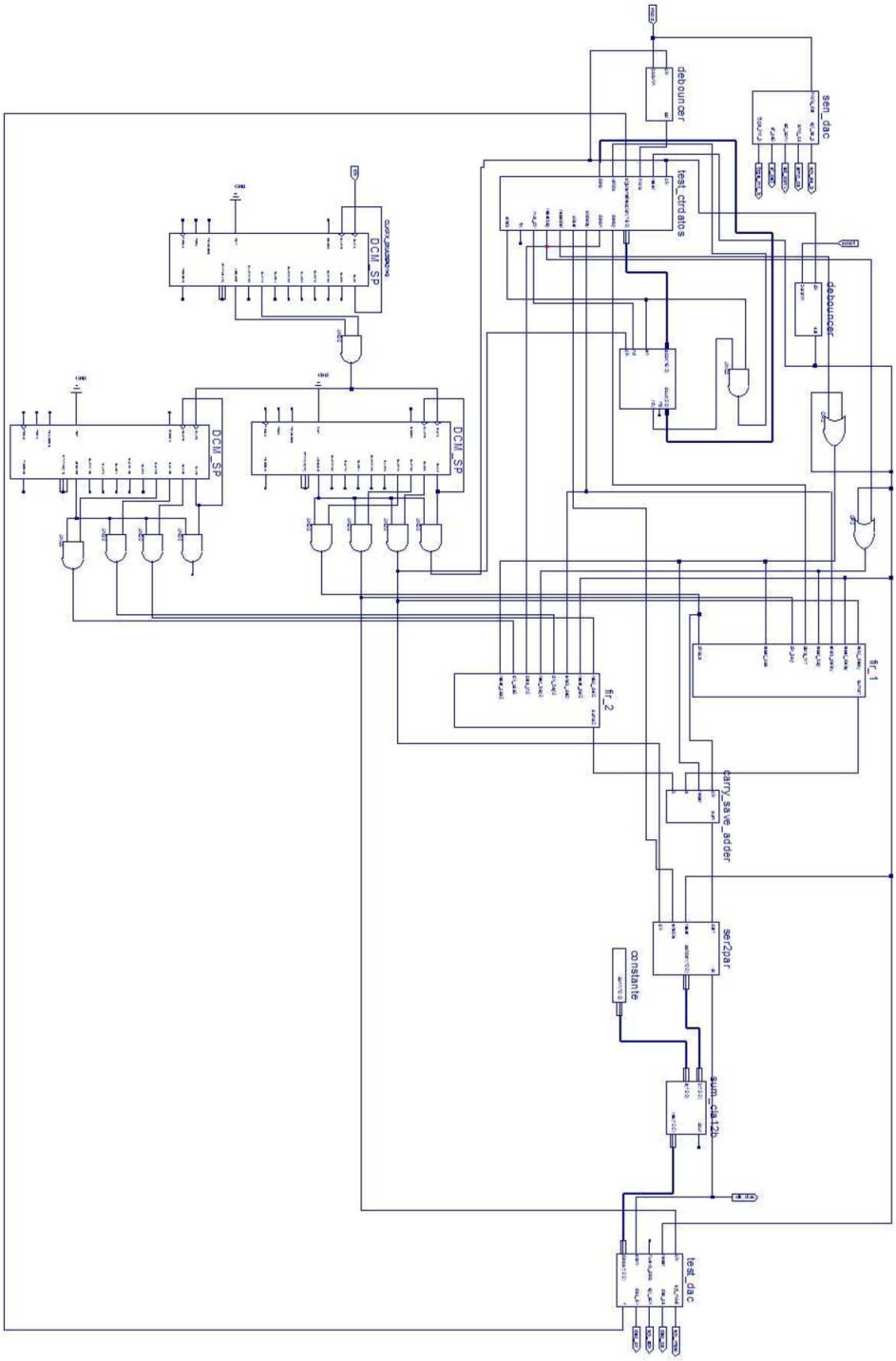


Figura 3.3: Esquemático del modulador en el programa ISE.

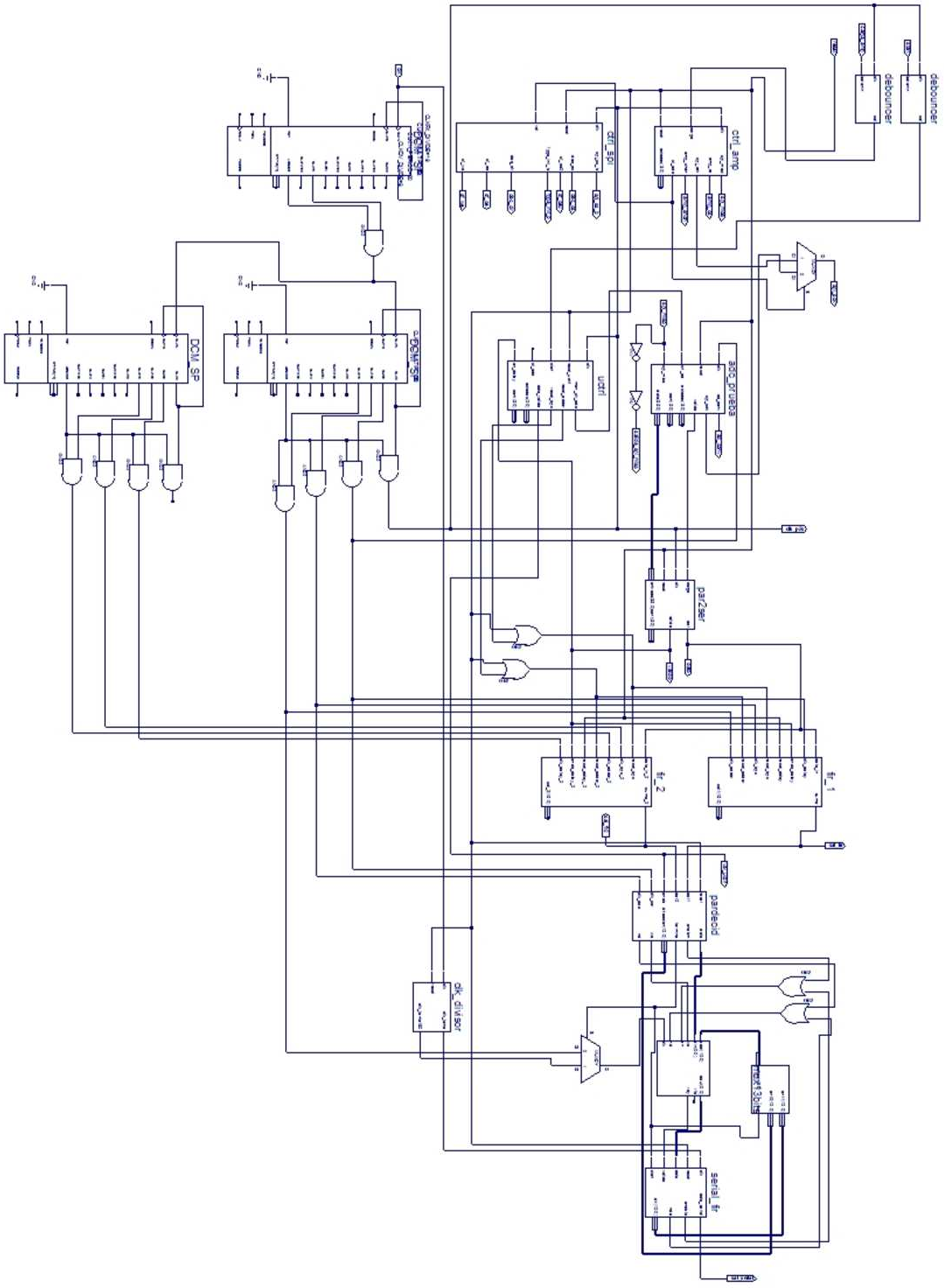


Figura 3.4: Esquemático del demodulador en el programa ISE.

	Ocupadas	Disponibles	Porcentaje de ocupación
Utilización de lógica			
Slices de flip-flops	4.467	9.312	47 %
LUTs de 4 entradas	2.577	9.312	27 %
Distribución de lógica			
Slices	2.868	4.656	61 %
LUTs de 4 entradas	2.619	9.312	28 %
IOBs	26	232	11 %
RAM16s	8	20	40 %
BUFGMUXs	8	24	33 %
DCMs	3	4	75 %

Tabla 3.4: Recursos ocupados en la implementación del demodulador sobre una plataforma FPGA Spartan3E.

3.5. Pruebas de funcionamiento

Las pruebas de funcionamiento realizadas consistieron en someter tanto el modulador como el demodulador a pruebas de transmisión de una secuencia conocida cuyos resultados prácticos, las salidas de cada bloque, fueron comparados con resultados teóricos con el propósito de verificar el correcto funcionamiento de la implementación desarrollada.

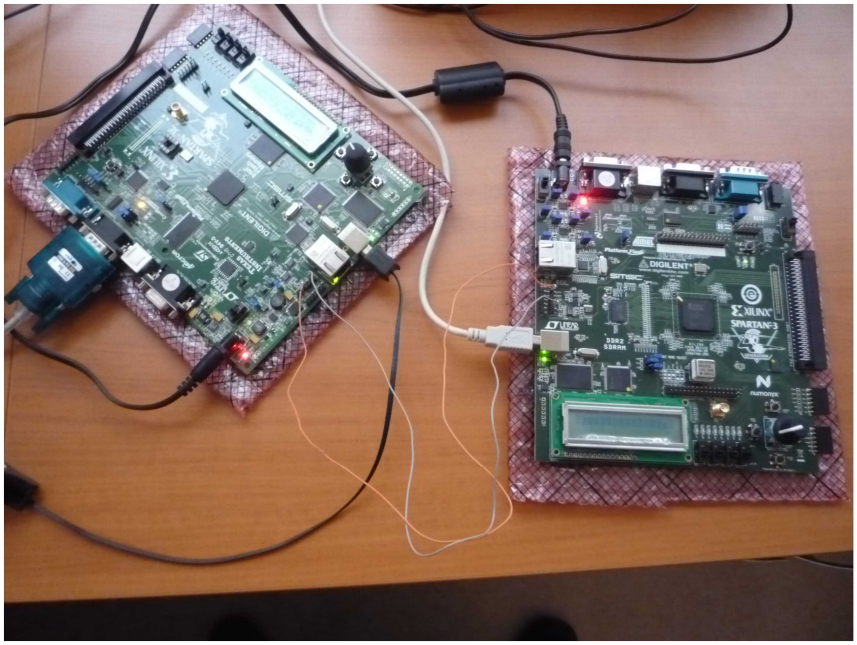


Figura 3.5: Disposición de las plataformas FPGA para realizar las pruebas de funcionamiento.

3.5.1. Prueba del sistema de modulación

Para probar el correcto funcionamiento del sistema de modulación, se creó una memoria que contiene una secuencia aleatoria de símbolos. A través del osciloscopio se observó la salida análoga desde el conversor digital a análogo A del integrado LTC2624 perteneciente a la plataforma Spartan3AN y se comprobó que el primer grupo de símbolos correspondiera al encabezado de la secuencia (figura 3.6) usando como referencia la forma de onda teórica obtenida a través de Matlab.

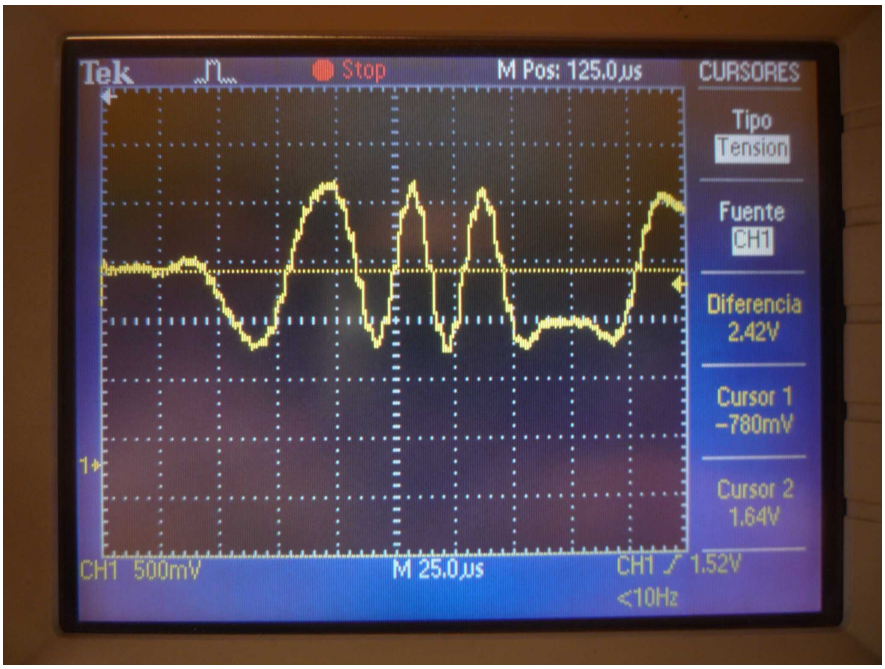


Figura 3.6: Símbolos modulados observados a través del osciloscopio.

3.5.2. Prueba del sistema de demodulación

La prueba de funcionamiento del sistema de modulación se dividió en dos partes: la primera de ellas corresponde a la verificación del funcionamiento de todos los bloques salvo la unidad de decisión y la segunda corresponde al funcionamiento del sistema completo. La prueba fue realizada colocando un valor de voltaje continuo en la entrada A del conversor análogo a digital, para luego chequear que éste entregara una lectura correcta del valor. Se prosiguió con el conversor paralelo a serie, en el cual se controló que el dato fuera bien convertido y que ademas la señal de control, que habilita el funcionamiento del arreglo de *delays* del filtro, estuviese bien sincronizada.

La prueba de funcionamiento del filtro FIR también se realizó comprobando el correcto funcionamiento de los sub-bloques que lo componen. Al ser un bloque donde no interviene la programación, se puso especial cuidado en verificar la sincronización de los sub-bloques, es decir, que el arreglo de *delays* entregase al mismo tiempo los datos para que cada multiplicador efectúe la operación y que estos últimos tuviesen similar comportamiento para con

el sumador. Como prueba final, se constató que el sumador entregara un dato correcto, y por lo tanto, validara el funcionamiento del filtro y por otro lado, que la unidad de control del sistema completo generara oportunamente la señal de control que indica la validez de un dato calculado.

Una vez finalizada esta parte se procedió a verificar el funcionamiento de la unidad de decisión, y por ende, del sistema completo. Para lo anterior se usó el modulador, cuyo correcto funcionamiento se verificó anteriormente, para enviar una secuencia de datos, demodularla y enviar los símbolos obtenidos a través del puerto serial RS-232.

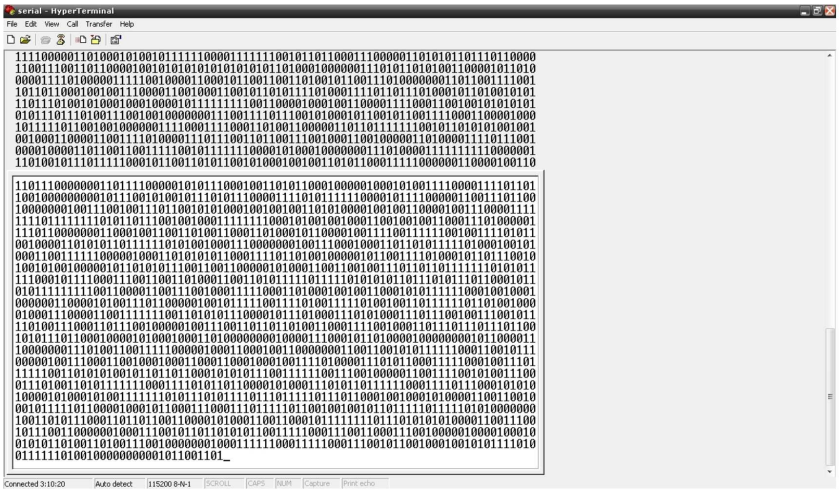


Figura 3.7: Símbolos demodulados observados a través de Hyperterminal (conexión serial).

Capítulo 4

Discusión y análisis

4.1. Implementación y funcionamiento del sistema

Durante la implementación del sistema y posterior ejecución de pruebas para verificar su funcionamiento, se presentaron variados problemas que fueron oportunamente resueltos, pero que sin embargo merecen ser detallados para poseer mayor información sobre el funcionamiento y las especificaciones de la solución creada.

Tanto el modulador como el demodulador poseen elementos que no son controlados por una máquina de estados y, por lo tanto, necesitan ser controlados por una unidad externa. Ello genera un problema que involucra la correcta sincronización de las señales de control para que el sistema funcione de forma correcta y los datos no se corrompan en algún bloque intermedio. Esto último hace referencia principalmente al manejo de señales de control para el funcionamiento del filtro FIR.

Al probar el funcionamiento del sistema se observó que para secuencias cortas de datos (< 400 símbolos) el sistema demodulaba sin errores la señal. Sin embargo, para secuencias de hasta 8192 símbolos el número de errores cometidos bordeaba la mitad de los bits demodulados. Ejecutando varias pruebas, recuperando la posición del símbolo en la cual se cometía el primer error y notando que este último siempre se encontraba en, aproximadamente una misma posición, se logró determinar que un problema de sincronización en las frecuencias de funcionamiento del modulador y el demodulador era el causante de tantos errores en la demodulación. La solución correspondió a calcular para cada parte del sistema el número de ciclos de reloj ocupados en el envío y adquisición de una muestra y con estos valores elegir los divisores correctos para la frecuencia de funcionamiento del modulador y demodulador.

$$t_{mod} = T_{50MHz} \cdot M_m \cdot 54. \tag{4.1}$$

$$t_{demod} = T_{50MHz} \cdot M_d \cdot 62. \tag{4.2}$$

donde M_m y M_d corresponden a los multiplicadores del modulador y demodulador, respecti-

vamente. Igualando ambas expresiones,

$$M_m \cdot 54 = M_d \cdot 62. \tag{4.3}$$

ejecutando simplificaciones y considerando que los multiplicadores deben tener un valor menor a uno, se llega a la siguiente solución:

$$M_m = \frac{9}{31}. \tag{4.4}$$

$$M_d = \frac{1}{3}. \tag{4.5}$$

Implementando los relojes del modulador y demodulador con los multiplicadores obtenidos el sistema experimentó una sustantiva disminución en el número de errores, alcanzando un desempeño perfecto, es decir, cero errores para una secuencia de 8192 símbolos. Dado el éxito de esta solución se implementaron memorias con un tamaño más grande con el fin de probar la robustez de este sistema de sincronización. Se logró, exitosamente, la modulación y demodulación, con cero errores, de secuencias de hasta 131.072 bits (128 Kb). Sin embargo considerando que no siempre los frecuencias estarán en la razón calculada, ya sea por que los relojes presentan *skew* o *jitter*, se pronostica que para algún símbolo cuya posición sea mayor a la de 131.072 puedan ocurrir errores. Una solución para este problema es la implementación de un algoritmo de resincronización, dentro de la unidad de decisión, una vez que se han demodulado cierto número de símbolos.

Otro punto a destacar es la optimización de área (porcentaje de los recursos lógicos ocupados) y búsqueda de mayor rendimiento del sistema. Respecto al primer punto, la síntesis (efectuado en el programa ISE) del circuito implementado fue realizada considerando un perfil *balanceado* es decir, considerando un equilibrio entre *slices* ocupadas y máxima frecuencia de funcionamiento del sistema. Este perfil puede ser cambiado dependiendo de la meta que se desee alcanzar. Otra arista de este problema radica en la optimización de la implementación del filtro FIR, ya que este último es el bloque clave del sistema. Una optimización de los coeficientes y una posterior reducción en los recursos ocupados para implementar el multiplicador pueden ayudar en la tarea de minimización en el área ocupada por todo el sistema.

El segundo punto, la mejora en el desempeño del sistema, se puede alcanzar, al igual que en el caso anterior, con la optimización de los bloques, ya que el *delay* de las operaciones está relacionado con la cantidad de compuertas necesarias para efectuar los cálculos, y mientras menor *delay* se obtenga mayor es la frecuencia a la que puede funcionar el sistema. Sin embargo, existe otra limitante que se relaciona con el rendimiento de los conversores (DAC y ADC) involucrados en el proceso de transmisión y recepción de datos. Los integrados que incorporan las plataformas FPGA tienen especificaciones [35], que en este caso, no permiten ocupar relojes con frecuencias mayores a 50 [MHz] (esta es la justificación de la condición impuesta para (4.4) y (4.5)) lo cual reduce la máxima velocidad de transferencia que puede lograr el sistema. La solución ofrecida, para el caso en que el sistema requiera funcionar con una mayor tasa de transferencia, es ocupar conversores externos y comunicarlos con las plataformas a través de los pines destinados para un uso de propósito general que ellas incorporan.

4.2. Análisis general del sistema

El desarrollo del sistema de modulación y demodulación efectuada en este trabajo podría perfectamente haberse implementado usando pulsos rectangulares, pero esto constituiría la elaboración de un sistema que funcionaría de forma perfecta en canales con gran ancho de banda y además no significaría ningún avance en el uso eficiente de los recursos. Como se propuso en la introducción de este trabajo, cada vez es necesario utilizar anchos de banda más pequeños y lograr velocidades de transferencias más grandes. Esta es la razón de utilizar *pulsos de Nyquist* para la modulación, ya que a medida que los pulsos tengan la capacidad de "traslaparse" entre sí, las velocidades de transferencia aumentarán y los anchos de banda disminuirán, pues dado el mismo efecto de "traslape" la truncación de los pulsos infinitos teóricos usados para la comunicación puede hacerse de forma mas amplia. Todo esto se refleja, por ejemplo, en la utilización de modulacion QAM en la banda ancha común que se usa para acceder a Internet, y en particular, apunta también al uso y mejora de los sistemas de comunicaciones de forma inalámbrica.

Capítulo 5

Conclusiones

El correcto funcionamiento, mostrado en los capítulos anteriores, de un sistema de modulación PAM binario usando *pulsos de Nyquist*, implementado sobre plataformas FPGA indica que el objetivo principal de este trabajo ha sido alcanzado satisfactoriamente. El cumplimiento de esta meta significa a su vez, el cumplimiento de todos los objetivos específicos planteados al comienzo de este trabajo.

La investigación teórica previa permitió conocer, en un primer acercamiento, las formas de modulación existentes y su definición a partir de ecuaciones, el desempeño que tienen éstas de forma teórica a través del análisis de probabilidades de ocurrencia de errores y permitió conocer una forma de mejorar el rendimiento de estos sistemas a través de la eliminación teórica de la interferencia intersimbólica usando una familia de funciones conocida como *pulsos de Nyquist*.

La revisión de temas referentes a tratamiento digital de datos consituyó una novedad y significó una gran obtención de conocimientos en esta área. Métodos para implementar sumas y multiplicaciones binarias constituyen la base para poder desarrollar cualquier otra operación aritmética más compleja y por ende sistemas de más alto nivel.

Finalmente, la implementación como tal permitió la maduración de las habilidades para describir un elemento mediante lenguaje Verilog y la adquisición de conocimiento en el uso de los periféricos integrados en las plataformas FPGA ocupadas. El primer punto destaca, puesto que durante el proceso de programación la ocurrencia de errores en la sintaxis, comportamiento incorrecto en la simulación de los bloques programados y mal funcionamiento de los sistemas de demodulación, permitió adquirir un mejor nivel de comprensión de las características del lenguaje, en la forma de transformar fórmulas en algoritmos de programación y en particular, de la utilización del software ISE de Xilinx. Otro punto a destacar es la adquisición de conocimientos en el uso de los periféricos asociados a una plataforma FPGA, en particular, en este trabajo se logró utilizar, de forma satisfactoria, los conversosres análogo a digital, digital a análogo y amplificadores presentes en las plataformas Spartan3E y Spartan3AN.

Las trabajos futuros que podrían generar extensiones del sistema aquí presentado corresponden a:

- Implementación y desarrollo de un sistema más complejo de modulación: La investigación previa mostró los variados sistemas de modulación existentes. Algunos de ellos pueden mejorar la tasa de transferencia de símbolos pero teniendo como costo la necesidad de implementar funciones más complejas tanto como para la modulación como para la demodulación.
- Optimización de los bloques de aritmética binaria: La revisión del estado del arte de la implementación de modems en plataformas FPGA reveló distintas maneras de obtener optimizaciones de área y/o rendimiento en los bloques de aritmética binaria. Esto, se traduciría en una posible disminución de tamaño en las FPGA ocupadas, en el caso de optimizar en área, y la obtención de sistemas más rápidos en el caso de optimizar rendimiento.
- Implementación de un sistema de modulación sin filtros adaptados: Nuevamente, la revisión de la literatura sobre el estado actual de las implementaciones reveló gran cantidad de investigaciones que hacen referencia a formas de demodular sin usar filtros adaptados. Lo anterior se traduce en buscar formas eficientes de implementar digitalmente funciones como CORDIC o el desarrollo de un PLL digital.
- Generación de un sistema de modulación con datos ingresados en tiempo real: El sistema aquí presentado trabaja con memorias para almacenar datos, un aspecto a mejorar del sistema implementado corresponde a la obtención, modulación y demodulación de datos ingresados por un usuario en tiempo real a través de interfaces como por ejemplo, un teclado.

Bibliografía

[1] J. F. Wakerly, *Digital Design: Principles and Practices*, 3rd ed. Prentice Hall, 2000.

[2] J. G. Proakis, *Digital Communications*, 4th ed. McGraw-Hill, 2000.

[3] R. Blahut, *Digital Transmission of Information*. Addison-Wesley Publishing Company, 1990.

[4] T. Rappaport, *Wireless Communications: Principles and Practice*, 2nd ed. Prentice Hall, 2001.

[5] D. L. Jones, "Advanced digital signal processing," University of Illinois, Urbana-Champaign, 2001, classnotes for the course ECE451.

[6] A. Antoniou, *Digital Signal Processing*. McGraw-Hill, 2005.

[7] A. Chorevas, D. Reisis, and E. Metaxakis, "An efficient digital FIR filter design for 64 QAM," in *Proceedings of the Third IEEE International Conference on Electronics, Circuits, and Systems (ICECS '96)*, vol. 2, Oct. 1996, pp. 900–903.

[8] C. Dick and F. Harris, "Configurable logic for digital communication: Some signal processing perspectives," *IEEE Communications Magazine*, vol. 37, no. 8, pp. 107–111, Aug. 1999.

[9] C.-J. Chou, S. Mohanakrishnan, and J. B. Evans, "FPGA implementation of digital filters," in *Proceedings of the Fourth International Conference on Signal Processing Applications and Technology*, 1993, pp. 80–88.

[10] I. Bravo, R. Rivera, A. Hernández, R. Mateos, A. Gardel, and F. J. Meca, "Implementación de filtros FIR en FPGA's," in *VI Congreso de Tecnologías Aplicadas a la Enseñanza de la Electrónica (TAEE'04)*, Valencia, España, 2004.

[11] P. Kollig and B. M. Al-Hashimi, "FPGA implementation of high performance FIR filters," in *Proceedings of 1997 IEEE International Symposium on Circuits and Systems (ISCAS '97)*, vol. 4, Jun. 1997, pp. 2240–2243.

[12] J. Valls, M. M. Peiró, T. Sansaloni, and E. Boemo, "Design and FPGA implementation of digit-serial FIR filters," in *IEEE International Conference on Electronics, Circuits and Systems*, 1998, pp. 191–194.

[13] T. Sasao, Y. Iguchi, and T. Suzuki, "On LUT cascade realizations of FIR filters," *Proceedings of the 8th Euromicro Conference on Digital System Design (DSD'05)*, pp. 467–475, 2005.

[14] H. M. Eissa, K. Sharaf, and H. Ragaie, "ARCTAN differentiated digital demodulator for FM/FSK digital receivers," in *Proceedings of the 45th Midwest Symposium on Circuits and Systems (MWSCAS-2002)*, vol. 2, Aug. 2002, pp. II–200 – II–203.

[15] L. Ferguson, "FPGA-based FIR filter using bit-serial digital signal processing," Atmel Corporation, 1999, Application note. [Online]. Available: http://www.atmel.com/dyn/resources/prod_documents/DOC0529.PDF

[16] R. J. Andraka, "FIR filter fits in an FPGA using a bit serial approach," *Proceedings of the 3rd Annual PLD conference and exhibit*, pp. 30–31, 1993.

[17] H. Sam and A. Gupta, "A generalized multibit recoding of two's complement binary numbers and its proof with application in multiplier implementations," *IEEE Transactions on Computers*, vol. 39, no. 8, pp. 1006 –1015, aug 1990.

[18] S. He and M. Torkelson, "FPGA implementation of FIR filters using pipelined bit-serial canonical signed digit multipliers," in *Proceedings of the IEEE 1994 Custom Integrated Circuits Conference*, May 1994, pp. 81–84.

[19] J. M. Luzuriaga, P. Cayuela, S. Olmedo, and G. Gutiérrez, "Desarrollo de filtros FIR en FPGAs," *Revista Tecnología y Ciencia*, no. 3, pp. 9–17, 2.

[20] G. Goslin and B. Newgard, "16-tap, 8-bit FIR filter applications guide," Xilinx Inc., Nov. 1994, Application note. [Online]. Available: http://www.xilinx.com/appnotes/fir_filt.pdf

[21] P. Longa and A. Miri, "Area-efficient FIR filter design on FPGAs using distributed arithmetic," *International Symposium on Signal Processing and Information Technology*, pp. 248–252, 2006.

[22] W. Sen, T. Bin, and Z. Jim, "Distributed arithmetic for FIR filter design on FPGA," in *International Conference on Communications, Circuits and Systems (ICCCAS 2007)*, Jul. 2007, pp. 620–623.

[23] D. L. Maskell, "Design of efficient multiplierless FIR filters," *IET Circuits, Devices Systems*, vol. 1, no. 2, pp. 175–180, 2007.

[24] S. Yang, Z. Wu, and G. Ren, "Design and implementation of FPGA-based FSK IF digital receiver," in *Proceedings of the 1st International Symposium on Systems and Control in Aerospace and Astronautics (ISSCAA 2006)*, Jan. 2006, pp. 819–821.

[25] L. Fang, K. Xizheng, and L. Qiang, "Design and implement of OQPSK modulator based on FPGA," in *Proceedings of the 8th International Conference on Electronic Measurement and Instruments (ICEMI '07)*, Jul. 2007, pp. 4–929 – 4–933.

[26] S. Vallabhaneni, S. Attri, N. Krishman, S. Sharma, and R. C. Chauhan. Design of an all-digital PLL (ADPLL) core on FPGA. [Online]. Available: http://klabs.org/mapld04/abstracts/sharma_a.doc

[27] A. H. Khalil, K. T. Ibrahim, and A. E. Salama, "Design of ADPLL for good phase and frequency tracking performance," in *Proceedings of the Nineteenth National Radio Science Conference (NRSC 2002)*, 2002, pp. 284–290.

[28] R. Stefo, J. Schreiter, J.-U. Schlussler, and R. Schuffny, "High resolution ADPLL frequency synthesizer for FPGA-and ASIC-based applications," in *Proceedings IEEE International Conference on Field-Programmable Technology (FPT)*, Dec. 2003, pp. 28–34.

[29] E. Magdaleno, M. Rodríguez, A. Ayala, O. González, B. Rodríguez, and S. Rodríguez, "Diseño y síntesis de un modulador BPSK, QPSK y FSK reconfigurable," in *XX Simposium Nacional de la Unión Científica Internacional de Radio (URSI 2005)*, Gandía, Valencia, España, Sep. 2005, pp. 73–76.

[30] H. Azmi, H. Elsimary, M. I. Youssef, and A. Safwat, "FPGA based multi-standard configurable FSK demodulator," *Integration, the VLSI Journal*, vol. 36, no. 3, pp. 145–154, 2003.

[31] C. Dick and F. Harris, "FPGA QAM demodulator design," in *Proceedings of the Reconfigurable Computing Is Going Mainstream, 12th International Conference on Field-Programmable Logic and Applications*. London, UK: Springer-Verlag, 2002, pp. 112–121.

[32] G. Elamary, G. Chester, and J. Neasham, "A simple digital VHDL QPSK modulator designed using CPLD/FPGAs for biomedical devices applications," in *Proceedings of the World Congress on Engineering 2009 Vol I*, London, UK, Jul. 2009, pp. 376–381.

[33] F. Ahamed and F. A. Scarpino, "An educational digital communications project using FPGAs to implement a BPSK detector," *IEEE Transactions on Education*, vol. 48, no. 1, pp. 191–197, Feb. 2005.

[34] R. Andraka, A. Berkun, and A. C. Group, "FPGAs make a radar signal processor on a chip a reality," in *IEEE Proceedings of the Asilomar Conference on Signals, Systems and Computers*, 1999, pp. 559–563.

[35] *Spartan-3E Starter Kit Board User Guide*, Xilinx, Inc., Mar. 2006, version 1.0a. [Online]. Available: http://www.xilinx.com/support/documentation/boards_and_kits/ug230.pdf

[36] J. P. Deschamp, G. J. A. Bioul, and G. D. Sutter, *Synthesis of Arithmetic Circuits - FPGA, ASIC and embedded systems*. Wiley-Interscience, 2006.

[37] "Spartan-3E FPGA family: complete data sheet," Xilinx, Inc., Apr. 2008, product specification, version 3.7. [Online]. Available: http://www.xilinx.com/support/documentation/data_sheets/ds312.pdf

Apéndice A

Demostraciones

A.1. Demostración resultado (2.23)

La secuencia de símbolos enviada corresponde a

$$c(t) = \sum_{l=-\infty}^{\infty} a_l s(t - lT) \tag{A.1}$$

donde la señal $s(t)$ es un *pulso de Nyquist*. Aplicando a la señal anterior un filtro $q(t)$ se obtiene la señal $y(t)$ definida como

$$y(t) = \int_{-\infty}^{\infty} \sum_{l=-\infty}^{\infty} a_l s(t - lT) q(t - \tau)\, d\tau \tag{A.2}$$

Lo cual lleva a una expresión de la forma

$$y(t) = \sum_{l=-\infty}^{\infty} a_l g(t - lT) \tag{A.3}$$

donde $g(t)$ es la respuesta del filtro.

Se desea muestrear este resultado cada T segundos, luego

$$y(l^*T) = \sum_{l=-\infty}^{\infty} a_l g((l^* - l)T) \tag{A.4}$$

Se desea recuperar el símbolo a_l cada T segundos, por ende la elección trivial de $g(t)$ corresponde a

$$g(t) = \begin{cases} 1 & \text{si } l^* = \text{l} \\ 0 & \text{si } l^* \neq l \end{cases} \tag{A.5}$$

Luego, en el caso $l^* = l$, la expresión para $y(t)$ será

$$y(l^*T) = a_l \int_{-\infty}^{\infty} s(\tau)q(-\tau)\,d\tau. \tag{A.6}$$

Para que el valor de la integral sea 1 y además considerando que $s(t)$ está definida en $[0, T]$ y es una señal con energía igual a 1, la elección de $q(t)$ corresponde a $s(-t)$

A.2. Demostración resultado (2.31)

La expresión para la razón señal a ruido corresponde a

$$\frac{S}{N} = \frac{[\int (H + \varepsilon F)Pdf]^2]}{\int N\left|H\right|^2 df + \varepsilon \int N[HF^* + H^*F]df + \varepsilon^2 \int N\left|F\right|^2 df} \tag{A.7}$$

donde H y F corresponden a las transformadas de Fourier de las señales $h(t)$ y $f(t)$ respectivamente, P corresponde a la potencia espectral del pulso $s(t)$ y N a la potencia espectral del ruido.

Despreciando el último término del denominador y utilizando la siguiente aproximación para $|x| << 1$,

$$\frac{1}{a + bx} = \frac{1}{a}\left(1 - x\frac{b}{a} + ...\right) \tag{A.8}$$

se obtiene

$$\frac{S}{N} = \frac{[\int (H + \varepsilon F)Pdf]^2}{\int N\left|H\right|^2 df}\left(1 - \varepsilon\left[\frac{\int N[HF^* + H^*F]df}{\int N\left|H\right|^2 df}\right] + ...\right) \tag{A.9}$$

expandiendo el primer término,

$$\frac{S}{N} = \frac{[\int HPdf]^2 + \varepsilon[\int HPdf \int F^*P^*df + \int H^*P^*df \int FPdf] + \varepsilon^2[\int FPdf]^2}{\int N\left|H\right|^2 df}\Bigg(1- \varepsilon\left[\frac{\int N[HF^* + H^*F]df}{\int N\left|H\right|^2 df}\right] + ...\Bigg). \tag{A.10}$$

Multiplicando, agrupando términos y despreciando el término asociado a ε^3 se obtiene la siguiente expresión

$$\frac{S}{N} = \frac{[\int HPdf]^2}{\int N\left|H\right|^2 df} + \varepsilon\left[\frac{\int HPdf \int F^*P^*df + \int H^*P^*df \int FPdf}{\int N\left|H\right|^2 df} - \frac{\int HPdf]^2 \int N[HF^* + H^*F]df}{[\int N\left|H\right|^2 df]^2}\right] + \varepsilon^2\left[\frac{[\int FPdf]^2}{\int N\left|H\right|^2 df} - \frac{[\int HPdf \int F^*P^*df + \int H^*P^*df \int FPdf][\int N[HF^* + H^*F]df]}{[\int N\left|H\right|^2 df]^2}\right]. \tag{A.11}$$

la función $H(f)$ es el filtro adaptado, luego

$$H(f) = \frac{P^*(f)}{N(f)}, \tag{A.12}$$

reemplazando en la expresión (A.11) los términos asociados a ε se cancelan, luego

$$\frac{S}{N} \approx \left(\frac{S}{N}\right)_{max} - \varepsilon^2 A \tag{A.13}$$

con

$$A = \left[-\frac{[\int FPdf]^2}{\int N\left|H\right|^2 df} + \frac{[\int HPdf \int F^*P^*df + \int H^*P^*df \int FPdf][\int N[HF^* + H^*F]df]}{[\int N\left|H\right|^2 df]^2} \right] \tag{A.14}$$