ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

DISEÑO E IMPLEMENTACIÓN DE UN MODULADOR Y UN DEMODULADOR N-QAM EMPLEANDO XILINX ISE, SYSTEM GENERATOR Y SIMULINK SOBRE UNA TARJETA DE ENTRENAMIENTO BASADA EN UN FPGA DE XILINX.

PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

JIMÉNEZ NÚÑEZ SIXTO ALEJANDRO

e-mail: ing_sixtojimenez-epn@yahoo.es

PANCHI CAMPOS DIEGO ENRIQUE

e-mail: dpanchi1986@hotmail.com

DIRECTOR: Ph.D ROBIN ÁLVAREZ R.

e-mail: arobin7es@yahoo.es

Quito, Mayo 2011

DECLARACIÓN

Nosotros, Sixto Alejandro Jiménez Núñez y Diego Enrique Panchi Campos, declaramos bajo juramento que el trabajo aquí descrito es de nuestra autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que hemos consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedemos nuestros derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

_____ _____

Sixto Jiménez Diego Panchi

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Sixto Alejandro Jiménez Núñez y Diego Enrique Panchi Campos, bajo mi supervisión.

_____

Ph.D Robin Álvarez R.
DIRECTOR DEL PROYECTO

AGRADECIMIENTO

Agradezco al Ph.D Robin Álvarez, por su guía que ha sido uno de los pilares fundamentales para el desarrollo y culminación exitosa del presente Proyecto de Titulación.

A nuestros profesores, que durante toda la Carrera nos han impartido sus conocimientos, para formarnos profesionalmente.

A mis compañeros y amigos, con quienes se ha compartido una cantidad de experiencias académicas y personales por tanto tiempo.

De manera especial, a mi madre, por la confianza que tuvo en mis capacidades y por el apoyo irrestricto recibido hasta el final mi carrera universitaria. Además a mi hermana, por su constante y desinteresada voluntad de ayuda brindada en momentos difíciles, cada semestre que pasamos juntos.

Sixto Jiménez

AGRADECIMIENTO

Agradezco a mis padres por el apoyo que me han brindado en todo momento para lograr conseguir mis objetivos, además quiero agradecer a mis amigos por su incondicional amistad y todos los grandes momentos que vivimos juntos durantes estos años de vida universitaria.

Diego Panchi

DEDICATORIA

A mi madre, Norma

A mi hermana, Diana

A mi padre, Sixto

Sixto Jiménez N.

DEDICATORIA

Dedico este proyecto de titulación a mis padres por todo el esfuerzo y sacrificio que han realizado día a día.

Diego Panchi C.

ÍNDICE DE CONTENIDOS

CONTENIDO

CAPÍTULO I

FUNDAMENTOS TEÓRICOS DE LA MODULACIÓN DIGITAL EN CUADRATURA

CAPÍTULO II

FPGA (FIELD-PROGRAMMABLE GATE ARRAY)

CAPÍTULO III

DESCRIPCIÓN DEL SOFTWARE

CAPÍTULO IV

DISEÑO Y SIMULACIÓN EN SIMULINK CON XILINX SYSTEM GENERATOR

CAPÍTULO V

IMPLEMENTACIÓN EN HARDWARE

CAPÍTULO VI

CONCLUSIONES Y RECOMENDACIONES

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

El diseño del modulador y demodulador n-QAM se desarrolló empleando las herramientas de software: Xilinx ISE, System Generator y Simulink, y su implementación se la realizó sobre una tarjeta de entrenamiento basada en un FPGA de Xilinx. El punto de partida para el diseño fue un diagrama de bloques generalizado, en el cual se dividió la funcionalidad del sistema complejo en subsistemas sencillos. A continuación se probaron cada uno de los subsistemas independientemente, y una vez evaluada su correcta operación se procedió a unificarlos; para finalmente implementar el modulador y el demodulador.

Este trabajo se divide en seis capítulos, además de anexos en los que se incluye un manual de usuario para el uso del software e información de los dispositivos a utilizar.

En el capítulo 1 se realiza una breve descripción de un sistema de telecomunicaciones, además de la modulación digital QAM y la modulación de múltiples estados.

En el capítulo 2 se presenta la evolución de los dispositivos lógicos programables hasta llegar al FPGA. Se detalla la tarjeta de entrenamiento que contiene el FPGA de Xilinx.

En el capítulo 3 se analizarán los bloques de System Generator necesarios en el diseño, y el software de descarga iMPACT.

En los capítulos 4 y 5 se diseña una estructura generalizada de un modulador y demodulador QAM en Simulink con System Generator presentándose las pruebas que validan su funcionamiento, para luego implementarla en hardware.

El capítulo 6 contiene las conclusiones y recomendaciones del presente proyecto.

PRESENTACIÓN

El presente Proyecto de Titulación tiene como objetivo destacar una parte fundamental en los sistemas de telecomunicaciones, como son el modulador para la transmisión y el demodulador para la recepción, lo cual es vital para un futuro desarrollo de un sistema completo implementado en su totalidad sobre FPGAs.

Por otra parte, la nueva herramienta de hardware reconfigurable (FPGA) es muy útil cuando existen varios tipos de aplicaciones debido a que es un equipamiento versátil y su funcionalidad está descrita únicamente en el software que utiliza. Al compartir una misma infraestructura se brindan distintos servicios y se soportan diferentes estándares o formas de modulación.

El estudio realizado en el presente Proyecto de Titulación, ha sido desarrollado en base a la evolución de las comunicaciones digitales, aprovechando las ventajas con respecto a su contraparte analógica, y enfocado principalmente a su futura aplicación a la radio y televisión digital.

CAPÍTULO I

1. FUNDAMENTOS TEÓRICOS DE LA MODULACIÓN DIGITAL EN CUADRATURA

1.1. VISIÓN GENERAL DE UN SISTEMA DE TELECOMUNICACIONES

En la actualidad, los sistemas de comunicación analógica están siendo reemplazados por los sistemas de comunicación digital dadas la múltiples ventajas que estos presentan. Ventajas como facilidad de procesamiento, regeneración de la señal, multicanalización e inmunidad al ruido han hecho posible que se consiga un notorio desarrollo tanto en la robustez como en la fiabilidad de los sistemas de telecomunicaciones, permitiendo a su vez ofrecer nuevas y mejores aplicaciones al usuario.

Un sistema de telecomunicaciones constituye la totalidad de los mecanismos que proporcionan el enlace de la información entre la fuente y el destino. Un sistema de comunicaciones básico está formado por tres componentes esenciales: transmisor, canal de transmisión y receptor; esta estructura básica se puede apreciar en la figura 1.1.

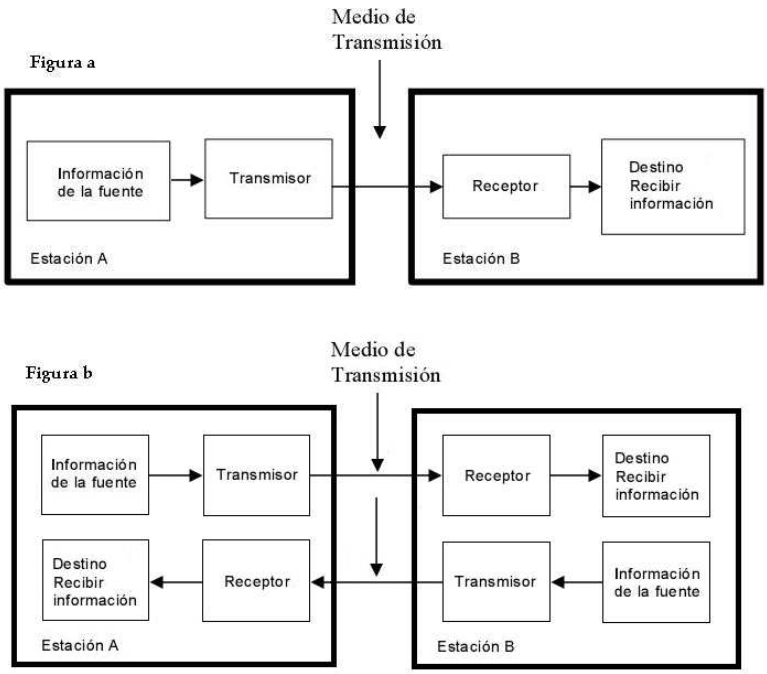


Figura 1.1. Sistema de Comunicaciones Básico (a) *Half-duplex*; (b) *Full-duplex*

En el proceso de transmisión, se debe tomar la información, adaptarla y finalmente transmitirla utilizando un canal o medio de transmisión. En la recepción se deben realizar los procesos inversos hasta restituir la información, considerando corregir las alteraciones y errores introducidos por la naturaleza de los canales de transmisión.

La adaptación de la señal al canal de transmisión está constituida por una serie de procedimientos que se los realiza dentro del transmisor, los cuales entre otros son frecuentemente la codificación y la modulación. La estructura de un transmisor se puede observar en la figura 1.2.

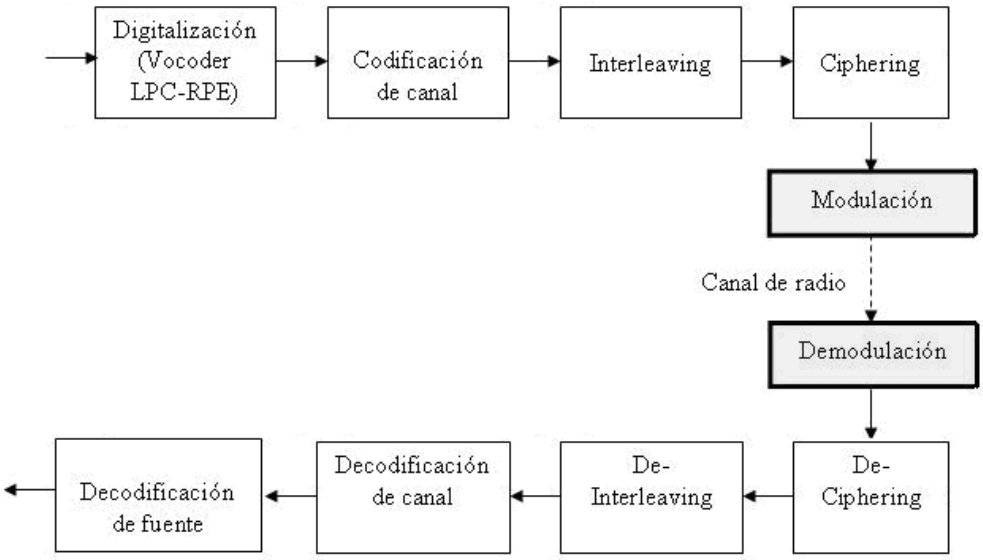


Figura 1.2. Estructura de un Transmisor de Comunicaciones

La modulación no es más que la modificación de uno o varios de los tres parámetros fundamentales que caracterizan a la señal portadora (amplitud, frecuencia y fase), siendo la modulación una de las etapas más importantes del procesamiento de una señal dentro de un sistema de telecomunicaciones, es por esto y por las exigencias de las nuevas aplicaciones que también ha sufrido una importante evolución abandonando sus tradicionales técnicas analógicas (AM y FM), pasando por sus similares digitales (ASK y FSK) hasta llegar a las técnicas de modulación en cuadratura como PSK y QAM, en la cuales se concentrará el desarrollo de éste capítulo.

Cabe mencionar el desarrollo de otras técnicas de modulación como TCM y OFDM, las cuales no se analizarán, debido a que están fuera del alcance de este proyecto de titulación.

Sin embargo, la modulación no sólo se clasifica por la naturaleza digital o analógica de la señal modulante o por el parámetro que contiene el mensaje en la señal modulada (amplitud, frecuencia o fase), sino que también se puede definir otra clasificación, en la que se toma en cuenta la relación lineal o no lineal entre el espectro producido por la señal modulada y el espectro del mensaje, generando así la modulación lineal o no lineal.

El objetivo de este proyecto de titulación es realizar la implementación de las etapas de modulación y demodulación de un Sistema de Telecomunicaciones sobre una tarjeta de entrenamiento basada en un FPGA, particularizando el sistema a la modulación digital QAM, sin embargo se dará una breve descripción de las técnicas básicas de modulación digital, para posteriormente pasar a un análisis de las técnicas de modulación digital en cuadratura y así particularizar a la modulación QAM.

1.2. TÉCNICAS BÁSICAS DE MODULACIÓN DIGITAL

En modulación digital se tienen tres técnicas básicas en concordancia con los parámetros antes mencionados que contienen la información: modulación por desplazamiento de amplitud (ASK), por desplazamiento de frecuencia (FSK) y por desplazamiento de fase (PSK). En todos estos casos, la señal modulada ocupa un ancho de banda centrado en torno a la frecuencia de la portadora.

Para permitir una clara observación de la variación de los parámetros que contiene la información en cada una de las modulaciones se realizarán simulaciones en el software Matlab, además, se presentarán las ecuaciones que rigen la señal modulada resultante en cada caso.

1.2.1. MODULACIÓN ASK

La figura 1.3 se ha obtenido a partir de una simulación realizada por medio de un script en el software MATLAB, en esta podemos apreciar una secuencia de datos [1 1 0 1 1 0 0 0 1], la señal portadora utilizada para la modulación, y la señal modulada ASK en el dominio del tiempo.

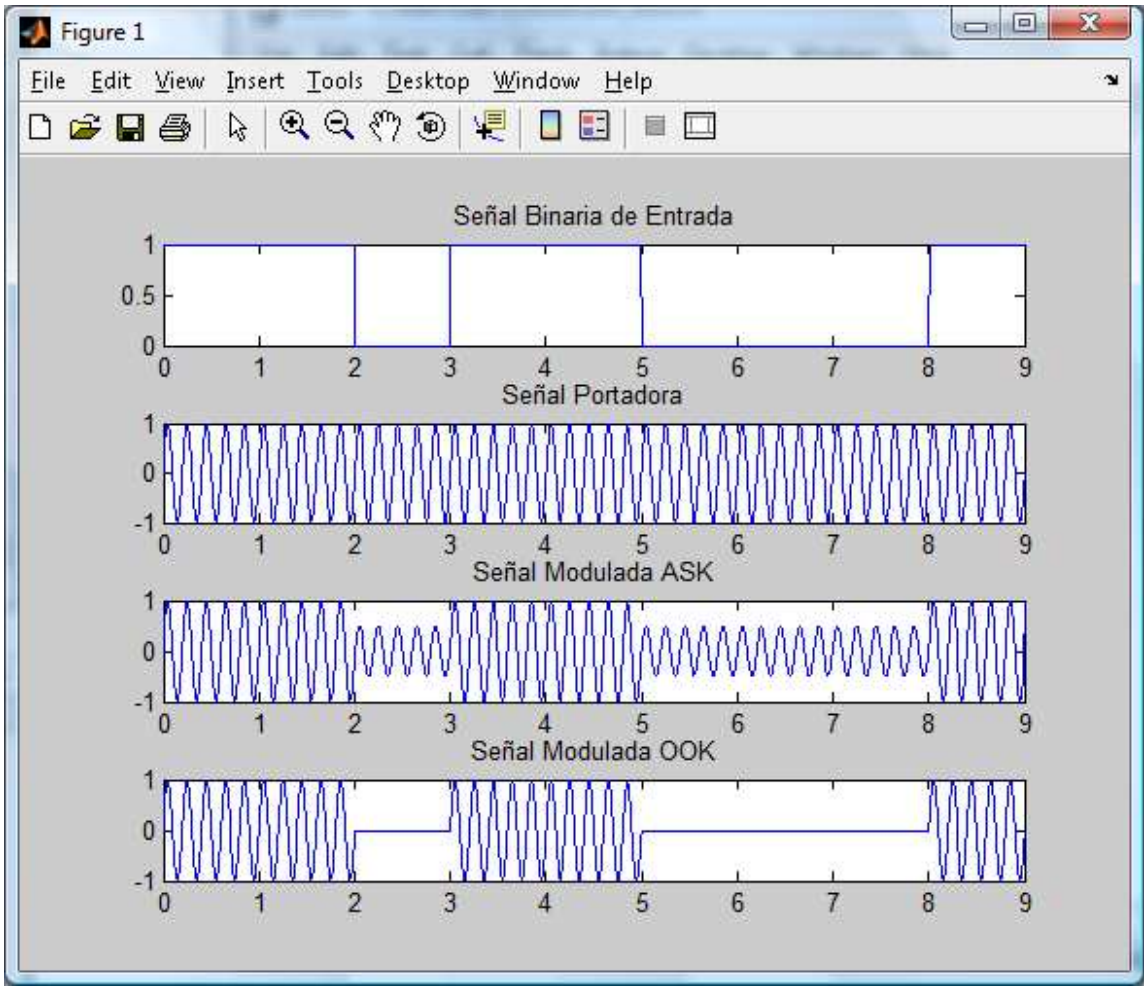


Figura 1.3. Simulación de la Modulación ASK

Es importante apreciar el cambio de amplitud en la señal modulada en el momento que se da una transición de un estado lógico a otro en la señal modulante. El cambio de amplitud para efectos didácticos y demostrativos se da de una amplitud de 1 en un estado 1L a una amplitud de 0.5 en un estado 0L. La señal modulada por lo tanto sería representada por la ecuación:

$$s(t) = \begin{cases} A_1 \cos(2\pi f_c t) \Rightarrow x = 1 \\ A_2 \cos(2\pi f_c t) \Rightarrow x = 0 \end{cases}$$

Donde:

s(t): Señal Modulada

A: Amplitud

f_c : Frecuencia de la portadora

x: estado lógico del bit de entrada

La última gráfica en la figura 1.3 corresponde a una variación de la modulación ASK que se conoce como OOK (On-Off Keying), en la cual se obtiene una señal modulada igual a cero cuando los bits del mensaje son 0L. La ecuación que representaría esta variante sería:

$$s(t) = \begin{cases} A_1 \cos(2\pi f_c t) \Rightarrow x = 1 \\ 0 \qquad\qquad\quad \Rightarrow x = 0 \end{cases}$$

1.2.2. MODULACIÓN FSK

Otra de las técnicas de modulación digital básicas es la FSK, en la cual la información está contenida en el cambio de frecuencia de la señal modulada, este cambio lo podemos apreciar en la figura 1.4, donde al igual que en ASK se realiza una simulación en MATLAB y se muestra en primer lugar los bits datos [1 1 0 1 1 0 0 0 1], seguidos por la señal portadora utilizada y finalmente la señal modulada en la cual se aprecia claramente el desplazamiento de frecuencia cuando se produce una transición de 1L a 0L en la señal binaria de entrada. La ecuación que representa la señal modulada FSK es:

$$s(t) = \begin{cases} A_1 \cos(2\pi (f_c) t) \Rightarrow x = 1 \\ A_1 \cos(2\pi (f_c + \Delta f) t) \Rightarrow x = 0 \end{cases}$$

Donde:

Δf : Desplazamiento de frecuencia

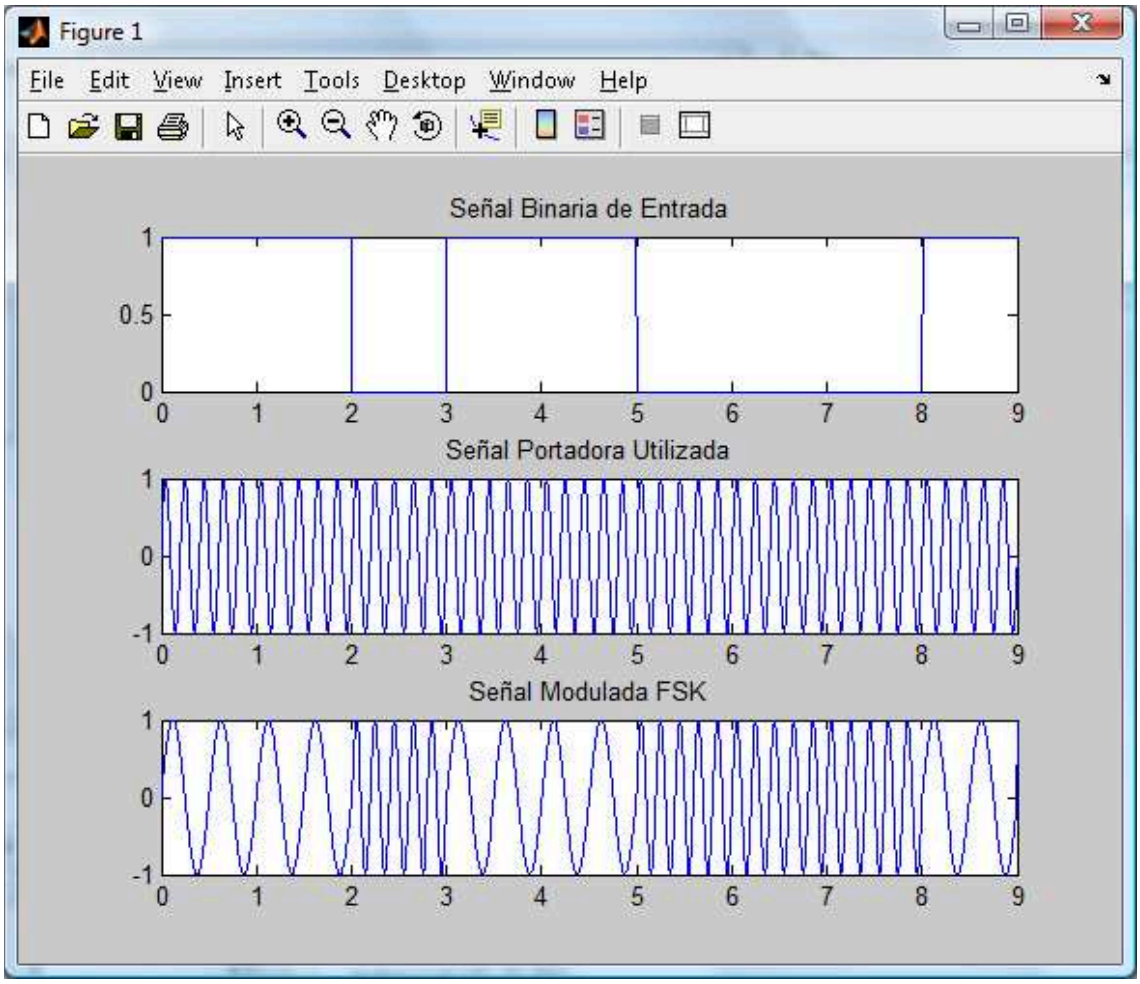


Figura 1.4. Simulación de la Modulación FSK

1.2.3. MODULACIÓN BPSK

Como última técnica básica de modulación digital se tiene la BPSK, en la cual la información se encuentra contenida en el desplazamiento de fase, es decir para un estado 0L la fase será una y para 1L la fase será otra. Esto se puede apreciar en la figura 1.5, en la cual se puede observar al igual que en los casos anteriores la señal binaria de entrada [1 1 0 1 1 0 0 0 1], la señal portadora utilizada para la modulación y la señal modulada apreciándose en esta última el desplazamiento de fase que en este caso es de 180º o π radianes ya que se está analizando únicamente la modulación PSK binaria es decir de dos estados.

La señal modulada BPSK se representaría de la siguiente manera:

$$s(t) = \begin{cases} A\cos(2\pi f_c t) = A\cos(2\pi f_c t) \Rightarrow x = 0 \\ A\cos(2\pi f_c t + \pi) = -A\cos(2\pi f_c t) \Rightarrow x = 1 \end{cases}$$

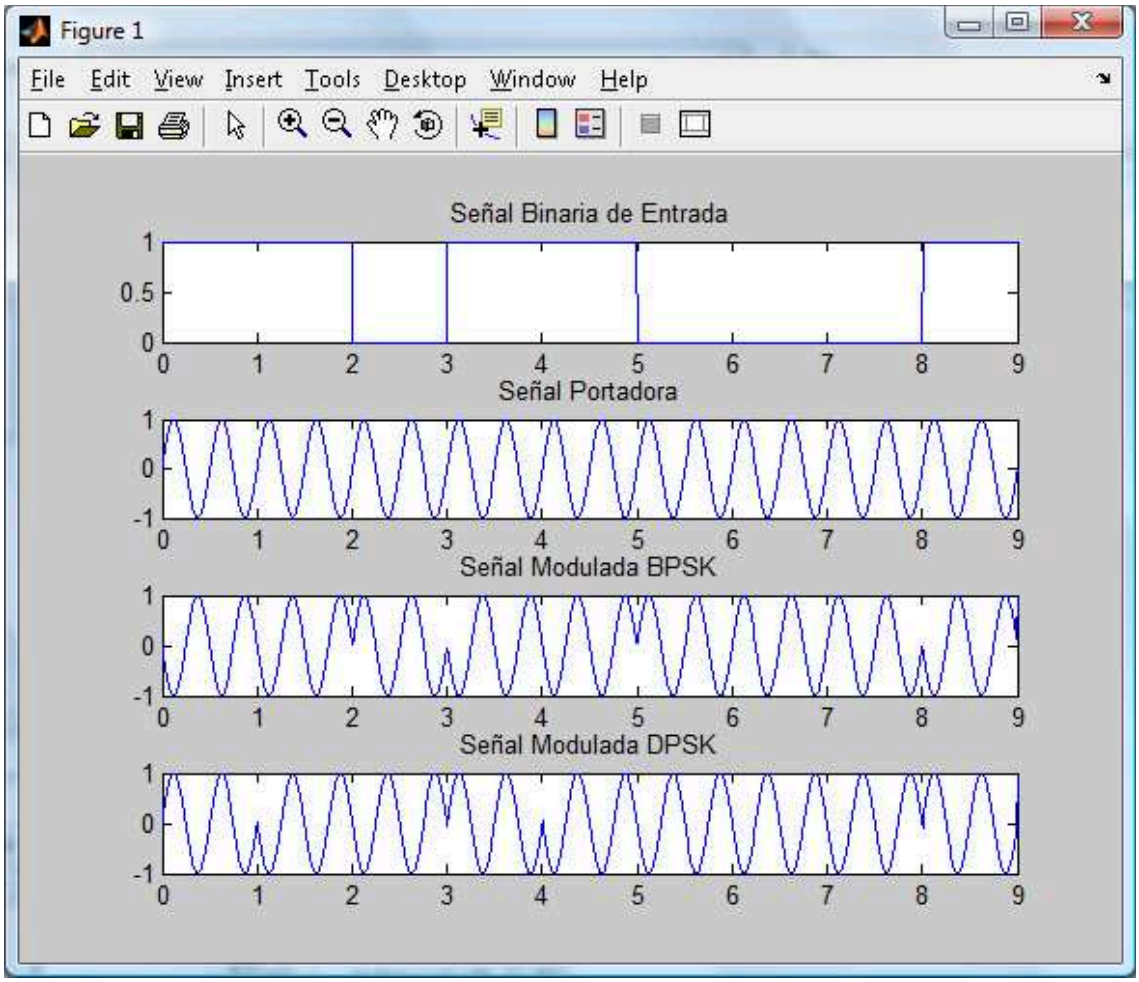


Figura 1.5. Simulación de la Modulación BPSK y DPSK

La señal DPSK es una alternativa a la modulación PSK, debido a que el desplazamiento de fase no se da sólo por una transición en la señal binaria de entrada sino que toma en cuenta el estado anterior, es decir, para un estado 0L en la señal binaria de entrada, se mantiene la fase del estado anterior de la señal modulada DPSK, mientras que para un estado 1L se produce un cambio de fase.

La principal ventaja de realizar modulación BPSK diferencial es que no se necesita recuperar la señal portadora y por lo tanto se puede utilizar en receptores no coherentes, además el detector decodifica la información digital basándose en diferencias relativas de fase.

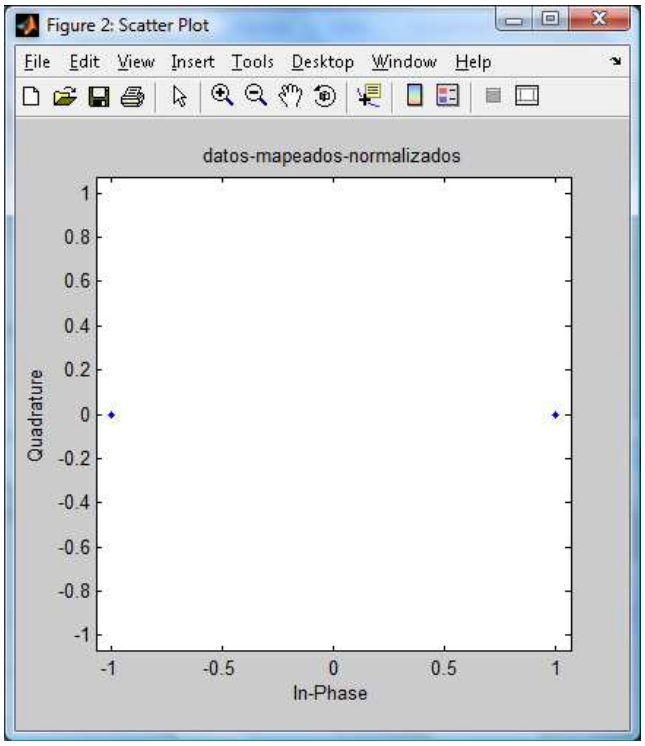


Figura 1.6. Diagrama de constelaciones de BPSK

La figura 1.6 muestra el diagrama de constelaciones obtenido en la simulación de la modulación BPSK.

En las secciones posteriores se analizará la modulación PSK multinivel tomándola como base para el desarrollo de las técnicas de modulación en cuadratura y así llegar finalmente a la modulación QAM, en la cual se centra el desarrollo de este proyecto de titulación.

1.3.MODULACIÓN DIGITAL PSK MULTINIVEL

Las técnicas de modulación multinivel se basan en lograr que cada elemento de señalización represente más de un bit y se emplean con el objetivo de conseguir un uso más eficaz del ancho de banda.

Con la modulación PSK multinivel se consigue obtener más de dos fases en la salida para representar los elementos de señalización o símbolos (combinación de más de un bit), este número de fases que se obtiene en la salida para diferenciar los símbolos creados debe ser una potencia de 2, es decir debe ser 2^n.

Sin embargo, este número no puede ser indefinido ya que la distancia entre los símbolos se vuelve cada más pequeña y esto provoca que el sistema sea más propenso a errores.

En el caso de la modulación PSK multinivel 16-PSK es un sistema raramente utilizado ya que la distancia angular entre los símbolos es de apenas 22.5^{o}, lo que a su vez significa que cada símbolo puede desplazarse 11.25^{o} en el medio de transmisión para mantener la integridad de la señal en la demodulación, tanto esta distancia angular como la que se presenta en todos los sistemas PSK multinivel se obtiene a partir de la ecuación:

$$\theta = \frac{360}{n}$$

Donde:

θ: Distancia angular entre los símbolos

n: Número de estados de n-PSK

El desplazamiento de fase que puede tener un símbolo en el canal de transmisión para mantener su integridad en demodulación se determina por medio de la ecuación:

$$\Delta\theta = \pm\frac{\theta}{2}$$

Donde:

$\Delta\theta$: Desplazamiento de fase que puede tener un símbolo para mantener su integridad.

Así también, el número de bits por símbolo necesarios para obtener los diferentes niveles de modulación se determina a través de la ecuación:

$$N = \log_2 n = \frac{\log n}{\log 2}$$

Donde:

N: Bits por símbolo

En la siguiente tabla se pueden ver los valores obtenidos a partir de las ecuaciones anteriores para los diferentes multiniveles de PSK.

n-PSK (multinivel PSK)	**n (# de fases posibles)**	**N (# de bits por símbolo)**	**θ Distancia entre los símbolos (°)**	**Δθ Desplazamiento angular entre símbolos (°)**
4-PSK	4	2	90	45
8-PSK	8	3	45	22.5
16-PSK	16	4	22.5	11.25

Tabla 1.1. Tabla de Valores de la Modulación n-PSK

Al igual que en las técnicas básicas de modulación digital se realizarán simulaciones en Matlab y se mostrará a partir de la figura 1.7 a la figura 1.12 tanto la señal en el tiempo, como los diagramas de constelaciones de las diferentes modulaciones n-PSK.

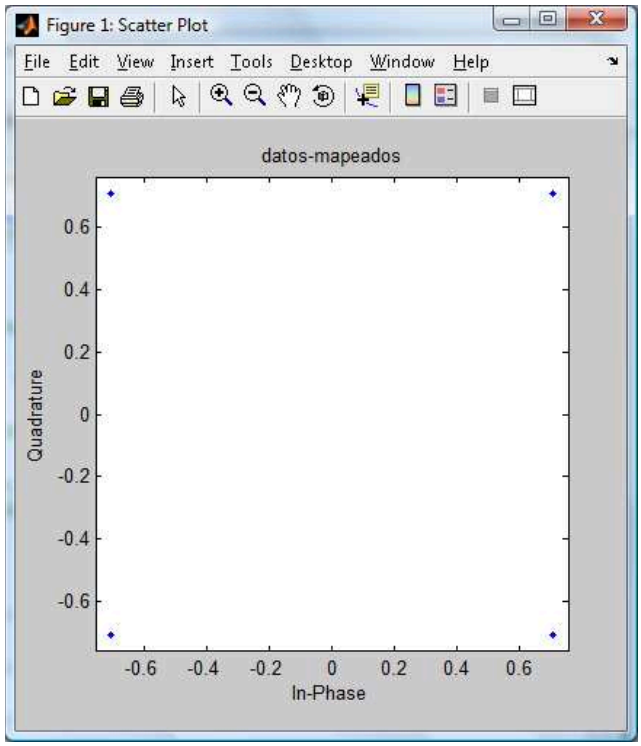


Figura 1.7. Diagrama de constelaciones de 4-PSK

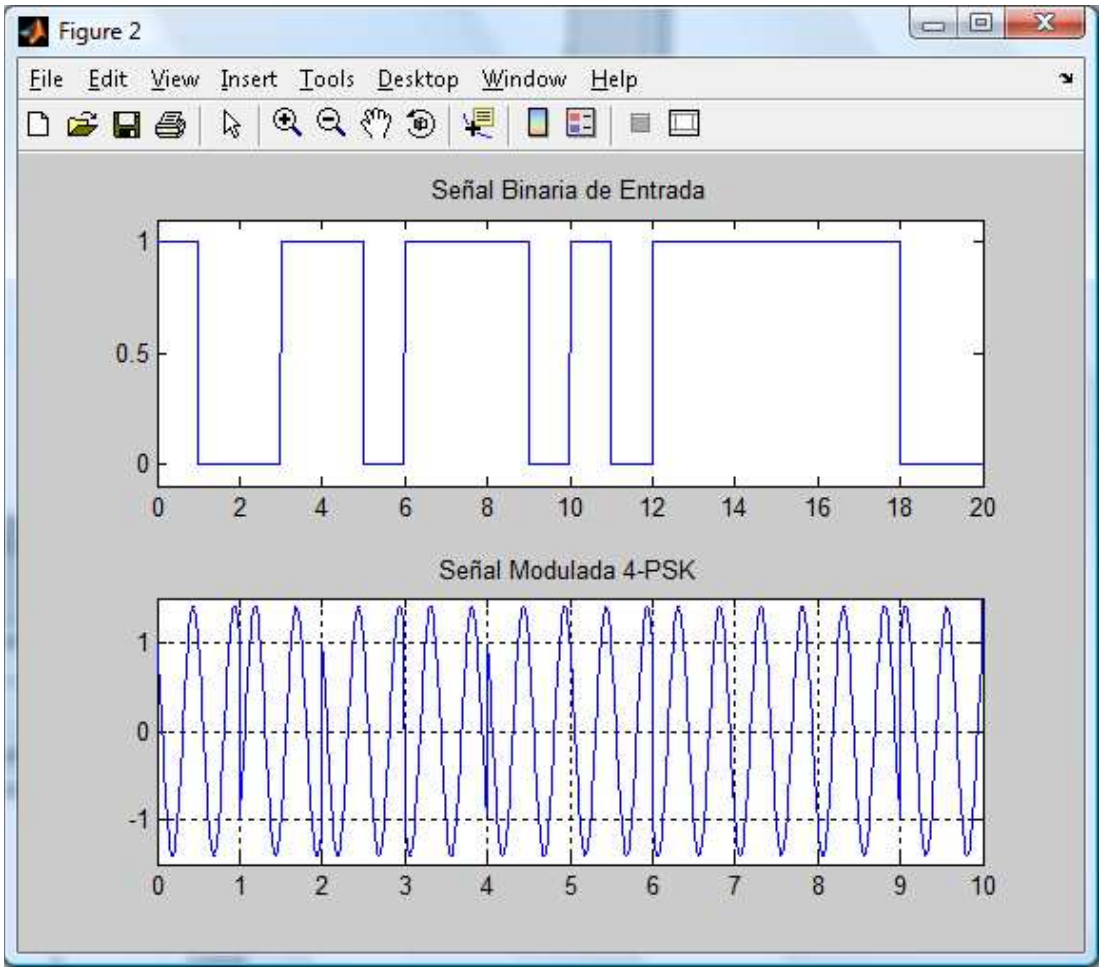


Figura 1.8. Señal Modulada 4-PSK

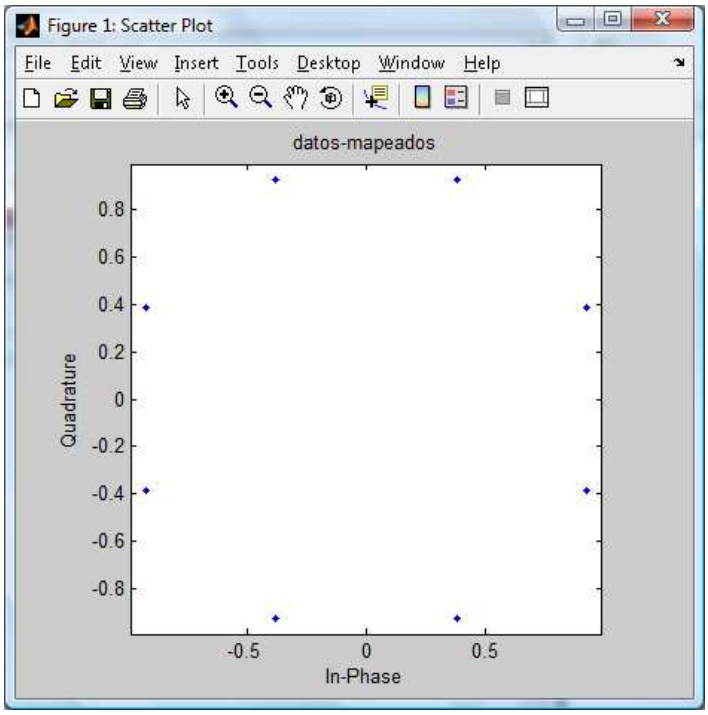


Figura 1.9. Diagrama de constelaciones de 8-PSK

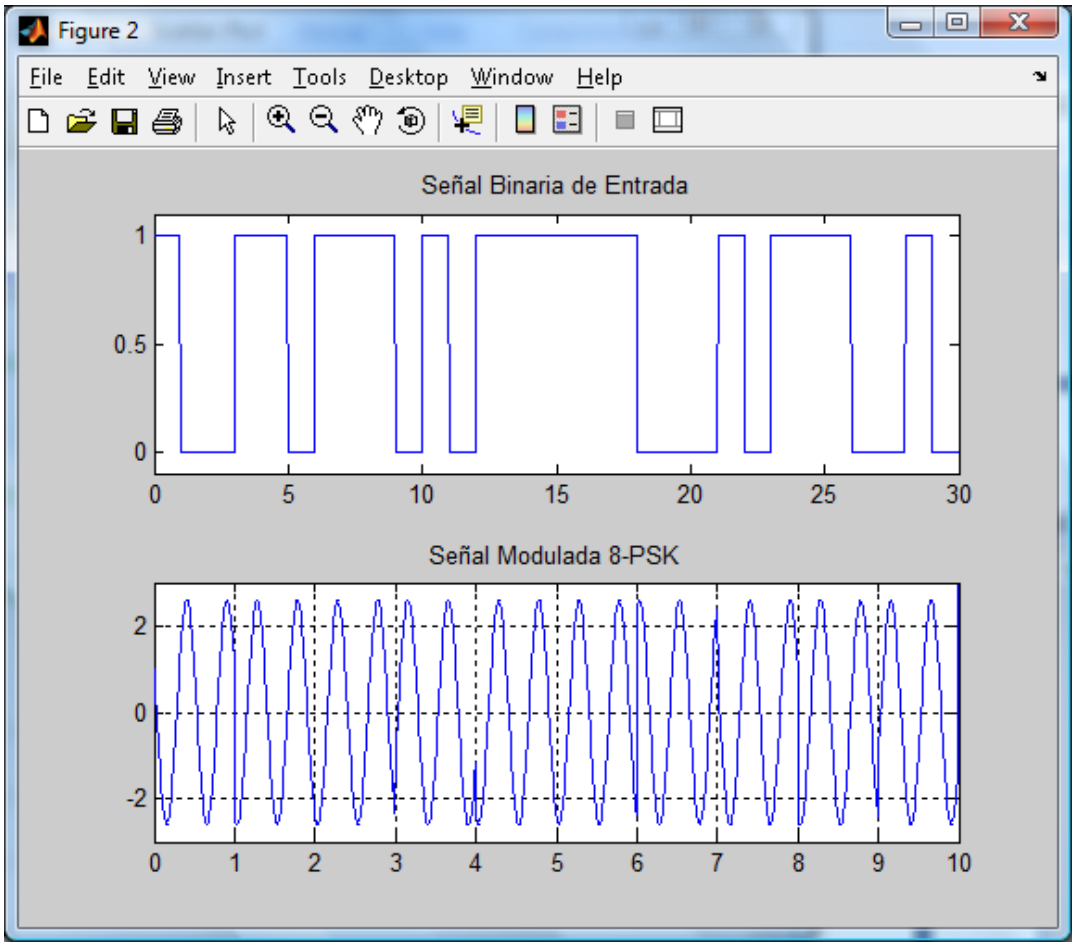


Figura 1.10. Señal Modulada de 8-PSK

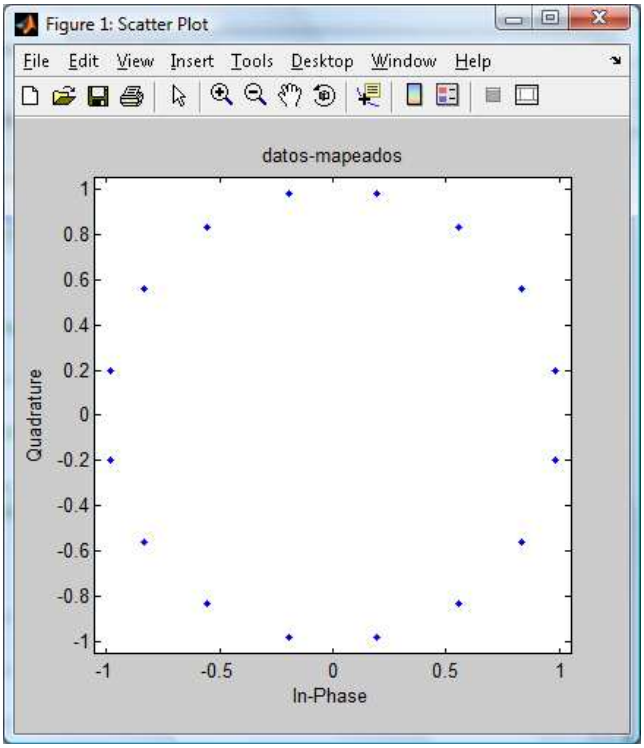


Figura 1.11. Diagrama de constelaciones de 16-PSK

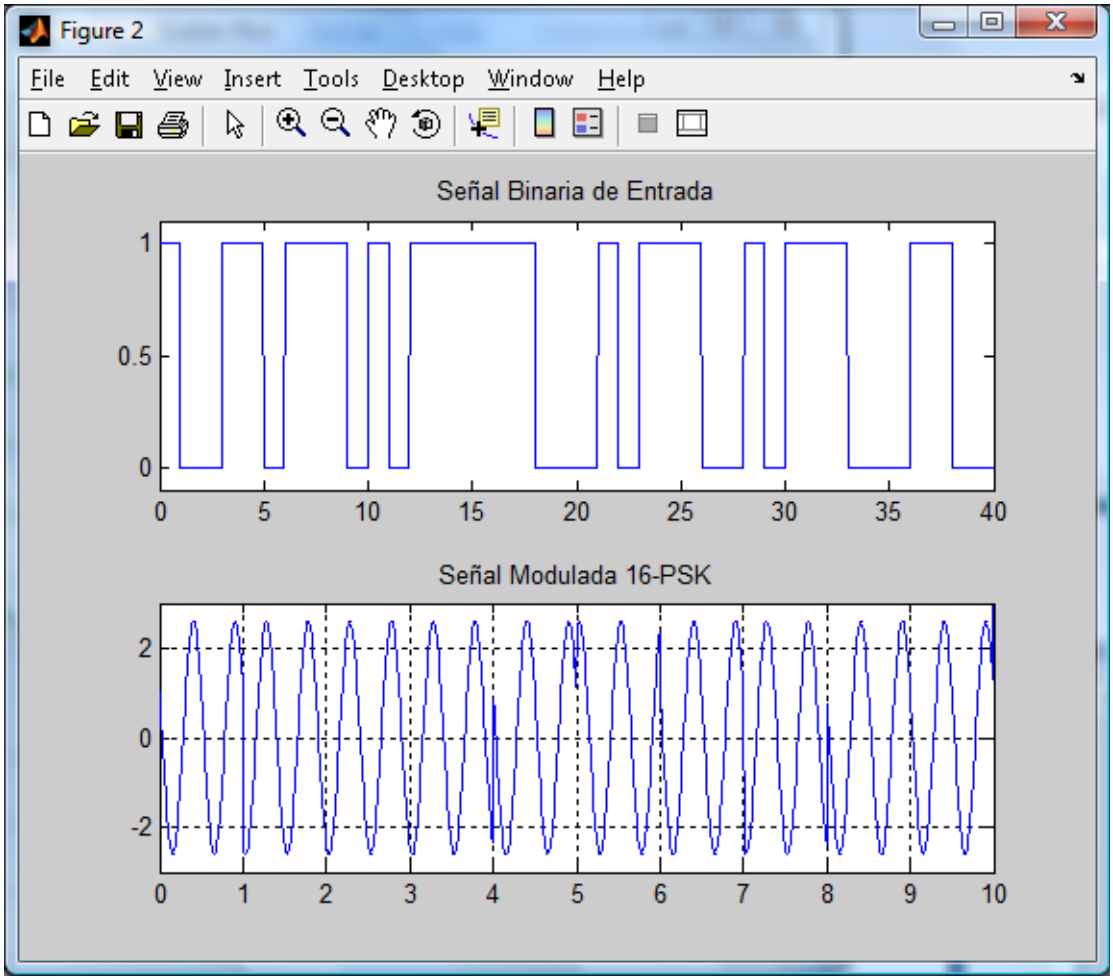


Figura 1.12. Señal Modulada 16-PSK

1.3.1. ESTRUCTURA DE UN MODULADOR n-PSK.

Para comprender la estructura del modulador PSK multinivel es necesario que se tenga claro cuando dos ondas están en fase o en cuadratura, es por esto que se da una breve definición de estos términos.

El que una onda este en fase o en cuadratura es un estado relativo en que se encuentran, en relación con la diferencia de fase entre las mismas, es decir si las dos ondas tienen una diferencia de fase igual a 0º, entonces se encuentran en fase, mientras que si presentan un desplazamiento de fase una respecto de la otra de 90º entonces están en cuadratura. A continuación se presenta un ejemplo con dos señales senoidales:

$$S_1 = A_1\, sen(2\pi ft)$$
$$S_2 = A_2\, sen(2\pi ft + d)$$

Donde:

d: Diferencia de fase entre S_1 y S_2

Si esta variable tiene un valor de 0, entonces las señales estarán en fase, caso contrario si esta variable tiene un valor diferente de cero existirá un desfase entre las señales, las mismas que estarán en cuadratura sólo cuando este desfase sea igual a 90º o (π/2) radianes. En la figura 1.13 se pueden apreciar los dos estados relativos antes descritos, así como también el estado relativo de oposición de fase, en el cual se tiene una diferencia de fase de 180º o π radianes.

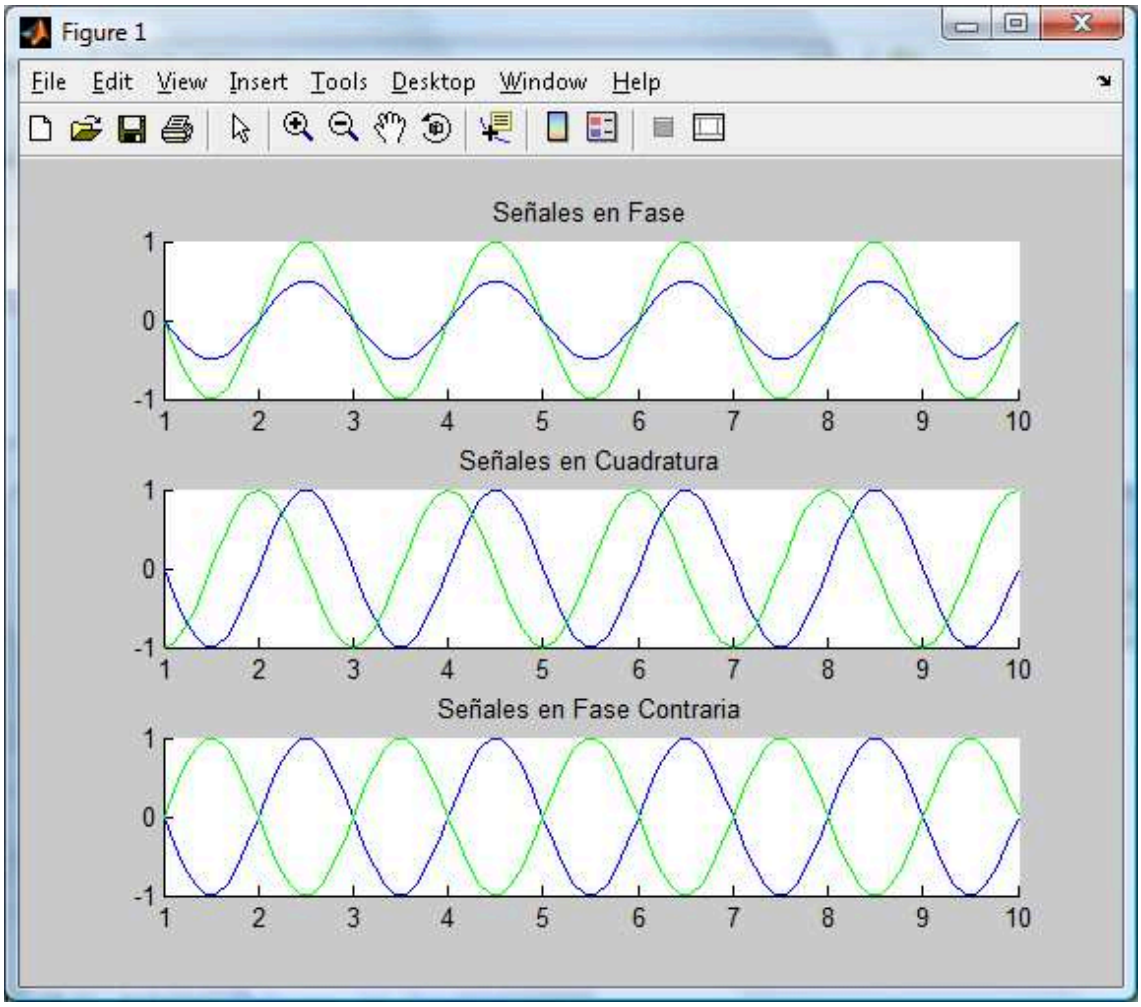


Figura 1.13.Estados Relativos de Ondas

En la figura 1.14 se puede observar el diagrama de bloques de un modulador PSK multinivel, ahí se puede reconocer etapas importantes para lograr esta modulación, como son: el conversor serie-paralelo, un oscilador para generar la portadora, el desplazamiento de fase de 90º, los multiplicadores para generar la señal modulada en los canales I (en fase) y Q (en cuadratura), el sumador para obtener la modulada con la adición de las dos señales de cada canal.

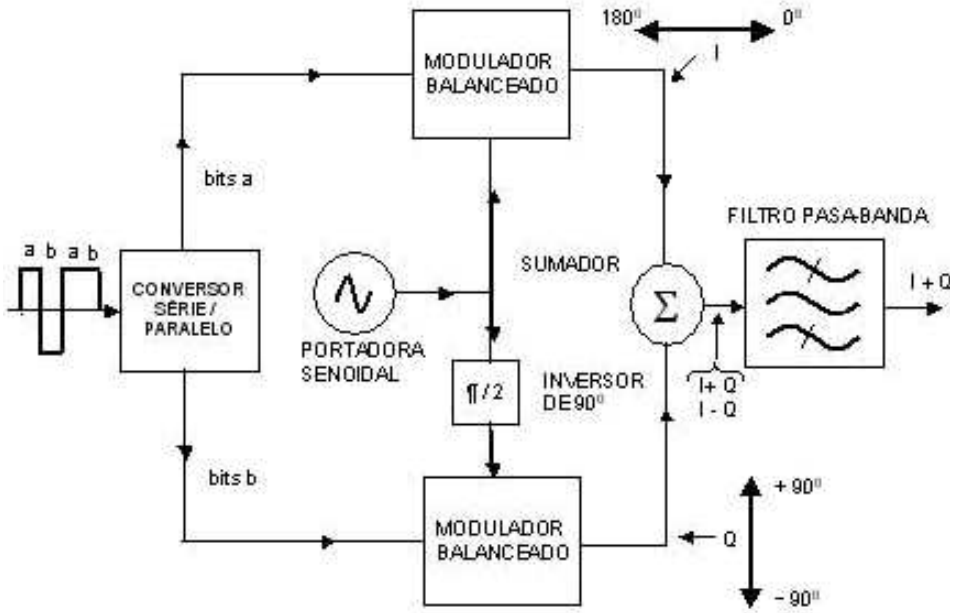


Figura 1.14. Estructura de un Modulador PSK multinivel

1.4. MODULACIÓN DIGITAL QAM

La técnica de modulación digital de amplitud en cuadratura (QAM), es una técnica en la cual el mensaje no está contenido únicamente en la variación de fase como en la modulación PSK antes descrita, debido a que en este caso la información se contiene además en las variaciones de amplitud que se generan en este tipo de modulación.

La modulación QAM se basa en la transmisión de dos mensajes independientes por un único camino. Esto se consigue modulando una misma portadora, desfasada 90º entre uno y otro mensaje. Esto supone la formación de dos canales ortogonales en el mismo ancho de banda, con lo cual se mejora en eficiencia de

ancho de banda. En la figura 1.15 se puede observar el diagrama de bloques de un modulador QAM.

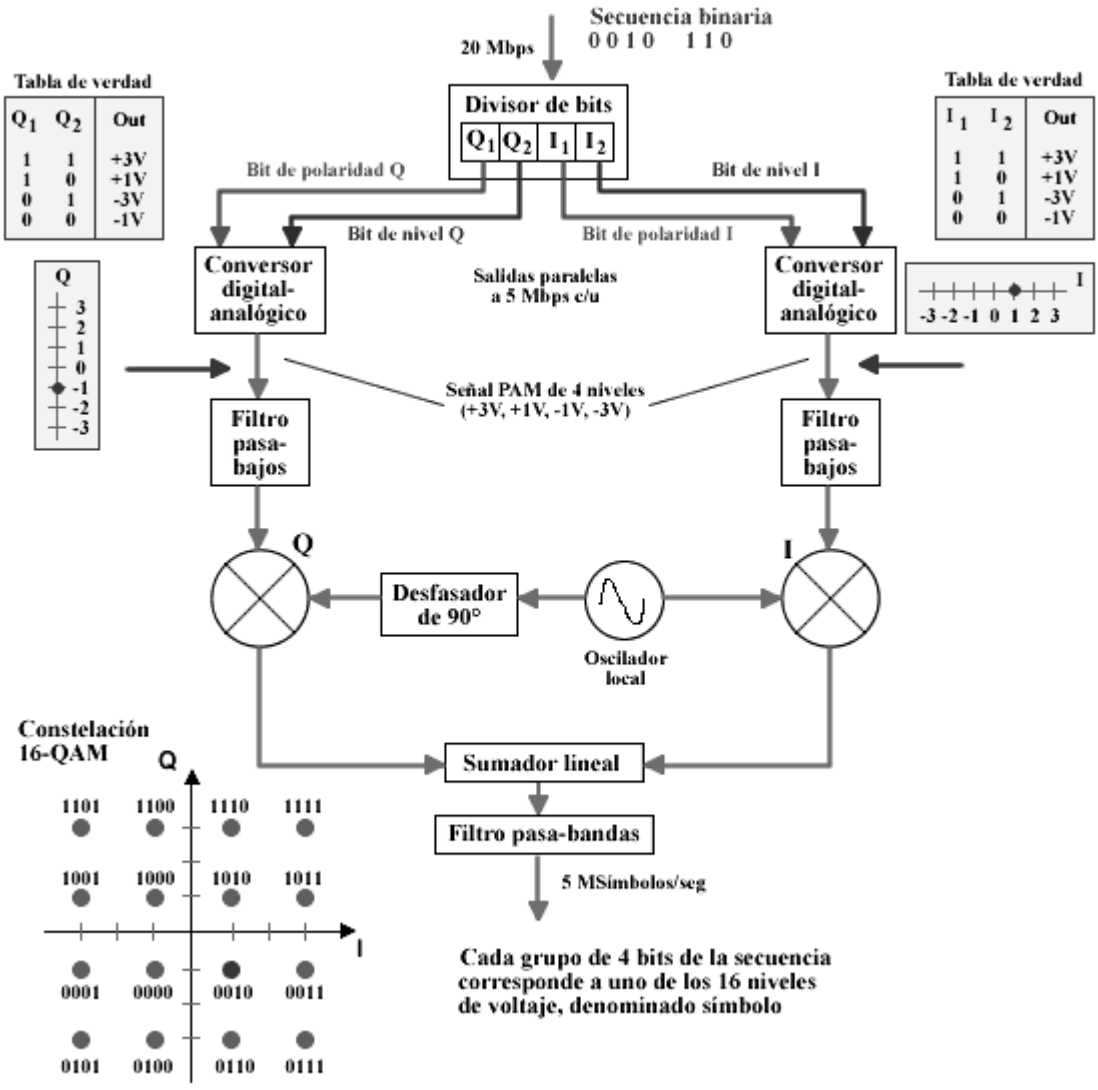


Figura 1.15. Diagrama de Bloques de un Modulador QAM

1.4.1. MODULACIÓN DIGITAL QAM MULTINIVEL

La modulación digital QAM es por naturaleza multinivel, ya que su menor número de estados es cuatro en donde sólo se presentan cambios de fase, y su máximo valor dependiendo de la aplicación puede llegar a ser hasta 256, en teoría se puede llegar a incrementar este número de estados, sin embargo mientras más estados se incremente es mayor también la probabilidad de error que se presenta.

Tomando esto como premisa se analizará únicamente la modulación 4-QAM, 16-QAM y 64-QAM.

Para la modulación QAM se debe analizar el número de bits por símbolo necesarios para derivar la mitad hacia cada canal (en fase y cuadratura), este número de bits está determinado por la ecuación:

$$N = \log_2 n,$$

Dado el número de bits por símbolo tanto la velocidad de señal como las consideraciones del ancho de banda en la modulación multinivel QAM son diferentes a la señal de entrada y se determinan por medio de las siguientes ecuaciónes.

$$Vs = \frac{Vin}{N},$$

Donde:

Vin: Velocidad de entrada de la señal binaria

Vs: Velocidad de señal a la salida del modulador.

$$AB = \frac{f_b}{N},$$

Donde:

f_b: Frecuencia de la señal binaria de entrada

AB: Ancho de banda de la señal modulada

Otro parámetro importante en la modulación QAM multinivel es la distancia existente entre puntos adyacentes la misma que se representa por:

$$d = \frac{\sqrt{2}}{L-1},$$

Donde:

d: Distancia entre los símbolos adyacentes QAM.

L: Número de niveles en cada eje.

En la siguiente tabla se presentará un resumen de valores de n-QAM:

n-QAM (multinivel QAM)	n (# de estados posibles)	N (# de bits por símbolo)	d Distancia entre los símbolos	Vs Velocidad de Señal (Baudios)	AB Ancho de Banda (Hz)
4-QAM	4	2	$\sqrt{2}$	$\frac{Vin}{2}$	$\frac{f_b}{2}$
16-QAM	16	4	$\frac{\sqrt{2}}{3}$	$\frac{Vin}{4}$	$\frac{f_b}{4}$
64-QAM	64	6	$\frac{\sqrt{2}}{7}$	$\frac{Vin}{6}$	$\frac{f_b}{6}$

Tabla 1.2. Tabla de Valores de la Modulación n-QAM

A continuación en las figuras desde la 1.16 hasta la 1.21 se mostrarán tanto las señales moduladas en el tiempo, como los diagramas de constelaciones obtenidos por medio de simulaciones realizadas en Matlab.

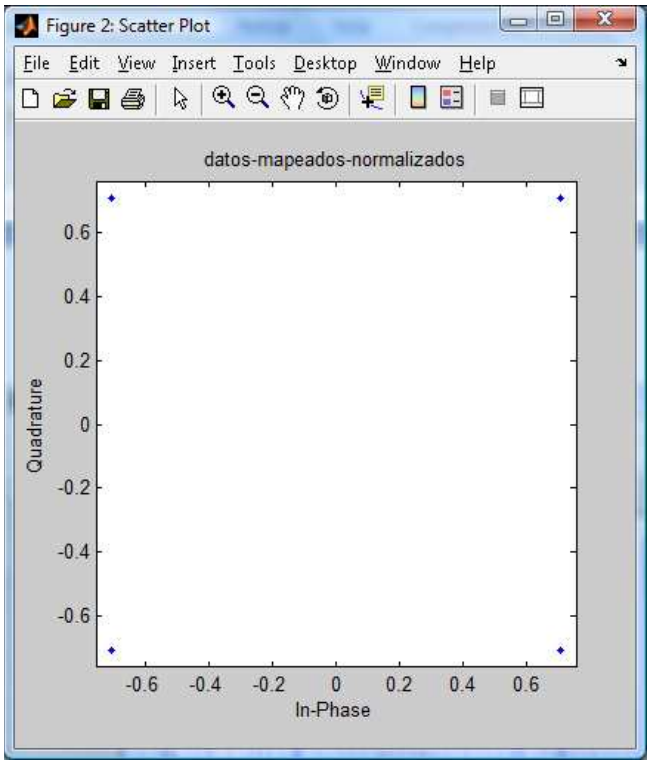


Figura 1.16. Diagrama de Constelaciones de 4-QAM

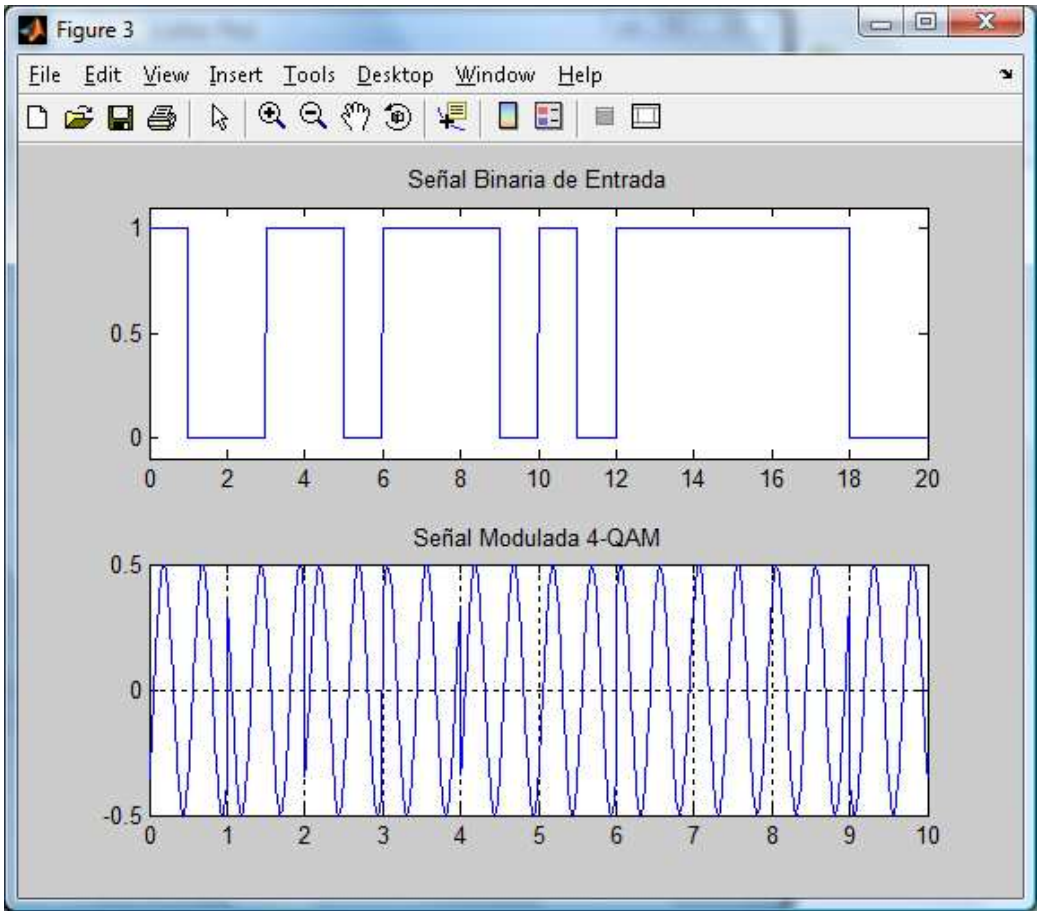


Figura 1.17. Señal Modulada 4-QAM

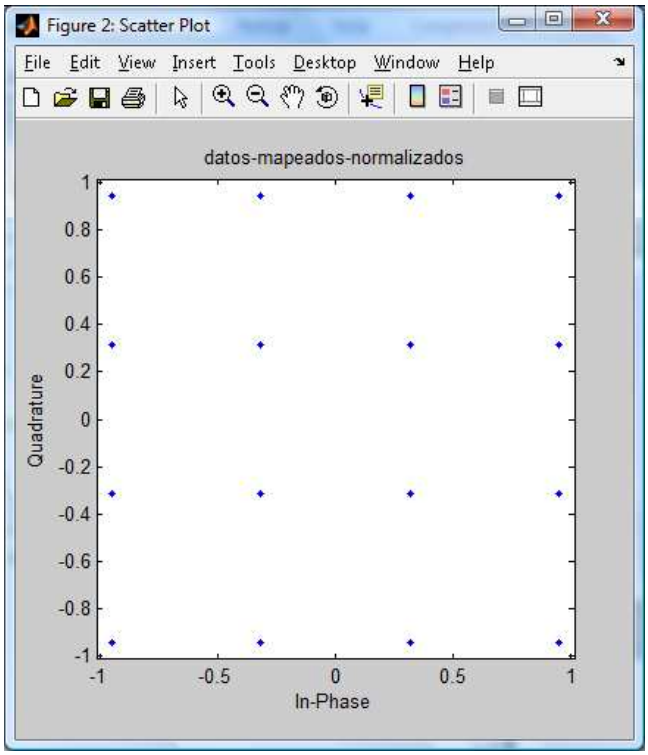


Figura 1.18. Diagrama de Constelaciones de 16-QAM

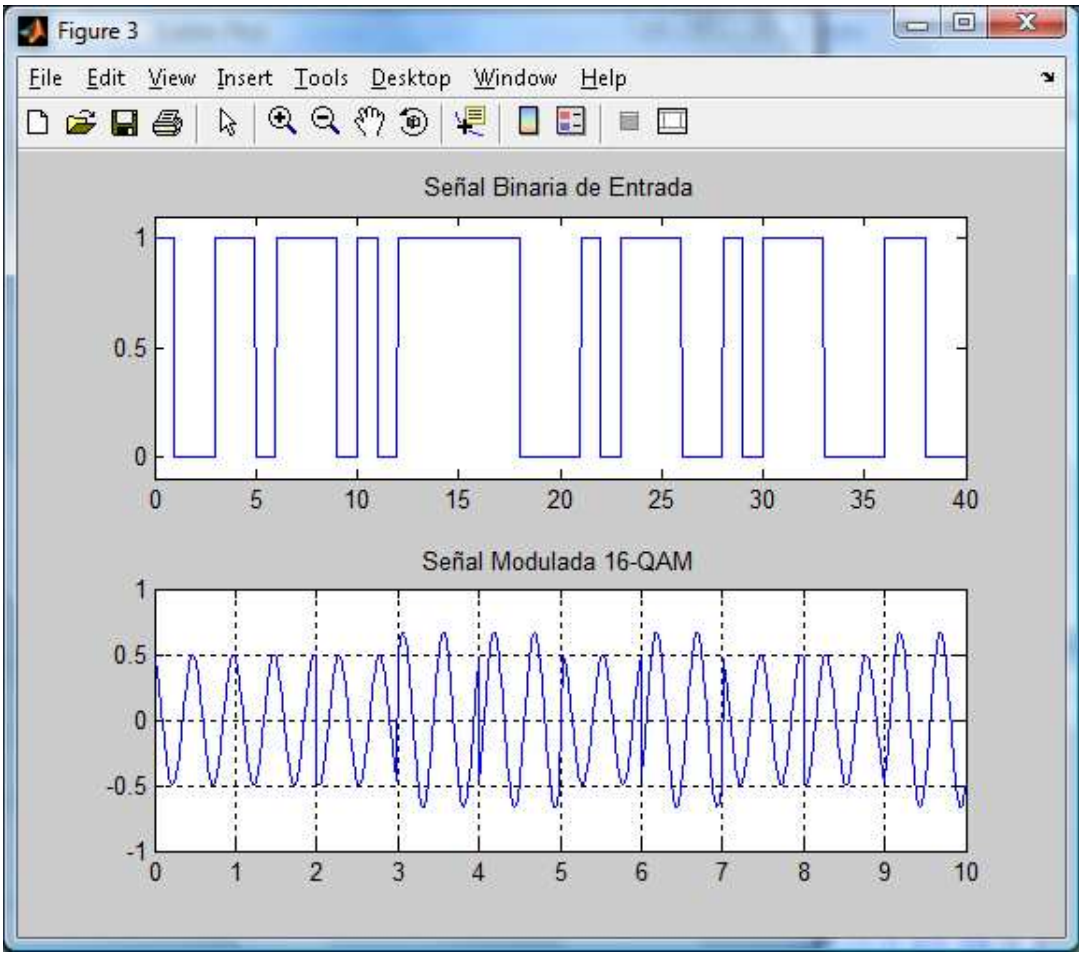


Figura 1.19. Señal Modulada 16-QAM

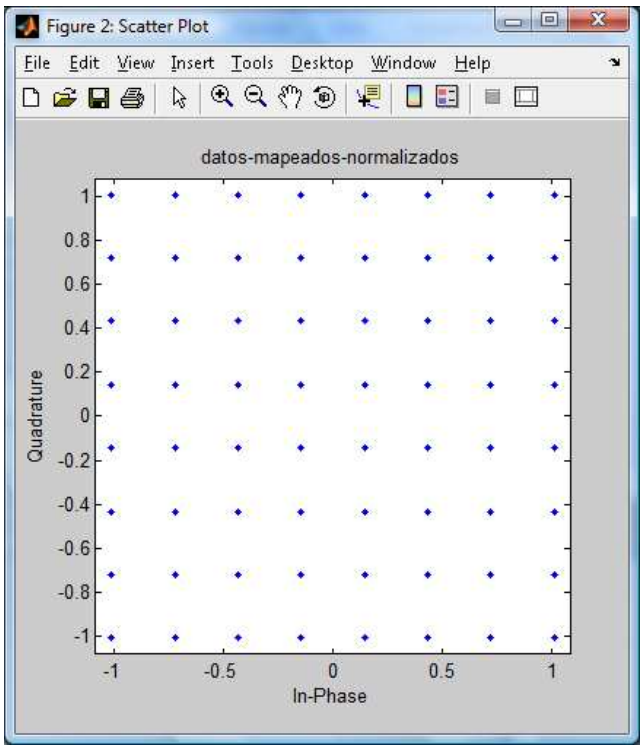


Figura 1.20. Diagrama de Constelaciones de 64-QAM

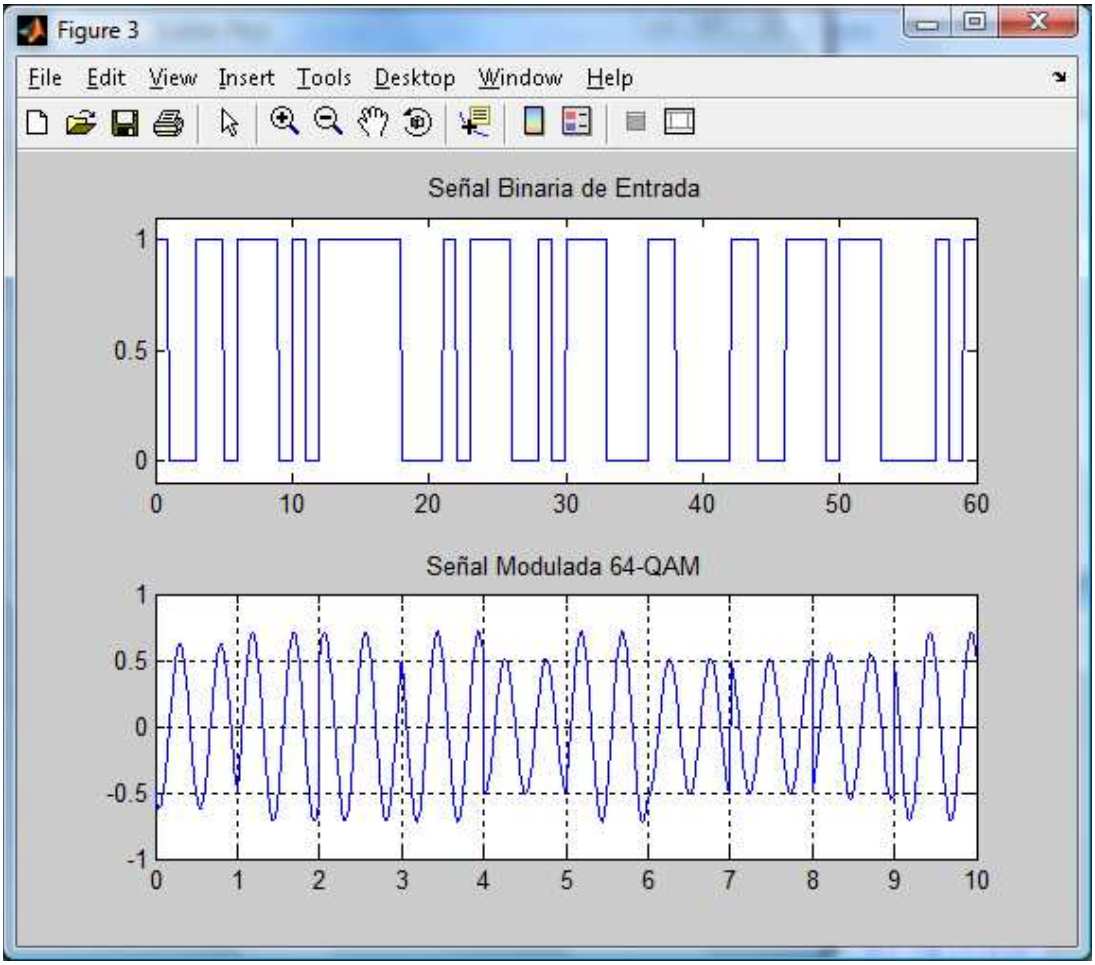


Figura 1.21. Señal Modulada 64-QAM

1.5.COMPARACIÓN DE PSK CON QAM

Una vez definidas las técnicas de modulación PSK y QAM es importante realizar una comparación de las mismas con el fin de determinar similitudes y diferencias así como también ventajas y desventajas de una u otra dependiendo de la aplicación para la que se utilice.

La primera similitud como ya se expuso anteriormente es que las dos son técnicas de modulación digital en cuadratura y que las dos permiten generar modulaciones n-arias es decir modulaciones multinivel.

Los sistemas de modulación 4QAM y QPSK son prácticamente los mismos ya que en este caso la modulación QAM no presenta variaciones de amplitud, es decir la

modulación se da únicamente en fase lo que a su vez produce que tanto el diagrama de constelaciones como la señal en el tiempo sea la misma para los dos casos.

A pesar de que en los dos casos de modulación la distancia entre los símbolos adyacentes es cada vez menor al aumentar 'n' el número de estados de la modulación, lo que a su vez provoca que aumenten los errores; en el caso de la modulación QAM esta distancia es mayor como se evidencia en los diagramas de constelaciones y por lo tanto la tasa de bits errados de esta modulación respecto a PSK es menor lo que representa la ventaja más notoria de QAM. Es así que la modulación PSK multinivel es usada raramente hasta 16PSK, mientras que sistemas de modulación 64QAM e incluso de mayor nivel son comúnmente utilizados en muchas aplicaciones. La diferencia entre la tasa de bits errados en uno y otro caso se puede apreciar en la figura 1.22.

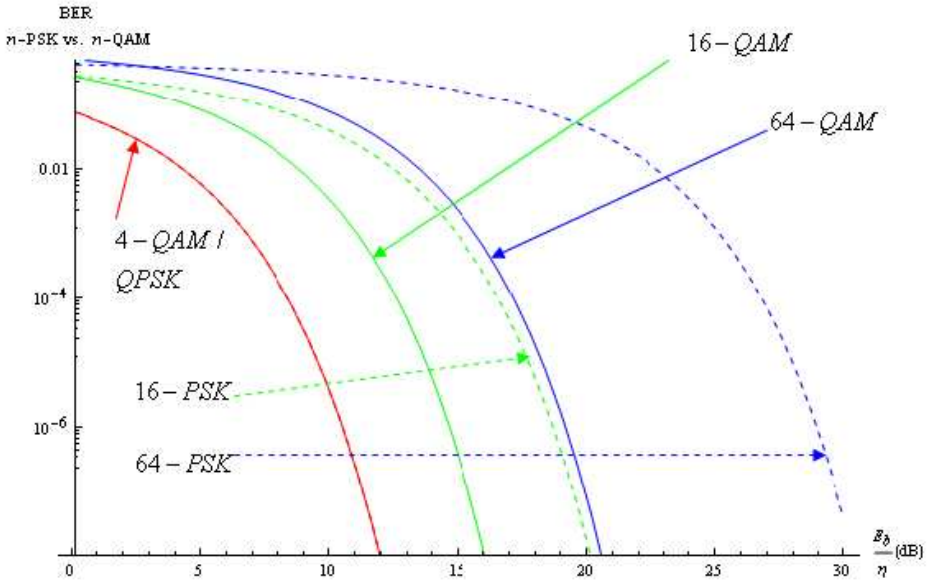


Figura 1.22. Comparación de BER entre n-PSK y n-QAM

Un parámetro determinante al elegir el método de modulación a implementarse es la complejidad del sistema lo que se traduce en costo. La complejidad de uno u otro sistema aumenta generalmente con el incremento de ciertas condiciones

como la eficiencia de ancho de banda. En la figura 1.23 se muestra una comparación de los diferentes tipos de modulación en relación a su complejidad.

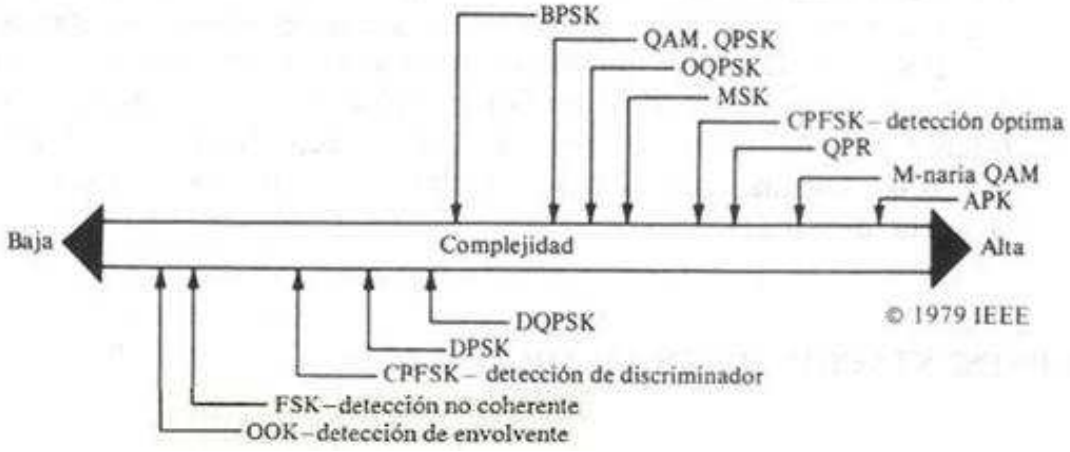


Figura 1.23. Comparación de BER entre n-PSK y n-QAM

1.6. ESTRUCTURA DE UN DEMODULADOR QAM.

La demodulación es el proceso a través del que se recupera la señal binaria de datos, para obtener una correcta recuperación de la información se debe seguir diferentes etapas, las cuales se observan en el diagrama de bloques de la figura 1.24.

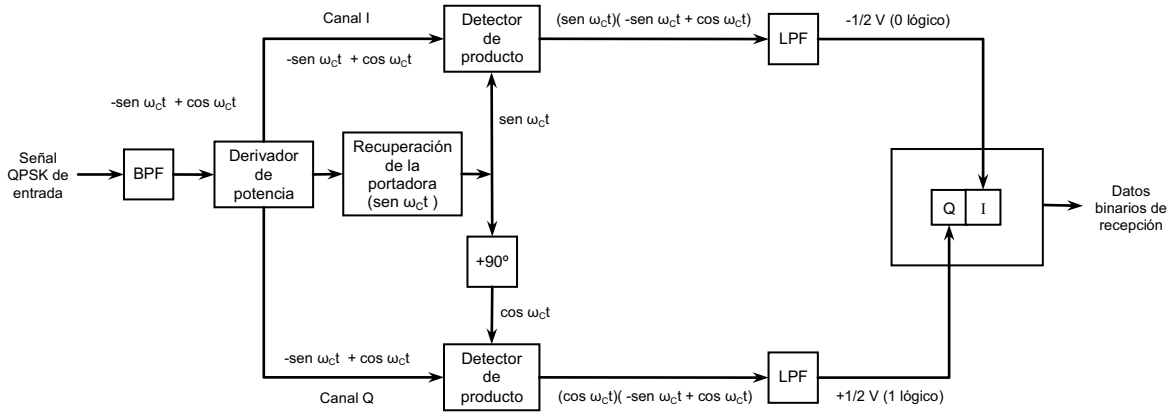


Figura 1.24. Diagrama de Bloques Demodulador

De la estructura del diagrama de bloques se puede observar que las etapas a cumplir son contrarias a las que se realizaron en la modulación, es decir se debe recuperar la señal portadora, filtrar las altas frecuencias, convertir de multinivel a señal binaria y finalmente unificar los bits obtenidos de los dos canales, para

obtener la señal de datos recuperada. Un factor clave a tomar en consideración en la recuperación de la señal de datos es la naturaleza ruidosa de los canales de transmisión, lo que obliga a tener algoritmos de decisión para detección y corrección de errores, de este tema se realizará una breve descripción a pesar de que el mismo no está dentro del alcance de este proyecto de titulación debido a que la transmisión electromagnética no forma parte del sistema implementado y por lo tanto no se tiene canal de transmisión que inserte ruido al sistema, además de que se realizarán pruebas de funcionamiento individuales tanto del modulador como del demodulador.

1.7. RECUPERACIÓN DE LA PORTADORA

En la recuperación de la señal portadora cabe mencionar el concepto de detección coherente, el mismo que hace referencia al traslado de frecuencia de la señal modulada recibida por medio del producto de esta por una señal de referencia de la misma frecuencia de la portadora y en fase con ella. Así también, en algunos sistemas especialmente los diferenciales se realiza detección no coherente es decir no se hace recuperación de portadora. En el caso particular de este proyecto de titulación se realizará detección coherente a pesar de que no se realice recuperación de portadora, ya que se generará una señal con la misma frecuencia y fase que la señal portadora por medio de un oscilador local.
Para el proceso de recuperación de portadora se emplea uno de los siguientes tres métodos: Circuito Cuadrado, Circuito de Costas Loop, y el Remodulador.

1.8. ALGORITMOS DE DECISIÓN

Los algoritmos de decisión permiten asignar a cada señal recibida la secuencia transmitida más probable, debido a que la señal recibida es una versión ruidosa de la señal transmitida. Estos algoritmos son básicamente de dos grupos: *hard* y *soft decision*.

Cabe mencionar que estos algoritmos son empleados en la etapa de decodificación del receptor, y no serán empleados en el diseño e implementación

de este proyecto de titulación debido a que en el mismo no se trabaja con canal de transmisión y por ende no se inserta ruido al sistema.

1.8.1. ALGORITMOS *HARD DECISION*

El demodulador obtiene una estimación de la secuencia codificada, comparando las salidas de los filtros adaptados con un conjunto de umbrales, y a partir de ésta el decodificador de canal obtiene la secuencia fuente más probable. La modalidad más empleada en este tipo de algoritmos está basada en la distancia de Hamming.

1.8.2. ALGORITMOS *SOFT DECISION*

El decodificador trabaja directamente con las salidas del filtro adaptado, esto es, con las variables de decisión. En este caso se le proporciona al decodificador una mayor cantidad de información, ya que no sólo obtiene la estimación de los bits de la secuencia codificada, sino una medida de la verosimilitud de cada posible estimación.

La decodificación basada en decisiones ponderadas proporciona mejores prestaciones en términos de la probabilidad de error final resultante. La contrapartida es un aumento de complejidad del decodificador, que debe trabajar con la información adicional obtenida del demodulador. Sin embargo, el aumento de complejidad es pequeño en el caso de decodificación convolucional si se utiliza el algoritmo de Viterbi, por lo que esta modalidad es la habitualmente empleada.

CAPÍTULO II

2. FPGA (FIELD-PROGRAMMABLE GATE ARRAY)

Un FPGA es un dispositivo programable de gran desarrollo en la actualidad que ofrece mejores prestaciones que otros dispositivos programables como son los PLAs y PLDs.

2.1. DISPOSITIVOS LÓGICOS PROGRAMABLES

Un dispositivo programable se define como un circuito de propósito general, que posee una estructura interna que puede ser modificada por el usuario para implementar una aplicación específica.

Con el pasar de los años los dispositivos programables han sufrido una evolución, comenzando por la memoria PROM, la cual es el primer dispositivo que cumplió con las características para ser programable, ya que esta puede implementar una función lógica descrita a través de una tabla de verdad, utilizando las líneas de direcciones como entradas y las de datos como salidas.

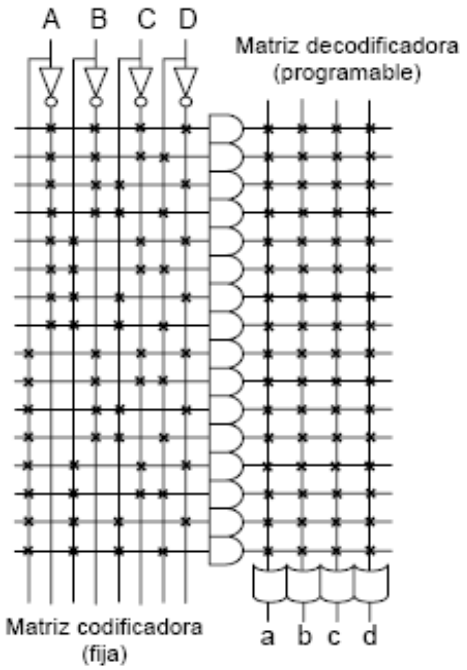


Figura 2.1. Memoria PROM a Nivel de Circuito

El PLD o Dispositivo Lógico Programable es una matriz de compuertas AND conectada a una matriz de compuertas OR, además de algunos biestables, lo que permite implementar cualquier función lógica como una suma de productos.

Su versión más básica es un PAL o Arreglo Lógico Programable, el cual está formado por los dos planos AND-OR, con conexiones programables únicamente en las compuertas AND, como se muestra en la figura 2.2.

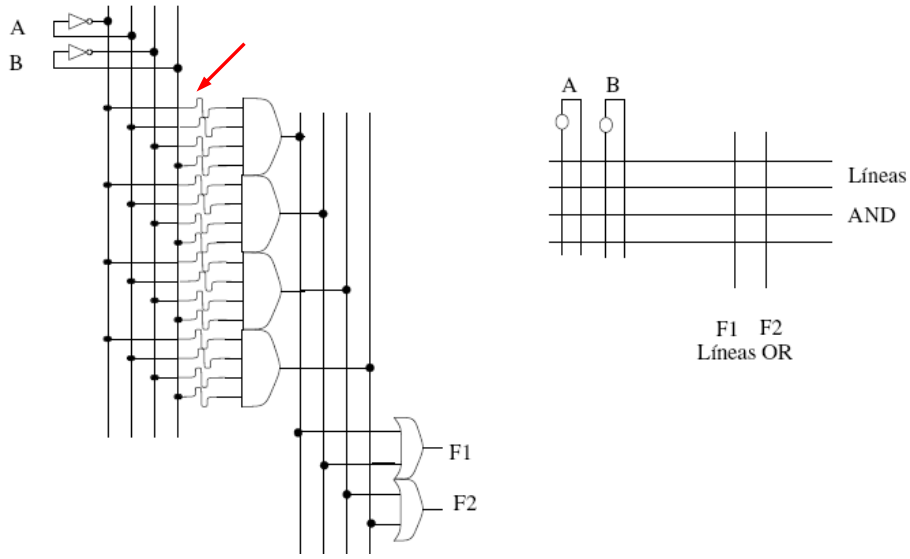


Figura 2.2. PAL a Nivel de Circuito

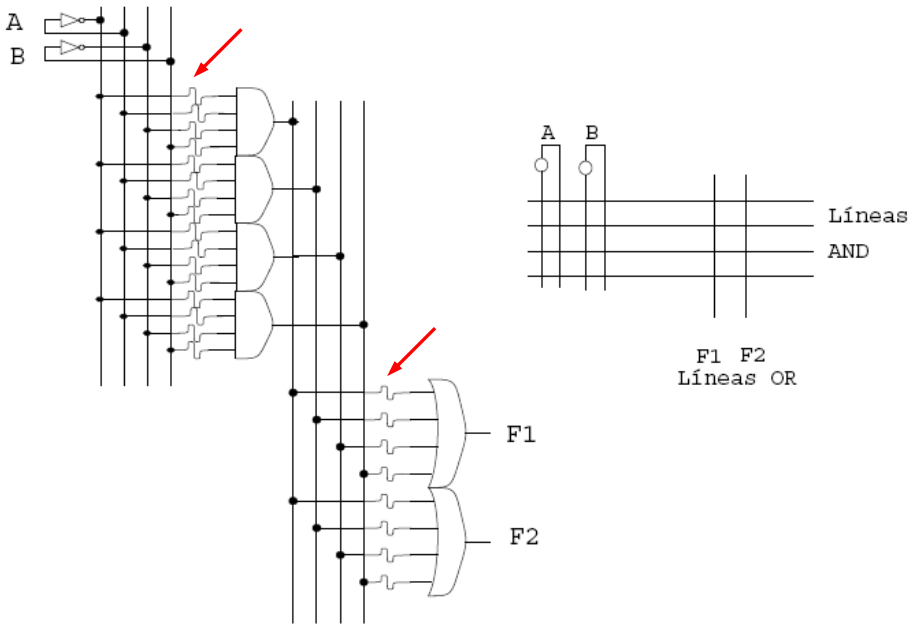


Figura 2.3. PLA a Nivel de Circuito

El PLA es más flexible con respecto al PAL, debido a que se pueden programar las conexiones entre los dos planos, lo que permite obtener buenos resultados en funcionalidades sencillas siempre y cuando éstas sean combinacionales. Aunque PLA también significa Arreglo Lógico Programable, se utiliza para diferenciarlo del PAL.

La ventaja de utilizar estos dispositivos PLD, radica en que son bastante eficientes cuando se implementan circuitos no superiores a centenas de compuertas, sin embargo están limitados a una arquitectura fija y un número fijo de biestables.

El GAL o Arreglo Lógico Programable Genérico es por su parte el primer circuito reprogramable, incluye flip-flops, lo que le permite no sólo implementar funciones combinacionales sino también secuenciales.

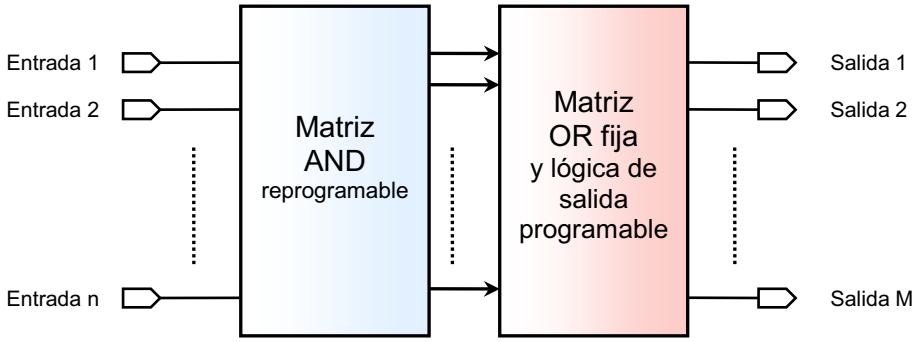


Figura 2.4. Estructura Básica de un dispositivo GAL

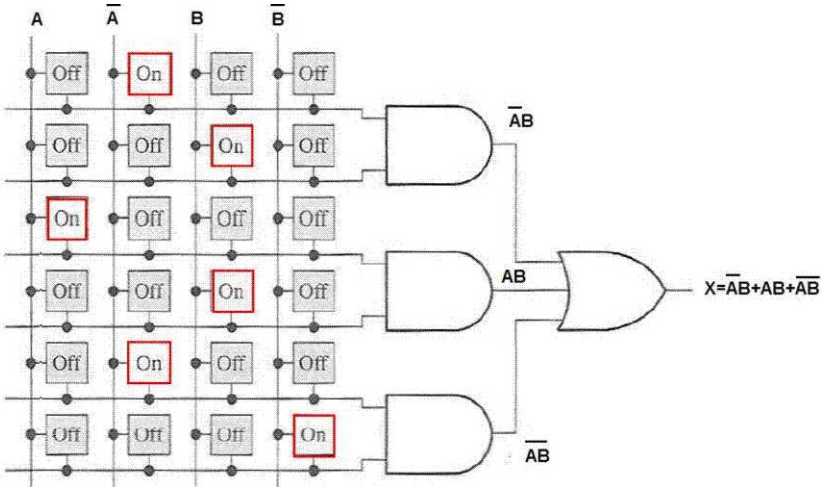


Figura 2.5. Esquema de Funcionamiento de un GAL de 2 entradas y una salida

Otro de los dispositivos lógicos programables que se conocen es el CPLD o Dispositivo Lógico Programable Complejo, el cual se forma con múltiples bloques lógicos, cada uno similar a un PLD. Los bloques lógicos se comunican entre sí utilizando una matriz programable de interconexiones (PIM), con lo cual el área del silicio utilizado se aprovecha de una mejor manera, teniendo así una mejor eficiencia a menor costo.

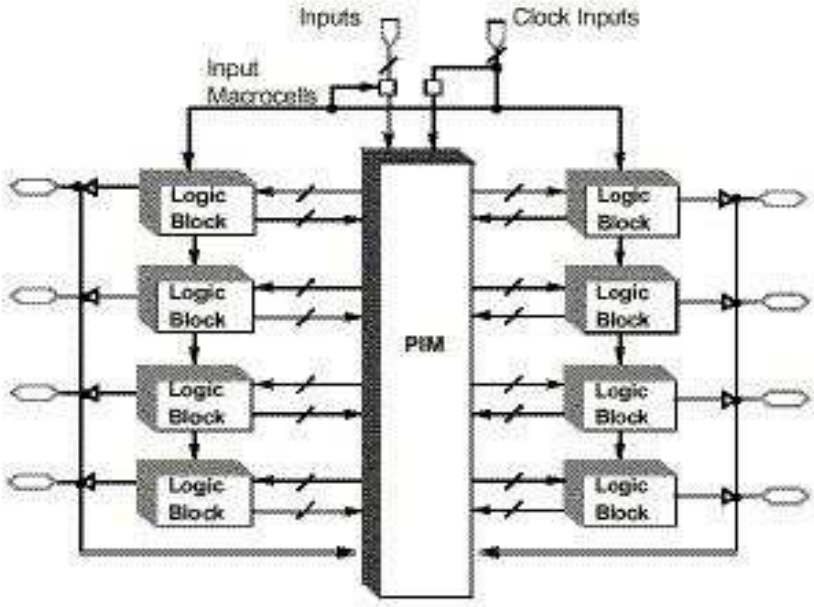


Figura 2.6. Arquitectura Básica de un CPLD

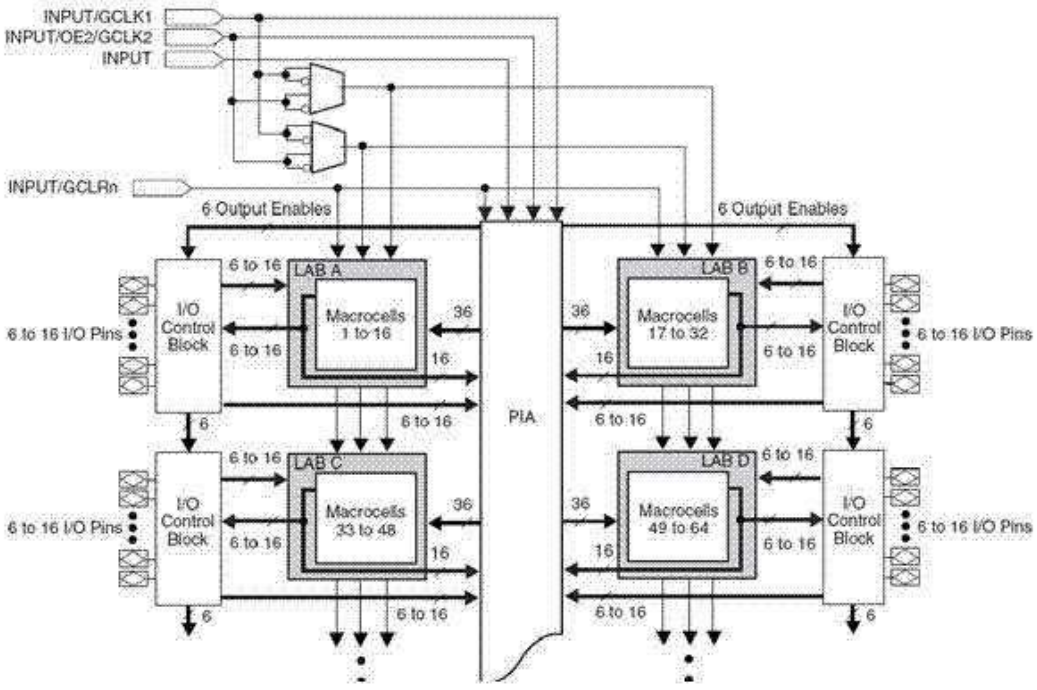


Figura 2.7. Arquitectura de Matriz de suma de productos

Existen dos tipos de arquitectura de los CPLD: Arquitectura de Matriz de suma de productos y Arquitectura Look-up Table (LUT) o tabla de datos.

En la Arquitectura de Matriz de suma de productos que se muestra en la figura 2.7, el CPLD puede contener similares PLD de baja densidad o "PAL", interconectados entre sí en un sólo chip.

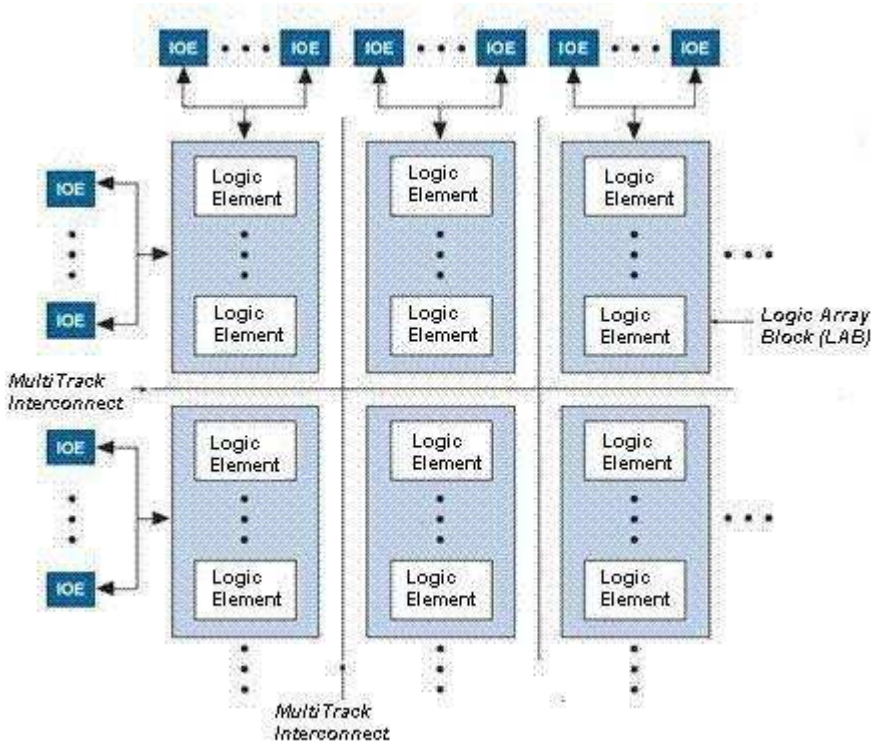


Figura 2.8. Arquitectura Look-up Table (LUT) o tabla de datos

Por otro lado, la arquitectura Look-up Table se basa en la implementación lógica del bus de interconexión en filas y columnas, que también proveen la conexión al bloque de arreglos lógicos (LAB). El LAB consiste de varios elementos lógicos (LE); según el modelo pueden ser 10 LE, por cada LAB. Los LE son una pequeña unidad lógica que proveen una eficiente implementación de funciones lógicas del usuario.

Un MPGA (Mask-Programmable Gate Array) está constituido por un conjunto de transistores más circuitos de E/S, que se unen por medio de pistas de metal conocidas como máscaras, las que se trazan de forma óptima.

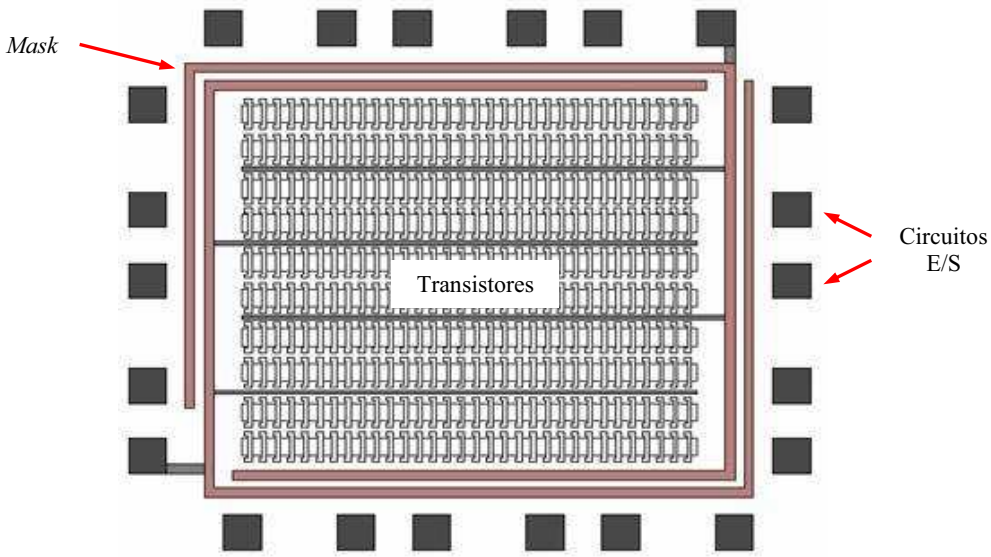


Figura 2.9. Esquema Básico de una MPGA

El FPGA (Field Programmable Gate Array) es el dispositivo programable por el usuario de más general espectro. También se denominan LCA (Logic Cell Array) y consiste en una matriz bidimensional de bloques configurables que se pueden vincular mediante recursos generales de interconexión. Estos recursos incluyen segmentos de pista de diferentes longitudes, más unos conmutadores programables que sirven para enlazar bloques a pistas o pistas entre sí.

Entre las características de estos dispositivos se tiene que son reconfigurables, cumplen con un proceso de fabricación estándar y poseen un bajo consumo de potencia.

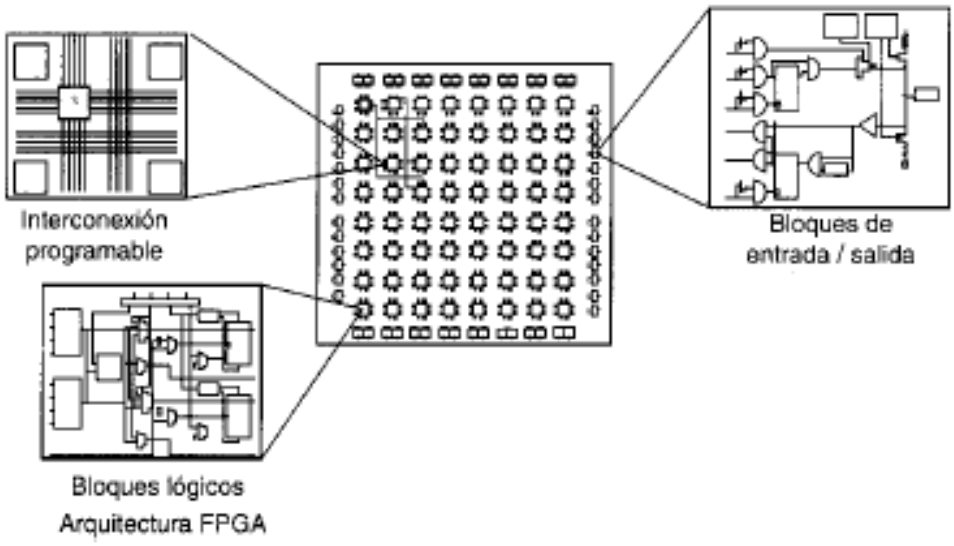


Figura 2.10. Estructura General de un FPGA

2.2. ESTRUCTURA GENERAL DE UN FPGA.

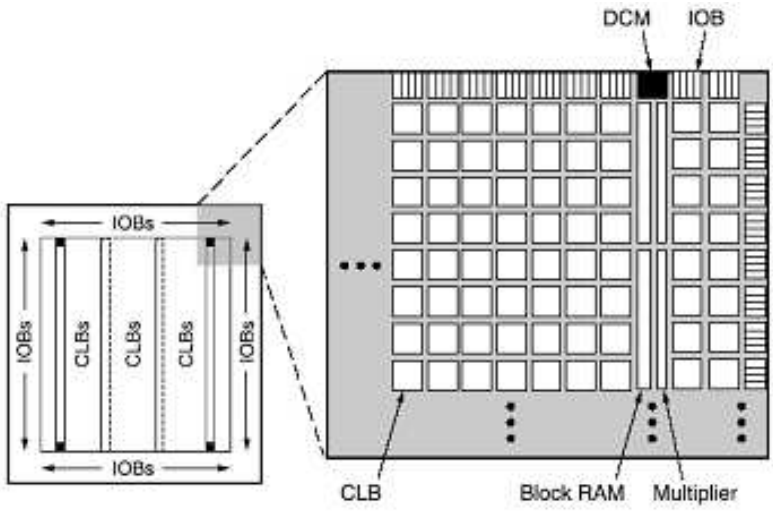


Figura 2.11. Estructura General de un FPGA de Xilinx

Los elementos básicos constituyentes de un FPGA se pueden ver en figura 2.11, y son los siguientes: Bloques lógicos, bloques de entrada/salida, recursos de interconexión y memoria RAM.

2.2.1. BLOQUES LÓGICOS.

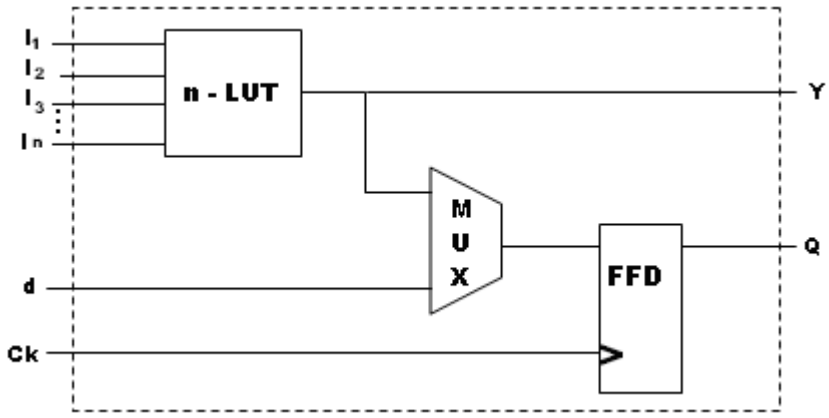


Figura 2.12. Estructura Básica de un Bloque Lógico

El bloque lógico consta de una parte combinacional que permite implementar funciones lógicas booleanas, más una parte secuencial formada por flip-flops, que permite sincronizar la salida con una señal de reloj externa, la cual es útil para realizar circuitos secuenciales y permite implementar registros.

La estructura de un bloque lógico varía de un fabricante a otro; sin embargo, la parte combinacional se basa principalmente en una Look-Up Table (LUT).

Una LUT es un componente de memoria que almacena una tabla de verdad, donde las direcciones de la memoria son las entradas de la función lógica a implementar, y en cada celda de dicha memoria se almacena el resultado de la combinación correspondiente de las entradas.

En una n-LUT es posible implementar cualquier función lógica de n entradas. En la figura 2.12 se muestra la estructura básica de un bloque lógico, en la cual se observa que el multiplexor puede ser configurado para aceptar la salida de la LUT o aceptar una entrada del bloque lógico directamente. En la figura 2.13 se muestra la estructura de una LUT.

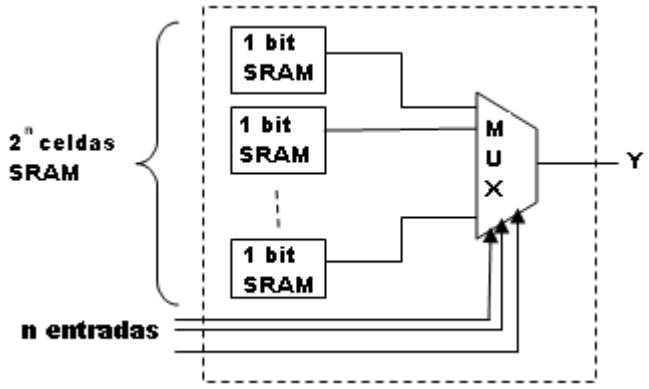


Figura 2.13. Estructura de una Look-up Table (LUT) de n entradas.

Para utilizar las celdas lógicas disponibles, se debe tener una red de interconexión flexible. Sin embargo, debe considerarse que cuanto mayor flexibilidad tenga la red de interconexión mayor será el consumo de potencia y menor la velocidad de operación del dispositivo.

2.2.2. RECURSOS DE INTERCONEXIÓN.

En general, las conexiones internas entre los elementos de un FPGA pueden realizarse de cuatro formas:

Líneas directas: Tienen lugar entre bloques adyacentes.

Conexiones de propósito general: Se realizan a través de una matriz de interconexión cuando se desean conectar dos bloques no adyacentes. Su misión es conectar canales verticales y horizontales que permiten llegar al punto final de la conexión. Cuanto más larga es la línea de ruteo, mayores serán los retrasos introducidos tanto por la propia longitud de la línea como por los elementos de interconexión utilizados a lo largo de la misma.

Líneas largas: Estas conexiones se utilizan cuando una señal debe recorrer una gran longitud, por ejemplo en la implementación de un bus interno; así se evita que la misma atraviese elementos de interconexión, evitando de esta forma retrasos introducidos por los mismos.

Líneas rápidas: Cuando los bloques lógicos contienen más de una LUT, las líneas rápidas son las encargadas de realizar las conexiones entre las distintas LUTs.

2.2.3. BLOQUES DE ENTRADA / SALIDA

La función de un bloque de entrada/salida es permitir el paso de una señal hacia dentro o hacia el exterior del dispositivo. Por este motivo, dependiendo de la interfaz eléctrica, el mismo debe contar con recursos tales como:

- Salidas configurables como *Tri-State* u *Open-Collector*.
- Entradas con posibilidad de *pull-up* o *pull-down* programables.
- Registros de salida.
- Registros de entrada.

Además, debido a que hoy en día los FPGA funcionan a altas velocidades de operación y que los mismos tienen desde cientos hasta miles de terminales de conexión internos, los bloques de entrada/salida deben ser capaces de proveer un adecuado acoplamiento de impedancias, de forma tal que se eviten las

reflexiones de las señales. Esto, que en el pasado se lograba conectando un resistor cercano al terminal del dispositivo, en la actualidad debe ser implementado dentro del propio FPGA, debido a la gran cantidad de terminales que estos dispositivos poseen.

2.2.4. BLOQUES DE MEMORIA

Los bloques de memoria consisten en varias RAM internas al FPGA de 18 Kbits cada una, que se comportan como un chip de memoria de doble puerto, cada puerto tiene sus propias señales de control para las operaciones de lectura y escritura como se muestra en la figura 2.14.

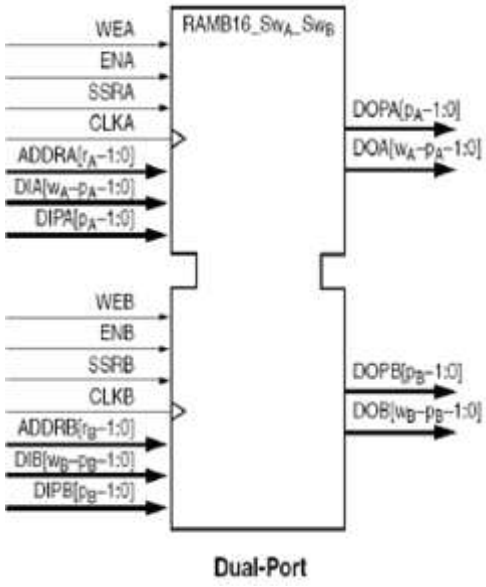


Figura 2.14. Memoria RAM de Doble Puerto

Las principales aplicaciones de estos bloques en un diseño implementado en FPGA son: líneas de retardo, registros de desplazamiento muy largos o muy cortos, buffers circulares, variables durante cálculos matemáticos por ejemplo coeficientes de filtros FIR, almacenamiento de programas para procesadores embebidos, contadores muy largos y muy rápidos, almacenamiento de formas de onda o tablas de funciones trigonométricas.

2.3. TIPOS DE CELDAS DE CONFIGURACIÓN DE UN FPGA

Un punto a tener en cuenta, es cómo se almacena la información correspondiente a las funciones lógicas que se desean implementar y las conexiones que deben hacerse entre bloques lógicos (tecnología de programación).

2.3.1. SRAM (RAM ESTÁTICA)

En estas, el contenido se almacena mediante un proceso de configuración en el momento de encendido del circuito que contiene al FPGA. Debido a que es SRAM, el contenido de la memoria se pierde cuando se deja de suministrar energía; la información binaria de las celdas SRAM generalmente se almacena en memorias seriales EEPROM conocidas como memorias de configuración. En el momento de encendido del circuito, toda la información binaria es transferida a los bloques e interconexiones del FPGA mediante el proceso de configuración el cual es generalmente automático, dado que el propio FPGA contiene un circuito interno que se encarga de efectuar la programación.

2.3.2. ANTIFUSE (ANTIFUSIBLE)

Un FPGA que utiliza este tipo de tecnología sólo puede ser programado una vez (One Time Programmed, OTP). Se las llama antifusible, porque a diferencia de un fusible, donde originalmente existe una conexión y mediante el paso de corriente dicha conexión es destruida, se tiene que la conexión debe ser creada. Una vez que es programado no se puede recuperar el estado original de la conexión.

2.3.3. FLASH

El avance experimentado en los últimos años en el diseño y prestaciones de las celdas de memoria flash ha permitido su incorporación a los dispositivos programables. Los FPGAs basados en celdas flash recogen las ventajas principales de las dos técnicas anteriores situándose en un punto intermedio. Su tamaño es bastante más reducido que el de una celda de SRAM, aunque sin

llegar al tamaño reducido de un antifusible; son reprogramables, aunque la velocidad de programación es bastante más lenta que en el caso de una SRAM y son no volátiles, por lo que no necesitan un dispositivo auxiliar para guardar la configuración interna, como en el caso de la SRAM.

2.4. CELDAS DE CONFIGURACIÓN DE XILINX

En el caso de los FPGAs de XILINX los elementos de programación se basan en células de SRAM que controlan transistores de paso, compuertas o multiplexores. En la figura 2.14 se puede ver esquemáticamente su estructura interna, dependiendo del tipo de conexión requerida se elegirá un modelo u otro de FPGA.

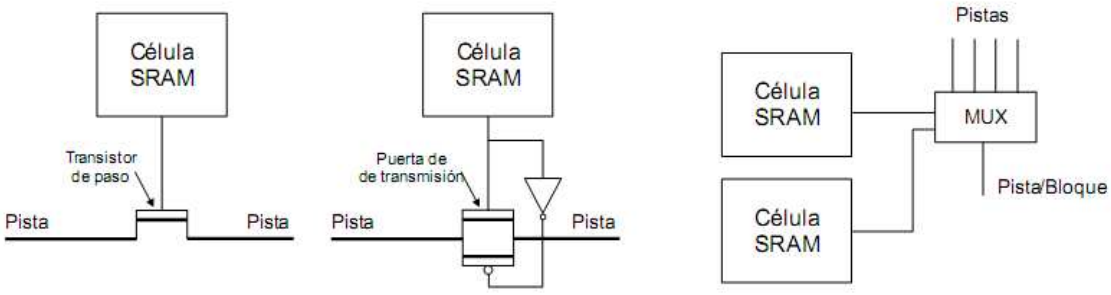


Figura 2.15. Tipos de conectores utilizados por Xilinx

Es importante destacar, que si se utilizan células SRAM la configuración del FPGA sería válida únicamente mientras esté conectada la alimentación, pues la memoria es volátil. En los sistemas finales, está claro que hace falta algún mecanismo de almacenamiento no volátil que cargue las células de RAM, esto se puede conseguir mediante memorias ROM.

2.5. TARJETA DE ENTRENAMIENTO SPARTAN-3E

Para describir la tarjeta Spartan-3E mostrada en la figura 2.16, se mostrará en primer lugar las características técnicas proporcionadas en su *datasheet* por el fabricante.

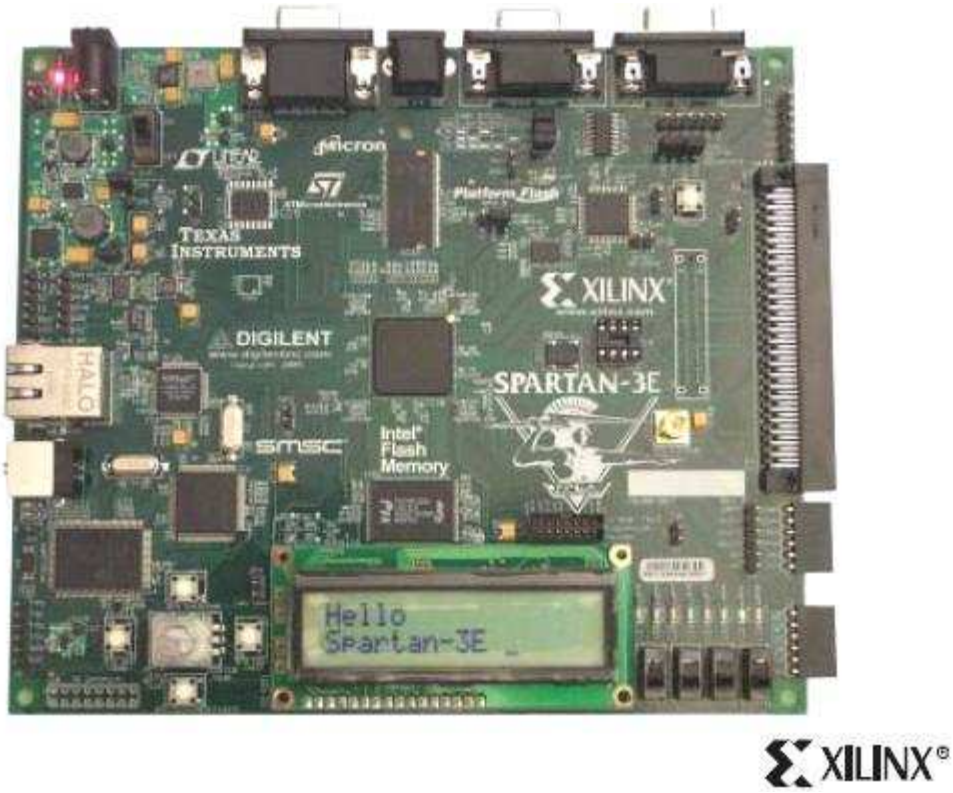


Figura 2.16. Tarjeta Spartan-3E

Dispositivos Xilinx:

FPGA (XC3S500E-4FG320C) de 320 pines y encapsulado BGA[1]

- Sobre las 10,000 celdas lógicas
- 232 pines de entrada/salida

CoolRunner™-II CPLD (XC2C64A-5VQ44C)

Platform Flash (XCF04S-VO20C)

Reloj: 50MHz reloj oscilador de cristal

Memoria:

128 Mbit Flash Paralela

16 Mbit SPI Flash

34 Mbyte DDR SDRAM

Conectores e Interfaces:

Ethernet 10/100

[1] Xilinx Spartan-3E UG 230 Starter Kit Board User Guide, pp. 12 y 13.

JTAG USB

2 Puertos Seriales 9-pin RS-232

Puerto PS/2 estilo mouse o teclado

Encoder giratorio con pulsador

4 Switches

8 LEDs

4 Pulsadores

Puerto de expansión de conexión de 100 pines

3 Conectores de expansión de 6 pines

Display:

LCD de 2 filas – 16 caracteres

2.5.1. FUENTES DE RELOJ

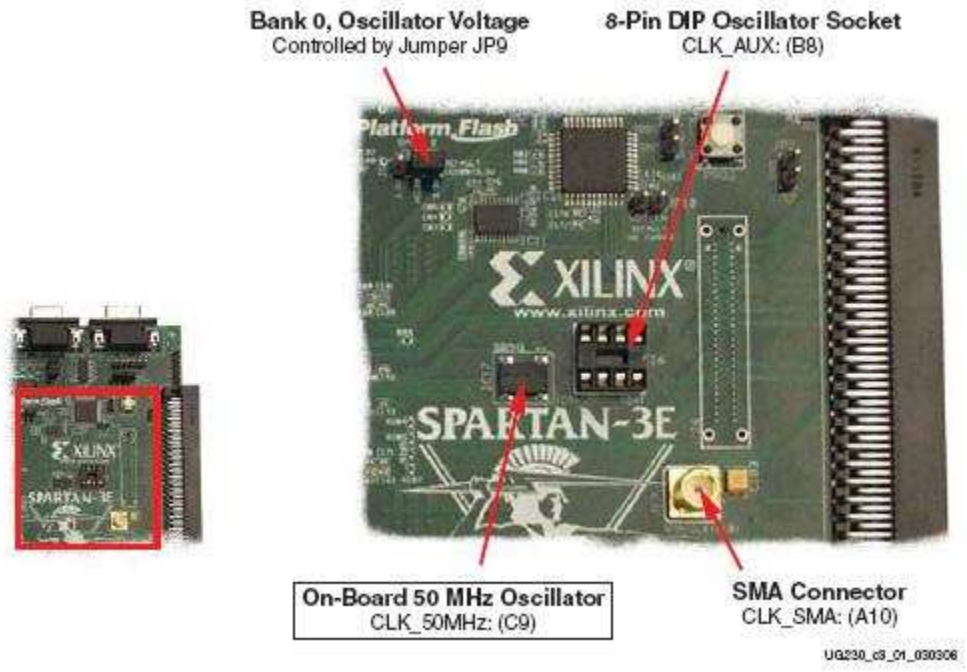


Figura 2.17. Fuentes de Reloj de la Spartan-3E

En este proyecto de titulación se utilizará la fuente de reloj integrada en la tarjeta de entrenamiento Spartan-3E de Xilinx, la misma que posee un oscilador de 50MHz con 40 a 60 % de duty cycle y una precisión de ± 2500 Hz o ± 50 ppm, y se encuentra ubicada físicamente en el pin C9 del FPGA utilizado.

Como se puede apreciar en la figura 2.16, ésta no es la única opción para obtener la señal de reloj requerida para un sistema ya que al trabajar con un valor único de frecuencia de reloj se tendría una limitante en la gran versatilidad que nos ofrece un dispositivo como un FPGA. Es así, que Xilinx ofrece dos posibilidades adicionales a su reloj integrado en la tarjeta, permitiendo así introducir externamente esta señal hacia el FPGA.

Como primera opción la Spartan-3E de Xilinx posee un socket de 8 pines para colocar un oscilador cuyo tipo de encapsulamiento sea DIP (Dual in-line package). Además posee también un conector SMA (SubMiniature version A), el cual es un tipo de conector roscado para cable coaxial. Los pines que ocupan físicamente estas dos opciones de reloj externo son el B8 y A10 respectivamente.

2.5.2. CONECTORES DE EXPANSIÓN.

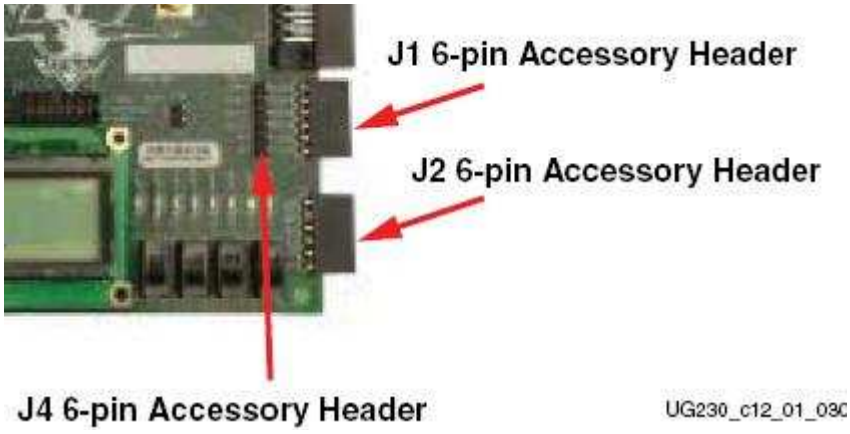


Figura 2.18. Conectores de Expansión de la Spartan-3E

Los conectores J1, J2 y J4 son conectores de 6 pines que proveen interfaces de I/O utilizando varios módulos periféricos de Digilent. Como se puede observar en la figura 2.17 los conectores J1 y J2 usan sockets hembra de 6 pines a 90º, mientras el J4 usa un conjunto de pines en estaca o espadines.

Los tres conectores tienen una estructura similar como la que se puede observar en la figura 2.18, en la cual cuatro pines están conectados a la FPGA, uno es tierra y el último está conectado a una fuente de alimentación de 3.3V.

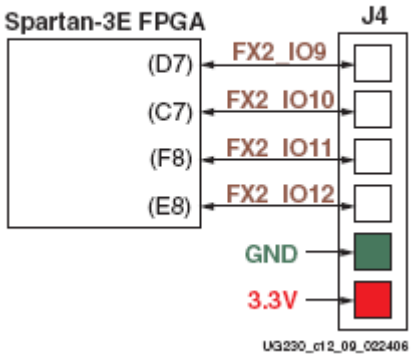


Figura 2.19. Estructura del Conector J4 de la Spartan-3E

2.5.3. INTERRUPTORES

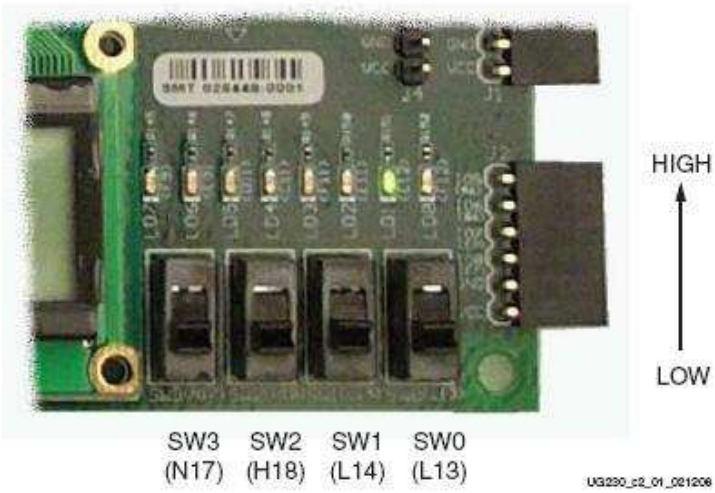


Figura 2.20. Interruptor desplazable de la Tarjeta Spartan-3E

La tarjeta de entrenamiento Spartan-3E de Xilinx tiene cuatro interruptores desplazables ubicados en la esquina inferior derecha de la tarjeta. El funcionamiento de este tipo de interruptores se basa en que si su estado es ON se conecta a una fuente de 3.3V estado lógico Alto, mientras que si su estado es OFF el mismo se conecta a tierra dando un nivel de voltaje bajo, cabe mencionar que presentan un rebote mecánico cercano a los 2ms y no hay circuitos activos para la eliminación de los rebotes.

CAPÍTULO III

3. DESCRIPCIÓN DEL SOFTWARE

3.1. INTRODUCCIÓN

System Generator o SysGen es un paquete computacional de Xilinx el cual se aloja en Simulink de Matlab. Permite realizar el diseño de sistemas, por medio del correcto acoplamiento de bloques, para obtener una funcionalidad requerida. El SysGen no es capaz de implementar un sistema individualmente, necesita la instalación del paquete ISE de Xilinx.

En este capítulo se hará la descripción general del software y de los bloques que serán utilizados posteriormente en el diseño del modulador y del demodulador.

3.2. SYSTEM GENERATOR

3.2.1. INSTALACIÓN Y COMPATIBILIDAD CON LA VERSIÓN DE MATLAB

La instalación de System Generator de Xilinx es un proceso muy sencillo; sin embargo presenta una serie de dificultades especialmente en la compatibilidad de versiones de los diferentes paquetes que deben ser instalados.

Es necesario que exista compatibilidad entre ISE, System Generator (paquetes de Xilinx) y Simulink de Matlab. Adicionalmente es importante validar que sea compatible el sistema operativo y los drivers del cable de programación de la tarjeta de entrenamiento.

Tomando en consideración esto, se decidió para el desarrollo del presente proyecto de titulación, la utilización de la versión 8.1 de System Generator e ISE de Xilinx y la versión 7.1 de Matlab (Simulink v6.3 (R14.3) Service Pack 3) corriendo sobre Windows XP Professional Service Pack 2. Este conjunto de versiones no genera ningún tipo de incompatibilidad; sin embargo a continuación

se presenta una lista de opciones de versiones, las cuales son compatibles y no producen ningún tipo de problema.

Las versiones de Matlab compatibles con System Generator 8.1 son:

- MATLAB v7.0.1/Simulink v6.1 (R14.1) Service Pack 1
- MATLAB v7.0.4/Simulink v6.2 (R14.2) Service Pack 2
- MATLAB v7.1.0/Simulink v6.3 (R14.3) Service Pack 3
- MATLAB v7.2/Simulink v6.4 (R2006a)

Los paquetes de Xilinx son compatibles en la misma versión, es decir System Generator 8.1 es compatible con ISE 8.1. La compatibilidad del software ISE 8.1i de Xilinx con el sistema operativo sobre el cual está corriendo, se muestra en la figura 3.1.

Platform Type	Version Number
Windows®	2000 Pro SP2, SP3, SP4 XP Pro SP1, SP2
Japanese Windows®	2000 Pro SP2, SP3, SP4 XP Pro SP1, SP2
Chinese Windows®	2000 Pro SP2, SP3, SP4 XP Pro SP1 XP Pro SP2
Korean Windows®	2000 Pro SP2, SP3, SP4 XP Pro SP1 XP Pro SP2
Linux®	Red Hat® Enterprise WS 3.0 32-bit/64-bit Red Hat Enterprise WS 4.0 32-bit/64-bit (Limited Support)
Solaris®	8, 9

Note: Limited support implies that limited testing was conducted on this operating system.

Figura 3.1. Compatibilidad de ISE8.1i de Xilinx con Sistemas Operativos

La figura 3.2 muestra la versión más reciente de Xilinx ISE, es decir la 12.1 y su compatibilidad con los diferentes sistemas operativos.

ISE Design Suite	Operating System Support									
	Windows XP Professional 32-Bit*	Windows XP Professional 64-Bit*	Windows Vista Business 32-Bit*	Windows Vista Business 64-Bit*	Red Hat Enterprise Linux 4 WS32-bit	Red Hat Enterprise Linux 4 WS64-bit	Red Hat Enterprise Linux 5 WS32-bit	Red Hat Enterprise Linux 5 WS64-bit	SUSE Linux Enterprise 11 32-bit	SUSE Linux Enterprise 11 64-bit
ISE Design Entry and Implementation Tools	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
ISE Simulator (ISim)	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
ISE WebPACK	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
ChipScope Pro and the ChipScope Pro Serial I/O Toolkit	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
Embedded Development Kit (EDK) and Platform Studio	✔	✔	✔		✔	✔	✔	✔	✔	✔
Software Development Kit (SDK)	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
System Generator for DSP	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
ModelSim Xilinx Edition-III (MXE-III)	✔	✔	✔	✔						

*US and Japanese

Note: Windows 7 Operating Systems will have only limted testing, resulting in limited Windows 7 support for ISE 12. Full support for Windows 7 is planned for ISE 13.

Figura 3.2. Compatibilidad de ISE12.1 de Xilinx con Sistemas Operativos

Una vez mostradas las opciones de compatibilidad de los distintos paquetes computacionales, se analizarán los pasos a seguir para la instalación de los mismos, debido a que para su correcto funcionamiento se debe seguir un orden específico y ciertas consideraciones.

Los pasos de instalación de System Generator son:

- Instalación del paquete computacional Matlab. Un detalle a tomar en cuenta en este paso, es que el directorio de instalación de Matlab debe ser C:\MATLAB71, además de no tener ningún espacio como se muestra en el directorio.

- El siguiente software que se debe instalar es el ISE de Xilinx. La instalación de este paquete computacional puede tardar varios minutos dependiendo de la capacidad del computador, pues en la mayoría de los casos este instalador no

sólo tiene el ISE8.1, sino un conjunto de software de Xilinx: ChipScope, EDK, Modelsim, entre otros programas que no entran en el alcance del presente proyecto de titulación.

- Finalmente se instala System Generator.

Una vez culminado el procedimiento descrito anteriormente, se tiene como resultado tres blocksets en el Simulink de Matlab, los cuales a su vez están compuestos por diez librerías, las que contienen una gran cantidad de bloques con los cuales se genera el diseño del sistema a implementarse en el FPGA.

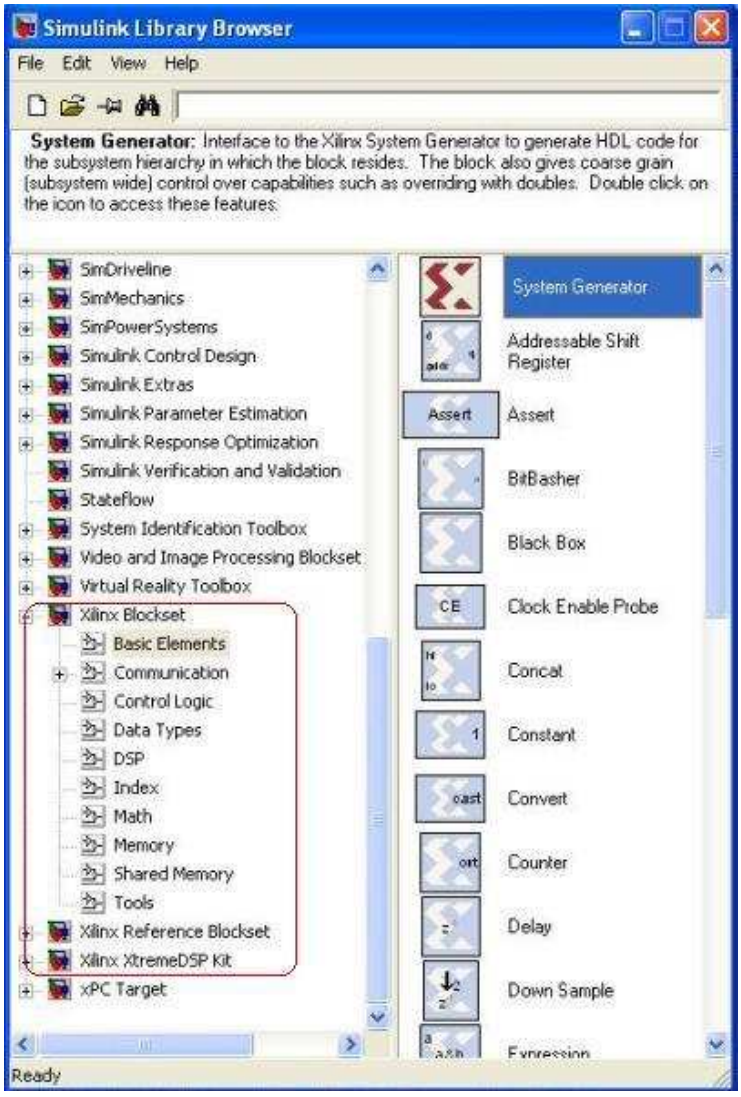


Figura 3.3. Blocksets de System Generator instalados en Simulink

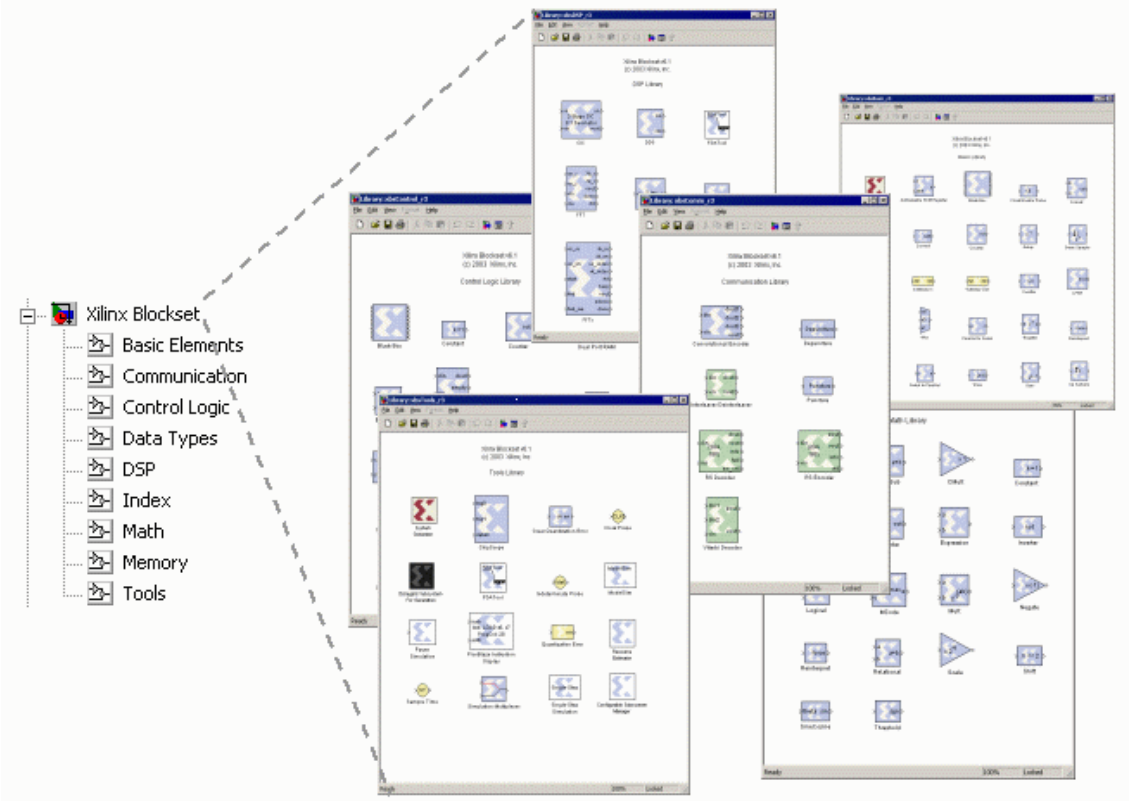


Figura 3.4. Bloques que conforman las librerías de Xilinx Blockset

Cabe mencionar que los bloques que componen las librerías de System Generator son de diferente clase, pues algunos son sintetizables (se pueden implementar en el FPGA) y otros no. La manera de identificarlos es por el color de fondo del bloque, tal como se presentan en las figuras 3.5 y 3.6.

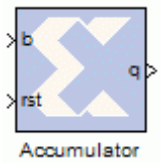


Figura 3.5. Ejemplo de Bloque Sintetizable

Figura 3.6. Ejemplo de Bloque No Sintetizable

3.2.2. INTEGRACIÓN DEL PROGRAMA SYSGEN CON XILINX ISE

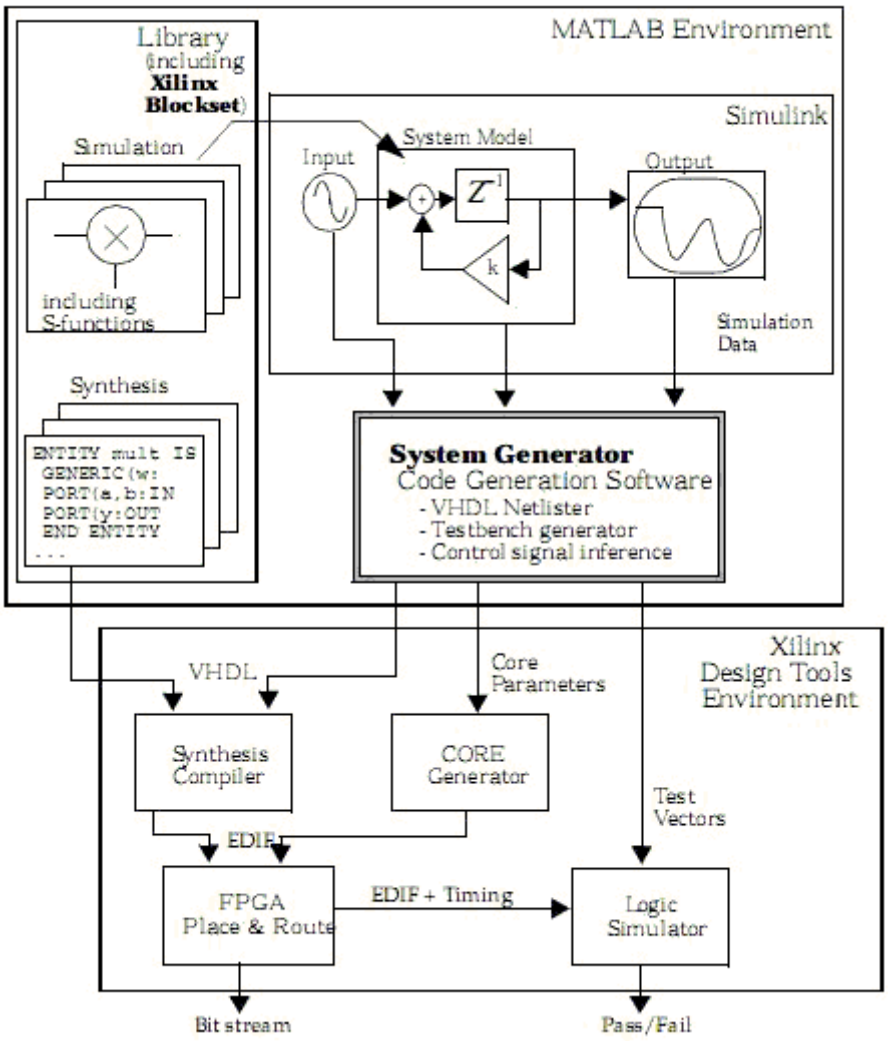


Figura 3.7. Diagrama de compilación

Cuando un sistema ha sido modelado en Simulink con la utilización de cualquier blockset de SysGen, se necesita compilar el modelo para obtener el archivo de configuración del FPGA, por medio de la integración de System Generator con Xilinx ISE, como se indica en la figura 3.7. Para la compilación, SysGen debe sintetizar o trasladar el sistema a VHDL, que es un lenguaje comprendido por Xilinx, para que posteriormente las herramientas del ISE procedan.

En esta compilación se originan una serie de archivos que ayudan finalmente a generar el *bitstream*, siendo este proceso y sus archivos transparentes para el programador.

3.3.OPCIONES COMUNES EN PARÁMETROS DE BLOQUES

Cada uno de los bloques de SysGen presenta parámetros configurables, a los cuales se accede haciendo doble clic sobre el bloque. Estos pueden ser específicos o comunes en la mayoría de los bloques. Por este motivo, en esta sección se detallan las características de los parámetros comunes.

3.3.1. REPRESENTACIÓN ARITMÉTICA

Dentro de este parámetro se puede escoger el tipo de dato que se va a tener en la salida. Las opciones que ofrece System Generator son: booleano, sin signo y con signo en complemento de dos.

3.3.2. NÚMERO DE BITS

El número de bits es un parámetro que permite indicar con cuántos bits se va a representar una muestra. El máximo valor que puede adoptar es 4096 bits.

3.3.3. PUNTO BINARIO

El parámetro punto binario indica el número de bits a la derecha del punto para el puerto de salida, estableciendo el tamaño de la parte fraccionaria. La posición del punto binario debe estar entre cero y el valor especificado en el número de bits.

Entonces, un número de punto fijo se caracteriza por los tres parámetros analizados anteriormente, es decir la representación aritmética, el número de bits y el punto binario.

3.3.4. CUANTIZACIÓN Y DESBORDAMIENTO

Las opciones para la cuantización son: *Truncate* para eliminar los bits que se encuentran a la derecha del bit menos significativo (LSB) que se pueda

representar; y *Round* que permite redondear al valor más cercano con el número de bits disponibles.

Con respecto al desbordamiento SysGen ofrece tres opciones: *Wrap* para descartar los bits que están a la izquierda del bit más significativo; *Saturate* que permite saturar al valor positivo más grande y al valor negativo más pequeño; y *Flag as error* que genera un error durante la simulación.

3.3.5. PERÍODO DE MUESTREO

El flujo de datos es procesado a una tasa específica de muestreo, siendo común que cada bloque detecte la tasa de muestreo de entrada y genere la correcta tasa de muestreo a la salida. Sin embargo SysGen permite modificar este parámetro con la opción *Specify Explicit Sample Period,* con la cual el usuario coloca el período de muestreo requerido para todas las salidas del bloque.

3.3.6. LATENCIA

Muchos elementos en el blockset de Xilinx tienen una opción de latencia, la cual define el número de períodos de muestreo que será retardada la salida del bloque. Cabe recalcar que un período de muestreo puede contener varios ciclos de reloj en la implementación del sistema.

3.4. DESCRIPCIÓN DE LOS BLOQUES UTILIZADOS EN EL MODULADOR Y DEMODULADOR

Para complementar la explicación de los otros parámetros usados por los bloques, se debe hacer referencia al punto 3.3.

3.4.1. TOKEN DE SYSTEM GENERATOR

Figura 3.8. Bloque Token de System Generator

Este bloque posee características esenciales para la creación y simulación de modelos en Simulink que contengan cualquier elemento que se tome del Blockset de Xilinx. Dichas características son el control del sistema mediante los parámetros de la simulación y la invocación del generador de código.

Las diferentes librerías del Blockset de Xilinx donde se encuentra este bloque son Basic Elements, Tools, e Index.

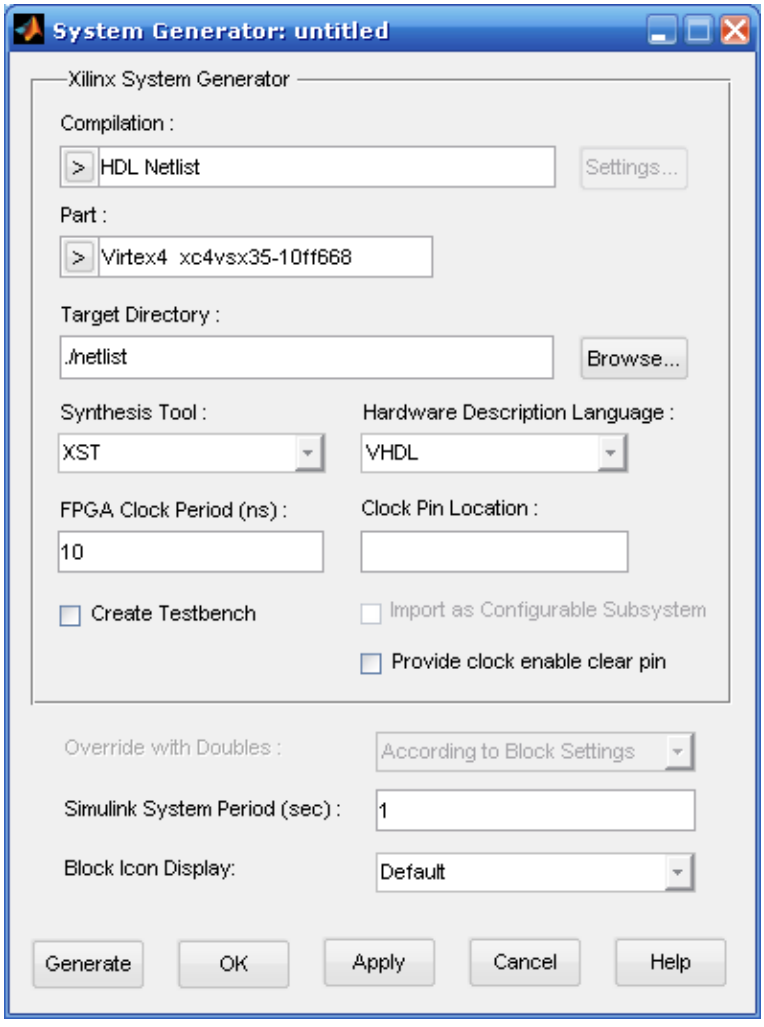


Figura 3.9. Parámetros del Token de System Generator

Los parámetros específicos mostrados en la figura 3.9 se detallan a continuación:

Compilation: Indica el tipo de compilación elegida para cuando se llame al generador de código.
Part: Define el tipo de FPGA que contiene la tarjeta de entrenamiento.
Target Directory: Muestra el directorio donde SysGen escribirá los resultados de la compilación.
Synthesis Tool: Especifica la herramienta que se utilizará para la síntesis del diseño. Teniendo tres posibilidades: Synplicity's Synplify Pro, Synplify y Xilinx's XST (incluida con la instalación de SysGen).
Hardware Description Language: Señala el tipo de lenguaje HDL que se empleará (VHDL o Verilog).
FPGA Clock Period: Determina el período en nanosegundos que adoptará el reloj en el hardware dentro del FPGA.
Clock Pin Location: Especifica el terminal que corresponde a la señal de reloj utilizada siendo diferente en cada tarjeta de entrenamiento.
Create Testbench: Indica a SysGen que cree un archivo de prueba (HDL testbench) con el cual se puede comparar el resultado obtenido en Simulink.
Import as Configurable Subsystem: Construye un bloque configurable que consta de varios sistemas de los cuales el subsistema fue derivado.
Provide clock enable clear pin: Proporciona un puerto de habilitación ce_clr en el reloj principal.
Override with Doubles: Determina que todos los cálculos sean realizados con aritmética de doble precisión.
Simulink System Period: Define el período del sistema en Simulink dado en segundos. Este es el máximo común divisor de los períodos de muestreo que aparecen en el modelo.
Block Icon Display: Especifica la información que se desea visualizar en los bloques cuando se simula el sistema. Existen varios tipos Default, Simple Rates, HDL port names, Input data types y Output data types.

3.4.2. RESOURCE ESTIMATOR

Figura 3.10. Bloque Resource Estimator

Este bloque genera una estimación rápida de los recursos del FPGA requeridos para implementar un modelo o un subsistema de System Generator. La estimación se hace tomando en cuenta bloques Xilinx, tablas de lookup (LUTs), flip flops, bloques de memoria, multiplicadores, buffers y puertos de entrada salida.

Cada bloque de Xilinx que requiere recursos del FPGA, posee un campo donde se almacena un vector, que contiene la cantidad de recursos necesarios. El bloque Resource Estimator puede acceder a este vector y de esta forma hacer un cálculo de los recursos utilizados en un subsistema o en un modelo.

Las diferentes librerías del Blockset de Xilinx donde se encuentra este bloque son Tools e Index.

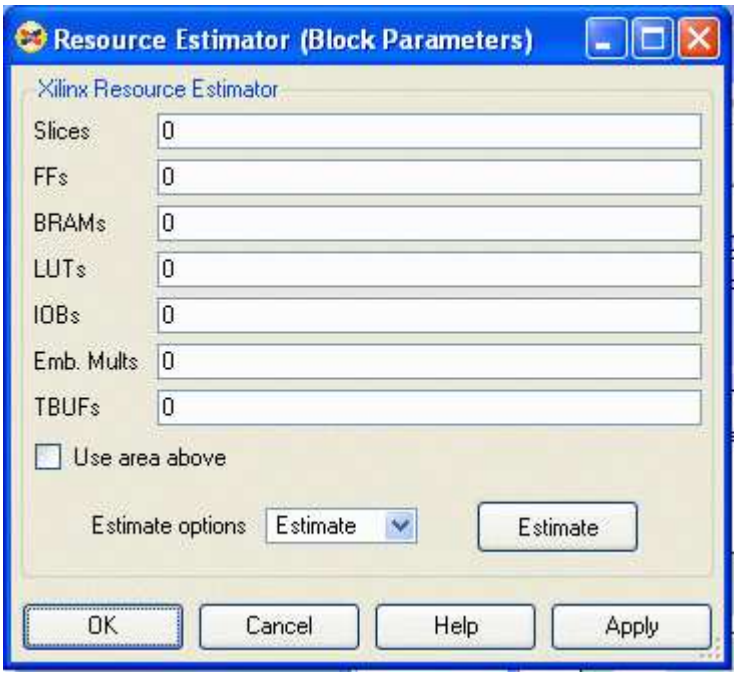


Figura 3.11. Parámetros del Bloque Resource Estimator

Los parámetros mostrados en la figura 3.11 se detallan a continuación:

Slices: Slices utilizados por los bloques del sistema. Consisten en dos flip-flops, dos LUTs y están asociados a un multiplexor.
FFs: Flip-flops utilizados por los bloques.
BRAMs: Bloques de memoria RAM utilizados por los bloques del sistema.
LUTs: Look-up Tables utilizadas por el sistema.
IOBs: Bloques de entrada/salida consumidos por el sistema.
Embedded Mults: Multiplicadores embebidos utilizados por los bloques del sistema.
TBUFs: Buffers de tres estados utilizados por el sistema.
Use Area Above: Cuando está casilla está marcada, cualquier estimación de recursos devolverá los números ingresados en los cuadros de edición.
Estimate Options: Permite seleccionar el método de estimación, entre las siguientes opciones: Estimate, Quick, Post Map y Read Mrp (los dos últimos están fuera del alcance del proyecto).
Estimate: Realiza la estimación de recursos, mostrando en el campo de recursos respectivo la cantidad utilizada por el modelo.

3.4.3. GATEWAY IN

Figura 3.12. Bloque Gateway In

Este bloque es la interfaz entre los bloques de Simulink y los de SysGen, debido a que convierte los datos de tipo entero, doble y punto fijo al formato de SysGen de punto fijo. Además cada bloque define una entrada de alto nivel en el diseño HDL generado.

Las diferentes librerías del Blockset de Xilinx donde se encuentra este bloque son Basic Elements, Data Types, e Index.

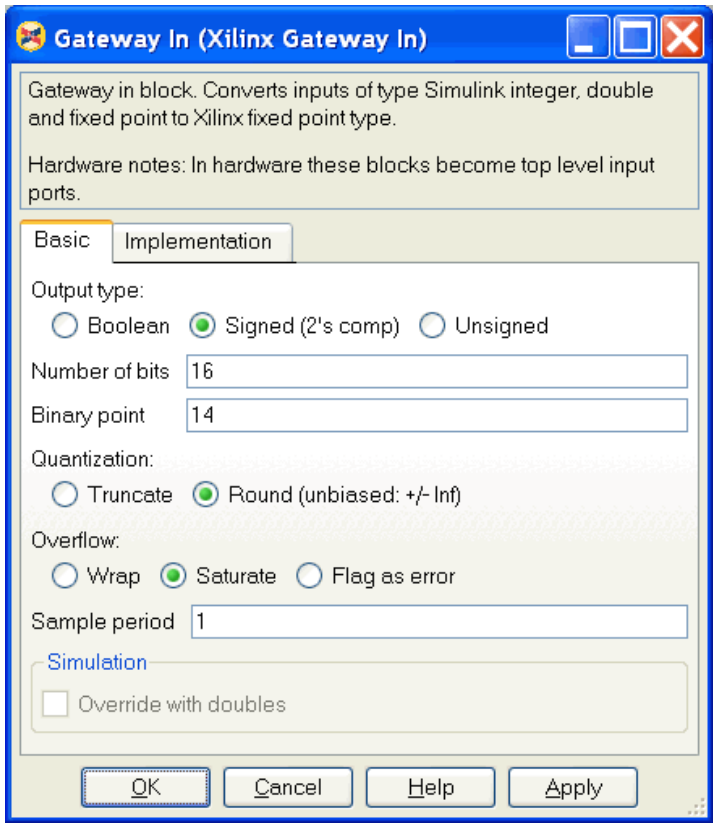


Figura 3.13. Parámetros Básicos del bloque Gateway In

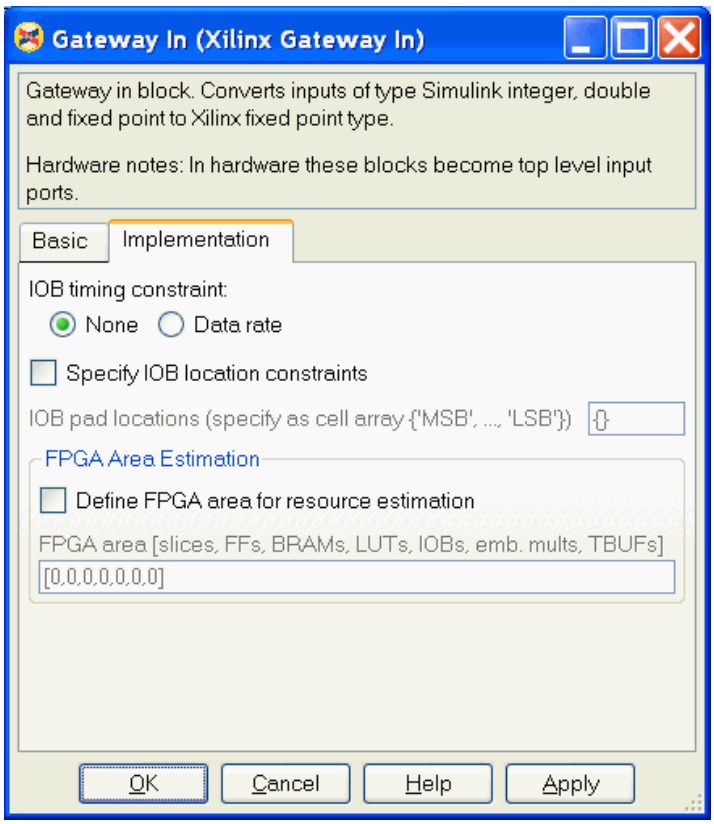


Figura 3.14. Parámetros de Implementación del bloque Gateway In

Los parámetros específicos mostrados en la figuras 3.13 y 3.14 se detallan a continuación:

IOB Timing Constraint: Cuando se traslada a hardware el bloque Gateway In, se construye con un conjunto de buffers de entrada/salida IOB, en los cuales se puede limitar su sincronización. Para esto existe None y Data Rate.
Si se marca None los IOBs hacia los elementos sincrónicos no se limitan. Pero al contrario si Data Rate es seleccionado, los IOBs son limitados y funcionarán con un tiempo de muestreo definido que depende de la Frecuencia del Reloj especificado en el Token del Sistema.
Specify IOB Location Constraints: Si se marca este cuadro, habilita un campo que permite al usuario definir la ubicación de la interfaz que representa este bloque dentro de la tarjeta de entrenamiento.
IOB Pad Locations: Aquí se describe la ubicación de la interfaz como un arreglo de caracteres tipo celda {'MSB',..., 'LSB'}.

3.4.4. GATEWAY OUT

Figura 3.15. Bloque Gateway Out

Este bloque es la interfaz entre los bloques de SysGen y los de Simulink, debido a que convierte los datos de tipo punto fijo entero de SysGen, en tipo doble de Simulink.

De acuerdo a su configuración, este bloque puede también definir un puerto de salida de alto nivel en el diseño HDL generado por SysGen o también ser usado simplemente como un punto de prueba.

Las diferentes librerías del Blockset de Xilinx donde se encuentra este bloque son Basic Elements, Data Types, e Index.

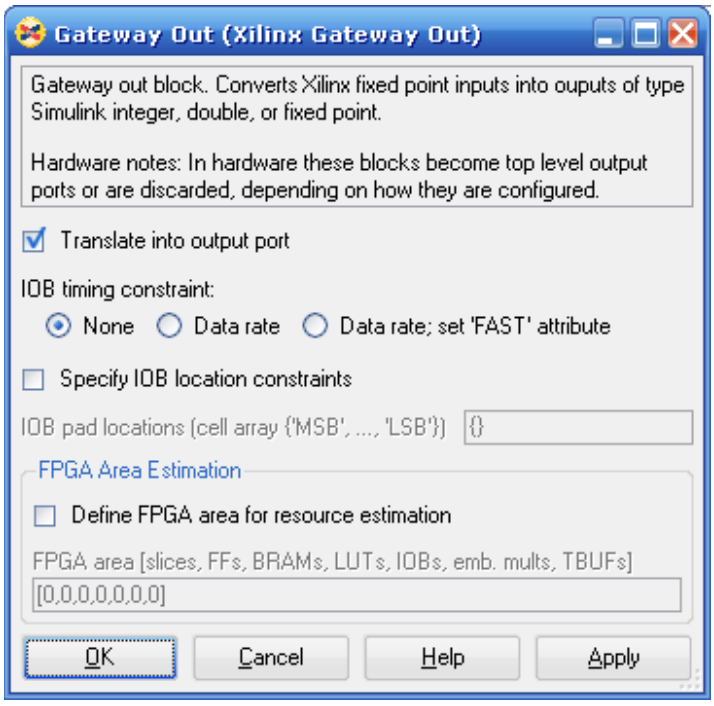


Figura 3.16. Parámetros de Implementación del bloque Gateway Out

Los parámetros mostrados en la figura 3.16 se detallan a continuación:

Translate into Output Port: En forma predeterminada está habilitada, pero si se deselecciona esta opción, se previene que llegue a trasladarse en un puerto en hardware. Cuando esto último pase, este bloque se tornará con un color gris.

IOB Timing Constraint: De forma similar a Gateway In, cuando se traslada al hardware el bloque Gateway Out se construye con un conjunto de buffers de entrada/salida IOB. Los tipos de sincronización son: None, Data Rate, y Data Rate; Set 'FAST' Attribute.

Si None es seleccionado los IOBs hacia los elementos sincrónicos no se limitan; si se selecciona Data Rate los IOBs son limitados y funcionarán con un tiempo de muestreo definido que depende de la Frecuencia del Reloj especificado en el Token del Sistema. Por último, Data Rate; Set 'FAST' Attribute reduce el retado producido al seleccionar la opción anterior, pero incrementa el ruido y el consumo de potencia.

Los otros dos parámetros **Specify IOB Location Constraints** e **IOB Pad Locations** se configuran de igual forma que en el bloque Gateway In.

3.4.5. MUX

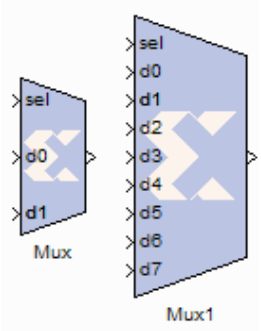


Figura 3.17. Bloque Mux

Este bloque es un multiplexor que posee entradas de datos configurables y una línea de selección, la misma que controla la salida que se obtiene en un determinado instante de tiempo.

Las diferentes librerías del Blockset de Xilinx donde se encuentra este bloque son: Basic Elements, Control Logic e Index.

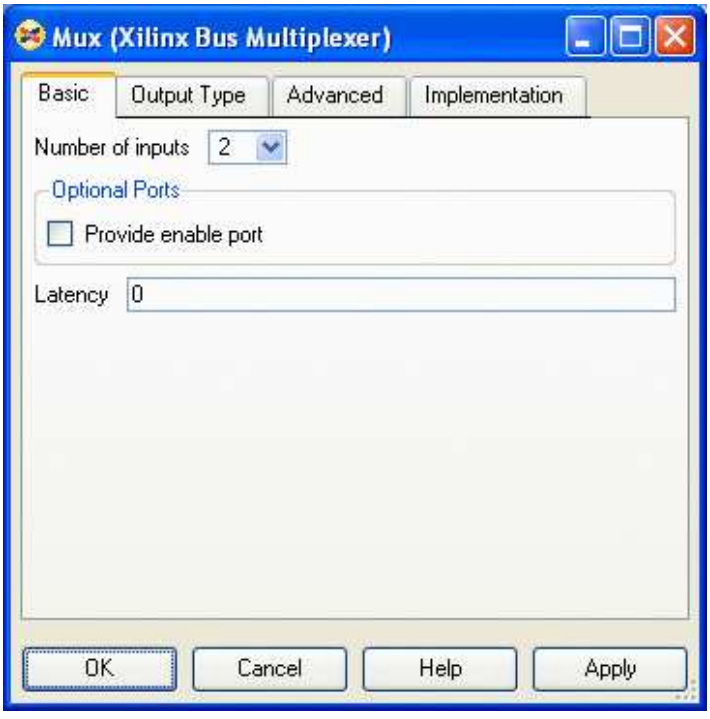


Figura 3.18. Parámetros Básicos del Bloque Mux

Los parámetros específicos mostrados en la figura 3.18 se detallan a continuación:

Number of Inputs: Este parámetro permite elegir el número de entradas que se tiene para multiplexar, este valor está comprendido en un rango de 2 a 32.

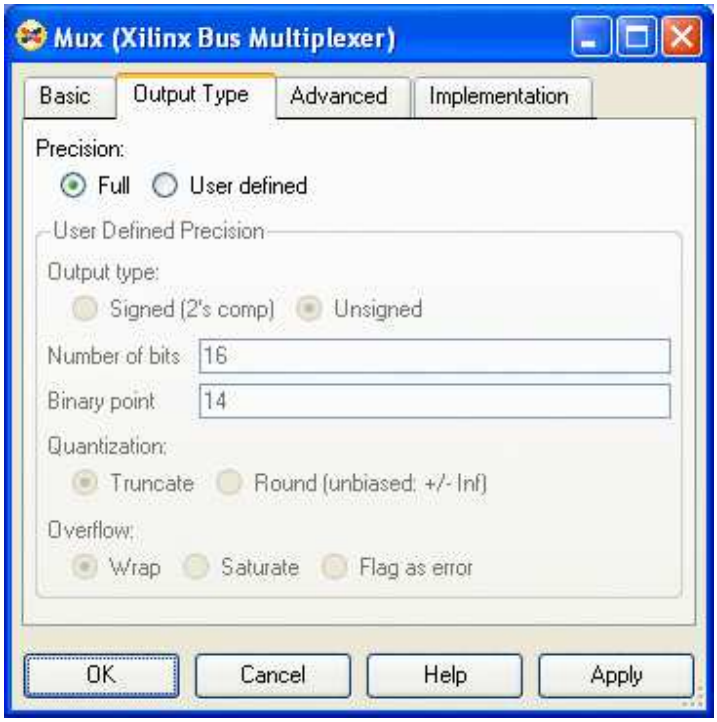


Figura 3.19. Parámetros de Salida del Bloque Mux

3.4.6. CONSTANT

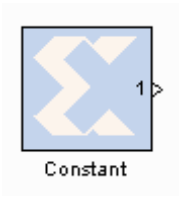


Figura 3.20. Bloque Constante

Este bloque permite asignar el valor de una constante, este valor debe ser asignado tomando en consideración el número de bits y el tipo de dato ya que puede ser booleano o de punto fijo.

Las diferentes librerías del Blockset de Xilinx donde se encuentra este bloque son: Basic Elements, Control Logic, Math e Index.

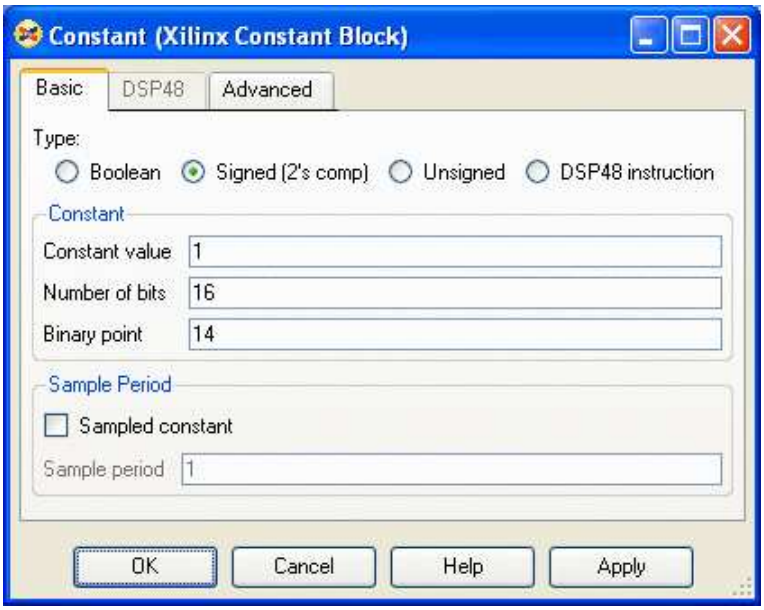


Figura 3.21. Parámetros Básicos del Bloque Constant

Los parámetros específicos mostrados en la figura 3.21 se detallan a continuación:

Constant Value: Permite especificar el valor de la constante.
Sample Constant: Al marcar esta opción se asocia un período de muestreo con la salida.

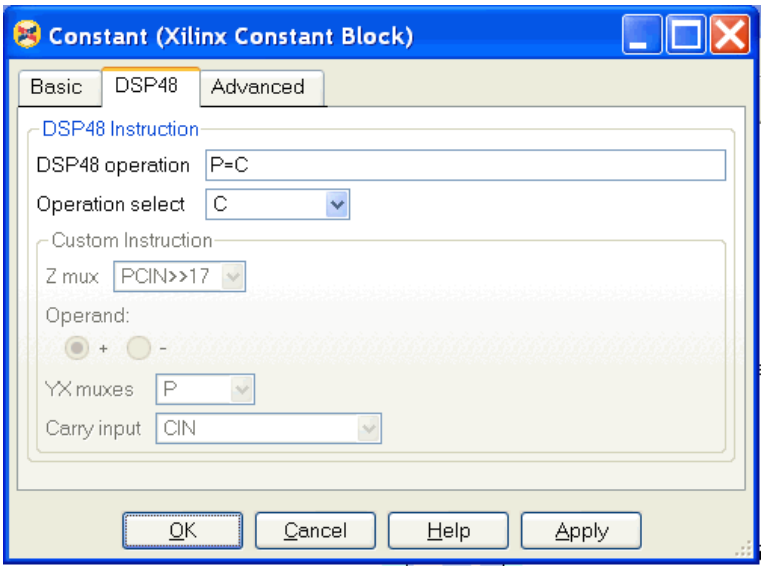


Figura 3.22. Parámetros de Instrucciones en Modo DSP48

La ficha DSP48 se activa sólo cuando en tipo se ha elegido modo DSP48, y permite escoger ciertas características de acuerdo a un formato de instrucciones de control, mismo que esta fuera del alcance de este proyecto dado que, como se verá posteriormente, en el diseño el DSP48 no ha sido utilizado y por ende sus instrucciones de control.

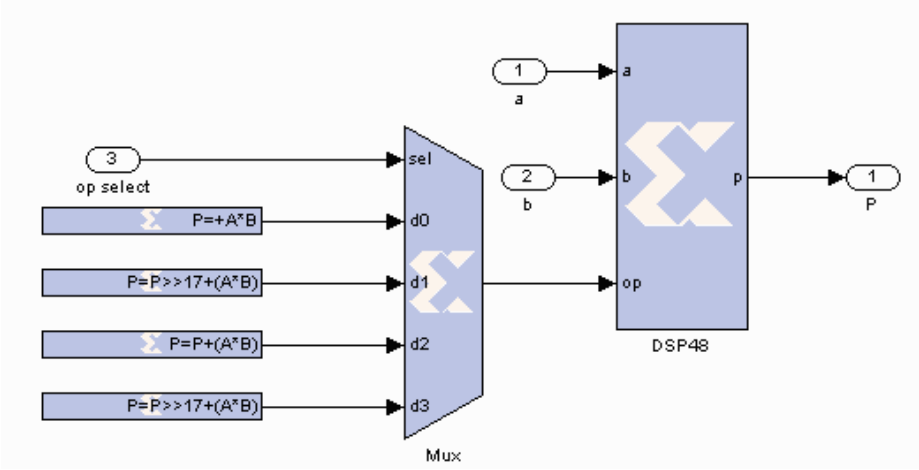


Figura 3.23. Ejemplo de Uso del Bloque Constante como Control de Instrucciones para DSP48

3.4.7. DELAY

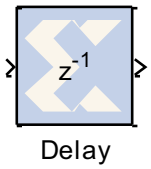


Figura 3.24. Bloque Delay

Este bloque permite que los datos de la señal de entrada se presenten después de un cierto tiempo en la salida. Este retardo se especifica por el usuario teniendo en cuenta el período de muestreo del sistema. El valor inicial de este bloque es cero y no es configurable.

Las diferentes librerías del Blockset de Xilinx donde se encuentra este bloque son: Basic Elements, Memory e Index.

Figura 3.25. Parámetros Básicos del Bloque Delay

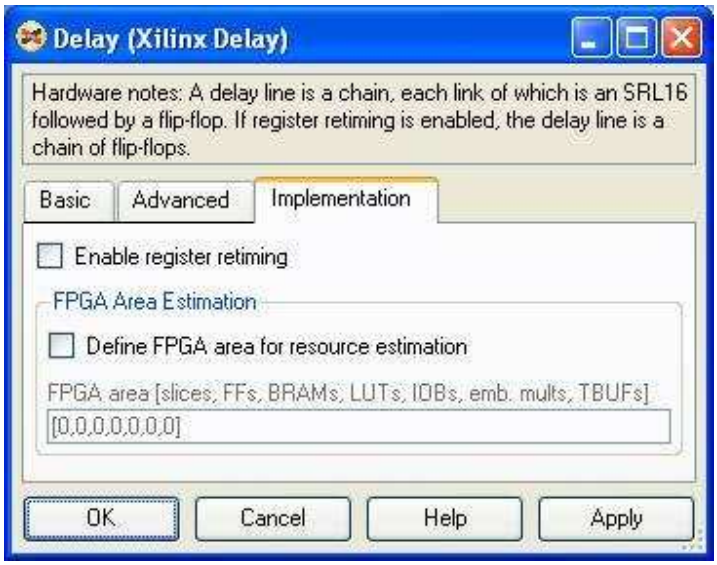


Figura 3.26. Parámetros de Implementación del bloque Delay

Los parámetros específicos mostrados en la figura 3.25 se detallan a continuación:

Enable register retiming: Al realizar la implementación se usará una cadena de Flip-Flops si se activa este campo. Cuando se sintetiza de esta forma, se hará uso de una gran cantidad de slices.

3.4.8. COUNTER

Figura 3.27. Bloque Counter

El bloque counter implementa contadores ascendentes, descendentes, o ascendentes / descendentes dentro del FPGA. Los contadores generados por este bloque pueden ser limitados a un valor determinado o de cuenta libre; sin embargo en el segundo caso su valor también se verá limitado por el número de bits que se maneje en la salida del contador. Adicionalmente, éste consume menos recursos que el contador limitado, pues no necesita un comparador.

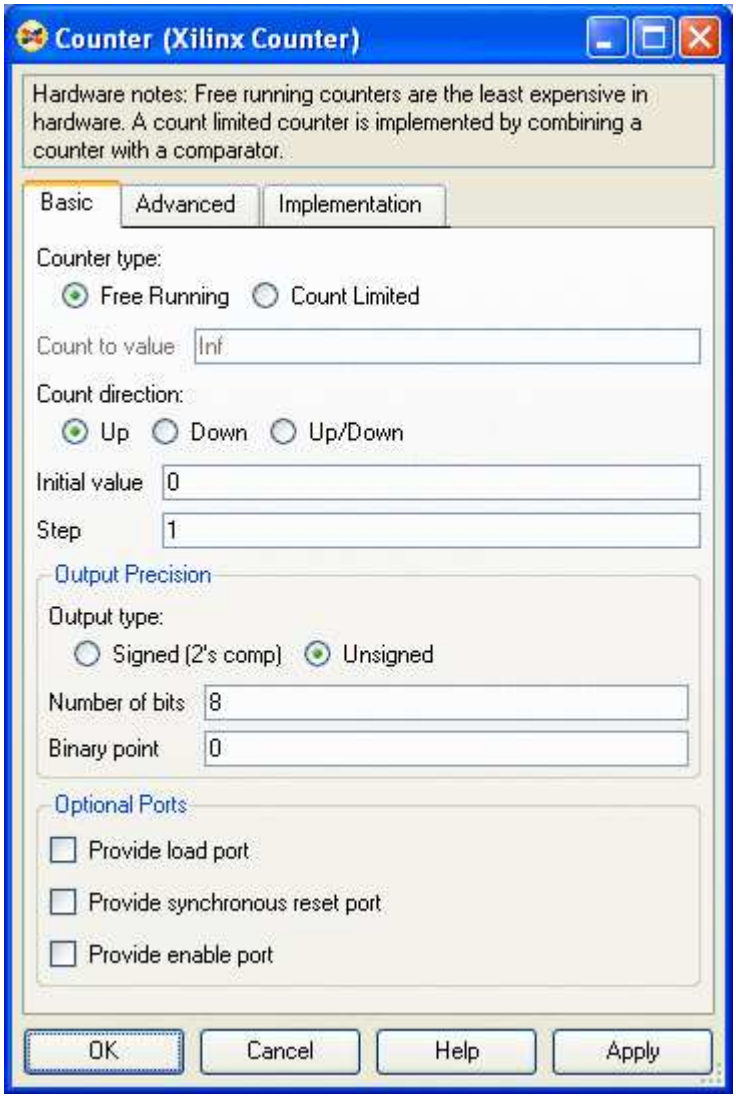


Figura 3.28. Parámetros Básicos del Bloque Counter

Las diferentes librerías del Blockset de Xilinx donde se encuentra este bloque son: Basic Elements, Control Logic, Math e Index.

Los parámetros específicos mostrados en la figura 3.28 se detallan a continuación:

Counter Type: Permite elegir entre contador de cuenta libre o un contador limitado a un valor determinado.
Count to value: En este campo se debe colocar el valor al cual se va a limitar el contador, cabe recalcar que en el caso de elegir un contador de cuenta libre éste campo estará deshabilitado.
Count Direction: Permite escoger si el contador es ascendente, descendente o ascendente/descendente.
Initial Value: En este campo se debe colocar el valor en el cual se iniciará la cuenta.
Step: Permite colocar el incremento o decremento en la cuenta, dependiendo la dirección elegida.
Provide load port: Esta opción es habilitada sólo para contadores de cuenta libre y permite obtener contadores con carga y entradas explícitas.

3.4.9. SERIAL TO PARALLEL

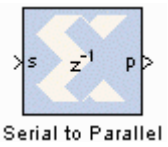


Figura 3.29. Bloque Serial to Parallel

Este bloque toma una cierta cantidad de datos en forma serial en su entrada y crea en su salida un único dato. El siguiente gráfico ilustrará mejor el comportamiento de este bloque.

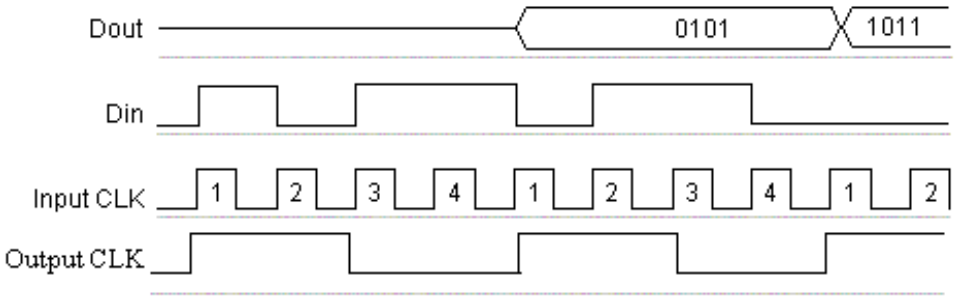


Figura 3.30. Ejemplo del comportamiento Serial to Parallel[2]

En este ejemplo se ilustra el caso en donde el ancho o la duración del dato en la entrada es 1, y la duración en la salida es 4. Aquí los datos son bits y el bloque está configurado para que el bit más significativo MSB sea el primero.

Las diferentes librerías del Blockset de Xilinx donde se encuentra este bloque son: Basic Elements, Data Types e Index.

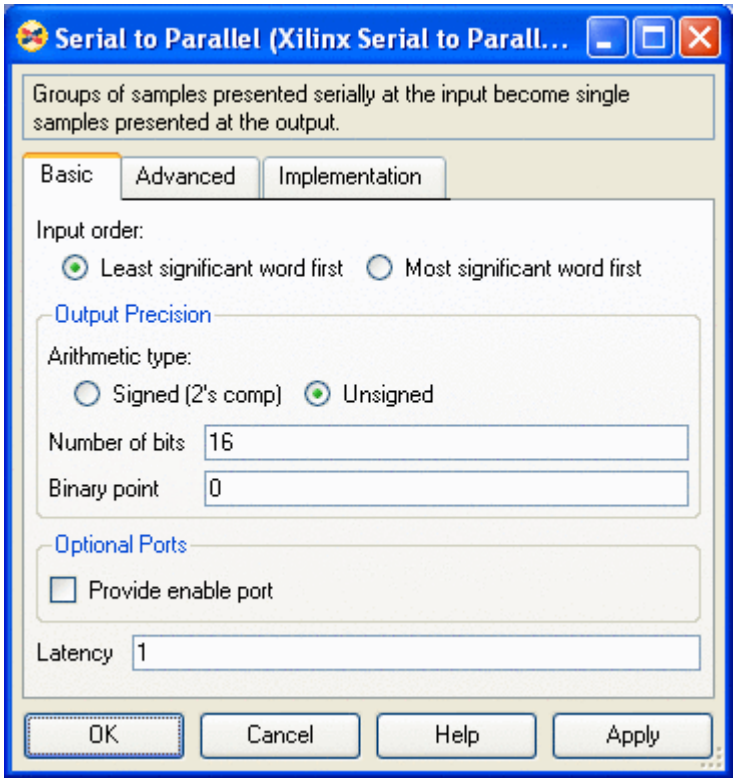


Figura 3.31. Parámetros de Básicos del bloque Serial to Parallel

[2] Matlab, Xilinx System Generator v8.1 User Guide, Blockset Reference Pages, Block Reference, Serial to Parallel, figure 1.

Este bloque tiene únicamente un puerto de entrada y un puerto de salida. Además el dato en la entrada puede tener cualquier tamaño, pero aquel en la salida es indicado en los parámetros.
Los parámetros específicos mostrados en la figura 3.31 se detallan a continuación:

Input order: El bit menos o más significativo es el primero.
Arithmetic type: Salida con signo en complemento de 2 o sin signo.
Number of bits: Ancho de la salida, el cual debe ser un múltiplo del número de bits que están en la entrada.
Binary point: Ubicación del punto binario en la salida.

Se muestra un error cuando el número de bits de salida no es divisible para el número de bits en la entrada.

3.4.10. PARALLEL TO SERIAL

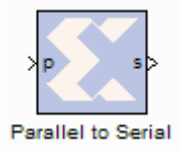


Figura 3.32. Bloque Parallel To Serial

Este bloque toma un dato de entrada y lo divide, obteniendo en la salida N partes multiplexadas en el tiempo, donde n es la relación entre el número de bits en la entrada y el número de bits en la salida. El mínimo retardo que admite este bloque es 1. El siguiente gráfico ilustrará mejor el comportamiento de este bloque.

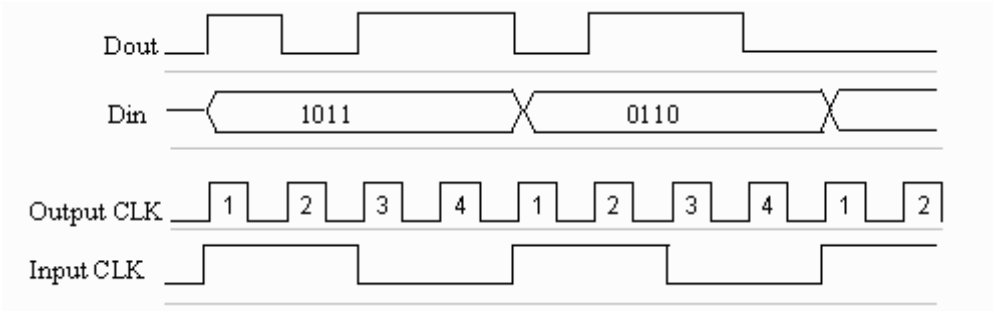


Figura 3.33. Ejemplo del comportamiento Parallel to Serial[3]

[3] Matlab, Xilinx System Generator v8.1 User Guide, Blockset Reference Pages, Block Reference, Parallel to Serial, figure 1.

En este ejemplo se ilustra el caso en donde el ancho o la duración del dato en la entrada es 4, y la duración en la salida de cada bit es 1. El bloque está configurado para que el bit más significativo MSB sea el primero.

Las diferentes librerías del Blockset de Xilinx donde se encuentra este bloque son: Basic Elements, Data Types, e Index.

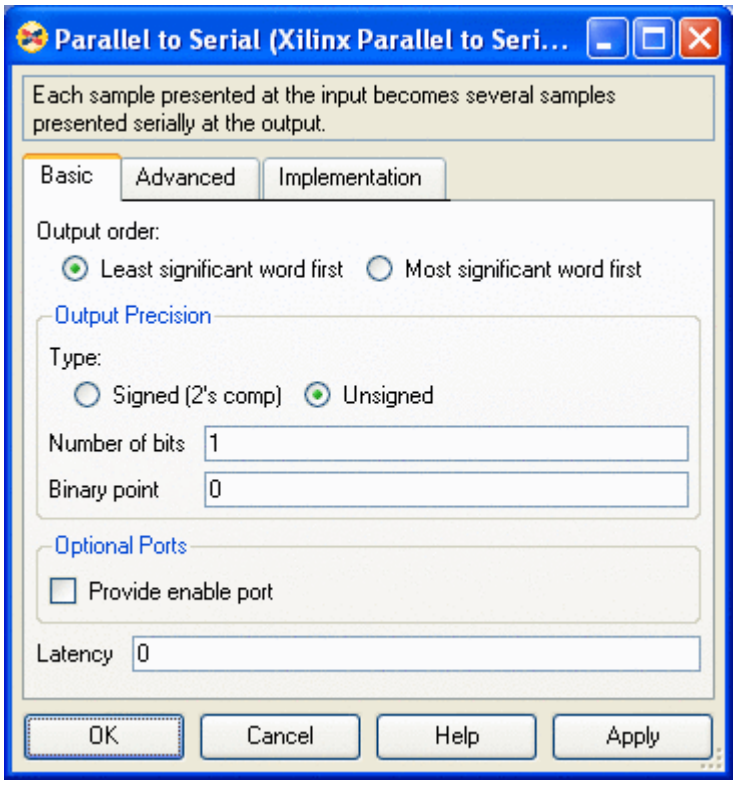


Figura 3.34. Parámetros Básicos del bloque Parallel to Serial

Los parámetros específicos mostrados en la figura 3.34 se detallan a continuación:

Output order: El bit menos o más significativo es el primero.
Type: Salida con signo en complemento de 2 o sin signo.
Number of bits: Ancho de la salida, el cual debe ser un número divisible para el número de bits que están en la entrada.
Binary point: Ubicación del punto binario en la salida.

3.4.11. DOWN SAMPLE

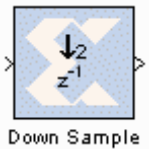


Figura 3.35. Bloque Down Sample

El bloque Down Sample realiza un proceso de decimación, es decir divide la tasa de muestreo que tiene la señal de entrada para un valor constante.

La señal de entrada es muestreada al inicio (el primer valor) o al final (el último valor) de la trama. El valor muestreado es presentado en el puerto de salida y sostenido hasta que la próxima muestra sea tomada.

Una trama Down Sample consiste en L muestras de entrada, donde L representa la velocidad de muestreo. En la figura 3.36 se presenta un ejemplo, donde la velocidad de muestreo es 4.

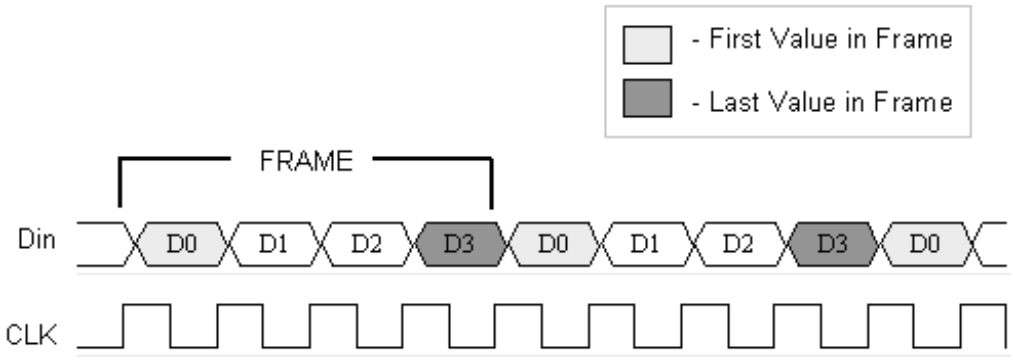


Figura 3.36. Ejemplo de una Trama del Bloque Down Sample[4]

Las diferentes librerías del Blockset de Xilinx donde se encuentra este bloque son: Basic Elements e Index.

[4] Matlab, Xilinx System Generator v8.1 User Guide, Blockset Reference Pages, Block Reference, Down Sample, figure 1.

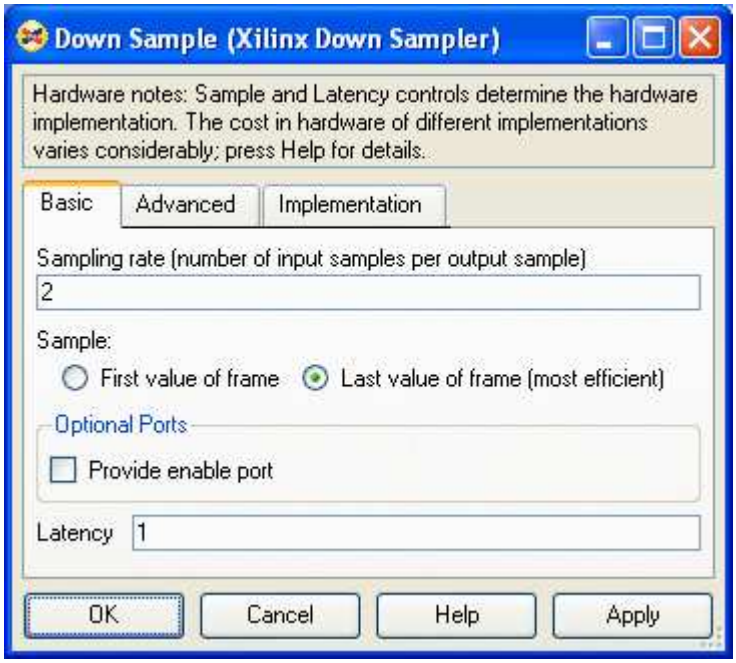


Figura 3.37. Parámetros Básicos del Bloque Down Sample

Los parámetros específicos mostrados en la figura 3.37 se detallan a continuación:

Sampling rate: En este campo se debe colocar cuantas muestras de la entrada se desea que se conviertan en la salida, de esta forma se realiza el proceso de decimación. Este valor debe ser mayor o igual a 2.

Sample: Este parámetro permite elegir si se tomará la primera o la última muestra de la trama.

3.4.12. UP SAMPLE

Figura 3.38. Bloque Up Sample

El bloque Up Sample incrementa la tasa de muestreo, mediante la división del período de muestreo y la velocidad de muestreo de la entrada.

Las diferentes librerías del Blockset de Xilinx donde se encuentra este bloque son: Basic Elements e Index.

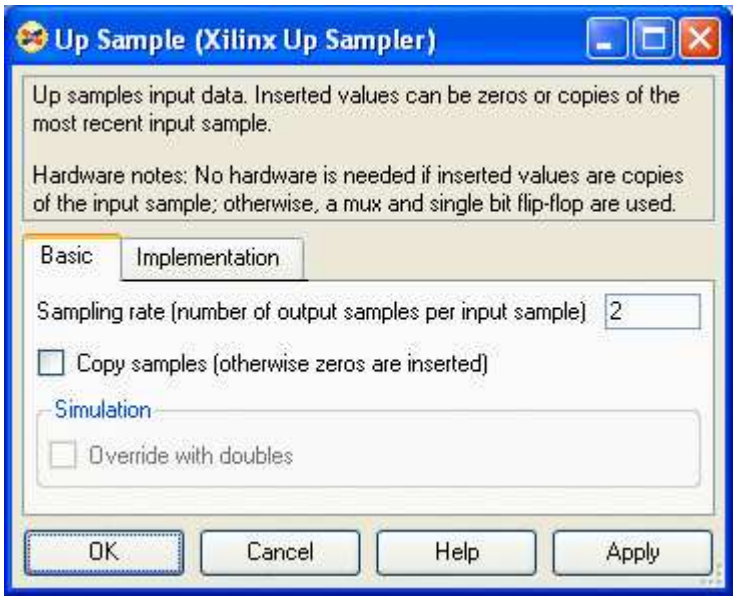


Figura 3.39. Parámetros Básicos del Bloque Up Sample

Los parámetros específicos mostrados en la figura 3.39 se detallan a continuación:

Sampling rate: En este campo se debe colocar cuantas muestras se desea obtener a la salida por cada muestra en la entrada del bloque, este valor debe ser un entero mayor o igual a 2.

Copy samples: Si esta casilla es seleccionada, las muestras que se insertan tienen el mismo valor de la muestra de entrada cuando ya se ha disminuido el período. Si la opción no es habilitada se insertan ceros como se indica en la figura 3.40.

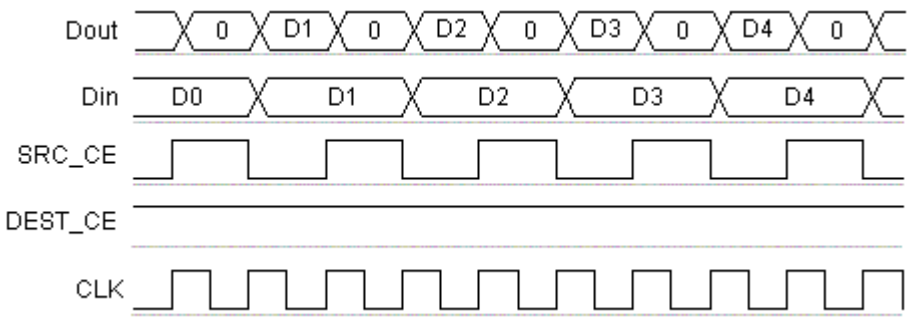


Figura 3.40. Ejemplo de funcionamiento del Bloque Up Sample[5]

[5] Matlab, Xilinx System Generator v8.1 User Guide, Blockset Reference Pages, Block Reference, Up Sample, figure 2.

3.4.13. ADDSUB

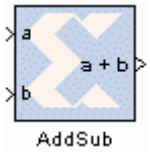


Figura 3.41. Bloque AddSub

Este bloque implementa un sumador o un restador dentro del FPGA según se elija o de acuerdo a una señal booleana de entrada.
Las diferentes librerías del Blockset de Xilinx donde se encuentra este bloque son: Math e Index.

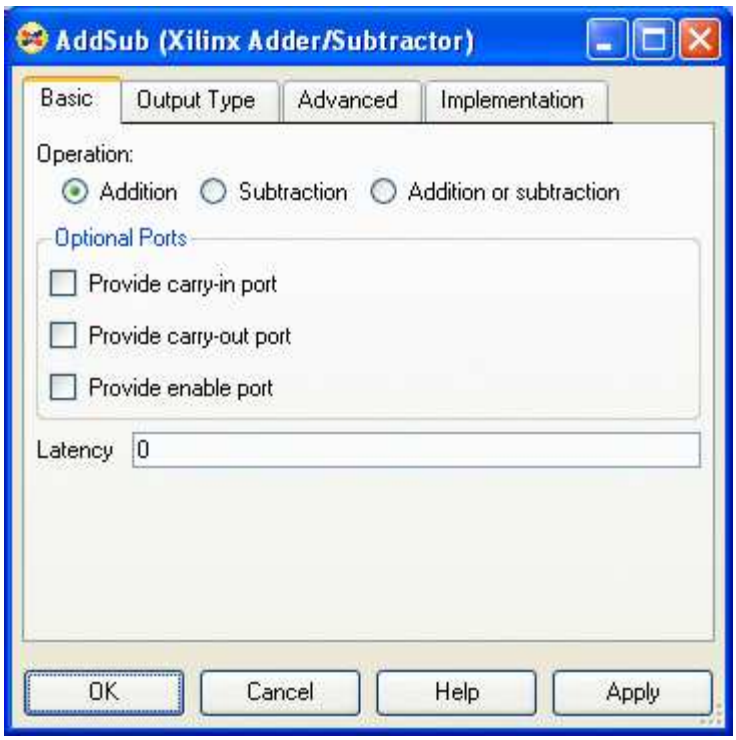


Figura 3.42. Parámetros Básicos del Bloque AddSub

Los parámetros específicos mostrados en la figura 3.42 se detallan a continuación:

Operation: Este parámetro permite definir la operación que realizará el bloque entre adición, sustracción, o adición/sustracción. En este último caso la operación se controlará por medio de una señal booleana, de tal forma que se realizará la adición cuando esta señal sea 0 y la sustracción cuando esté en 1.

Provide carry-in Port: Al seleccionar esta casilla, se crea una entrada adicional *carry-in (cin)*.

Provide carry-out Port: Al seleccionar esta casilla, permite tener una salida de *carry-out (cout)*.

3.4.14. MULT

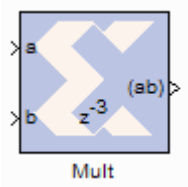


Figura 3.43. Bloque Mult

Este bloque implementa un multiplicador, que calcula el producto de los datos que se encuentran en los dos puertos de entrada (a y b).

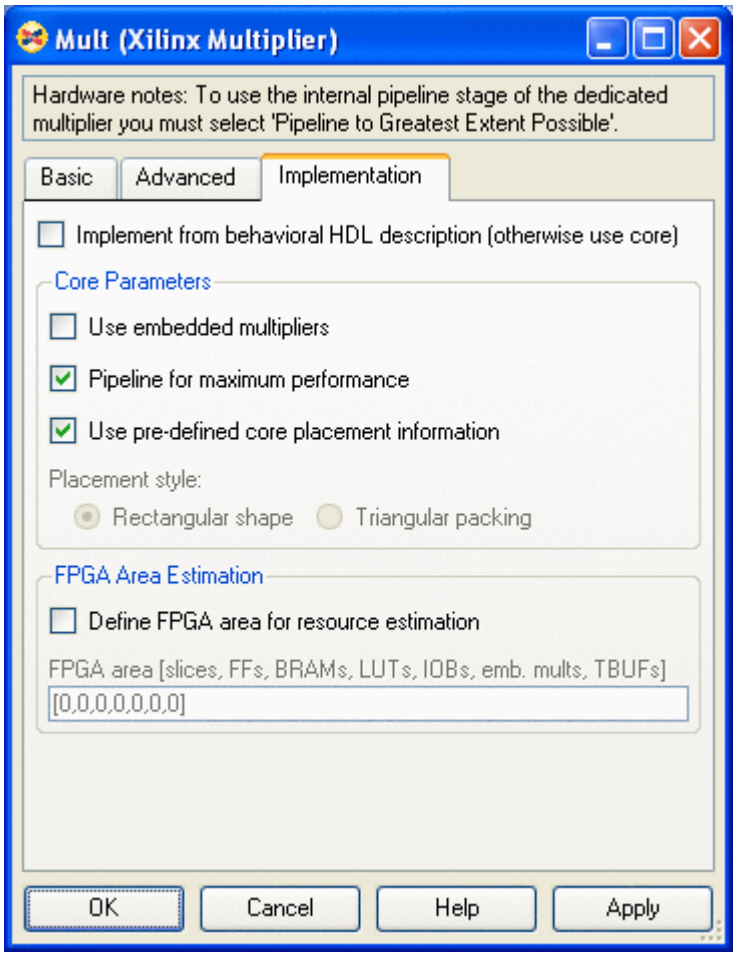


Figura 3.44. Parámetros de Implementación del bloque Mult

Las diferentes librerías del Blockset de Xilinx donde se encuentra este bloque son: Math e Index.

Los parámetros específicos mostrados en la figura 3.44 se detallan a continuación:

Use embedded multipliers: Si se selecciona, el bloque utiliza multiplicadores embebidos. Únicamente disponible en tarjetas de entrenamiento que contienen FPGAs tipo Virtex-II, Virtex-4 y Spartan-3.

Pipeline for maximum performance: Al seleccionar esta casilla, el multiplicador se implementa con la máxima eficiencia posible, haciendo que mejore la velocidad de procesamiento.

3.4.15. ROM

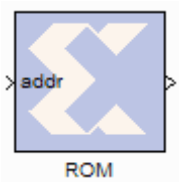


Figura 3.45. Bloque ROM

Este bloque es una memoria de sólo lectura (*Read Only Memory*), que almacena valores en sus distintas localidades. Cada valor está asociado con una dirección, la cual es un número sin signo y de tipo entero desde 0 hasta d-1; donde d (*depth*) es el número de valores o palabras que aloja la ROM.

Como particularidad de este bloque, se tiene que todos los valores almacenados en la ROM poseen la misma representación aritmética.

Las diferentes librerías del Blockset de Xilinx donde se encuentra este bloque son: Control Logic, Memory, e Index.

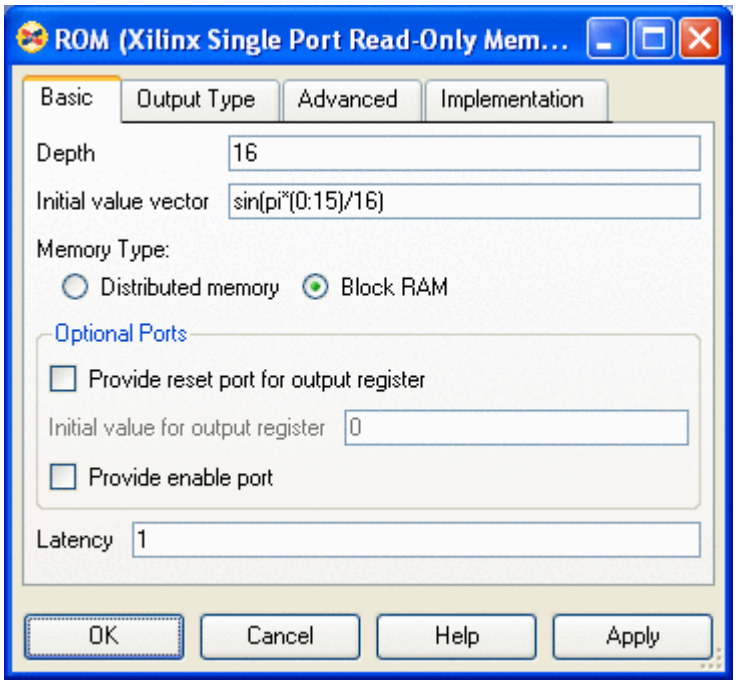


Figura 3.46. Parámetros Básicos del bloque ROM

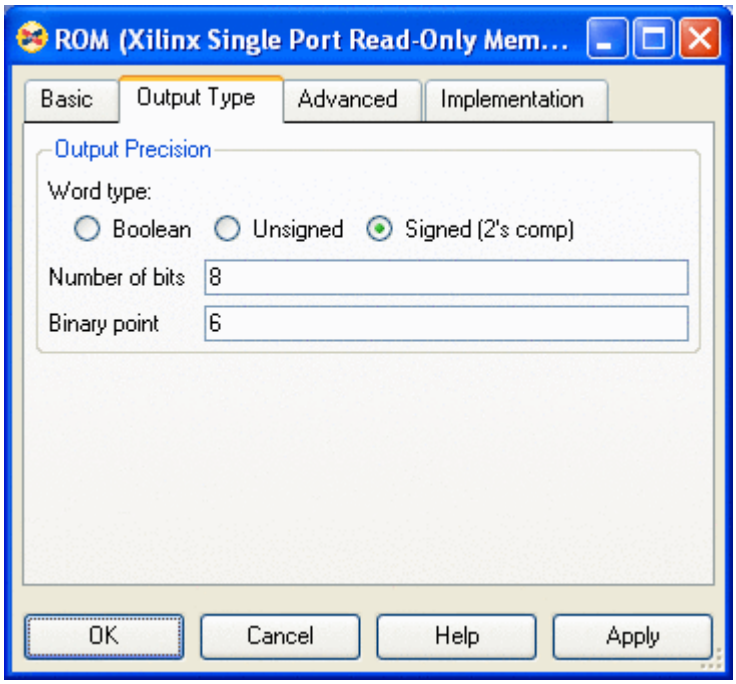


Figura 3.47. Parámetros del Tipo de Salida del bloque ROM

Los parámetros específicos mostrados en la figuras 3.46 y 3.47 se detallan a continuación:

Depth: Determina el número de palabras que se almacenan en la ROM, este debe ser un número entero positivo.

Initial value vector: Indica el valor inicial de la ROM. Si el tamaño del vector es mayor que el número de palabras (*depth*), los elementos que conforman el exceso son descartados. Si al contrario, el número de palabras de la ROM es mayor que el tamaño del vector su contenido es fijado en cero.
Memory Type: Determina las dos posibles formas de implementación que son: *Distributed Memory* y *Block Memory*.
Provide reset port for output register: Proporciona un puerto de reset para el bloque, siempre y cuando la latencia sea 1.
Initial value for output register: Fija el valor inicial en la salida cuando el reset se activa. Esta opción está disponible en tarjetas de entrenamiento que contienen FPGAs tipo Virtex-II, Virtex-II Pro, Virtex-4 y Spartan-3.
Word type: Tipo de datos, Booleano, con signo en complemento de 2 y sin signo.
Number of bits: Tamaño de la palabra de la ROM.
Binary point: Ubicación del punto binario en la salida en cada palabra.

Cuando se escoge *Block Memory*, el máximo número de palabras depende del tamaño de la palabra y es diferente en cada familia de FPGAs. En la figura 3.48 se proporcionan los valores máximos para diferentes familias.

Número de Palabras	Tamaño en Bits
2 to 2048	256
2049 to 4096	192
4097 to 8192	96
8193 to 16K	48
16K+1 to 32K	24
32K+1 to 64K	12
64K+1 to 128K	6

Figura 3.48. Máximo Número de Palabras según el Tamaño (Virtex/Virtex-E/Spartan-3)

Al contrario, cuando se opta por *Distributed Memory*, el número de palabras debe estar entre 16 y 65536 inclusive para la familia Spartan-3.

3.4.16. DAFIR v9_0

Figura 3.49. Bloque DAFIR v9_0

Este bloque implementa un filtro digital de respuesta finita al impulso FIR, con coeficientes distribuidos aritméticamente. El número de coeficientes del filtro 'N' se denomina también longitud del filtro y además N-1 es el orden del filtro. En el bloque DAFIR de la figura 3.49, se indica la longitud del filtro como 2-tap.

Es posible configurar el filtro en modo multicanal para procesar señales en paralelo, si se requieren varios filtros con la misma respuesta.

Las diferentes librerías del Blockset de Xilinx donde se encuentra este bloque son: DSP e Index.

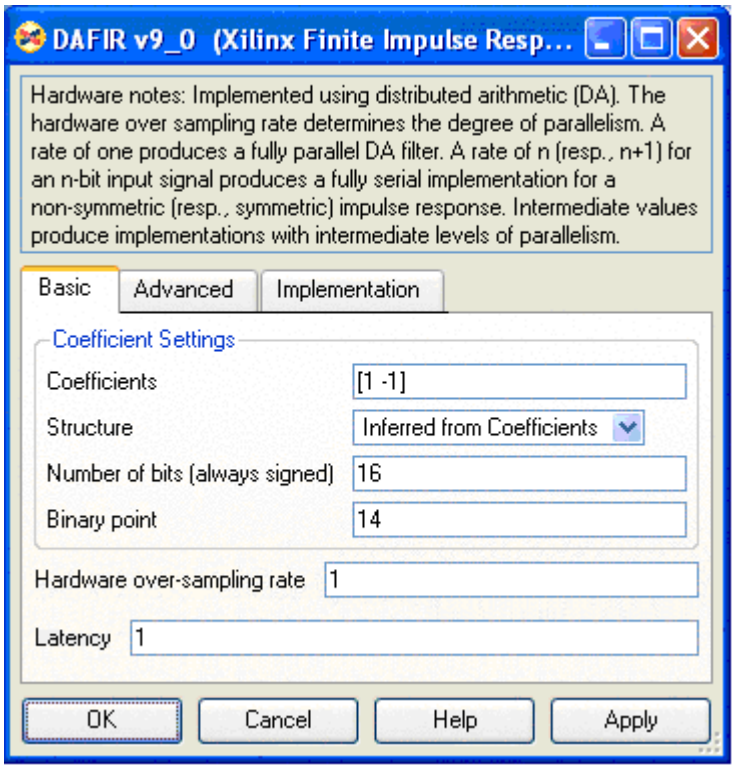


Figura 3.50. Parámetros Básicos del Bloque DAFIR v9_0

Los parámetros mostrados en la figuras 3.50 y 3.51 se detallan a continuación:

Coefficients: Define el vector de coeficientes que utiliza el filtro, estos valores serán generados por el bloque FDATool explicado más adelante.
Structure: La estructura de los coeficientes se puede escoger entre varias opciones, pero si los coeficientes se van a determinar con FDATool la más apropiada es *Inferred from coefficients* (inherente a la naturaleza de los coeficientes).
Number of bits: Indica el número de bits para representar los coeficientes del filtro, los cuales son de punto fijo y siempre con signo.
Binary Point: Especifica cuantos bits serán usados para la parte fraccionaria de los coeficientes.
Hardware over-sampling Rate: En la implementación, define el número de ciclos de reloj que se utilizan para procesar una muestra.

Figura 3.51. Parámetros Avanzados del Bloque DAFIR v9_0

Number of channels: Se puede elegir entre uno y ocho canales. Para filtros multicanal, el tiempo de muestreo de las señales de entrada debe ser igual para todos los canales.
Serial input: En modo multicanal, la entrada puede estar en serie o paralelo.

Polyphase behavior: Determina el comportamiento en la salida del filtro, según las muestras que existan en la salida, las opciones son: *Single rate, Decimation e Interpolation*.

3.4.17. FDATool

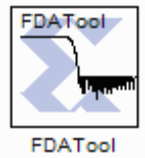


Figura 3.52. Bloque FDATool

En realidad este bloque no es una función independiente, es una interfaz de FDATool que es parte del *Toolbox Signal Processing* de MATLAB, y no funcionará si este toolbox no está instalado. Esta interfaz proporciona un medio para definir y almacenar un objeto de FDATool en un modelo de SysGen para determinar un filtro.

La explicación de cada uno de los campos, funciones y parámetros de la interfaz gráfica de diseño en FDATool no se detalla aquí, debido a que está fuera del alcance de este proyecto. Sin embargo, los campos utilizados en el diseño del filtro para este proyecto, se muestran en la modelación del mismo en el capítulo 4. Para una referencia completa de la interfaz gráfica se debería utilizar la ayuda de MATLAB.

El objetivo del bloque FDATool de Xilinx es almacenar en una estructura interna los coeficientes del filtro diseñado, para después extraerlos con las funciones proporcionadas por System Generator que son: *xlfda_numerator('FDATool')* y *xlfda_denominator('FDATool')*.

3.5. IMPACT

Realiza dos funciones básicas que son: la configuración del dispositivo a través de un cable de programación y la generación de ficheros de configuración en

diferentes formatos. En el caso de la Spartan-3E se utiliza un cable USB para la programación, a pesar de utilizar una interfaz JTAG.

En este proyecto de titulación se aplicará sólo la primera función, por lo tanto se describirá el procedimiento de configuración y descarga.

En primer lugar se debe abrir el programa iMPACT, para después configurar el FPGA en la ventana de inicio que se muestra en la figura 3.53. En esta, se debe seleccionar *Automatically connect to cable and Identify Boundary-Scan,* de acuerdo con el manual de la propia tarjeta de entrenamiento, para luego dar clic en finalizar.

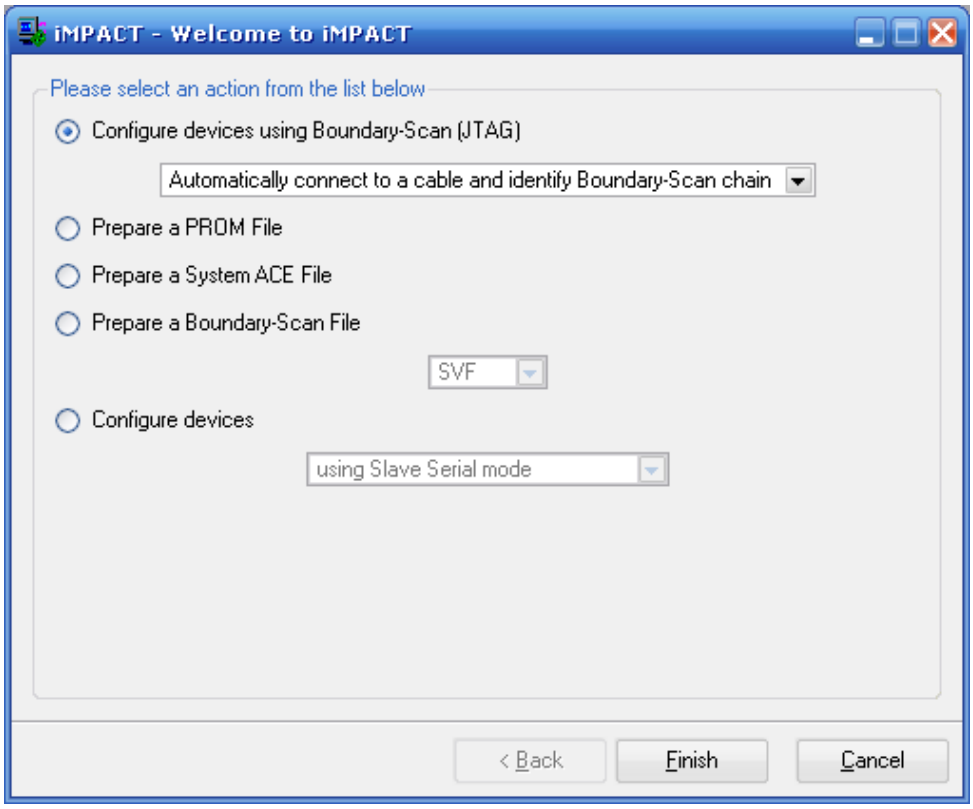


Figura 3.53. Ventana inicial de iMPACT

Luego, se seleccionará el archivo de configuración del dispositivo FPGA, el cual es un archivo bitstream con extensión *.bit*.

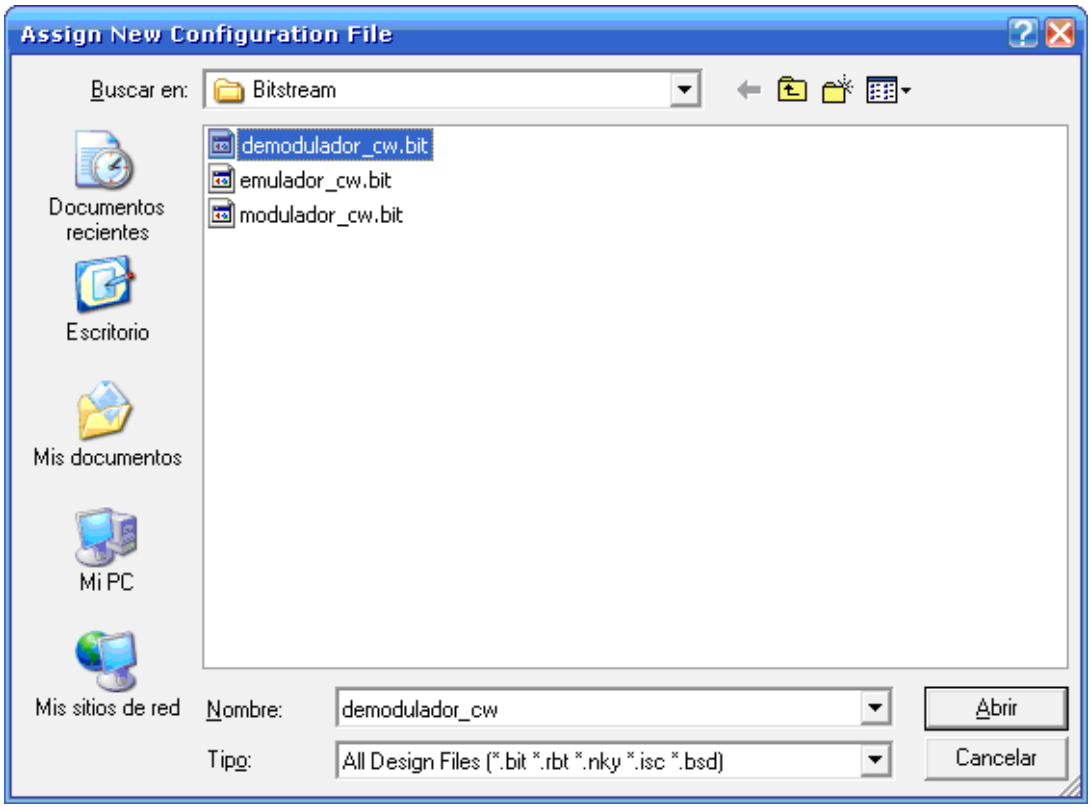


Figura 3.54. Ventana de Exploración para la Asignación del Bitstream

En las siguientes dos ventanas que aparecerán se debe presionar bypass, ya que la cadena de programación incluye también un archivo para la memoria flash y otro para el CPLD, los cuales en este proyecto de titulación no van a ser usados.

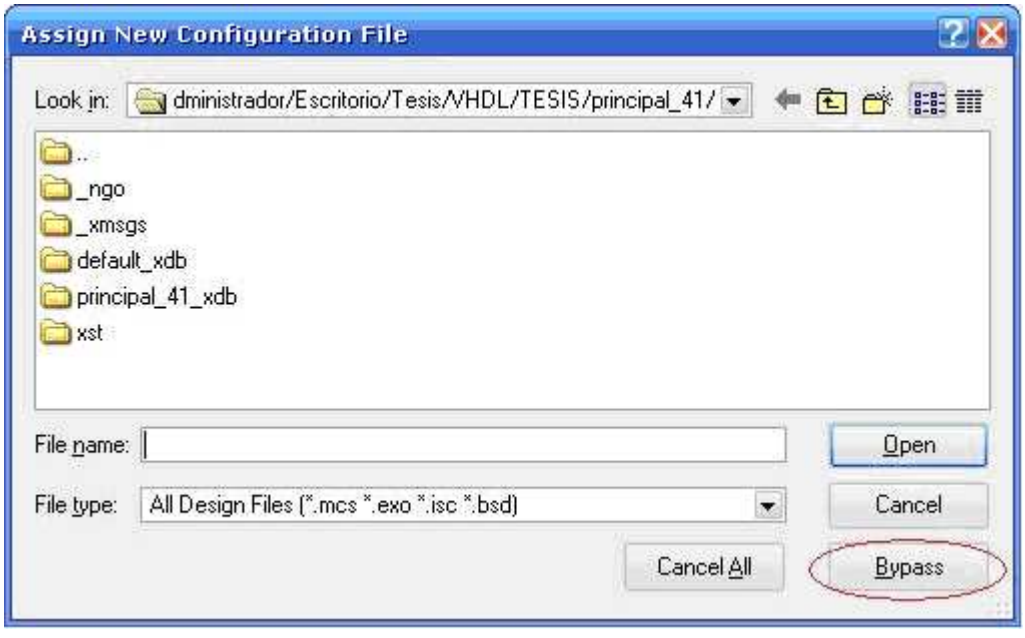


Figura 3.55. Ventana de Exploración del Archivo de Configuración, memoria Flash y CPLD

Una vez terminado este proceso, aparecerá la ventana de programación del dispositivo en la que se debe dar clic derecho sobre el FPGA y escoger la opción de programación.

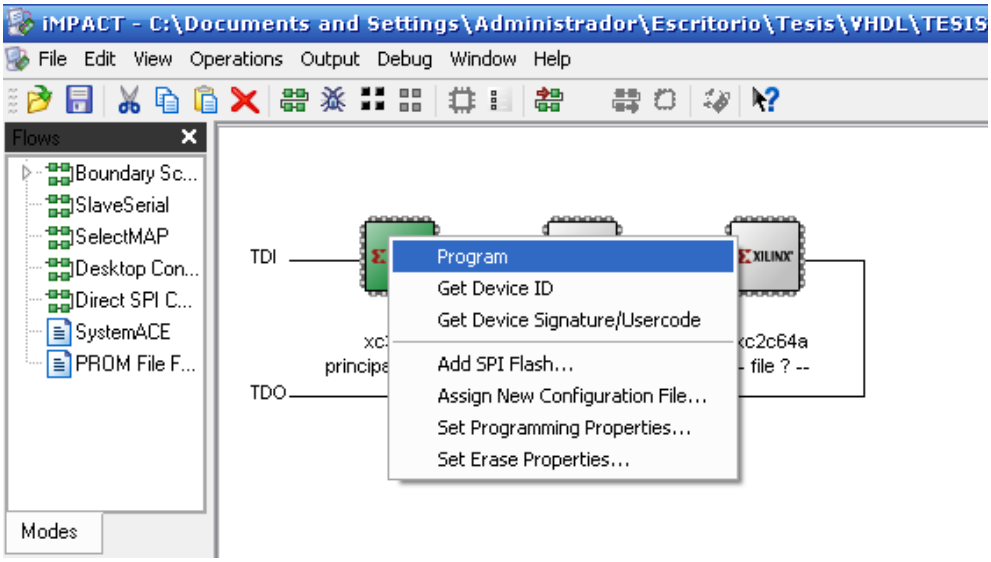


Figura 3.56. Ventana de Programación de iMPACT

Se acepta las condiciones de programación y se descarga el archivo de configuración del dispositivo.

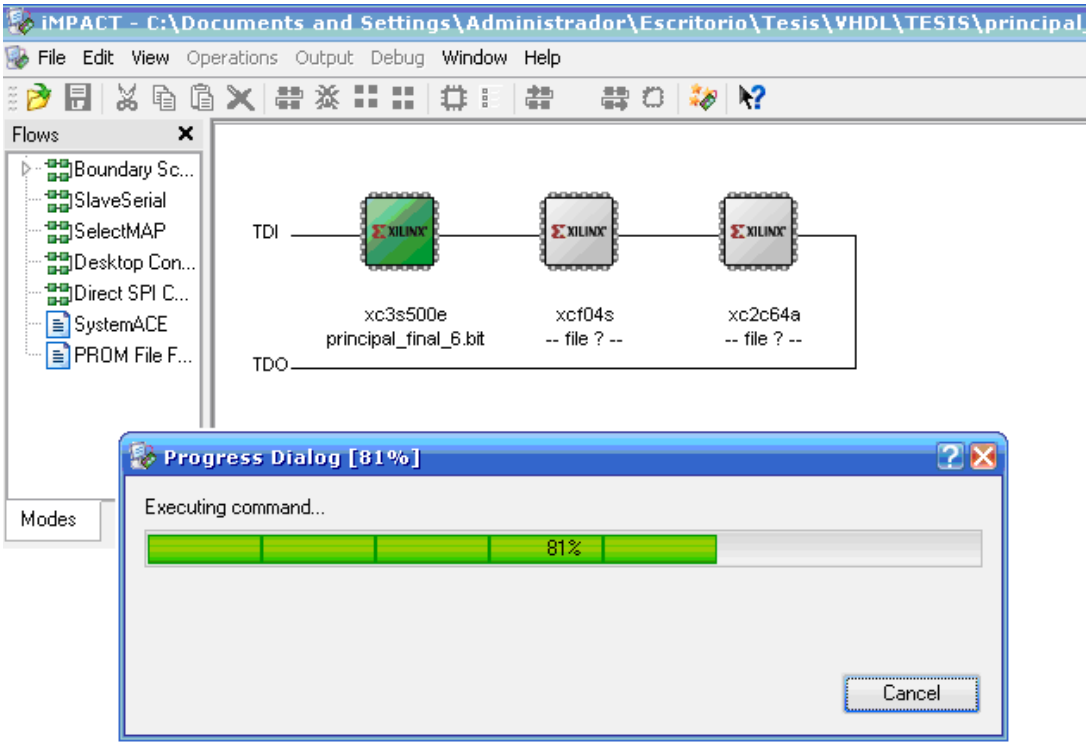


Figura 3.57. Ventana del Proceso de Descarga de iMPACT

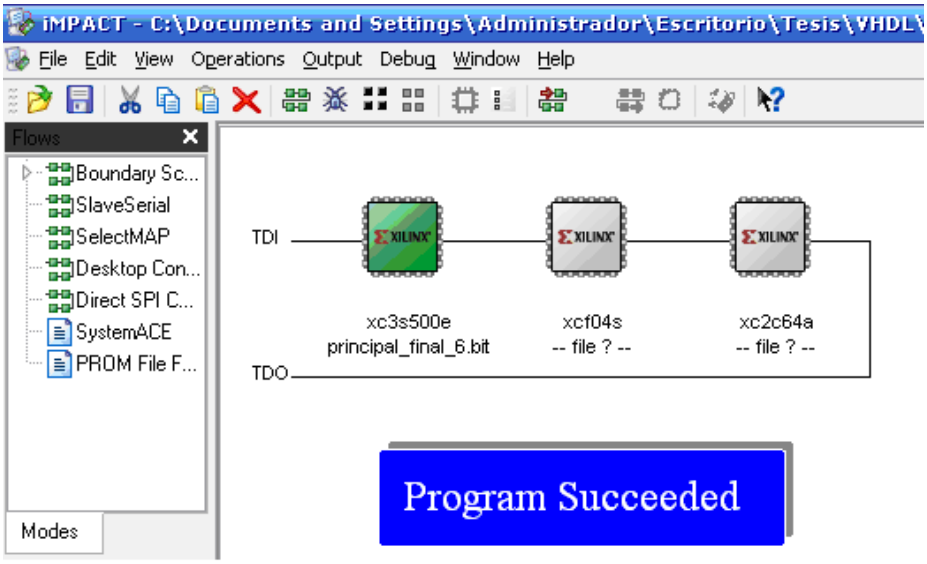


Figura 3.58. Ventana de Programación exitosa o fallida de iMPACT

Por último, se observa el mensaje de éxito o falla en el procedimiento de descarga del bitstream, finalizando así con la implementación del proyecto.

CAPÍTULO IV

4. DISEÑO Y SIMULACIÓN EN SIMULINK CON XILINX SYSTEM GENERATOR

4.1. DISEÑO DEL MODULADOR Y DEMODULADOR n-QAM

Para iniciar el diseño, es necesario segmentar la funcionalidad tanto del modulador como del demodulador para facilitar la abstracción del problema y su posterior solución. Además, la visualización de un gran sistema como un conjunto de componentes o subsistemas simplifica el análisis y la depuración del diseño. De este modo, el esquema imprescindible para utilizarlo en el estudio y la implementación de estos sistemas es un diagrama de bloques. En las figuras 4.1 y 4.2 se muestran los diagramas de bloques del modulador y del demodulador QAM respectivamente.

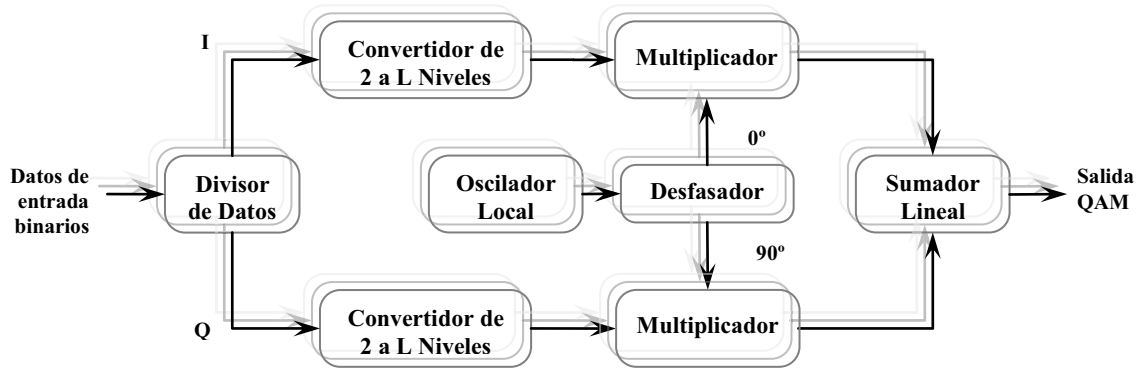


Figura 4.1. Diagrama General de Bloques del Modulador QAM

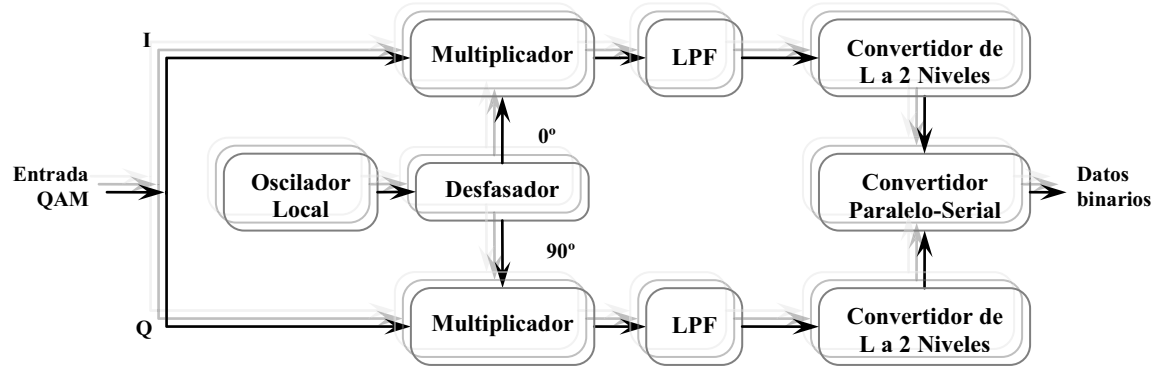


Figura 4.2. Diagrama General de Bloques del Demodulador QAM

Estos dos diagramas, además de facilitar el análisis, también establecen la estructura generalizada que se utilizará como base para el diseño de todos los tipos de modulación QAM que se implementarán en el proyecto.

4.1.1. CONDICIONES GENERALES DE DISEÑO

Dentro de las condiciones que de forma general delimitan el diseño están: los valores que toma 'n' para determinar los múltiples estados de QAM, la frecuencia del Oscilador Local y la ausencia del módulo de recuperación de portadora en el demodulador.

4.1.1.1. Delimitación de n-QAM

Es necesario precisar cuáles valores tomará 'n' para determinar el alcance del proyecto y delimitar en su funcionalidad tanto al modulador como al demodulador QAM. Entonces, como se tiene una especificación general de n-ario, se establece que n toma los valores de 4, 16 y 64; esto quiere decir que se diseñará un modulador 4, 16 y 64-QAM y su respectivo demodulador sobre una estructura generalizada.

4.1.1.2. Frecuencia de la Portadora

Dentro de este aspecto, una de las condiciones impuestas para el diseño, es que la frecuencia generada por el oscilador local es única e invariable. Además, como se utiliza una tarjeta de entrenamiento basada en FPGA, la cual contiene una señal de reloj f_{ck}, restringe la frecuencia de cualquier onda generada en ella.

Por ejemplo, una señal cuadrada periódica puede tener como máximo una frecuencia f_{ck}, a causa de aquello, una señal senoidal que consta de 'x' muestras para ser creada por período, tendría una frecuencia f_{ck}/x; la cual es obviamente 'x' veces inferior a f_{ck}. Tanto f_{ck} como la frecuencia de muestreo utilizada por la portadora se determinarán más adelante en el Token de System Generator.

4.1.1.3. Recuperación de la Portadora

En los sistemas con portadora suprimida, como PSK y QAM, se requieren métodos sofisticados de recuperación de la portadora, como un Circuito Cuadrado, un Circuito de Costas o un Remodulador[6].

Debido a que dicho estudio no forma parte del alcance del presente proyecto se ha prescindido de este bloque; sin embargo, una referencia local en el demodulador es necesaria, por lo que en lugar del bloque Recuperación de Portadora se coloca un Oscilador Local idéntico al del modulador, tanto en frecuencia como en fase, esto se muestra en la figura 4.2.

De esta manera, la portadora ya no se extrae de los datos modulados, sino por el contrario se origina localmente, logrando así la referencia requerida para la demodulación coherente y de esta manera la recuperación correcta de los datos binarios que fueron enviados.

4.1.2. DISEÑO Y SIMULACIÓN DE BLOQUES DEL MODULADOR

En el diagrama general de bloques del modulador de la figura 4.1, se muestran todos los componentes o subsistemas que se van a modelar. A continuación, se observarán tres secciones para la descripción de cada subsistema que son: el diseño, la configuración de los parámetros y la simulación en Simulink (FPGA).

4.1.2.1. Divisor de Datos

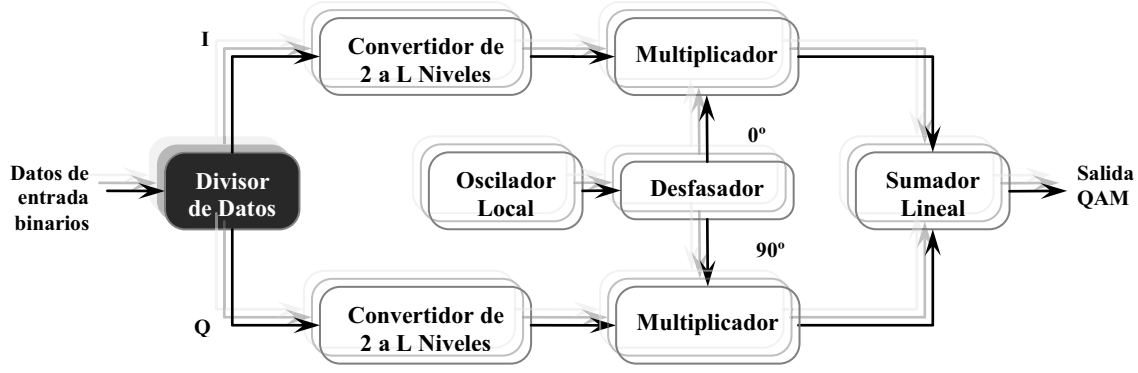


Figura 4.3. Divisor de Datos en el Modulador

[6] TOMASI, Wayne, Sistemas de Comunicaciones Digitales, p. 491.

Los bits de datos introducidos en forma serial llegan en primer lugar al divisor de datos, también llamado derivador, el cual permite la agrupación de N bits en la entrada del modulador n-QAM y su posterior separación en N/2 bits hacia cada canal I-Q.

Esta agrupación o combinación de 'N' bits en la entrada, se realiza para obtener 'n' condiciones o estados diferentes en la salida del modulador, como lo demuestra la siguiente relación:

$$n = 2^N$$

Donde:

n: Número de multiestados QAM

N: Número de bits agrupados en la entrada

Si en la relación anterior se despeja la variable 'N' se obtiene la siguiente expresión matemática:

$$N = \log_2 n$$

Por medio de la cual, se determina el valor de 'N' para cada tipo de modulación QAM, así como se muestra en la tabla 4.1.

n	4	16	64
N	2	4	6
N/2	1	2	3

Tabla 4.1. Agrupación de bits en la entrada para n-QAM

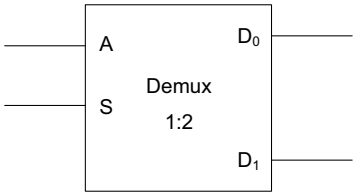
Entonces, se puede determinar que el derivador dirige 1, 2 o 3 bits a cada canal I-Q como se ilustra en la tabla 4.2.

N	Secuencia de bits	Canal I	Canal Q
2	$\ldots Q_2 I_2 Q_1 I_1$	$\ldots I_2 \_ I_1$	$\ldots Q_2 \_ Q_1 \_$
4	$\ldots Q_4 Q_3 I_4 I_3 Q_2 Q_1 I_2 I_1$	$\ldots \_ \_ I_4 I_3 \_ \_ I_2 I_1$	$\ldots Q_4 Q_3 \_ \_ Q_2 Q_1 \_ \_$
6	$\ldots Q_3 Q_2 Q_1 I_3 I_2 I_1$	$\ldots \_ \_ \_ I_6 I_5 I_4 \_ \_ \_ I_3 I_2 I_1$	$\ldots Q_6 Q_5 Q_4 \_ \_ \_ Q_3 Q_2 Q_1 \_ \_ \_$

Tabla 4.2. Separación de bits de entrada en canales I-Q

La secuencia de bits en realidad es cualquiera, pero se distingue en bits I-Q para facilitar su identificación y además los bits ingresan al modulador de derecha a izquierda. Se muestra de forma clara como los 6 bits, en el caso de 64-QAM, se separan de la secuencia de entrada hacia su canal respectivo en grupos de 3 bits ($I_3I_2I_1$ y $Q_3Q_2Q_1$) y de forma similar en los otros dos casos, para 4 y 16-QAM.
Los subguiones que separan cada grupo de N/2 bits, representan el tiempo que permanece sin recibir datos la entrada de un canal, mientras el otro canal está recibiendo datos.

El funcionamiento del bloque derivador de bits se describe en la tabla 4.2 y con una rápida inspección se determinaría que para lograr esto bastaría un demultiplexor 1:2 separando el flujo de bits, según la señal que se aplique en la selección, como se muestra en la figura 4.4. Este bloque no está disponible en las librerías de Xilinx, por lo que es necesario utilizar una combinación de los bloques que se estudiaron en el capítulo 3 para implementarlo.

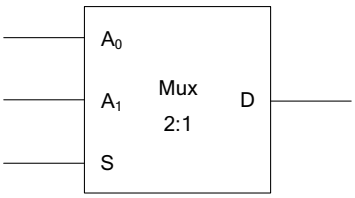


A	S	D1	D0
I	0	0	I
Q	1	Q	0

(a) (b)

Figura 4.4. (a) Bloque Demultiplexor 1:2; (b) Tabla de verdad del Demux 2:1

Si se observa con atención la salida D0 en la tabla de verdad de la figura 4.4, dependiendo del valor de la selección S, en un caso está presente a la salida 'I' y en el otro caso '0', tal como ocurre en la función de un multiplexor. Un comportamiento semejante se tiene en la salida D1.



A1	A0	S	D0
0	I	0	I
0	I	1	0

A1	A0	S	D1
Q	0	0	0
Q	0	1	Q

(a) (b) (c)

Figura 4.5. (a) Bloque Multiplexor 2:1; (b) Tabla de verdad del Mux 2:1 para el canal 'I'
(c) Tabla de verdad del Mux 2:1 para el canal Q

Entonces, se puede plantear la solución con dos multiplexores 2:1, con una entrada conectada a un valor constante (en este caso '0') y la otra en la entrada de datos. En la figura 4.5 (b) y (c) se definen las tablas de verdad de cada multiplexor para dividir la secuencia de bits de entrada. Además, mediante la variación de la señal aplicada en la entrada de selección 'S', se pueden obtener los bits requeridos en cada canal según la modulación QAM elegida. Si se aplica una señal cuadrada con las condiciones descritas a continuación se podrá generalizar este subsistema:

$$\frac{T_{sel}}{2} = \frac{1}{2} N \cdot tb_{in}$$

Donde:

T_{sel} : Período de la señal cuadrada de selección

N: Número de bits agrupados en la entrada

tb_{in} : Tiempo de bit, en la señal de entrada

La conexión de estos dos multiplexores 2:1 y la señal de selección, que se realiza en el ambiente de Simulink, se muestra en la figura 4.6.

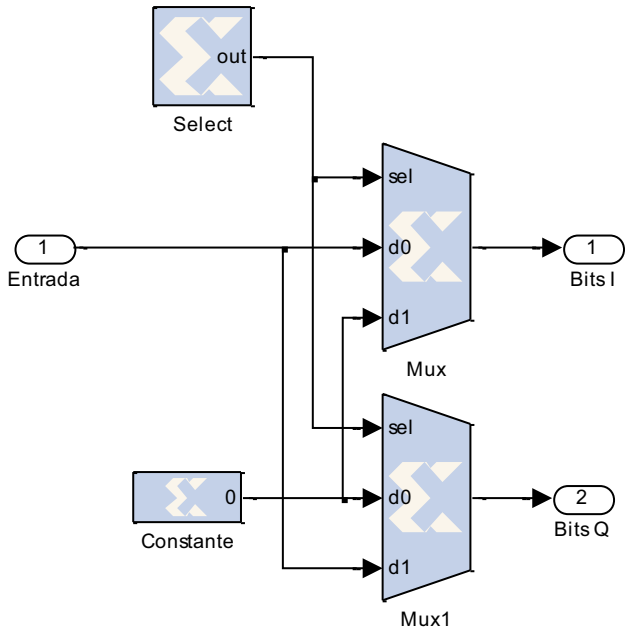


Figura 4.6. Modelación en Simulink con SysGen del bloque Divisor de Datos

Los datos ingresan hacia las dos entradas d0-d1 del multiplexor respectivo, para separarse en bits I-Q según la selección, que es la misma para ambos multiplexores. Las otras entradas d1-d0 se conectan a una constante para obtener el resultado de la tabla 4.2. y la señal cuadrada de selección, se obtiene mediante un contador ascendente módulo 2.

La configuración de parámetros de cada bloque que se utiliza en este subsistema, se muestra resumida en la tabla 4.4.

BLOQUE	PESTAÑA	PARÁMETROS	VALOR
Constante	*Basic*	*Type*	*Boolean*
		Constant value	*0*
Select	*Basic*	*Counter type*	*Free Running*
		Count direction	*Up*
		Initial Value	*0*
		Step	*1*
		Output type	*Unsigned*
		Number of bits	*1*
		Binary point	*0*
	Advanced	*Explicit Sample Period*	*1/2*log2(n)*
Mux y Mux1	*Basic*	*Number of inputs*	*2*
	Output Type	*Precision*	*User Defined*
		Output type	*Unsigned*
		Number of bits	*1*
		Binary point	*0*

Tabla 4.3. Configuración de los parámetros del Divisor de Datos

Para las pruebas del modulador, se utiliza una fuente que contiene el mensaje ‘nQAM’, el cual es representado en código ASCII con 8 bits por letra. La creación del subsistema Divisor de Datos en el ambiente de Simulink se muestra en la figura 4.7, así como la conexión del osciloscopio para la simulación.

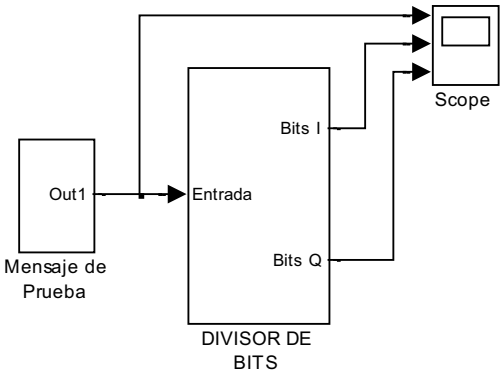


Figura 4.7. Subsistema del bloque Divisor de Datos creado en Simulink

Para 4-QAM:

Mensaje:	n	Q	A	M
Secuencia:	01101110	01010001	01000001	01001101
Bits I:	0_1_1_1_	0_0_0_0_	0_0_0_0_	0_0_1_0_
Bits Q:	_1_0_1_0	_1_1_0_1	_1_0_0_1	_1_0_1_1

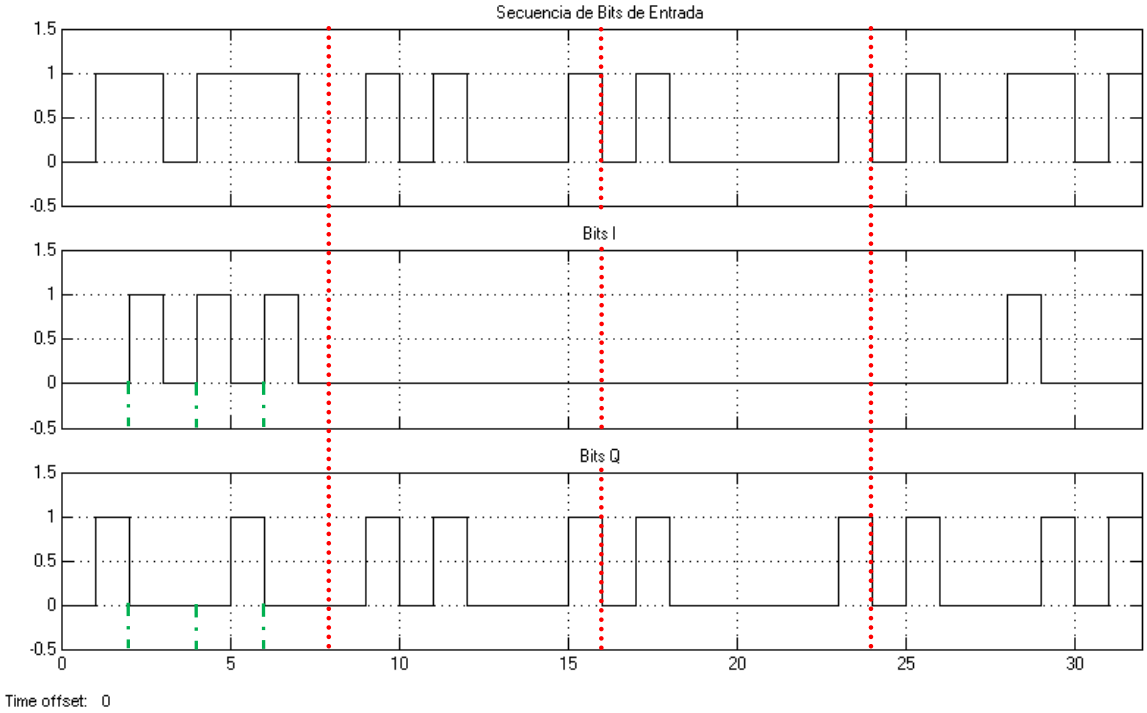


Figura 4.8. Simulación del bloque Divisor de Datos para 4-QAM

Para 16-QAM:

Secuencia:	01101110	01010001	01000001	01001101
Bits I:	01__11__	01__00__	01__00__	01__11__
Bits Q:	__10__10	__01__01	__00__01	__00__01

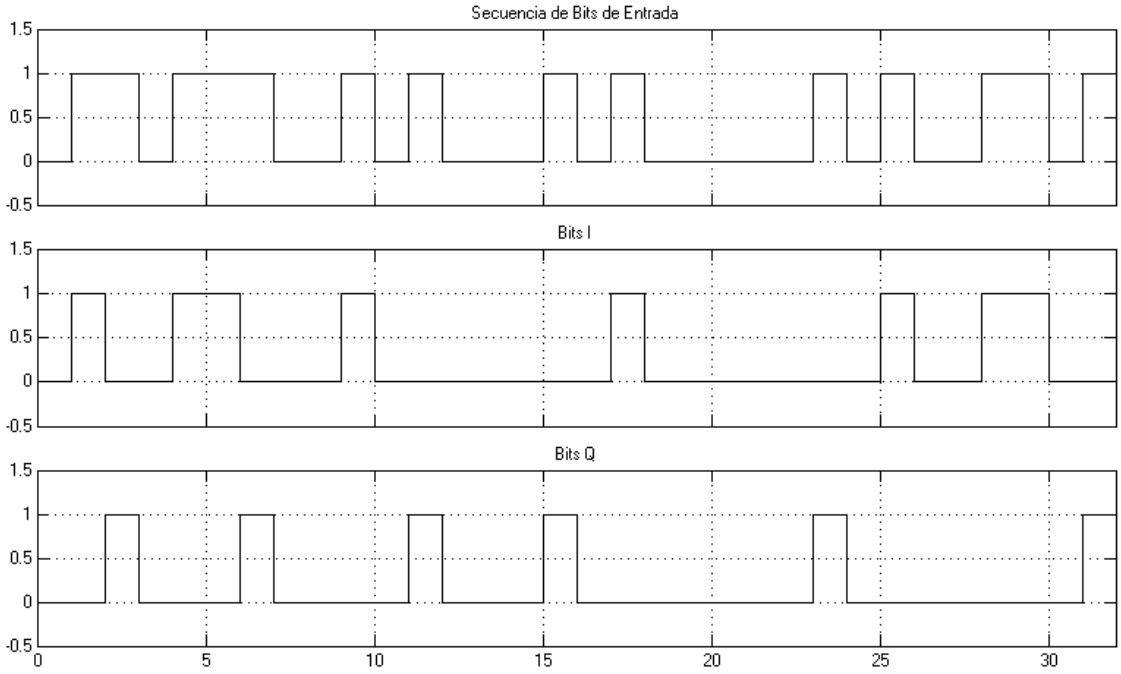


Figura 4.9. Simulación del bloque Divisor de Datos para 16-QAM

Para 64-QAM:

Secuencia:	01101110	01010001	01000001	01001101
Bits I:	011___10	0___000_	__000___	010___01
Bits Q:	___011__	_101___1	01___001	___011__

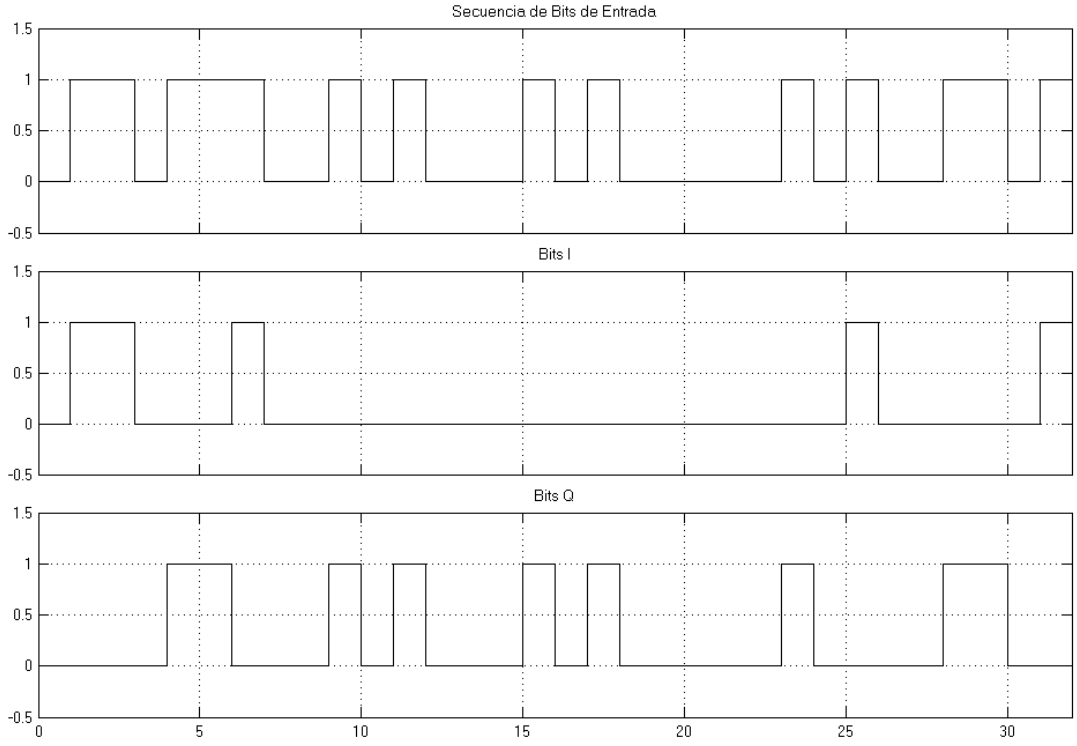


Figura 4.10. Simulación del bloque Divisor de Datos para 64-QAM

En las figuras 4.8, 4.9 y 4.10 la duración de cada bit es un segundo, por lo tanto un carácter tiene la duración de 8s y para el mensaje completo el tiempo de simulación es 32s.

Los resultados de la simulación en el divisor de datos son los previstos para cada modulación QAM, comprobando así el correcto funcionamiento y desempeño de este subsistema.

4.1.2.2. Convertidor de 2 a L Niveles

Una vez que se tiene el flujo serial de bits en una salida paralela de dos canales (canal I o en fase y el canal Q o en cuadratura), el bloque convertidor de 2 a L niveles, genera una señal multinivel según el número de bits y la combinación que se tenga en cada canal.

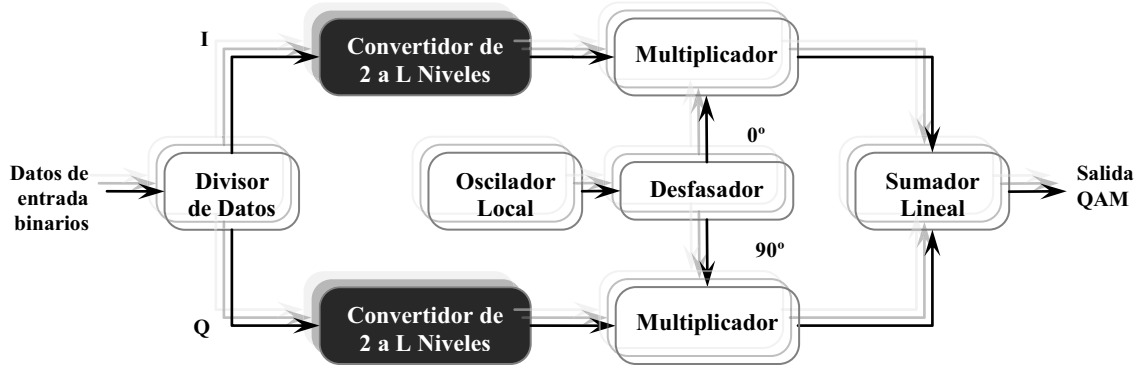


Figura 4.11. Convertidor de Niveles en el Modulador

n	Secuencia de bits	Canal I	Canal Q
4	… $Q_3I_3Q_2I_2Q_1I_1$	…_I_3_I_2_I_1 …_I_3_I_2_I_1 …$I_3I_3I_2I_2I_1I_1$	…_Q_3_Q_2_Q_1_ …_Q_3_Q_2_Q_1_ …$Q_3Q_3Q_2Q_2Q_1Q_1$_
16	… $Q_4Q_3I_4I_3Q_2Q_1I_2I_1$	…_ _I_4I_3_ _I_2I_1 …_ _$I_{3\text{-}4}$_ _$I_{1\text{-}2}$ … $I_{3\text{-}4}I_{3\text{-}4}$ $I_{1\text{-}2}I_{1\text{-}2}$	…_ _Q_4Q_3_ _Q_2Q_1_ _ …_ _$Q_{3\text{-}4}$_ _$Q_{1\text{-}2}$_ _ … $Q_{3\text{-}4}Q_{3\text{-}4}$ $Q_{1\text{-}2}Q_{1\text{-}2}$_ _
64	… $Q_3Q_2Q_1I_3I_2I_1$	…_ _ _$I_6I_5I_4$_ _ _$I_3I_2I_1$ …_ _ _$I_{4\text{-}5\text{-}6}$_ _ _$I_{1\text{-}2\text{-}3}$ … $I_{4\text{-}5\text{-}6}I_{4\text{-}5\text{-}6}$ $I_{1\text{-}2\text{-}3}I_{1\text{-}2\text{-}3}$	…_ _ _$Q_6Q_5Q_4$_ _ _$Q_3Q_2Q_1$_ _ _ …_ _ _$Q_{4\text{-}5\text{-}6}$_ _ _$Q_{1\text{-}2\text{-}3}$_ _ _ … $Q_{4\text{-}5\text{-}6}Q_{4\text{-}5\text{-}6}$ $Q_{1\text{-}2\text{-}3}Q_{1\text{-}2\text{-}3}$_ _ _

Tabla 4.4. Agrupación de bits en canales I-Q para la obtención de una señal unipolar multinivel

La idea que se tiene para diseñar este bloque es representar el grupo de bits en un número decimal sin signo, para tener así una señal multinivel unipolar y luego restarle un valor, para así obtener la señal multinivel bipolar requerida.

La tabla 4.4 permite visualizar como se va a obtener la señal multinivel unipolar de los canales I-Q, además de la eliminación de los bits introducidos.

Anteriormente se mencionó que los subguiones representan el tiempo en que los canales I-Q no reciben datos válidos. Ahora se tiene la certeza que esos datos no válidos son ceros introducidos por los multiplexores que conforman el bloque derivador cuando separa los bits en grupos a cada canal.

Entrada I_1/Q_1	Salida I_1/ Q_1
0	0
1	1

(a)

Entradas		Salida
I_1/Q_1	I_2/Q_2	I_{1-2}/ Q_{1-2}
0	0	0
0	1	1
1	0	2
1	1	3

(b)

Entradas			Salida
I_1/Q_1	I_2/Q_2	I_3/Q_3	I_{1-2-3}/ Q_{1-2-3}
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

(c)

Tabla 4.5. Tablas de verdad para los valores de la Señal Multinivel Unipolar de L niveles (L=2, 4 y 8). (a) 4-QAM; (b) 16-QAM; (c) 64–QAM.

Como se observó en la tabla 4.4, se toma un grupo de 1, 2 o 3 bits según el valor de N/2, obteniéndose un número decimal, el cual se encuentra representado como: I_1 o Q_1, I_{1-2} o Q_{1-2} e I_{1-2-3} o Q_{1-2-3} según el canal que corresponda y en función del número de bits que se agrupen.

Los multiniveles de la señal unipolar se han escogido convenientemente, debido a que la representación del conjunto de bits en binario es su equivalente en decimal. La asignación para cada nivel de la señal se muestra en la tabla 4.5; además se debe notar que el bit I_1/Q_1 es el más significativo MSB y el que ingresa primero al subsistema.

En el Blockset de Xilinx existe un bloque con la funcionalidad de representar muestras simples en un valor único. En el capítulo 3, se encuentra la descripción del bloque *Serial To Parallel*, el cual es necesario para realizar la conversión según se muestra en la tabla 4.4, por ejemplo pasar de I_2I_1 a $I_{1\text{-}2}$ en 16-QAM. Este bloque requiere como valor principal el número de bits agrupados por muestra, el cual es:

$$\frac{N}{2} = \frac{1}{2}\log_2 n$$

Pero aún se necesitan eliminar los ceros introducidos por el subsistema divisor de bits, como se describió en la tabla 4.4, es decir para pasar de _ $I_{1\text{-}2}$ a $I_{1\text{-}2}I_{1\text{-}2}$, lo cual se hace duplicando la muestra actual para que remplacen los ceros. Para conseguir esto, se hace que la muestra permanezca en su nivel, el doble de su tiempo original y es lo que precisamente hace este bloque disminuyendo la frecuencia de muestreo. Este bloque se denomina *Down Sample*, el cual permite que las muestras de la señal unipolar I_1, $I_{1\text{-}2}$ e $I_{1\text{-}2\text{-}3}$, o Q_1, $Q_{1\text{-}2}$ y $Q_{1\text{-}2\text{-}3}$ permanezcan en dichos valores el doble de su duración inicial.

Para obtener la señal bipolar simétrica necesaria, a partir de la señal unipolar conseguida de acuerdo a lo descrito anteriormente, se debe restar de ella un valor constante que se calcula de la siguiente manera:

$$\rightarrow k = \frac{Vmáx - Vmín}{2} = \frac{Vmáx - 0}{2} \Rightarrow k = \frac{Vmáx}{2}$$

Donde:

k : Valor constante

Vmáx : Nivel de voltaje máximo asignado para la Señal Multinivel Unipolar

$$\text{Para:} \qquad 4-\text{QAM} \rightarrow k = \frac{1}{2} = 0.5$$

$$16-\text{QAM} \rightarrow k = \frac{3}{2} = 1.5$$

$$64-\text{QAM} \rightarrow k = \frac{7}{2} = 3.5$$

Para la generalización de esta parte del subsistema se necesita expresar 'k' en función del parámetro 'n'. En primer término, se nota que el voltaje máximo asignado Vmáx se relaciona con el número de combinaciones, que se tiene cuando se agrupan 1, 2 o 3 bits respectivamente.

$$\text{Debido a que las combinaciones son: } 2^{\frac{N}{2}} = \sqrt{n}$$

$$\text{Se tiene que:} \qquad k = \frac{2^{\frac{N}{2}} - 1}{2} = \frac{\sqrt{n} - 1}{2}$$

Para realizar la resta de la señal unipolar multinivel con la constante 'k', se necesita un bloque *AddSub* (Sumador-Restador), el cual está disponible en el Blockset de Xilinx y también se describe en el capítulo 3.

Finalmente en la salida de este subsistema, se obtienen los valores detallados en la tabla 4.6 para cada modulación QAM.

I/Q	Salida
0	-0.5
1	0.5

(a)

I_1/Q_1	I_2/Q_2	Salida
0	0	-1.5
0	1	-0.5
1	0	0.5
1	1	1.5

(b)

I_1/Q_1	I_2/Q_2	I_3/Q_3	Salida
0	0	0	-3.5
0	0	1	-2.5
0	1	0	-1.5
0	1	1	-0.5
1	0	0	0.5
1	0	1	1.5
1	1	0	2.5
1	1	1	3.5

(c)

Tabla 4.6. Tablas de verdad para los valores de la Señal Multinivel Bipolar de L niveles (L=2, 4 y 8). (a) 4-QAM; (b) 16-QAM; (c) 64–QAM.

Estos valores, no corresponden a los planteados en el capítulo 1 en el diagrama de constelaciones, pero se nota claramente que duplicando sus valores, se obtendría lo deseado. Con el fin de no utilizar más recursos (una constante y un multiplicador), se decidió complementarlo con los subsistemas oscilador local, generando una portadora con una amplitud igual a dos, y el multiplicador.

La conexión de los bloques que conforman este subsistema para cada canal, se realiza en el ambiente de Simulink y se muestran en las figuras 4.12 y 4.13.

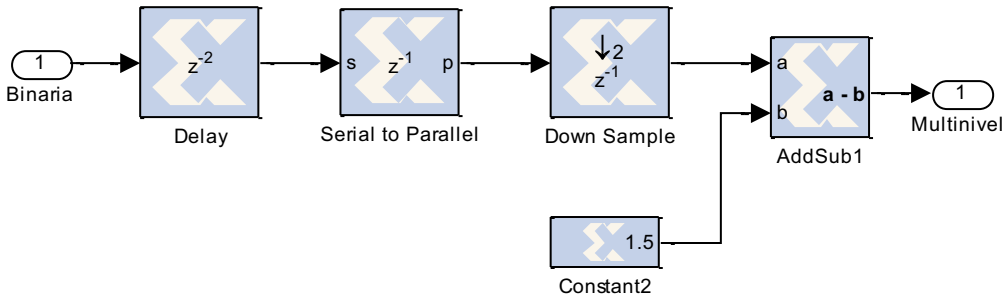


Figura 4.12. Modelación en Simulink con SysGen del bloque Convertidor de 2 a L Niveles para el Canal I

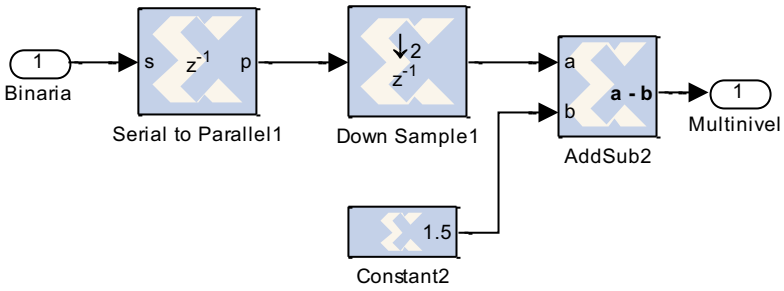


Figura 4.13. Modelación en Simulink con SysGen del bloque Convertidor de 2 a L Niveles para el Canal Q

Nótese que en la entrada al convertidor del canal I existe un retardo representado por el bloque *delay*, cuya función es ubicar a I_1 en el mismo instante que Q_1, para procesarlos paralelamente y tenerlos listos para la obtención del diagrama de constelaciones. El retardo es exactamente igual al número de bits agrupados que es $N/2 = 1/2\log_2 n$.

La configuración de parámetros de cada bloque que se utiliza en este subsistema, se muestra resumida en la tabla 4.7, que es la misma para los dos canales. Es importante destacar que los valores indicados en el parámetro *Latency* de los

bloques *Serial to Parallel* y *Down Sample* son los mínimos requeridos para su funcionamiento.

BLOQUE	PESTAÑA	PARÁMETROS	VALOR
Delay	*Basic*	*Latency*	*1/2*log2(n)*
Serial to Parallel	*Basic*	*Input order* *Output type* *Number of bits* *Binary point* *Latency*	*Most significant word first* *Unsigned* *1/2*log2(n)* *0* *1*
Down Sample	*Basic*	*Sampling rate* *Sample* *Latency*	*2* *First value of frame* *1*
Constant	*Basic*	*Type* *Constant value* *Number of bits* *Binary point*	*Unsigned* *(sqrt(n) - 1) / 2* *1/2*log2(n)* *1*
AddSub	*Basic*	*Operation*	*Subtraction*

Tabla 4.7. Configuración de los parámetros del Convertidor de 2 a L Niveles

Para 4-QAM:

Mensaje	n	Q	A	M
Secuencia	01101110	01010001	01000001	01001101
Bits I	00101010	00000000	00000000	00001000
Delay	_ 00101010	00000000	00000000	00001000
Serial to Parallel	_ _00101010	00000000	00000000	00001000
Down Sample	_ _ _ _ 00 11 11 11	00000000	00000000	00 00 11 00
Señal Multinivel Unipolar	0 1	0	0	0 1 0
Señal Multinivel Bipolar	-0.5 +0.5	-0.5	-0.5	-0.5 +0.5 -0.5

Tabla 4.8. Valores previstos a la salida de cada bloque en el Canal I

La simulación muestra en primer lugar, las señales de cada uno de los bloques constitutivos del canal I para 4-QAM en la figura 4.14, para paso a paso aclarar su función dentro del subsistema. Comprendido esto, la señal multinivel bipolar se grafica para cada modulación QAM en los canales I-Q.

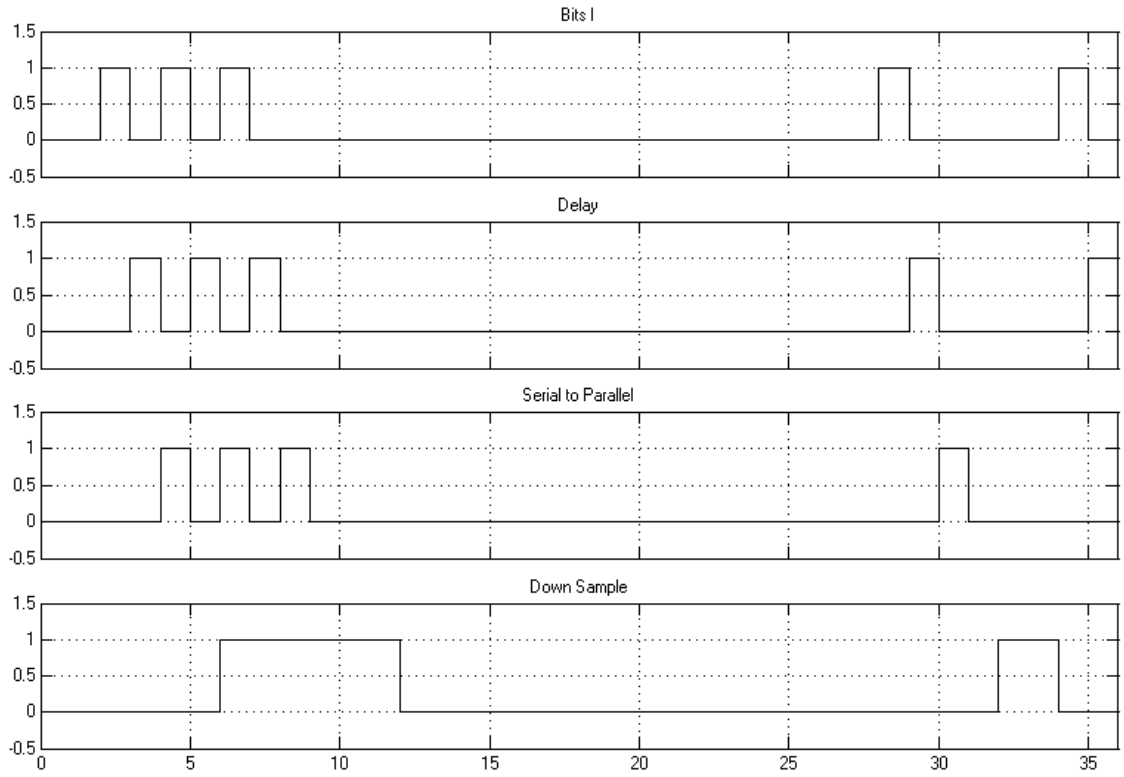


Figura 4.14. Simulación del bloque Convertidor de 2 a L Niveles para 4-QAM en detalle para el Canal I

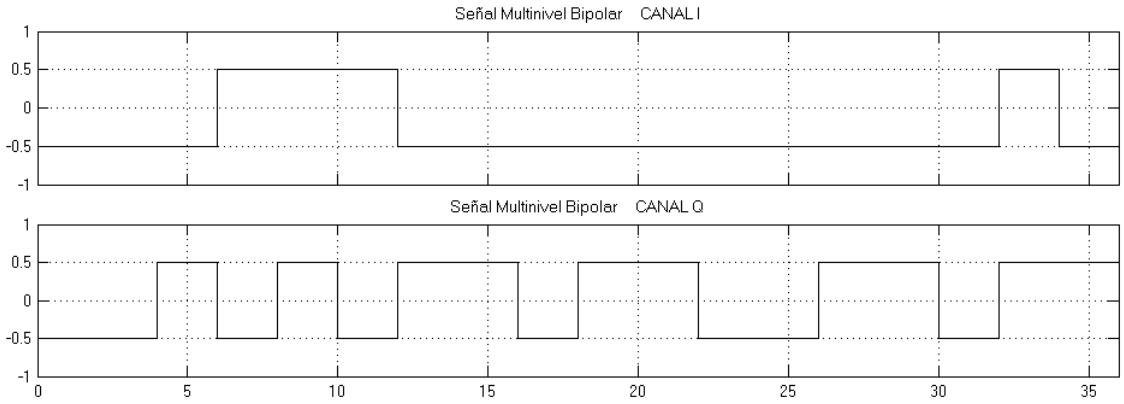


Figura 4.15. Simulación del bloque Convertidor de 2 a L Niveles para 4-QAM canales I-Q

Para 16-QAM:

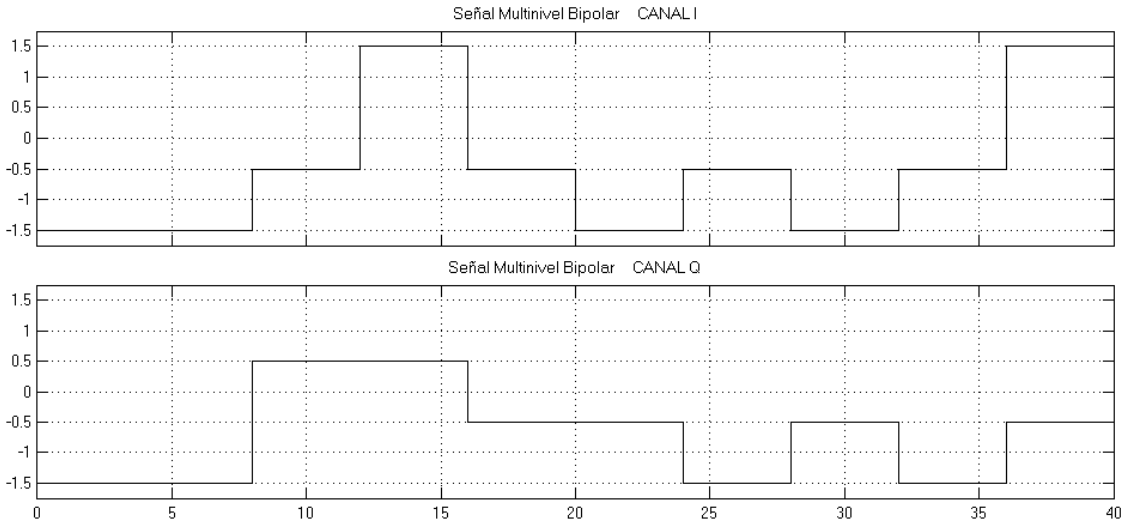


Figura 4.16. Simulación del bloque Convertidor de 2 a L Niveles para 16-QAM canales I-Q

Para 64-QAM:

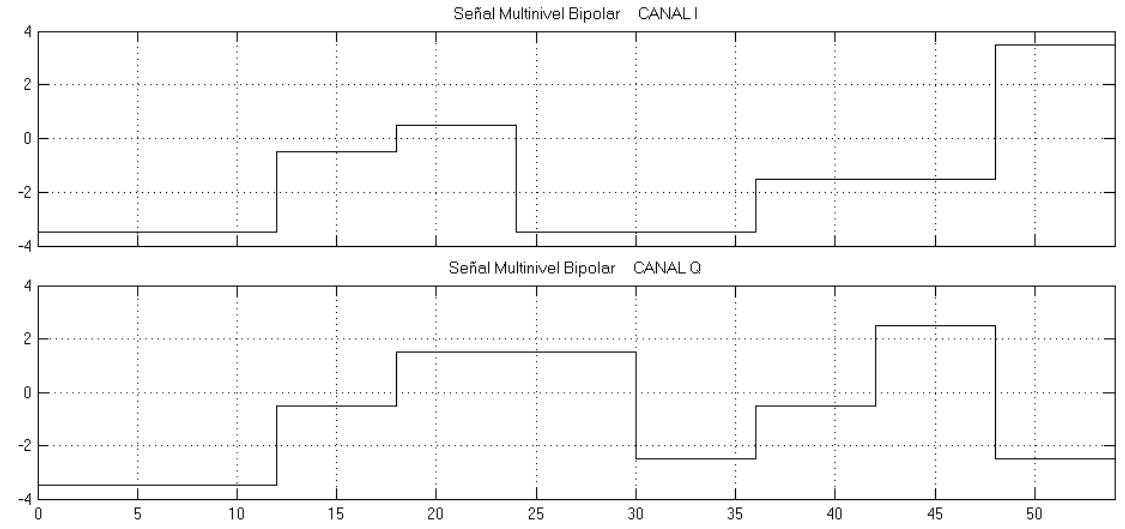


Figura 4.17. Simulación del bloque Convertidor de 2 a L Niveles para 64-QAM canales I-Q

4.1.2.3. Oscilador Local y Desfasador

En este subsistema se pretende generar la portadora, para trasladar en frecuencia la información contenida en los canales I-Q, debido a que frecuencias superiores permiten mejor radiación de la energía y mayor capacidad de transmisión, implicando un más amplio ancho de banda[7].

[8] HIDALGO, Pablo, Sistemas Digitales, Modulación Digital, p. 1.

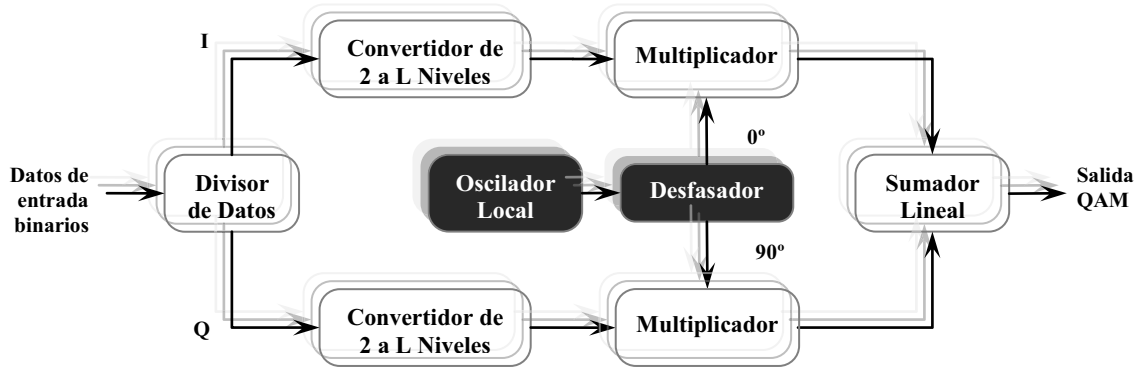


Figura 4.18. Oscilador Local y Desfasador en el Modulador

El oscilador de referencia local, se construye con la utilización de una memoria ROM que almacena en sus localidades valores discretos o muestras de una onda coseno, la cual será utilizada directamente para el canal I.

El tamaño de la memoria ROM requerida, depende del número de muestras por período 'x' que tiene la señal portadora; pero para determinar esto, se debe primero escoger la frecuencia de la portadora f_C, debido a que se relacionan así:

$$x = f_M \cdot \frac{1}{f_C}$$

Donde:

x : Número de muestras por período

f_M : Frecuencia de muestreo

f_C : Frecuencia de la portadora

Para escoger la frecuencia de la portadora, se recurre al teorema del muestreo:

$$f_M = 2f_{máx}$$

Pero para una mejor visualización de la señal en el tiempo se considera:

$$f_M >> 2f_{máx} \quad ; \quad f_{máx} = f_C$$
$$f_M > 10f_C \quad ; \quad f_M = f_{DAC} = 500\,KHz$$
$$f_C < \frac{500\,KHz}{10}$$
$$f_C < 50\,KHz$$

La frecuencia f_{DAC} se incluye en el cálculo, debido a que se utilizará un conversor digital analógico para visualizar las señales en el osciloscopio.

Con las condiciones determinadas anteriormente, se encontrará el valor de 'x' considerando que f_c = 12,5KHz:

$$x = 500\ KHz \cdot \frac{1}{12,5\ KHz}$$
$$\rightarrow x = 40\ muestras/p\ eríodo$$

Las líneas de direccionamiento 'D' necesarias en la ROM, se calculan así:

$$x \leq 2^D$$
$$40 \leq 2^D$$
$$D \geq 5,322 \rightarrow D = 6$$

Entonces, la ROM debe tener 40 localidades de memoria y 6 líneas de direccionamiento.

Para el canal Q, se necesita una referencia ortogonal a la utilizada por el canal I, por lo que se debe obtener la señal seno de la referencia local. El bloque Desfasador que realiza esta tarea, produce una diferencia de fase de π/2 con respecto a la onda coseno. En el diseño, se planea hacer este bloque con un retardo de un cuarto del período, que es justamente π/2; así se tiene lo siguiente:

$$t_R = \frac{T_C}{4} = \frac{1}{4f_C}$$

Donde:

t_R : Retardo aplicado a la señal portadora

T_C : Período de la señal portadora

f_C : Frecuencia de la señal portadora

Entonces,

$$t_R = \frac{1}{4f_C} = \frac{1}{4(12,5\ Khz)}$$
$$\rightarrow t_R = 20\,\mu s$$

Este valor t_R representado en número de muestras 'y' es:

$$y = \frac{x}{4} = \frac{40}{4} \rightarrow y = 10\,\text{muestras}$$

La conexión de los bloques que conforman estos subsistemas, se realiza en el ambiente de Simulink y se la muestra en las figuras 4.19 y 4.20.

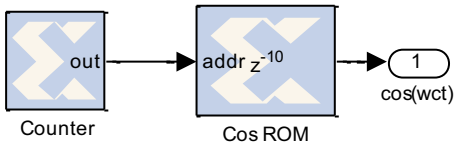


Figura 4.19. Modelación en Simulink con SysGen del bloque Oscilador Local

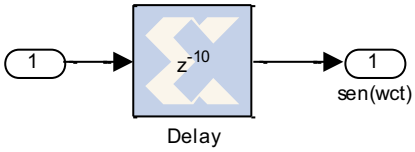


Figura 4.20. Modelación en Simulink con SysGen del bloque Desfasador

Para el direccionamiento, se utilizó un contador ascendente de módulo 40, el cual permite un acceso secuencial a la memoria.

La configuración de parámetros de cada bloque que se utiliza en los subsistemas, se muestra resumida en la tabla 4.9.

El valor de 1/80 en el parámetro *Explicit Sample Period* del bloque *Counter,* obedece a una condición impuesta de tener 2 oscilaciones de la portadora por bit de entrada, por lo que resulta 80 muestras cada 2 períodos.

Previo a la simulación de estos bloques, la configuración del *Token* de SysGen debe ser llevada a cabo. El parámetro más importante en la simulación de estos es '*Simulink System Period'*, cuyo valor es igual a 1/80, debido al período de muestreo que posee el contador de la memoria ROM para el direccionamiento. Con esta modificación, el resultado de la simulación para estos dos subsistemas se muestra en la figura 4.21.

SUBSISTEMA	BLOQUE	PESTAÑA	PARÁMETROS	VALOR
Oscilador Local	*Counter*	*Basic*	*Counter type*	*Count Limited*
			Count to value	*40-1*
			Count direction	*Up*
			Initial Value	*0*
			Step	*1*
			Output type	*Unsigned*
			Number of bits	*6*
		Advanced	*Explicit Sample Period*	*1/80*
	Cos ROM	*Basic*	*Depth*	*40*
			Initial value vector	*-2*sin(2pi [0:40-1]/40)*
			Latency	*10*
		Output Type	*Word type*	*Signed (2*'s comp*)*
			Number of bits	*10*
			Binary point	*7*
Desfasador	*Delay*	*Basic*	*Latency*	*10*

Tabla 4.9. Configuración de los parámetros del Oscilador Local y del Desfasador

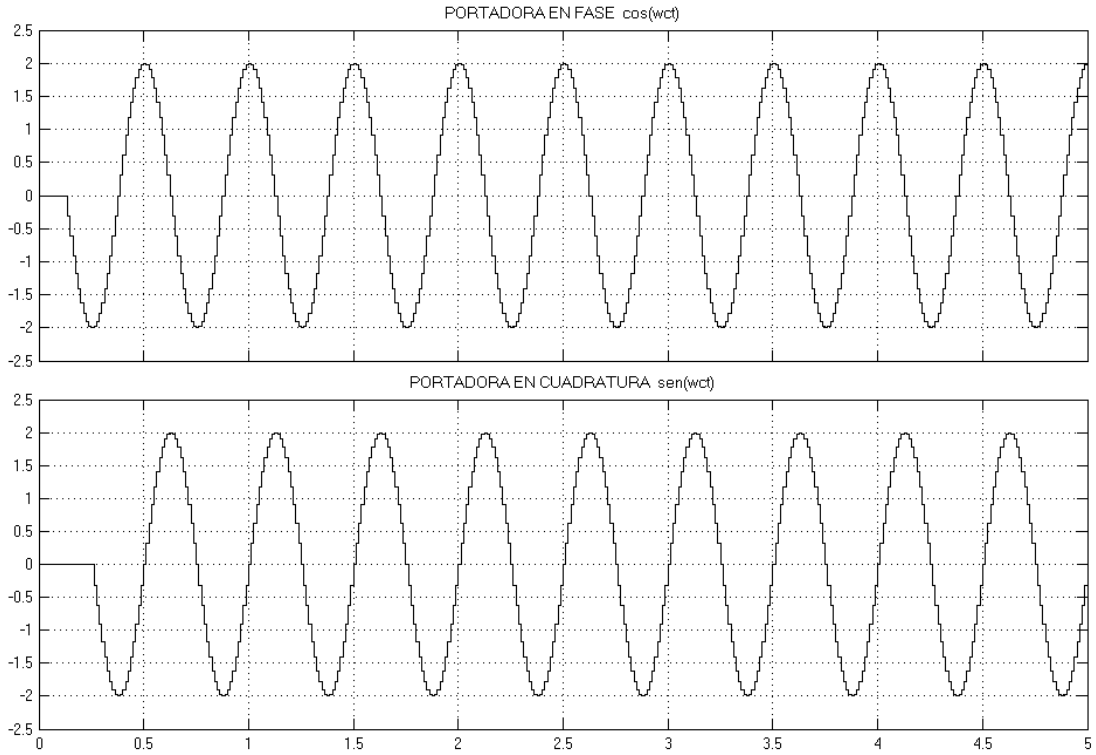


Figura 4.21. Simulación de los bloques Oscilador Local y Desfasador

Claramente se observa que la portadora en fase corresponde a la señal coseno, mientras la señal seno es la portadora en cuadratura. Ambas tienen una amplitud de 2, esto se hizo con el objetivo de complementar a la señal multinivel bipolar que se obtuvo en el convertidor y llegar a los valores inicialmente propuestos.

4.1.2.4. Multiplicador

Es también llamado Mixer, ya que mezcla o multiplica las señales de cada canal, con la onda en fase o en cuadratura según corresponda.

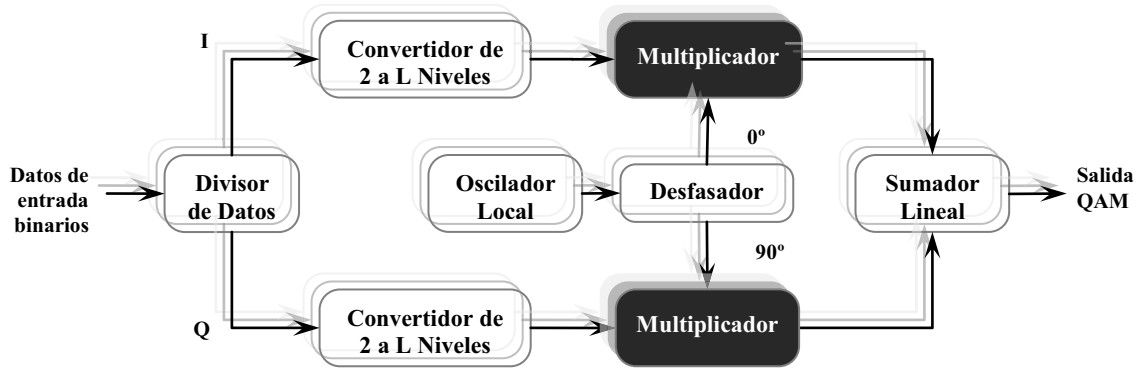


Figura 4.22. Mixer en el Modulador

En el Blockset de Xilinx existe un bloque multiplicador de dos entradas sin restricción al tipo de datos. En el capítulo 3, se encuentra la descripción del bloque *Mult*, el cual es necesario para obtener el producto entre la señal multinivel bipolar del canal I con la onda coseno y la señal del canal Q con la onda seno.

La configuración de los parámetros del bloque multiplicador, que es la misma para ambos canales, se muestra resumida en la tabla 4.10.

BLOQUE	PESTAÑA	PARÁMETROS	VALOR
Mult	*Basic*	*Precision*	*Full*
		Latency	*2*

Tabla 4.10. Configuración de los parámetros del Multiplicador

El valor del parámetro *Latency* es el mínimo requerido para su funcionamiento y su posterior implementación.

La conexión del bloque multiplicador que se realiza en el ambiente de Simulink, se muestra en la figura 4.23.

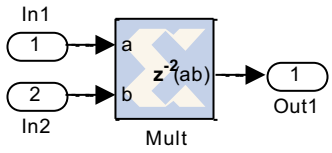


Figura 4.23. Modelación en Simulink con SysGen del bloque Multiplicador

La simulación de las figuras 4.24, 4.25 y 4.26, muestra el resultado del producto entre la señal multinivel bipolar de cada canal I-Q con portadora en fase o en cuadratura, según corresponda, para cada modulación QAM.

Para 4-QAM:

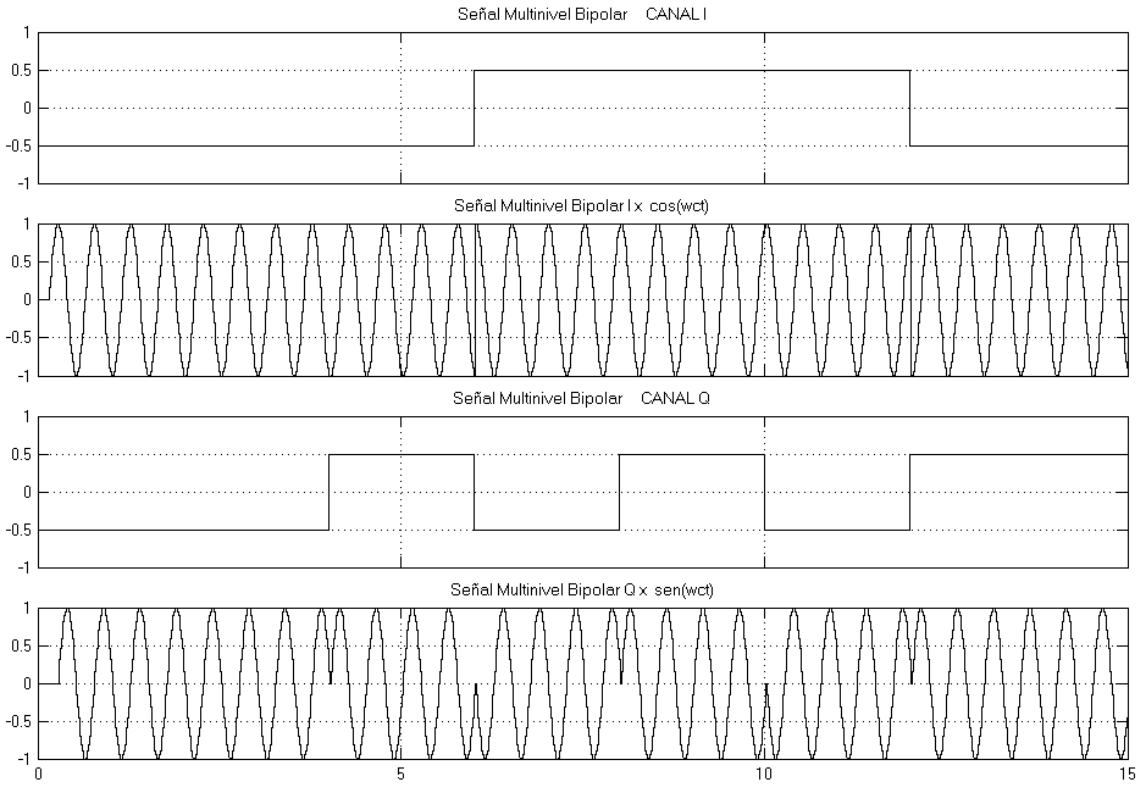


Figura 4.24. Simulación del Multiplicador, para 4-QAM canales I-Q

Para 16-QAM:

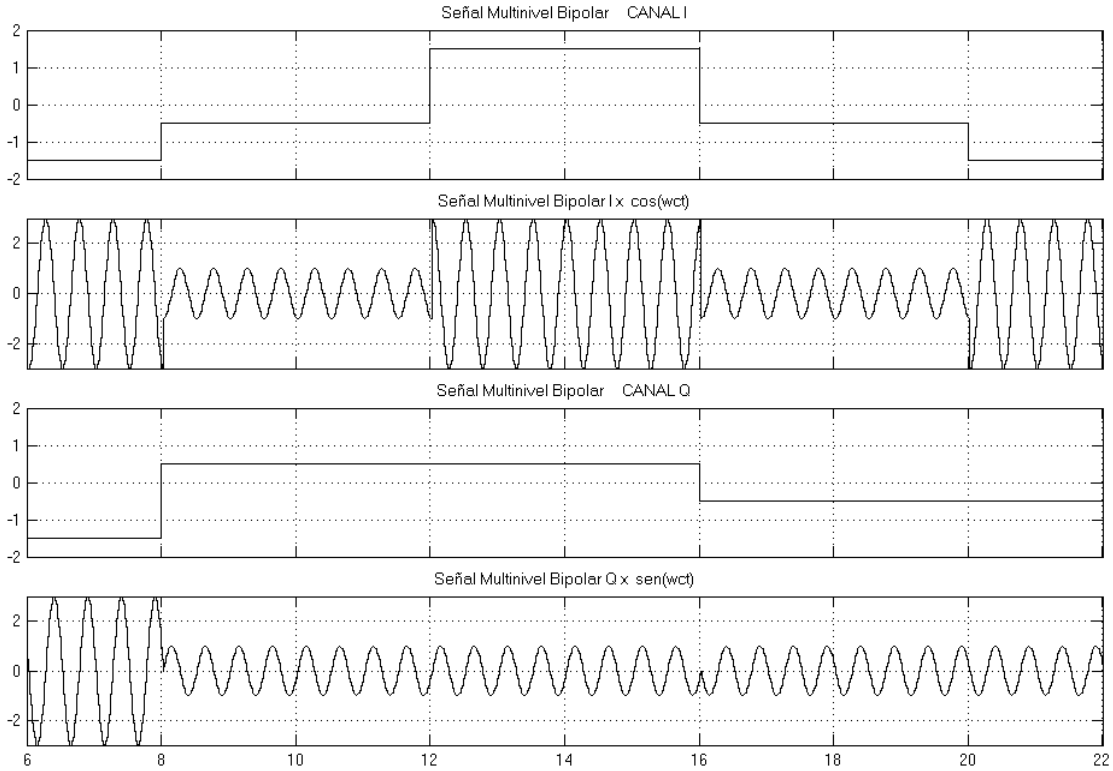


Figura 4.25. Simulación del Multiplicador, para 16-QAM canales I-Q

Para 64-QAM:

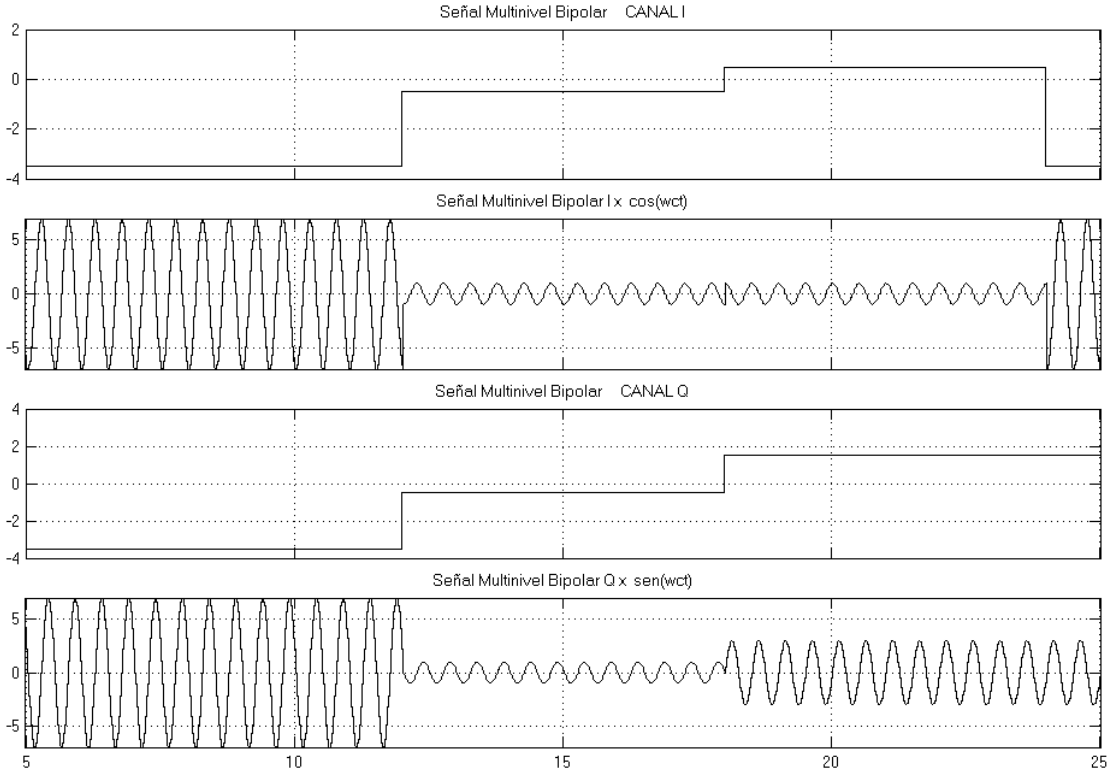


Figura 4.26. Simulación del Multiplicador, para 64-QAM canales I-Q

4.1.2.5. Sumador Lineal

El último bloque del modulador, suma las señales del canal I con las señales del canal Q, para obtener una onda modulada que contenga la información tanto en la amplitud como en la fase, resultando así la modulación QAM.

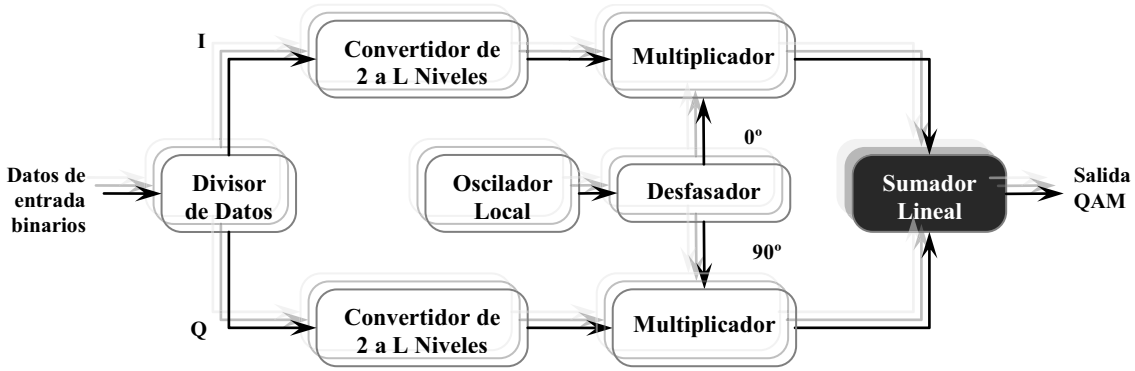


Figura 4.27. Sumador Lineal en el Modulador

En el Blockset de Xilinx existe un bloque Sumador-Restador de dos entradas. En el capítulo 3, se encuentra la descripción del bloque *AddSub*, el cual es necesario para obtener la suma de los canales I-Q y para cumplir con el objetivo planteado en cuanto al modulador QAM.

La configuración de parámetros del bloque sumador se muestra resumida en la tabla 4.11.

BLOQUE	PESTAÑA	PARÁMETROS	VALOR
AddSub	*Basic*	*Operation*	*Addition*
	Output Type	*Precision*	*Full*

Tabla 4.11. Configuración de los parámetros del Sumador Lineal

La conexión del bloque sumador que se realiza en el ambiente de Simulink, se muestra en la figura 4.28.

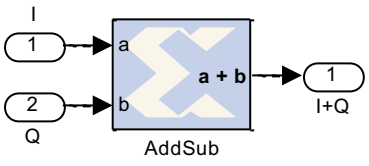


Figura 4.28. Modelación en Simulink con SysGen del bloque Sumador Lineal

La simulación de las figuras 4.29, 4.30, 4.31, 4.32, 4.33 y 4.34 muestra la señal modulada y su correspondiente diagrama de constelaciones para 4, 16 y 64-QAM.

Para 4-QAM:

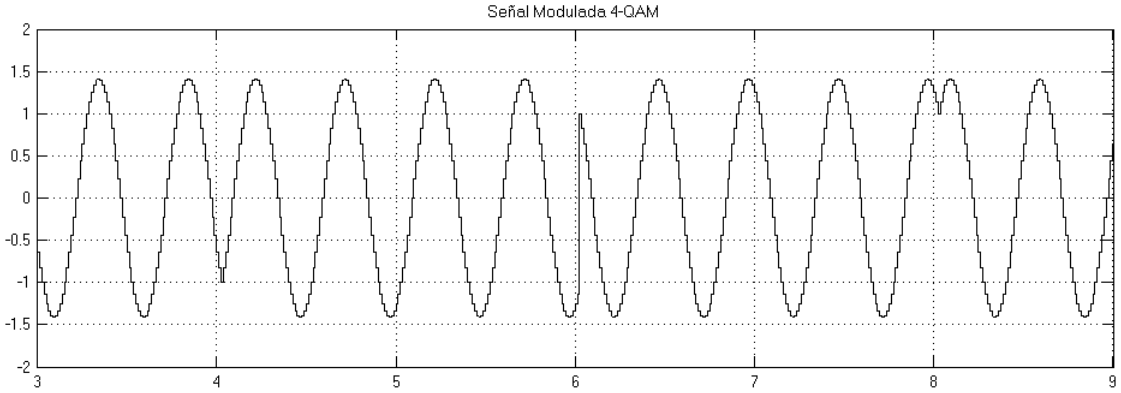


Figura 4.29. Simulación del Sumador Lineal para 4-QAM

Para 16-QAM:

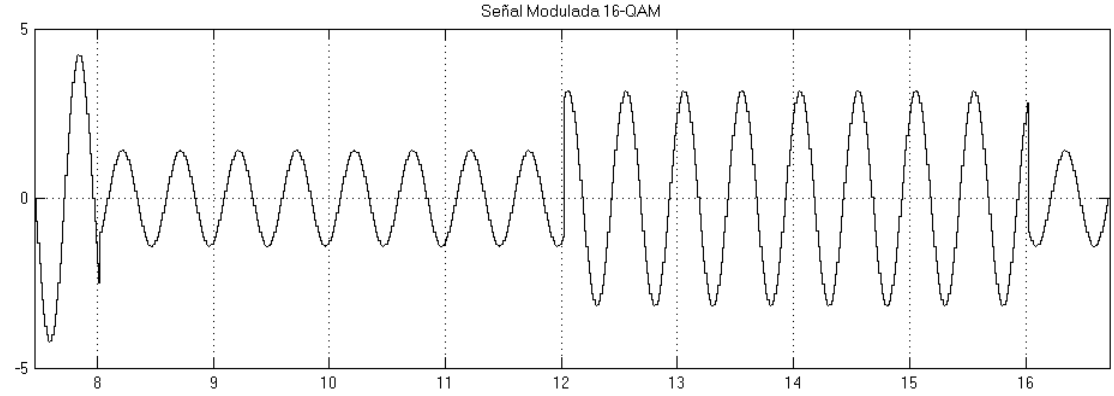


Figura 4.30. Simulación del Sumador Lineal para 16-QAM

Para 64-QAM:

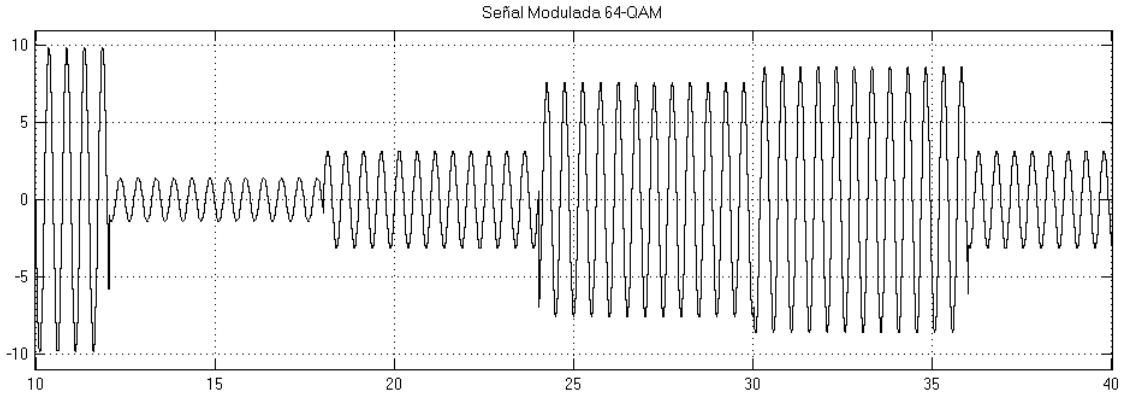


Figura 4.31. Simulación del Sumador Lineal para 64-QAM

Para 4-QAM:

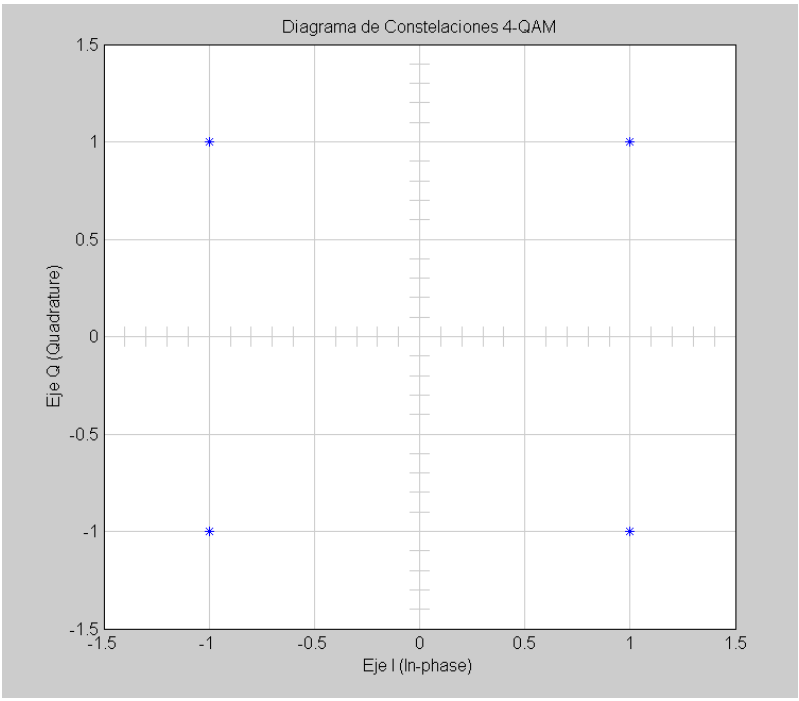


Figura 4.32. Diagrama de Constelaciones para 4-QAM

Para 16-QAM:

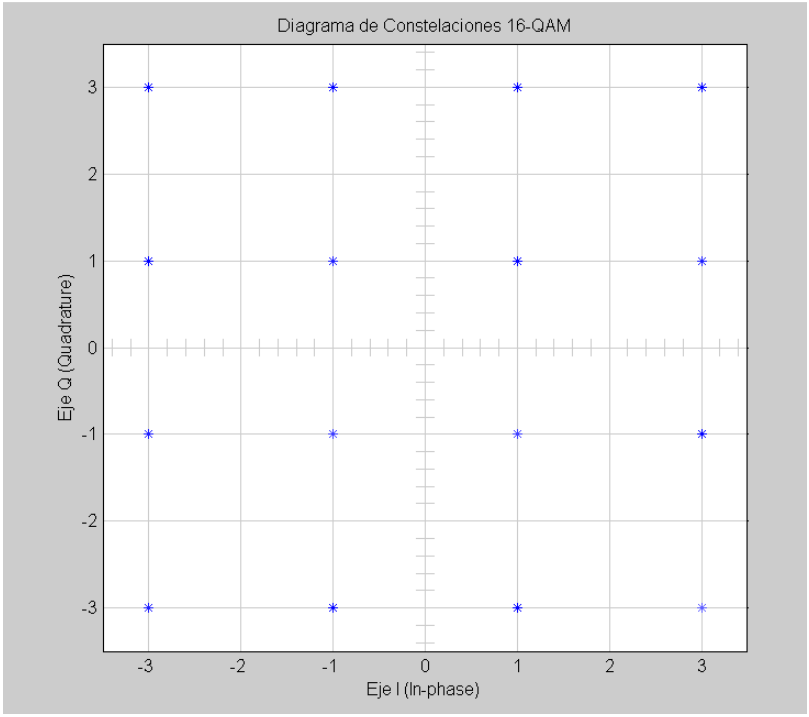


Figura 4.33. Diagrama de Constelaciones para 16-QAM

Para 64-QAM:

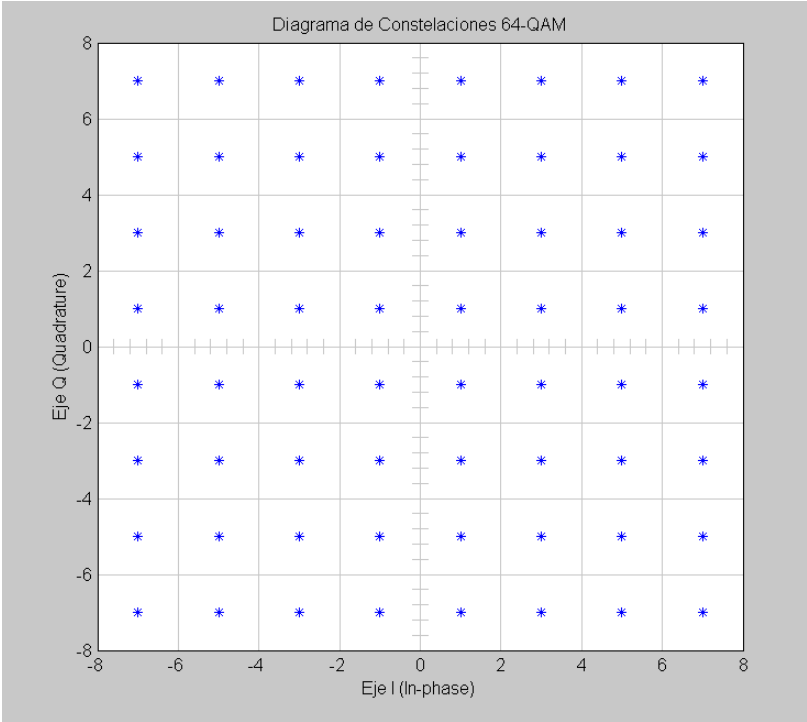


Figura 4.34. Diagrama de Constelaciones para 64-QAM

En las figuras que muestran la modulación, se ha tomado sólo una parte de la señal para que se puedan visualizar los cambios de amplitud y de fase.

4.1.3. CONSIDERACIONES DE ANCHO DE BANDA

4.1.3.1. Velocidad de Entrada

La tasa o la velocidad de los bits en la entrada V_{IN} es la misma para todos los moduladores n-QAM diseñados. El cálculo se realiza con la condición impuesta de tener dos oscilaciones de la portadora por cada bit de entrada, por lo que se tiene lo siguiente:

Como $f_C = 12{,}5\,KHz$

$$V_{IN} = \frac{f_C}{2} = \frac{12{,}5\,KHz}{2}$$

$$\rightarrow V_{IN} = 6{,}25\,Kbps$$

4.1.3.2. Velocidad de Modulación

En el modulador n-QAM existe un cambio en la salida, para cada N bits en la entrada. Por lo tanto, la velocidad con la que cambian los símbolos en la salida según el tipo de modulación QAM es:

$$V_{MOD} = \frac{V_{IN}}{N}$$

Con esta expresión se calcula la velocidad de modulación, obteniéndose los resultados para la modulación n-QAM resumidos en la tabla 4.12.

Modulación	N	V_{MOD} [Kbaudios]
4-QAM	2	3,125
16-QAM	4	1,563
64-QAM	6	1,042

Tabla 4.12. Velocidad de Modulación para n-QAM

4.1.3.3. Ancho de Banda

Para el cálculo del mínimo ancho de banda (AB) necesario para la propagación de la señal modulada, se recurre a la expresión de Nyquist para el mínimo AB:

$$AB = \frac{V_{MOD}}{2}$$

Pero, debido a que el espectro de salida de un modulador QAM es una señal con doble banda lateral DBL y portadora suprimida, el mínimo ancho de banda de Nyquist en este caso es el doble, como se muestra en la tabla 4.13.

Modulación	AB_{DBL} [KHz] $AB_{DBL} = 2AB = V_{MOD}$
4-QAM	3,125
16-QAM	1,563
64-QAM	1,042

Tabla 4.13. Ancho de Banda para n-QAM

4.1.3.4. Eficiencia Espectral

La eficiencia del ancho de banda, eficiencia espectral o densidad de información, a menudo se utiliza para comparar el rendimiento de una técnica de modulación digital con otra[8].

Se calcula con la siguiente expresión:

$$\delta = \frac{V_{IN}}{AB_{DBL}}$$

La eficiencia espectral para cada modulación QAM se resume en la tabla 4.14, teniendo en cuenta que la velocidad de entrada es de 6,25 Kbps para todos los moduladores.

Modulación	δ [bps/Hz]
4-QAM	2
16-QAM	4
64-QAM	6

Tabla 4.14. Densidad de Información para n-QAM.

4.1.4. DISEÑO Y SIMULACIÓN DE BLOQUES DEL DEMODULADOR

En el diagrama general de bloques del demodulador de la figura 4.2, se muestran todos los subsistemas que se van a modelar en este apartado. En resumen, su funcionamiento involucra operaciones inversas a las realizadas por cada uno de los subsistemas existentes en el modulador. Las tres secciones para la descripción de cada subsistema, se mantienen como en el caso del modulador.

Es importante recordar que una de las condiciones generales implantadas al inicio del capítulo, establece la ausencia de la recuperación de la portadora, en consecuencia el módulo del oscilador local y el desfasador para el demodulador son idénticos a los utilizados por el modulador, excepto por su amplitud que es igual a la unidad.

[8] TOMASI, Wayne, Sistemas de Comunicaciones Digitales, Pág 489.

4.1.4.1. Multiplicador

La señal QAM que ingresa se mezcla en el multiplicador con la portadora en fase del canal I y con la portadora en cuadratura en el multiplicador Q, respectivamente, resultando la señal en banda base más una componente de alta frecuencia.

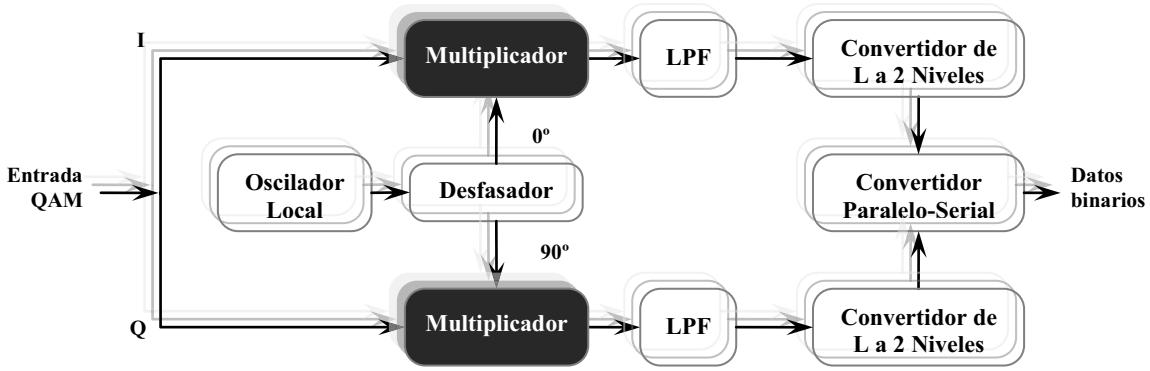


Figura 4.35. Divisor de Datos en el Demodulador

Para realizar el producto, se utiliza un bloque *Mult* (Multiplicador) del Blockset de Xilinx. La configuración de parámetros del bloque multiplicador, tanto para I como para Q, se muestra resumida en la tabla 4.15.

BLOQUE	PESTAÑA	PARÁMETROS	VALOR
Mult	*Basic*	*Precision*	*User defined*
		Output type	*Signed (2*'s comp*)*
		Number of bits	*12*
		Binary point	*7*
		Quantization	*Truncate*
		Overflow	*Wrap*
		Latency	*3*

Tabla 4.15. Configuración de los parámetros del Multiplicador

El valor del parámetro *Latency* es el mínimo requerido para el funcionamiento del bloque y su posterior implementación.

La conexión del bloque multiplicador, que se realiza en el ambiente de Simulink, se muestra en la figura 4.23.

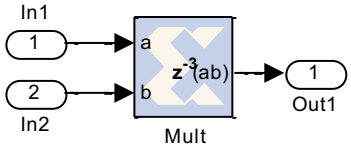


Figura 4.36. Modelación en Simulink con SysGen del bloque Multiplicador

La simulación muestra el resultado del producto entre la señal QAM obtenida del modulador con su correspondiente portadora, para cada modulación QAM, en las figuras 4.37, 4.38 y 4.39.

Para 4-QAM:

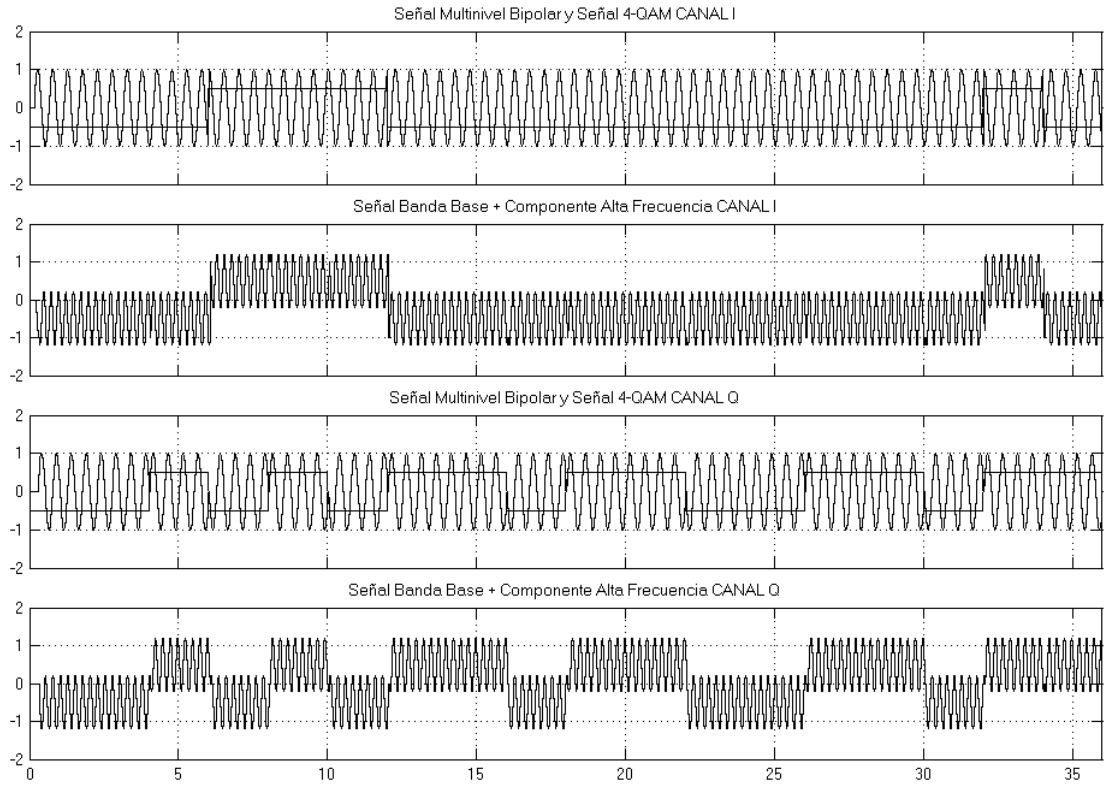


Figura 4.37. Simulación del Multiplicador, para 4-QAM canales I-Q

La pantalla del osciloscopio se encuentra dividida en cuatro partes, para poder realizar una evaluación rápida del funcionamiento del multiplicador.

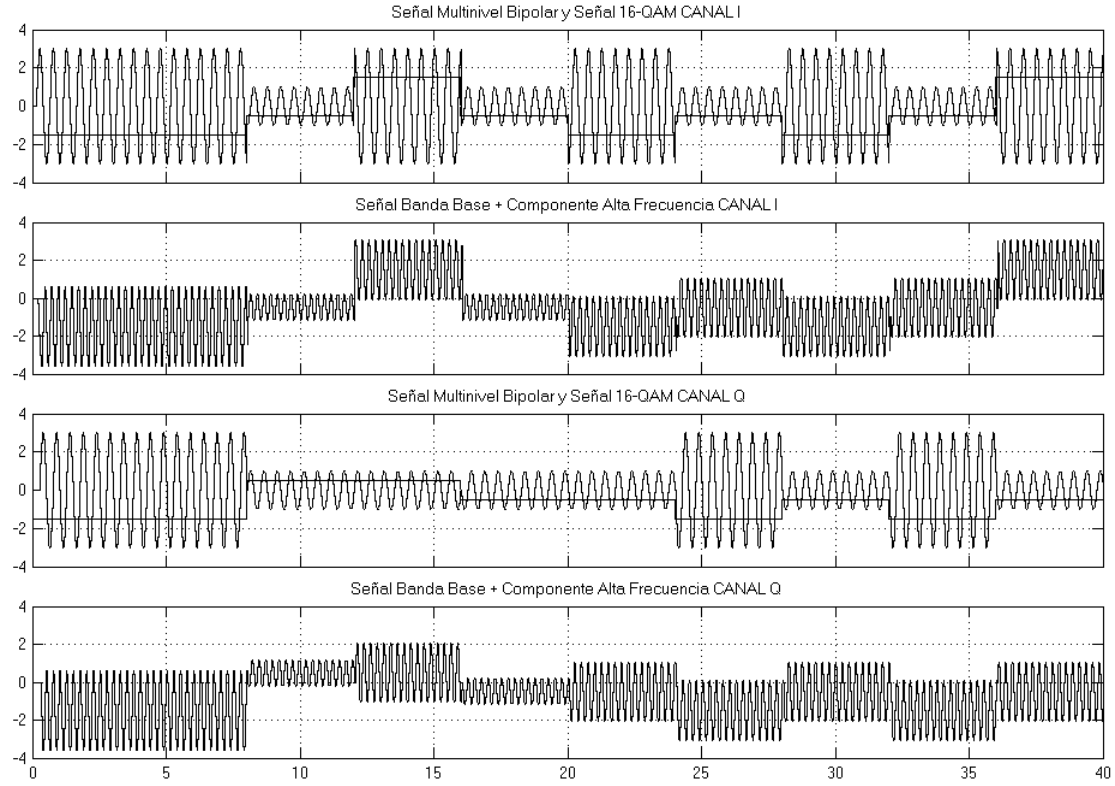


Figura 4.38. Simulación del Multiplicador, para 16-QAM canales I-Q

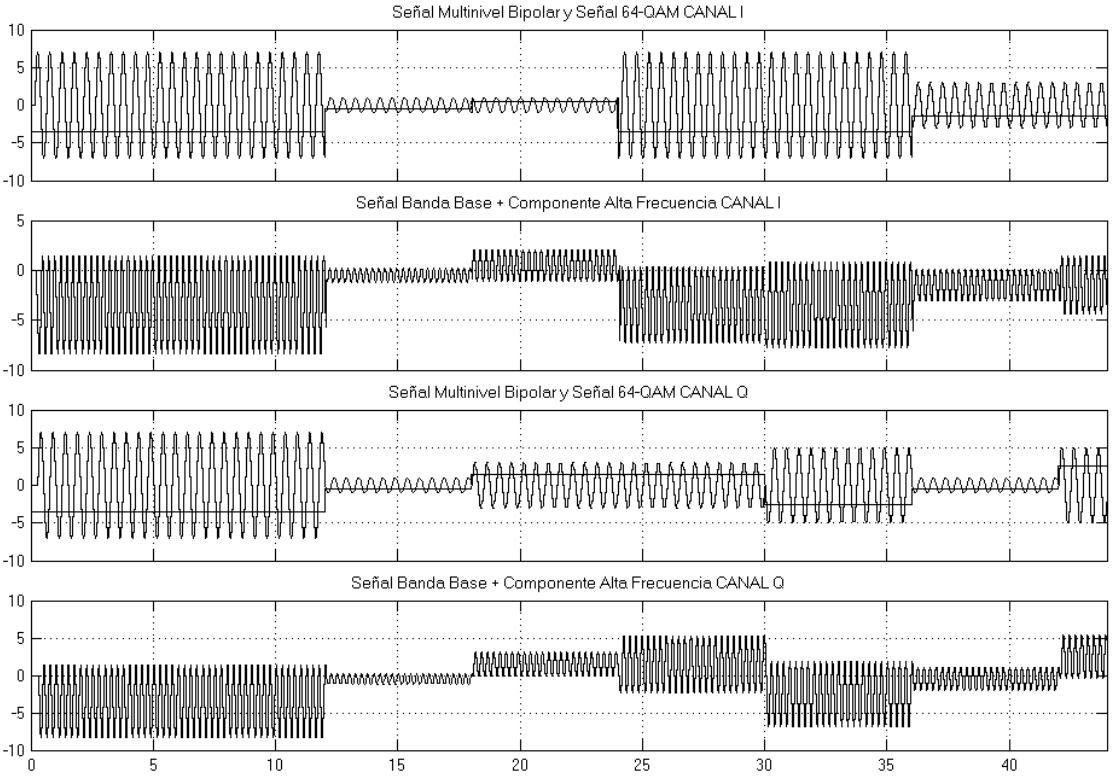


Figura 4.39. Simulación del Multiplicador, para 64-QAM canales I-Q

En la primera división está la señal bipolar en el canal I que se desea recuperar, y la multiplicación de ésta con la portadora en fase. En la segunda división se tiene la señal resultante del bloque multiplicador del canal I, que es una de mayor frecuencia que la portadora más una componente continua. En las divisiones tercera y cuarta se tienen los mismos resultados pero para el canal Q.

4.1.4.2. Filtro LPF

Las salidas de los multiplicadores producen una señal sinusoidal bipolar, como se evidencia en la simulación del subsistema anterior. Esta señal contiene componentes de frecuencia que deben ser filtradas para conseguir la señal multinivel bipolar. Para ilustrar el proceso de filtración, se va a tomar la modulación 4-QAM, en la cual se tiene una de las cuatro fases que se representan así:

$$\text{Entrada } 4\text{-QAM} \Rightarrow \begin{cases} +\cos\omega_c t + sen\omega_c t \\ +\cos\omega_c t - sen\omega_c t \\ -\cos\omega_c t + sen\omega_c t \\ -\cos\omega_c t - sen\omega_c t \end{cases}$$

Considerando la segunda entrada para la modulación 4-QAM, la salida del multiplicador en el canal Q es:

$$\begin{aligned} Q &= \underbrace{(+\cos\omega_c t - sen\omega_c t)}_{\text{Entrada 4-QAM}} \cdot \underbrace{sen(\omega_c t)}_{\text{Portadora}} \\ &= (\cos\omega_c t)(sen\omega_c t) + (-sen\omega_c t)(sen\omega_c t) \\ &= (\cos\omega_c t)(sen\omega_c t) - sen^2\omega_c t \\ &= \frac{1}{2}sen(\omega_c + \omega_c)t + \frac{1}{2}sen(\omega_c - \omega_c)t - \frac{1}{2}(-\cos 2\omega_c t) \\ &= \frac{1}{2}sen(2\omega_c)t + \frac{1}{2}sen(0) - \frac{1}{2} + \frac{1}{2}\cos 2\omega_c t \end{aligned}$$

$$\begin{aligned} &= \overbrace{\frac{1}{2}sen2\omega_c t}^{\text{Cero al filtrarlo}} - \frac{1}{2} + \overbrace{\frac{1}{2}\cos 2\omega_c t}^{\text{Cero al filtrarlo}} \\ &= -\frac{1}{2} \end{aligned}$$

Para eliminar las componentes de frecuencia, basta con diseñar un filtro pasa bajos LPF (*Lowpass Filter*) y recuperar así la señal multinivel bipolar.

El LPF tiene una frecuencia de punto de corte mucho más baja que $2\omega_C$ y en consecuencia, bloquea la segunda armónica de la portadora y pasa sólo la componente constante[9].

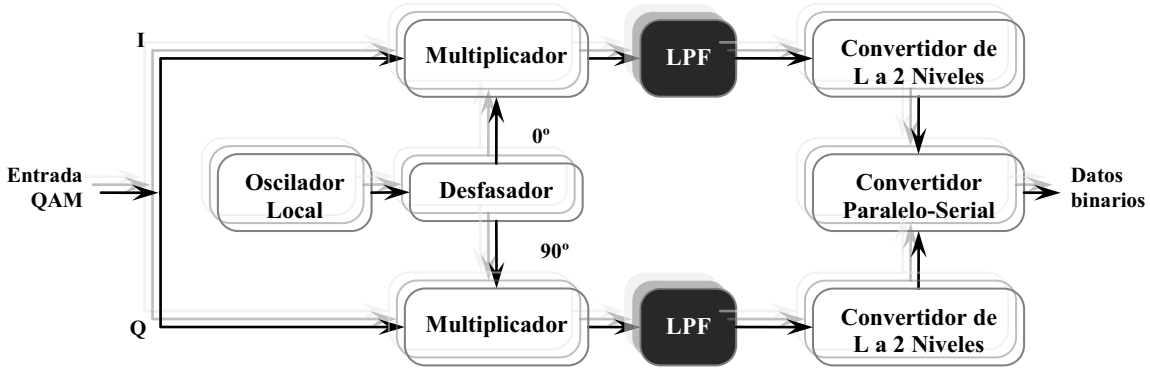


Figura 4.40. Filtro Pasa Bajos LPF en el Demodulador

Para la implementación del filtro LPF se utiliza el bloque DAFIR en ambos canales y para el diseño del filtro digital se recurre a la herramienta FDATool como se indicó en el capítulo 3. Las frecuencias necesarias para el diseño son la frecuencia de corte o de paso (*pass*) f_P y la frecuencia de parada (*stop*) f_S, que se determinan así:

El ancho de banda más amplio entre los tipos de modulación QAM es:

$$4\text{-QAM} \quad \Rightarrow \quad AB_{DBL} = 3{,}125\,KHz$$

Esto quiere decir que en Banda Base:

$$AB_{BLU} = 1{,}563\,KHz$$
$$\Rightarrow f_P = 1{,}563\,KHz$$

Para f_S se considera:

$$f_S < 2f_C - AB_{BLU}$$
$$f_S < 25\,KHz - 1{,}563\,KHz$$
$$f_S < 23{,}437\,KHz$$

Para que el orden del filtro no sea alto, la transición debe ser suave

$$\Rightarrow f_S = 10\ KHz$$

[9] TOMASI, Wayne, Sistemas de Comunicaciones Digitales, p. 467.

Las frecuencias f_P y f_S se determinaron para el peor caso que es el mayor ancho de banda utilizado, por lo tanto son válidas también para 16-QAM y 64-QAM.
La configuración de parámetros para el diseño del filtro y la obtención de sus coeficientes se muestra en la figura 4.41.

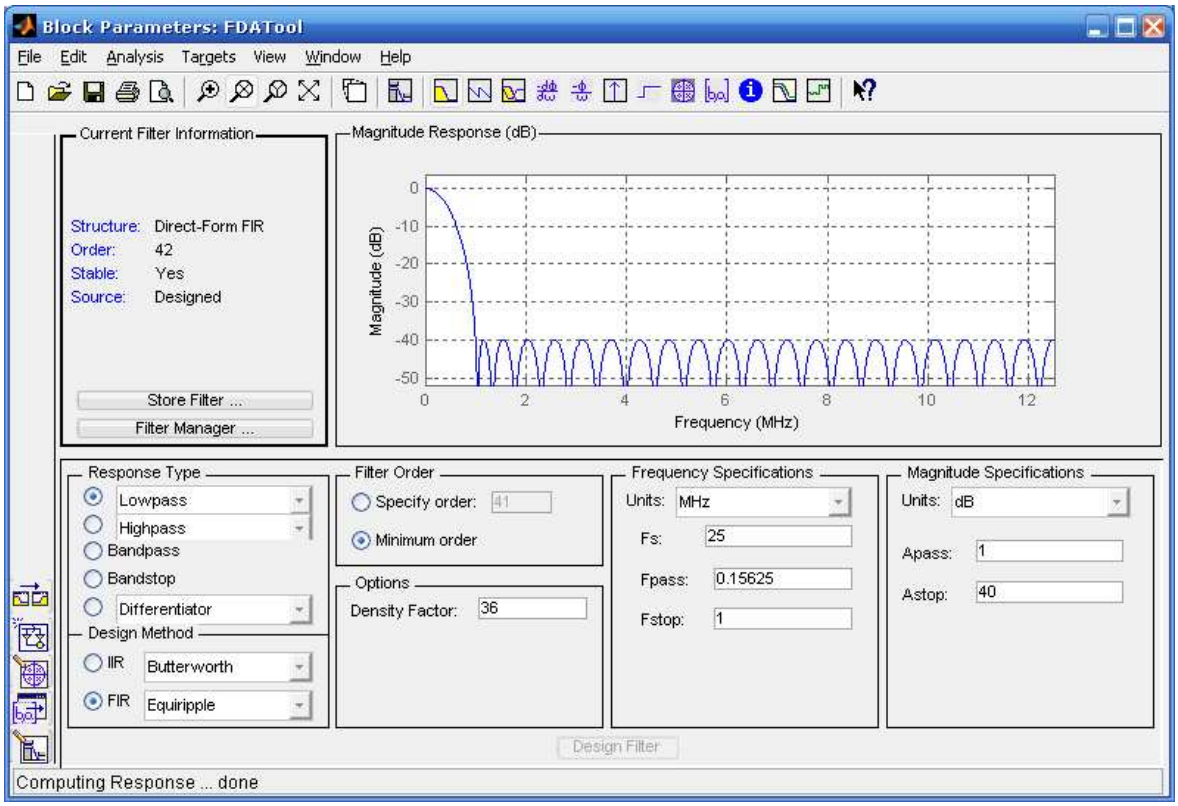


Figura 4.41. Diseño del Filtro Pasa Bajos con FDATool

La respuesta del filtro que se diseñó es la misma para los dos canales, como se puede deducir del desarrollo matemático del filtrado en 4-QAM, esto permite el empleo del modo multicanal en el bloque DAFIR.
Como la frecuencia de muestreo en las señales de entrada al filtro es la máxima posible 50 MHz, y el bloque DAFIR las va a procesar paralelamente, se debe acondicionar estas entradas para que su f_M sea menor o igual a 50 MHz / 2. Por ello se ubica un bloque *Down Sample* configurado con un valor de 2, en cada entrada al filtro.

Por último, la frecuencia de muestreo de la señal bipolar multinivel en cada canal, debe ser igual a la original (salida del bloque convertidor de 2 a L niveles), por lo

que se disminuye la frecuencia de muestreo ahora a la salida del filtro, también con un bloque *Down Sample.* El valor de configuración depende de la f_M en la portadora que es 80 y de la duración de cada nivel. Ésta última en general es *log2(n)*, pero como se acondicionó la señal en la entrada del filtro, éste valor se divide para 2.

La conexión de los bloques que conforman este subsistema, se realiza en el ambiente de Simulink y se muestra en la figura 4.42.

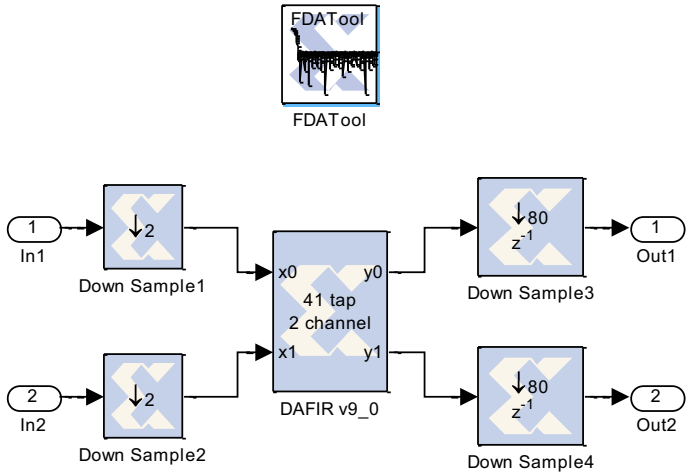


Figura 4.42. Modelación en Simulink con SysGen del bloque LPF

BLOQUE	PESTAÑA	PARÁMETROS	VALOR
Down Sample1 *Down Sample2*	*Basic*	*Sampling rate* *Sample*	*2* *First value of frame*
DAFIR v9_0	*Basic*	*Coefficients* *Structure* *Number of bits* *Binary point* *Latency*	*xlfda_numerator(´FDATool´)* *Inferred from coefficients* *16* *14* *33*
Down Sample3 *Down Sample4*	*Basic*	*Sampling rate* *Sample* *Latency*	*1/2*log2(n)*80* *Last value of frame* *1*

Tabla 4.16. Configuración de los parámetros del LPF

La configuración de parámetros de cada bloque que se utiliza en este subsistema, se muestra resumida en la tabla 4.16. Es importante destacar que los valores indicados en el parámetro *Latency* de los bloques DAFIR y *Down Sample* son los mínimos requeridos para su funcionamiento.

La simulación muestra la señal filtrada para el canal I-Q y las salidas del subsistema LPF para 4, 16 y 64-QAM.

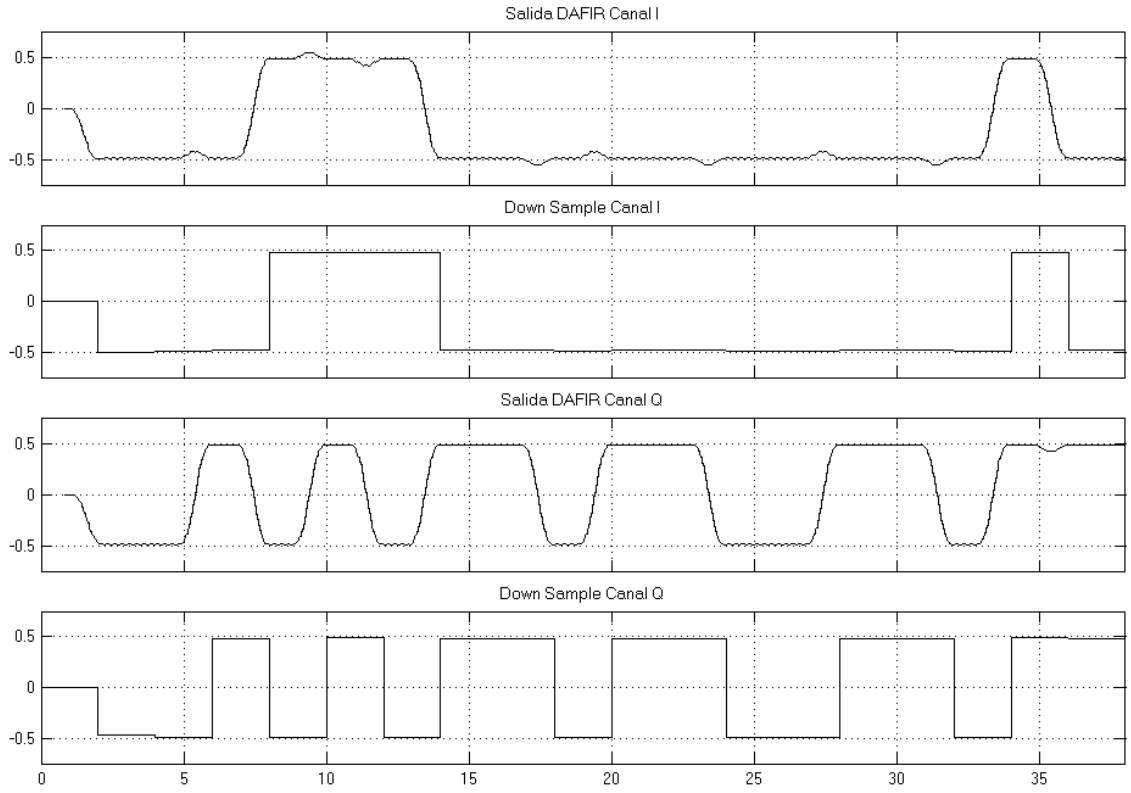


Figura 4.43. Simulación del LPF, para 4-QAM canales I-Q

En las figuras 4.43, 4.44 y 4.45, se observa que la señal filtrada no tiene niveles bien determinados, como se esperaría en un filtro real.

Además el uso de los bloques Down Sample a la salida del DAFIR funcionan correctamente porque la transición de nivel es vertical y mantiene el nivel en el valor adecuado.

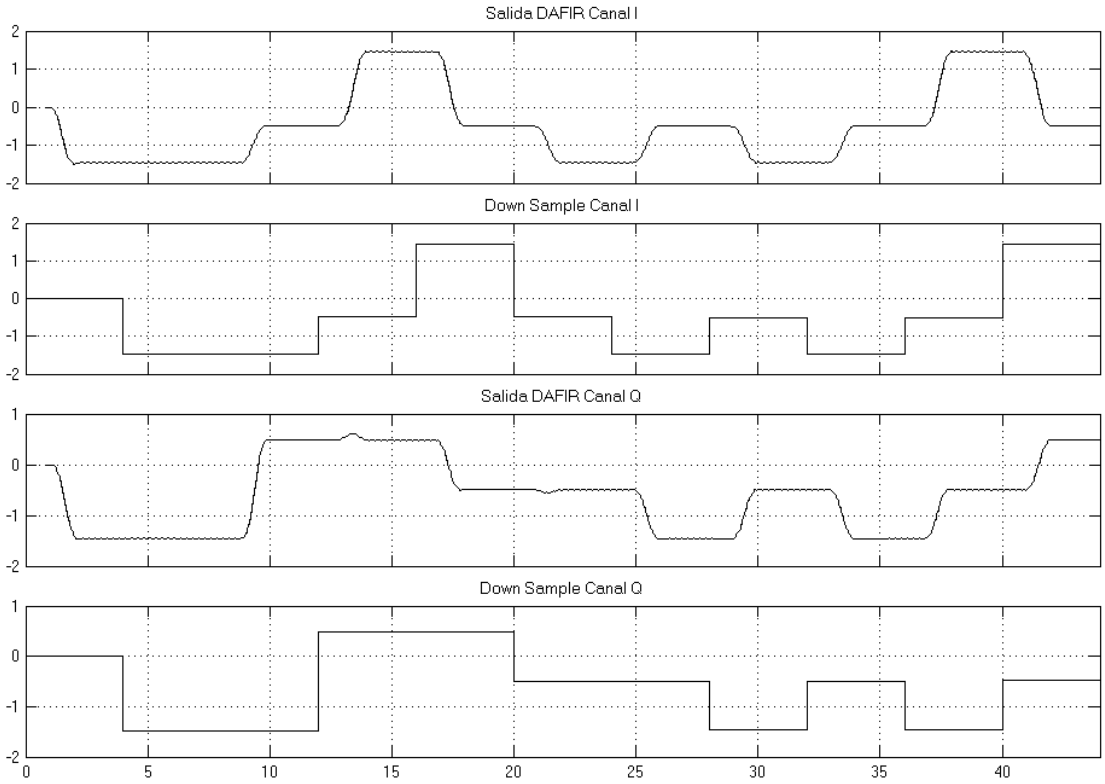


Figura 4.44. Simulación del LPF, para 16-QAM canales I-Q

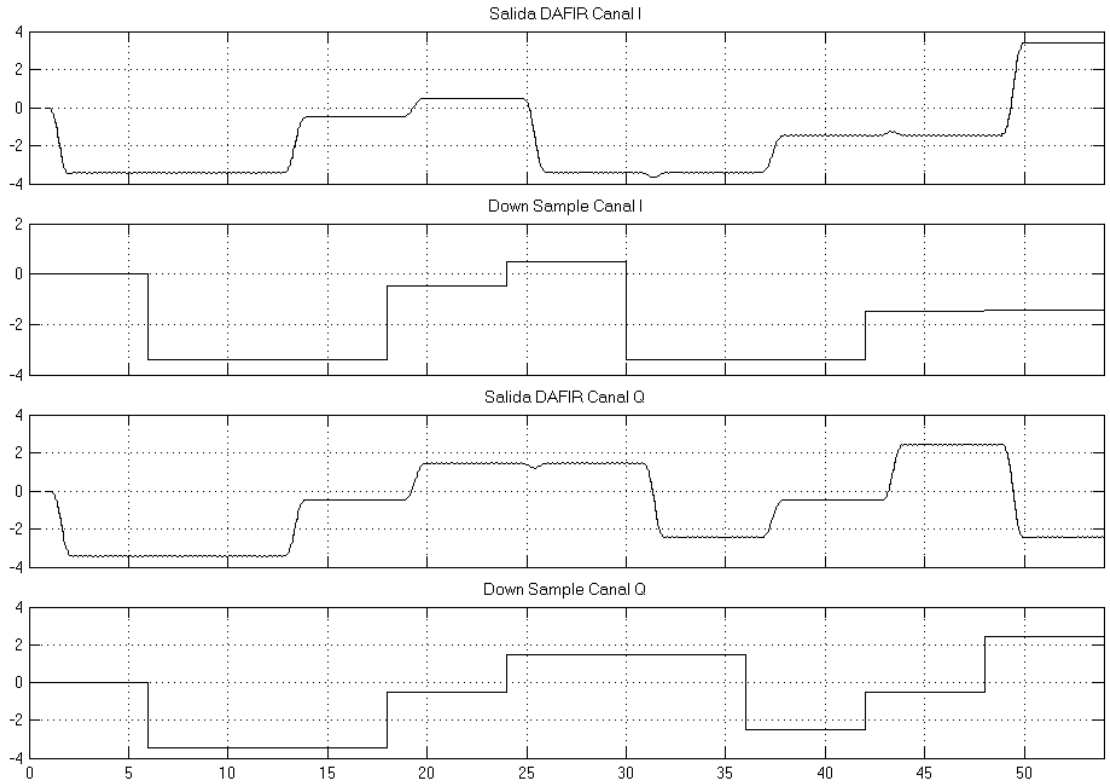


Figura 4.45. Simulación del LPF, para 64-QAM canales I-Q

4.1.4.3. Convertidor de L a 2 Niveles

Para diseñar este bloque, se planea seguir similares pasos a los realizados en el convertidor del modulador. En esta ocasión, se tiene como entrada a la señal bipolar multinivel (L niveles), de la cual se obtendrán los bits de información del canal respectivo.

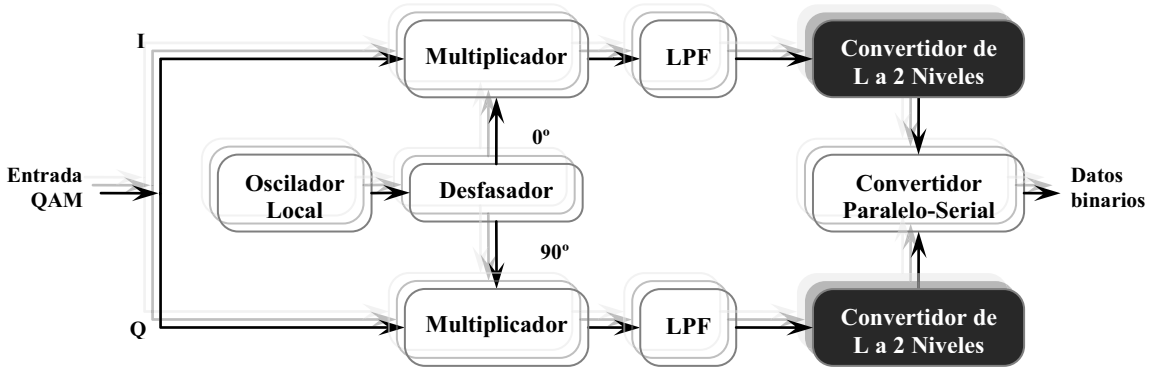


Figura 4.46. Convertidor de L a 2 Niveles en el Demodulador

Para conseguir esto, primero se debe obtener una señal unipolar, que convenientemente es la representación en decimal de los bits contenidos en cada símbolo, como se indicó en la tabla 4.5, para después convertirlos en su equivalente binario.

Entonces, si se suma un valor constante k a la señal bipolar, que sea igual al valor absoluto del nivel mínimo asignado, se obtiene la señal unipolar requerida. Pero como los niveles resultantes a la salida del filtro, no son exactamente los de la tabla 4.5, se debe realizar un ajuste. Esta constante tiene el mismo valor que en el caso del modulador para cada caso de modulación QAM.

Entonces, para obtener la señal unipolar se necesita un bloque *AddSub* (Sumador-Restador) y un bloque *Constant* configurado con el valor de k, de la misma forma que en el modulador.

El procedimiento de ajuste que se mencionó, se realiza con un cambio de la representación aritmética de cada nivel unipolar. Este procedimiento, permite modificar la parte decimal en el número ya sea por eliminación o redondeo. Por

ejemplo en 16-QAM, si se observa que el nivel mínimo a la salida del sumador es 3.0625, cuando se lo represente sólo con la parte entera se tendrá un nivel de 3.

El bloque disponible en el Blockset de Xilinx que efectúa este proceso, se denomina *Convert*, y es configurado para mostrar únicamente la parte entera del número con 1, 2 o 3 bits según la modulación.

El próximo paso es tener el equivalente binario del número decimal, lo que se realizará siguiendo la descripción de la tabla 4.17.

n	Canal I	Canal Q	Secuencia de bits Recuperada
4	…$I_3I_3I_2I_2I_1I_1$ …_I_3_I_2_I_1 …_I_3_I_2_I_1	…$Q_3Q_3Q_2Q_2Q_1Q_1$ …_Q_3_Q_2_Q_1 …_Q_3_Q_2_Q_1_	… $Q_3I_3Q_2I_2Q_1I_1$
16	… $I_{3-4}I_{3-4}$ $I_{1-2}I_{1-2}$ …_ _I_{3-4}_ _I_{1-2} …_ _I_4I_3_ _I_2I_1	… $Q_{3-4}Q_{3-4}$ $Q_{1-2}Q_{1-2}$ …_ _Q_{3-4}_ _Q_{1-2} …_ _Q_4Q_3_ _Q_2Q_1_ _	… $Q_4Q_3I_4I_3Q_2Q_1I_2I_1$
64	… $I_{4-5-6}I_{4-5-6}$ $I_{1-2-3}I_{1-2-3}$ …_ _ _I_{4-5-6}_ _ _I_{1-2-3} …_ _ _$I_6I_5I_4$_ _ _$I_3I_2I_1$	… $Q_{4-5-6}Q_{4-5-6}$ $Q_{1-2-3}Q_{1-2-3}$ …_ _ _Q_{4-5-6}_ _ _Q_{1-2-3} …_ _ _$Q_6Q_5Q_4$_ _ _$Q_3Q_2Q_1$_ _ _	… $Q_3Q_2Q_1I_3I_2I_1$

Tabla 4.17. Separación en bits de una señal unipolar multinivel de cada canal I-Q

El número decimal en cuestión es la señal unipolar multinivel, que se encuentra representado como: I_1 o Q_1, I_{1-2} o Q_{1-2} e I_{1-2-3} o Q_{1-2-3} según el canal y la modulación n-QAM que corresponda.

Primero, para pasar de $Q_{1-2}Q_{1-2}$ a _ _Q_{1-2} se debe incrementar la frecuencia de muestreo, dividiendo así la representación del número decimal en dos, para luego eliminar una parte de ella introduciendo ceros. El bloque *Up Sample* proporciona estas dos funciones a la vez, configurando una tasa de muestreo de 2 y no seleccionando *Copy Samples*.

Y el último paso, es conseguir que el número decimal representado como un conjunto de bits, sea desarticulado para formar una serie de bits independientes. Con el bloque *Parallel to Serial* se consigue esta tarea de manera efectiva configurando como uno el número de bits de salida.

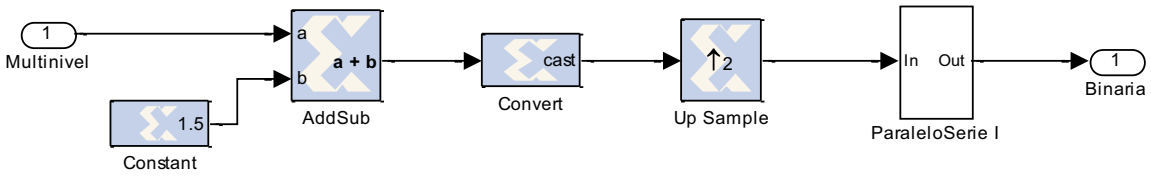


Figura 4.47. Modelación en Simulink con SysGen del bloque Convertidor de L a 2 Niveles para el Canal I

La configuración de parámetros de cada bloque que se utiliza en este subsistema, se muestra resumida en la tabla 4.18, que es la misma para los dos canales.

BLOQUE	PESTAÑA	PARÁMETROS	VALOR
AddSub	*Basic*	*Operation*	*Subtraction*
Constant	*Basic*	*Type* *Constant value* *Number of bits* *Binary point*	*Unsigned* *(sqrt(n) - 1) / 2* *1/2*log2(n)* *1*
Convert	*Basic*	*Type* *Number of bits* *Binary point* *Quantization* *Overflow*	*Unsigned* *1/2*log2(n)* *0* *Round* *Wrap*
Up Sample	*Basic*	*Sampling rate*	*2*
Delay	*Basic*	*Latency*	*1/2*log2(n)*

Tabla 4.18. Configuración de los parámetros del Convertidor de L a 2 Niveles

Para la generalización de este subsistema se tuvo un inconveniente con el bloque *Parallel to Serial*, éste no permite la visualización de bits individuales en la salida si la señal de entrada está representada con un número de bits igual a 1, como es

el caso de la modulación 4-QAM. Debido a esto se necesita un subsistema adicional que discrimine la entrada al bloque *Parallel to Serial*.

Este subsistema llamado ParaleloSerie I-Q, se compone de un multiplexor 3:1 para la selección de la entrada, el control se realiza con una constante que depende del parámetro 'n'. El bloque principal *Parallel to Serial* también está incluido pero únicamente en 16 y 64-QAM.

La conexión de los bloques que conforman este subsistema para cada canal, se realiza en el ambiente de Simulink y se muestra en las figuras 4.48 y 4.49.

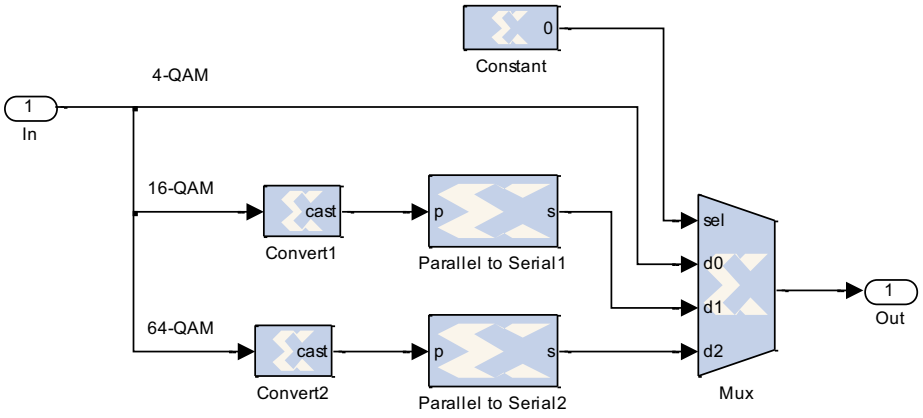


Figura 4.48. Modelación en Simulink con SysGen del subsistema ParaleloSerie I

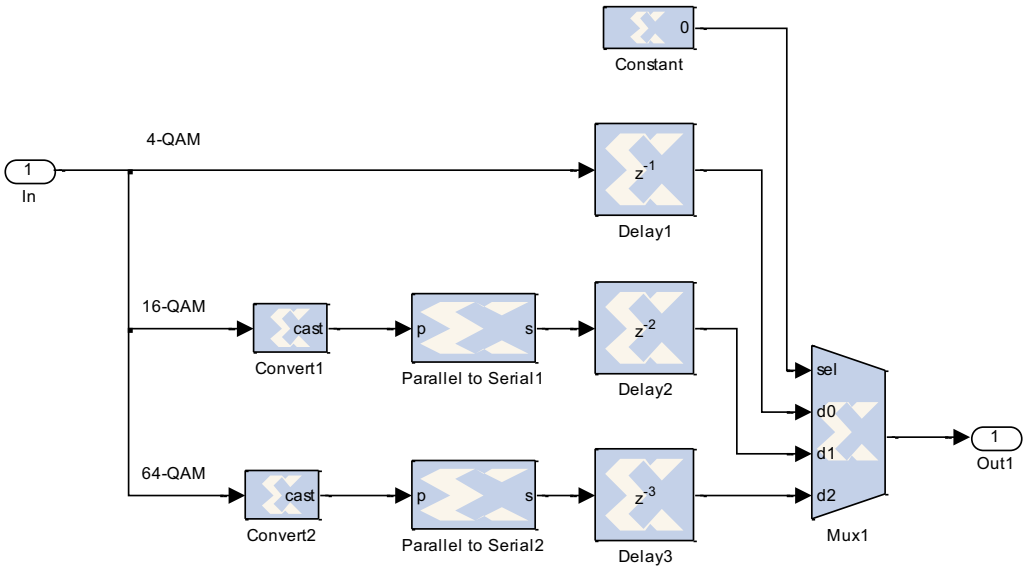


Figura 4.49. Modelación en Simulink con SysGen del subsistema ParaleloSerie Q

El bloque *Delay* en el canal Q tiene la función de agregar un retardo según la modulación, para que en un determinado tiempo en el canal I existan bits válidos mientras que en el canal Q ceros y viceversa.

La configuración de parámetros de cada bloque en este subsistema, se muestra resumida en la tabla 4.19.

BLOQUE	PESTAÑA	PARÁMETROS	VALOR
Convert1	*Basic*	*Type* *Number of bits* *Binary point* *Quantization* *Overflow*	*Unsigned* *2* *0* *Truncate* *Wrap*
Convert2	*Basic*	*Number of bits*	*6 - 1/2*log2(n)*
Parallel to Serial1 *Parallel to Serial2*	*Basic*	*Output order* *Output type* *Number of bits* *Binary point*	*Most significant word first* *Unsigned* *1* *0*
Delay1	*Basic*	*Latency*	*1*
Delay2	*Basic*	*Latency*	*2*
Delay3	*Basic*	*Latency*	*3*
Constant	*Basic*	*Type* *Constant value* *Number of bits* *Binary point*	*Unsigned* *1/2*log2(n) - 1* *2* *0*
Mux	*Basic*	*Number of inputs*	*3*

Tabla 4.19. Configuración de parámetros, subsistema ParaleloSerie I-Q

La simulación en la figura 4.50, muestra en primer lugar la función de cada bloque de la figura 4.47 para 16-QAM, para paso a paso aclarar su función dentro del

subsistema. Comprendido esto, los bits recuperados en cada canal son graficados para cada modulación QAM en las figuras 4.51, 4.52 y 4.53.

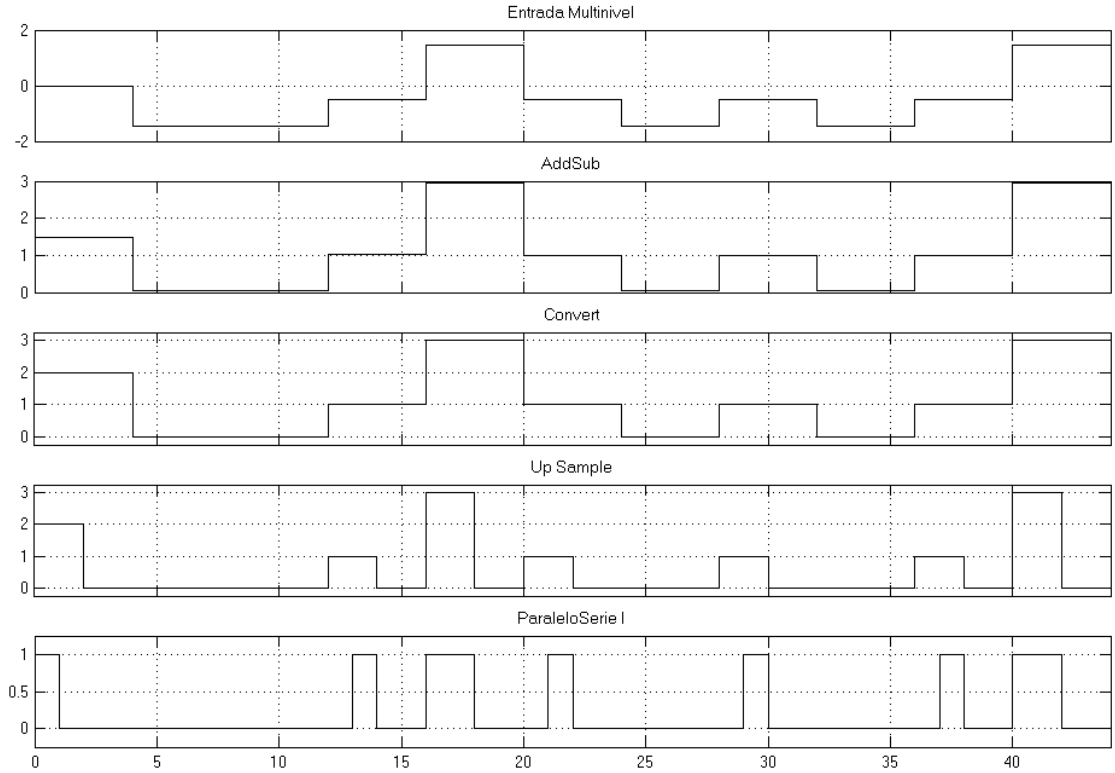


Figura 4.50. Simulación del bloque Convertidor de L a 2 Niveles para 16-QAM en detalle para el Canal I

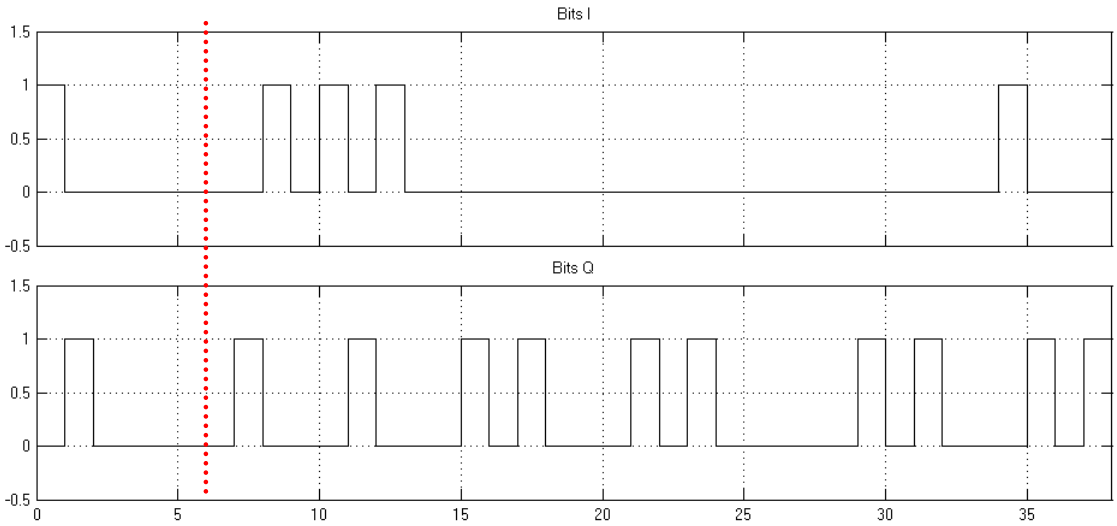


Figura 4.51. Simulación del bloque Convertidor de L a 2 Niveles para 4-QAM canales I-Q

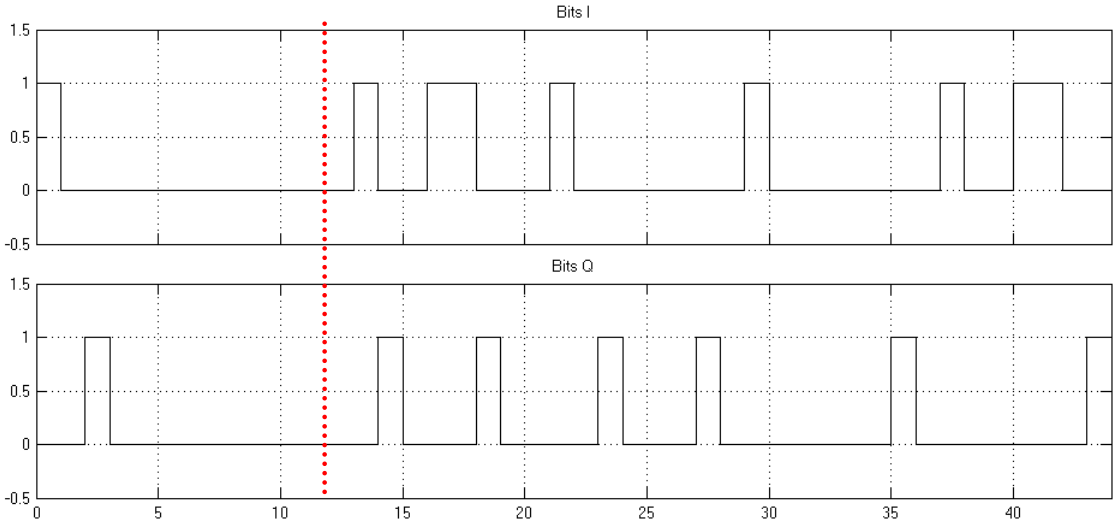


Figura 4.52. Simulación del bloque Convertidor de L a 2 Niveles para 16-QAM canales I-Q

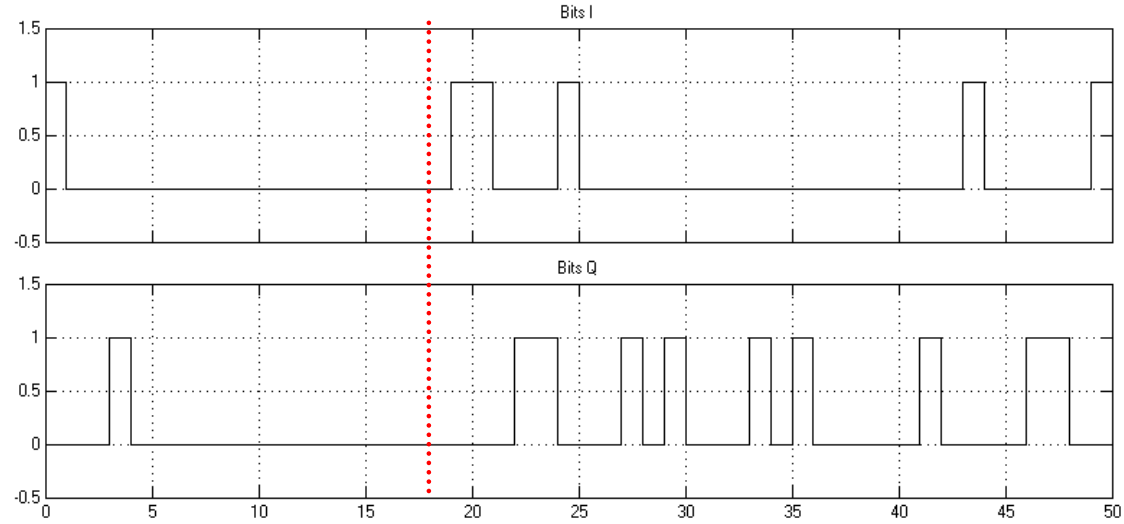


Figura 4.53. Simulación del bloque Convertidor de L a 2 Niveles para 64-QAM canales I-Q

Los bits que se encuentran a la derecha de la línea punteada corresponden a información válida, resultado de la demodulación del mensaje. El resto es efecto de la latencia introducida por el modulador y el demodulador.

4.1.4.4. Conversor Paralelo - Serial

En este subsistema, los canales de datos I-Q, que son paralelos llegan a ser nuevamente un sólo flujo de datos de binarios.

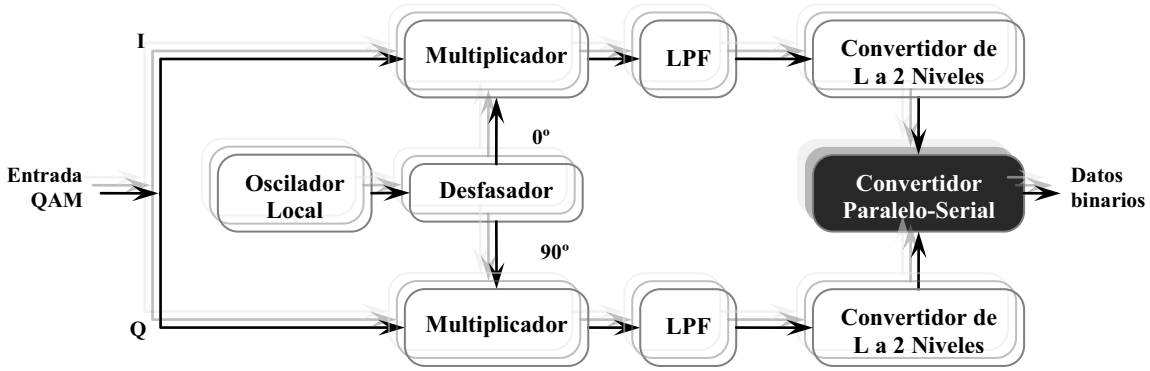
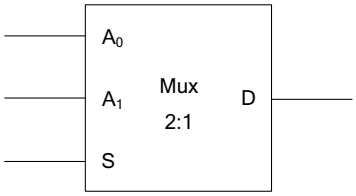


Figura 4.54. Convertidor Paralelo-Serial en el Demodulador

Para este subsistema, se necesita un multiplexor de dos entradas, que alterne los bits contenidos en los canales I-Q, para tener un flujo serial de datos. El diagrama del multiplexor 2:1 y su tabla de verdad se muestran en la figura 4.55.



(a)

A1	A0	S	D
0	I	0	I
Q	0	1	Q

(b)

Figura 4.55. (a) Bloque Multiplexor 2:1; (b) Tabla de verdad del Mux 2:1 para el conversor paralelo-serial

Si se observa con atención la salida D, en la tabla de verdad de la figura 4.55, en un caso está presente el canal ‘I’ y en el otro ‘Q’ en la salida, dependiendo del valor de la selección S.

Mediante la variación de la señal aplicada en la entrada de selección ‘S’, se pueden dejar pasar los bits de los canales I-Q como se indicó en la tabla 4.17 la secuencia de bits recuperada. El diseño en Simulink de esta parte del subsistema es idéntico al realizado en el divisor de datos del modulador.

La conexión del multiplexor y la señal de selección que se realiza en el ambiente de Simulink, se muestra en la figura 4.56.

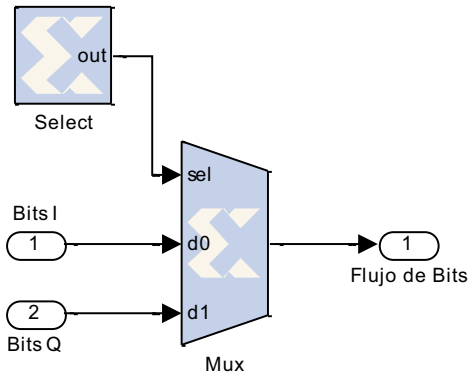


Figura 4.56. Modelación en Simulink con SysGen del bloque Convertidor Paralelo-Serial

La configuración de parámetros de cada bloque en este subsistema, se muestra resumida en la tabla 4.20.

BLOQUE	PESTAÑA	PARÁMETROS	VALOR
Select	*Basic*	*Counter type*	*Free Running*
		Count direction	*Up*
		Initial Value	*0*
		Step	*1*
		Output type	*Unsigned*
		Number of bits	*1*
		Binary point	*0*
	Advanced	*Explicit Sample Period*	*1/2*log2(n)*
Mux	*Basic*	*Number of inputs*	*2*
	Output Type	*Precision*	*User Defined*
		Output type	*Unsigned*
		Number of bits	*1*
		Binary point	*0*

Tabla 4.20. Configuración de los parámetros del Convertidor Paralelo-Serial

La simulación tiene cinco partes para cada modulación QAM, en primer lugar la señal demodulada QAM esto quiere decir los bits recuperados, abajo están los bits del mensaje 'nQAM' original, en la tercera y cuarta parte se encuentran los bits de cada canal I-Q obtenidos del subsistema anterior y por último la señal de

selección que envía a la salida del multiplexor grupos de 1, 2 o 3 bits a la salida de forma alternada.

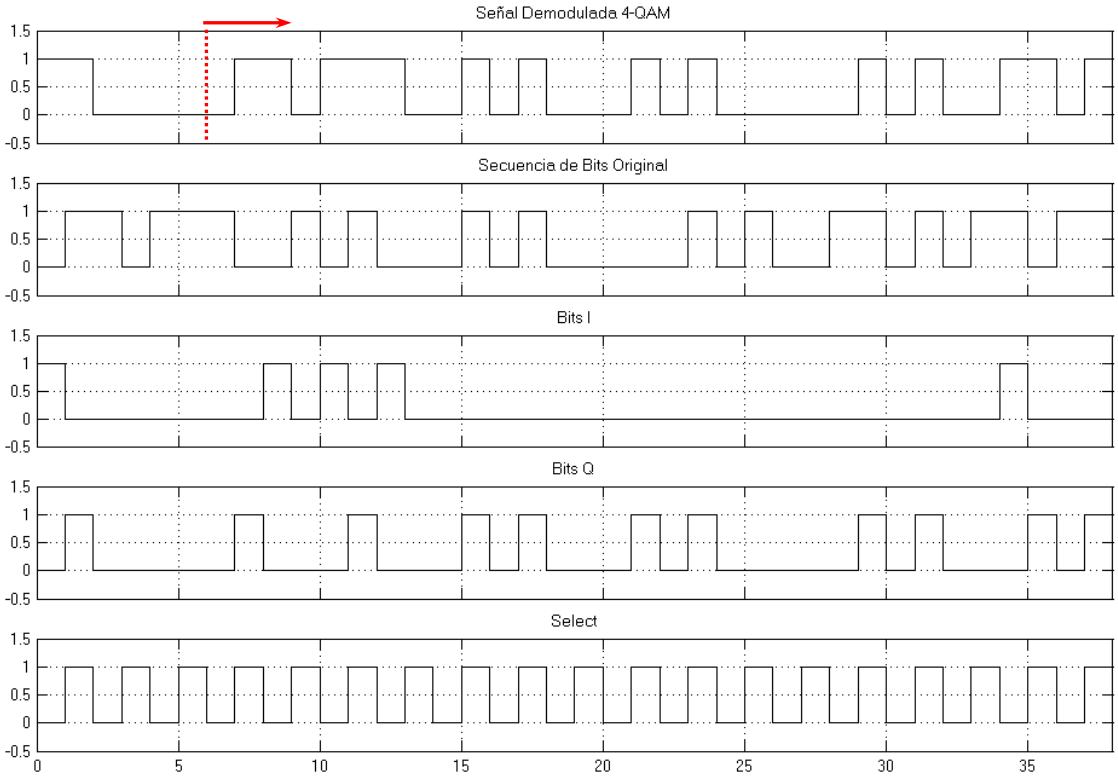


Figura 4.57. Simulación del bloque Conversor Paralelo-Serial para 4-QAM

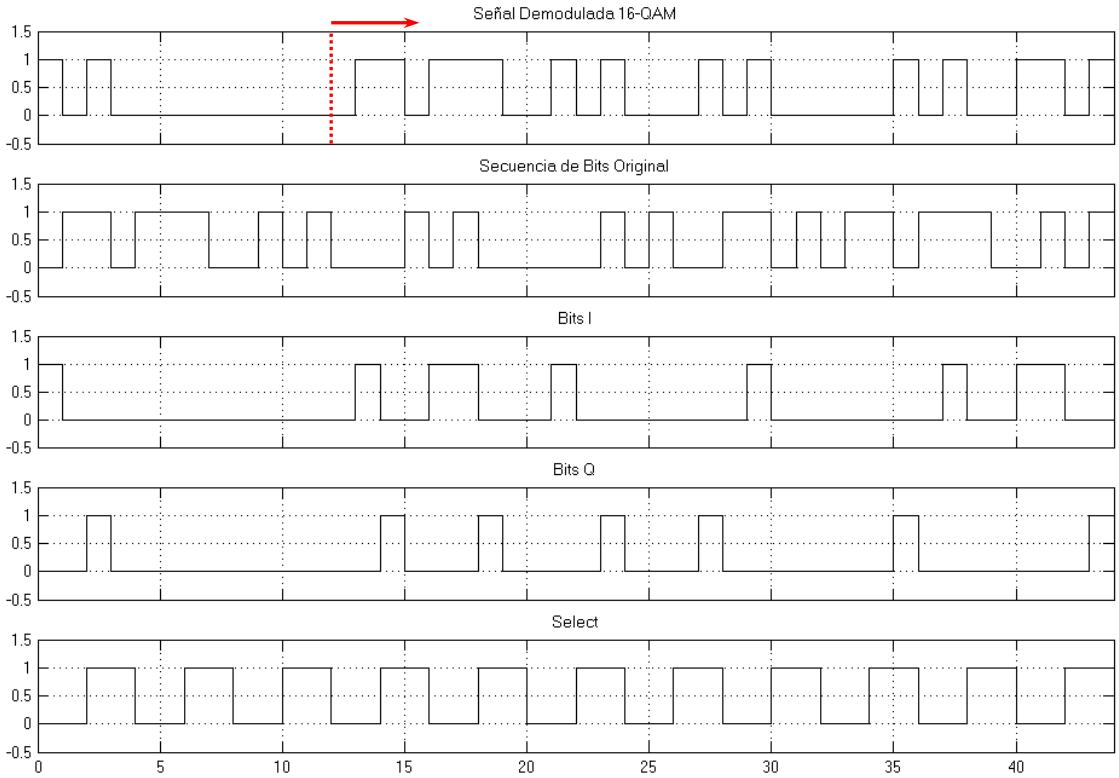


Figura 4.58. Simulación del bloque Conversor Paralelo-Serial para 16-QAM

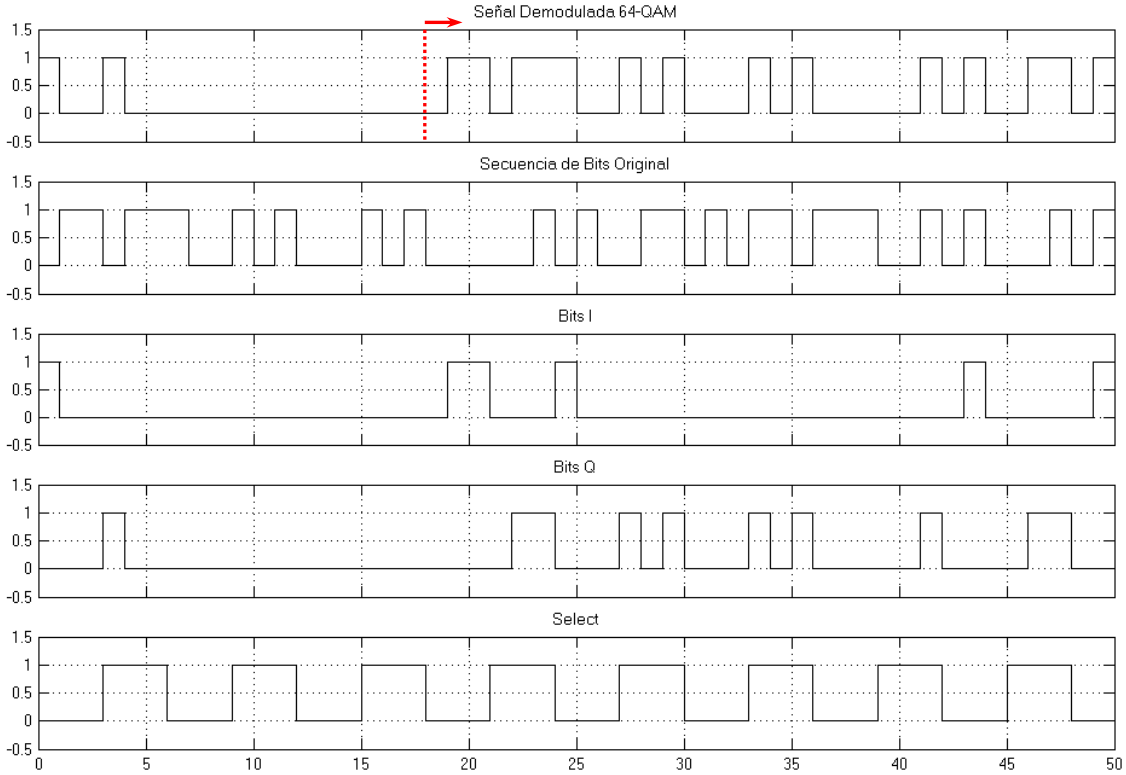


Figura 4.59. Simulación del bloque Conversor Paralelo-Serial para 64-QAM

Es claro y concluyente que el proceso de la demodulación es exitoso debido a que la secuencia de los bits, que se construyeron a partir del mensaje ‘nQAM’ representados en ASCII y la secuencia recuperada es idéntica.

CAPÍTULO V

5. IMPLEMENTACIÓN EN HARDWARE

5.1. INTRODUCCIÓN

Terminadas las etapas de diseño y simulación para ambos sistemas con total satisfacción, se realizará en esta sección la co-simulación en hardware y la implementación sobre tarjetas de entrenamiento basadas en un FPGA de Xilinx.

La tarjeta de entrenamiento utilizada para co-simular e implementar el modulador y el demodulador será la Spartan-3E, la que se describió en el capítulo 2. Los componentes que se utilizarán de esta tarjeta se muestran en la figura 5.1 y se enumeran a continuación.

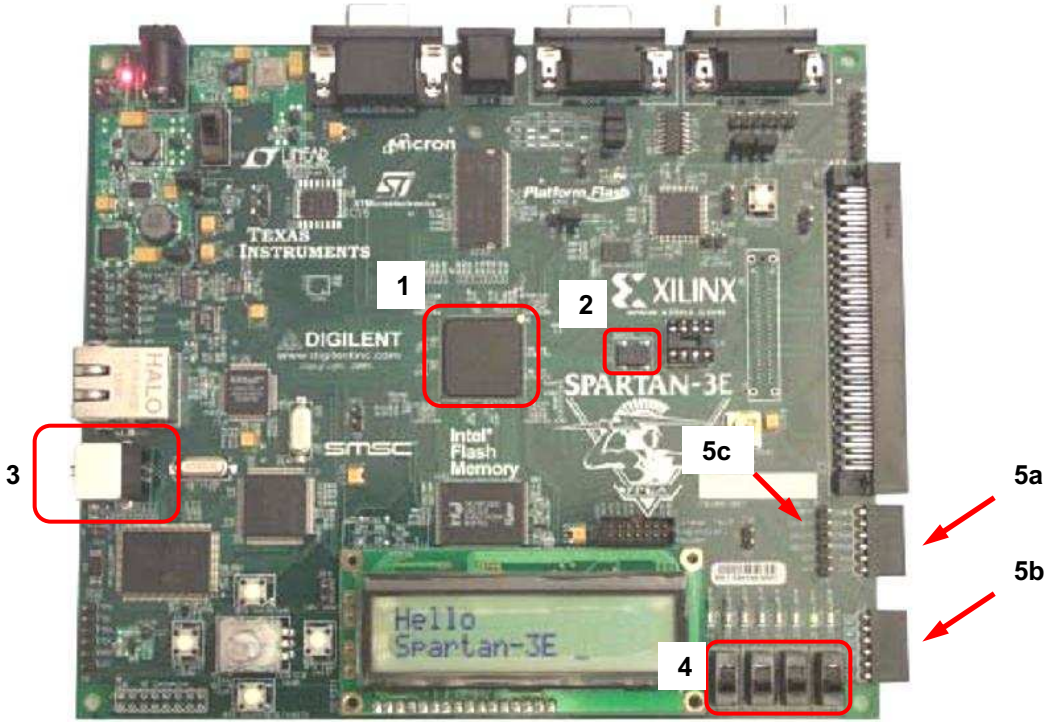


Figura 5.1. Tarjeta de desarrollo Spartan-3E

1. FPGA xc3s500e-4fg320c
2. Reloj oscilador de 50 Mhz.
3. Interfaz de programación USB.

4. Interruptores SW3, SW2, SW1 y SW0.
5. Conectores de expansión o pines de entrada/salida: J1, J2 y J4.

5.2. CO-SIMULACIÓN EN HARDWARE

Este proceso permite ejecutar un sistema diseñado en software sobre una plataforma de hardware para observar su comportamiento y desempeño desde un computador. *System Generator* facilita la co-simulación en hardware, debido a que proporciona un vínculo directo entre la simulación en Simulink del sistema y el FPGA, mediante la interfaz genérica JTAG (*Joint Test Action Group*) y el cable de programación de Xilinx (*Platform Cable USB*).

5.2.1. ESTIMACIÓN DE RECURSOS

Una herramienta que resulta muy útil al momento de trasladar el sistema modelado de SysGen a hardware es el bloque *Resource Estimator* descrito en el capítulo 3. Este bloque permite determinar el área utilizada en el FPGA, según el número de *slices,* flip-flops (FFs), bloques de memoria (BRAM), *lookup tables* (LUTs), buffers entrada/salida (IOB), multiplicadores y buffers tri-estados necesarios, para así conocer si es posible la implementación en la tarjeta disponible.

La estimación de los recursos utilizados en cada uno de los sistemas, se resume en la tabla 5.1.

RECURSOS	MÁXIMO DISPONIBLE	MODULADOR			DEMODULADOR		
		4-QAM	16-QAM	64-QAM	4-QAM	16-QAM	64-QAM
Slices	4656	98	112	123	2608	2760	2885
Flip-Flops	9312	126	151	161	4840	5127	5394
BRAMs	20	1	1	1	4	4	8
LUTs	9312	100	121	124	4142	4431	4671
Mults	20	2	2	2	2	2	2

Tabla 5.1. Estimación de Recursos

En todos los casos se utiliza una cantidad de recursos menor al máximo disponible, por lo tanto cada modulador y demodulador se puede implementar en una tarjeta Spartan-3E.

5.2.2. BLOQUE JTAG

Para realizar la co-simulación del sistema modelado con SysGen, se debe crear el bloque JTAG. La creación de este bloque se la hace en dos pasos por medio del Token, así se tiene que escoger primero la plataforma de co-simulación y luego invocar el generador de código.

La creación del bloque JTAG para el modulador es idéntica también para el demodulador, por lo que se indicará de forma general el procedimiento realizado para el modulador 4-QAM.

La selección de la plataforma utilizada en este proyecto en el Token de SysGen se muestra en la figura 5.2.

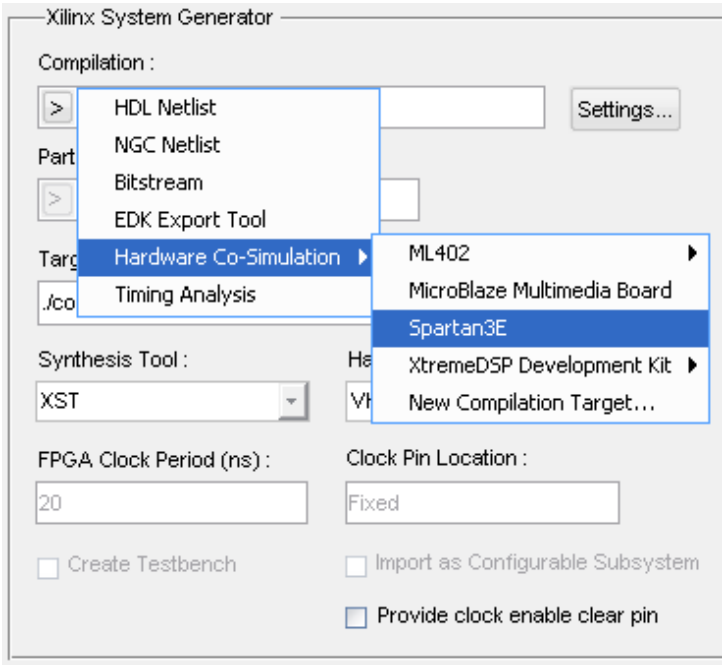


Figura 5.2. Selección de la Plataforma para Co-simulación

Para la invocación del generador de código se pulsa el botón *Generate,* con lo cual se compila el modelo y produce un archivo *bitstream* de configuración para el FPGA, que se asociará con el bloque JTAG.

En este archivo *bitstream,* se agrega automáticamente una interfaz lógica que permite la comunicación entre SysGen y el diseño sobre el FPGA, cuando se utilice la interfaz física (cable USB) entre la tarjeta y el computador[10].

Finalizada la compilación, se crea una librería en Simulink que contiene el bloque JTAG del sistema seleccionado. En la figura 5.3 se muestra el bloque JTAG para el modulador 4-QAM.

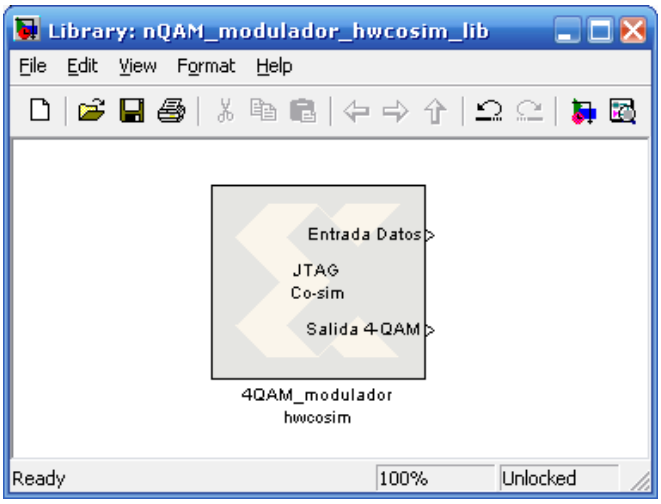


Figura 5.3. Bloque JTAG para el modulador 4-QAM

El bloque JTAG creado se lo puede manipular como cualquier otro de Simulink, por lo que se puede recurrir tanto a fuentes como a visualizadores para hacer las pruebas de co-simulación y además copiarlo al modelo de origen para comparar su desempeño en hardware con la simulación.

5.2.3. PRUEBAS DE CO-SIMULACIÓN

Para una mejor evaluación del comportamiento y desempeño de cada sistema, el bloque JTAG creado y el modelo en SysGen del cual proviene, se simularán en

[10] Matlab, Xilinx System Generator v8.1 User Guide, A Tutorial Introduction, Hardware Co-Simulation.

forma conjunta, por lo tanto las pruebas de co-simulación se realizarán según el diagrama de la figura 5.4.

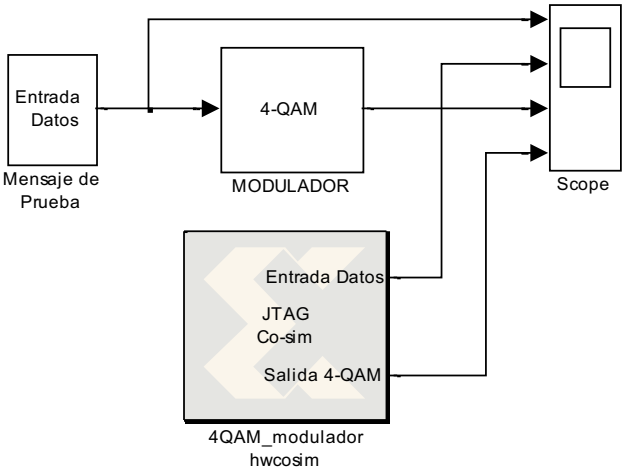


Figura 5.4. Diagrama General para Pruebas de Co-simulación

El bloque JTAG requiere el cambio de configuración en cuanto al cable de programación, ya que por defecto está *Parallel Cable IV* y se utilizará *Platform USB* como se indica en la figura 5.5.

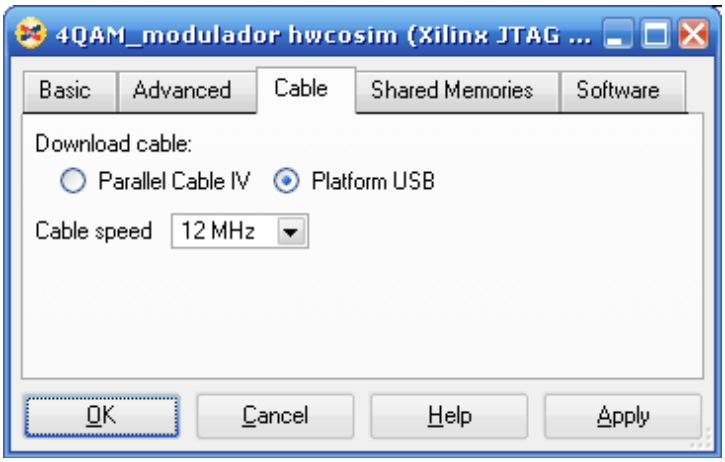


Figura 5.5. Configuración del Bloque JTAG

Para iniciar la co-simulación, primero se debe conectar la tarjeta de entrenamiento al computador mediante el cable USB, luego encenderla y por último se debe esperar que el sistema operativo reconozca la plataforma.

Los resultados de la co-simulación para el modulador y demodulador se indican en las figuras 5.6 a 5.11.

Para 4-QAM:

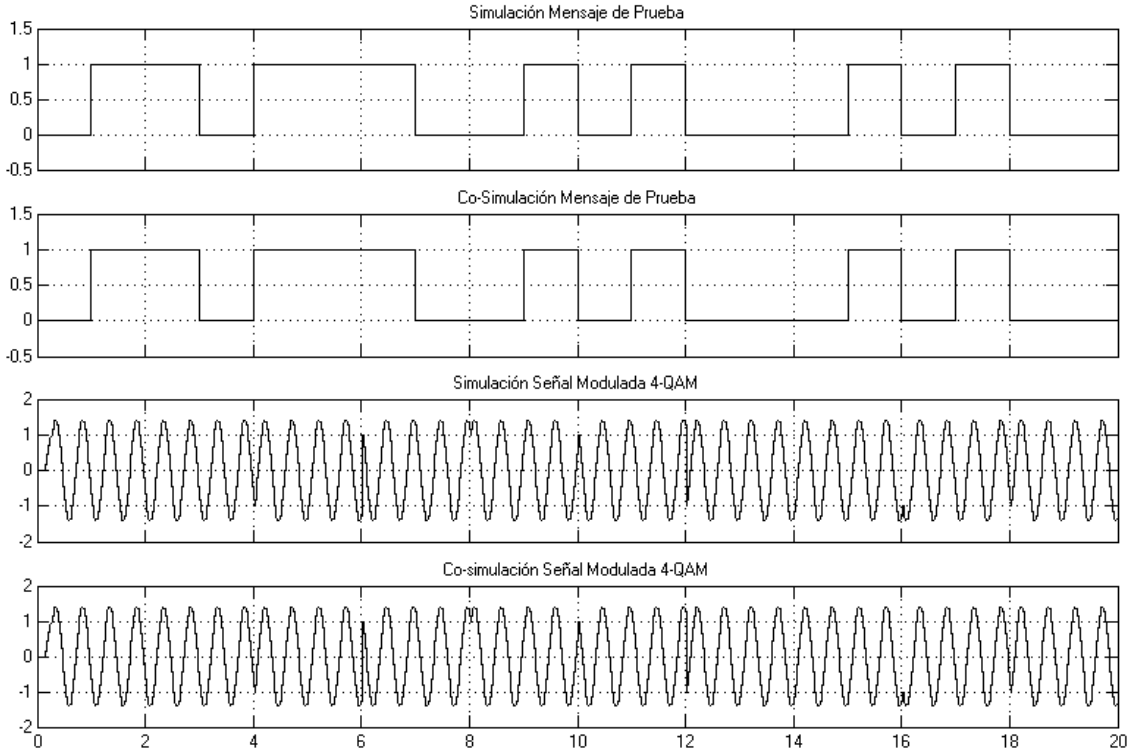


Figura 5.6. Co-simulación del Modulador 4-QAM

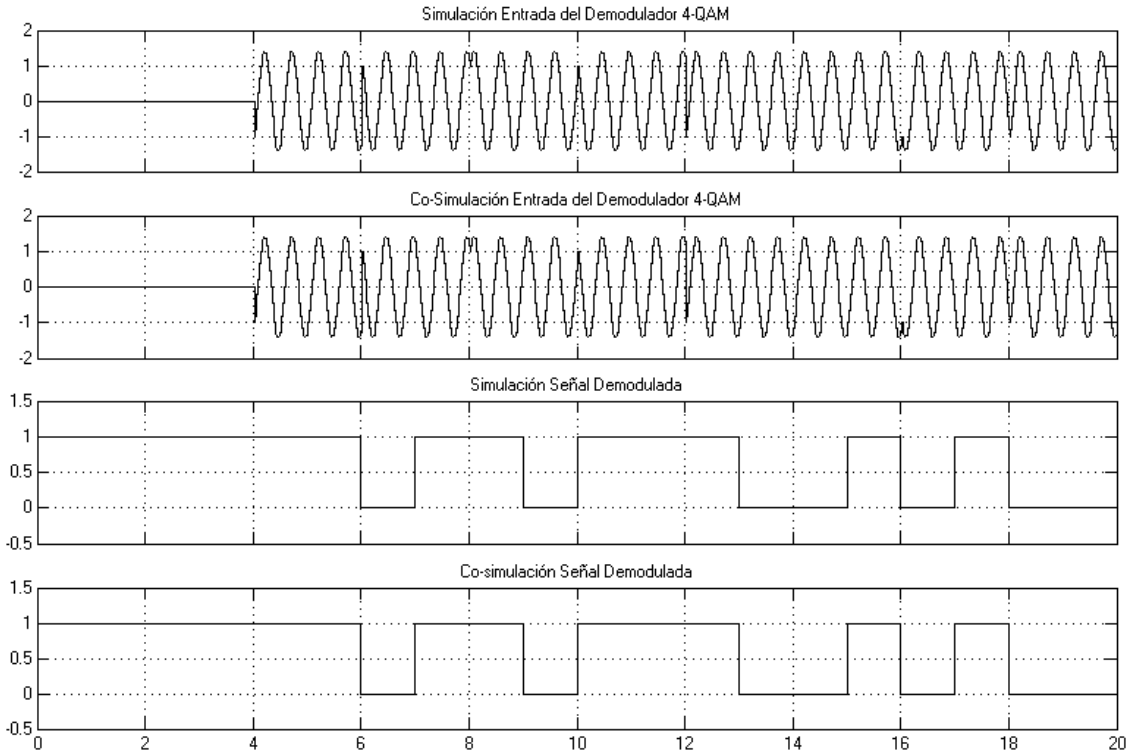


Figura 5.7. Co-simulación del Demodulador 4-QAM

Para 16-QAM:

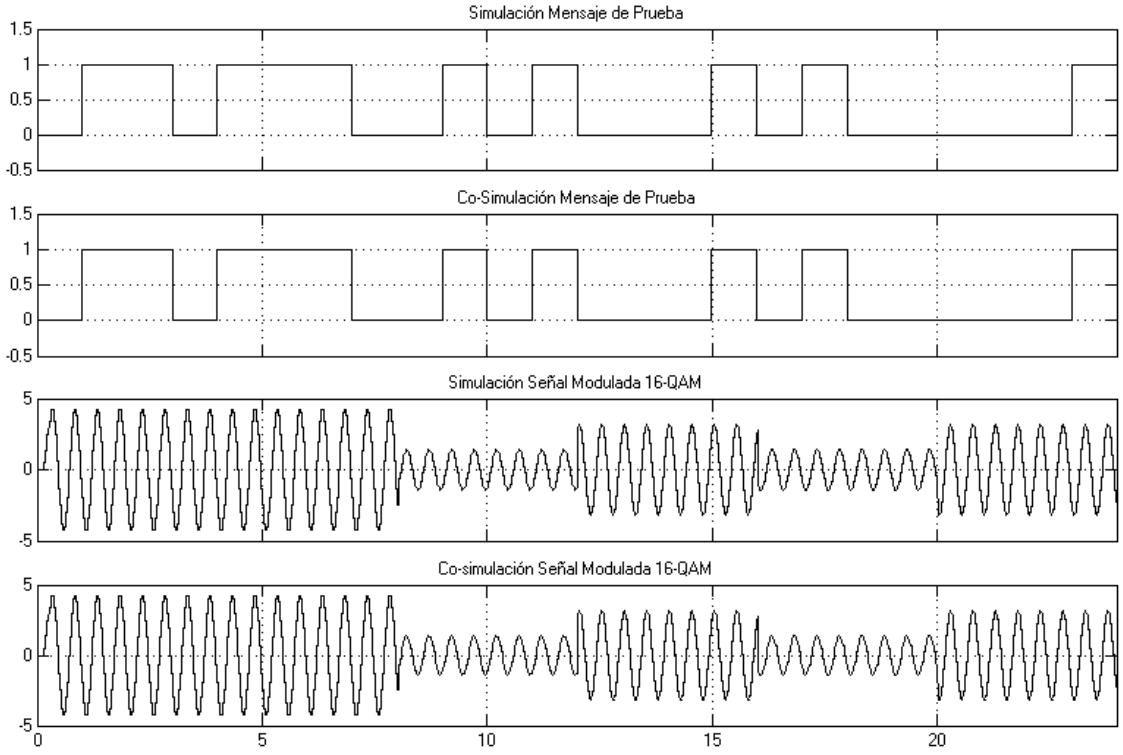


Figura 5.8. Co-simulación del Modulador 16-QAM

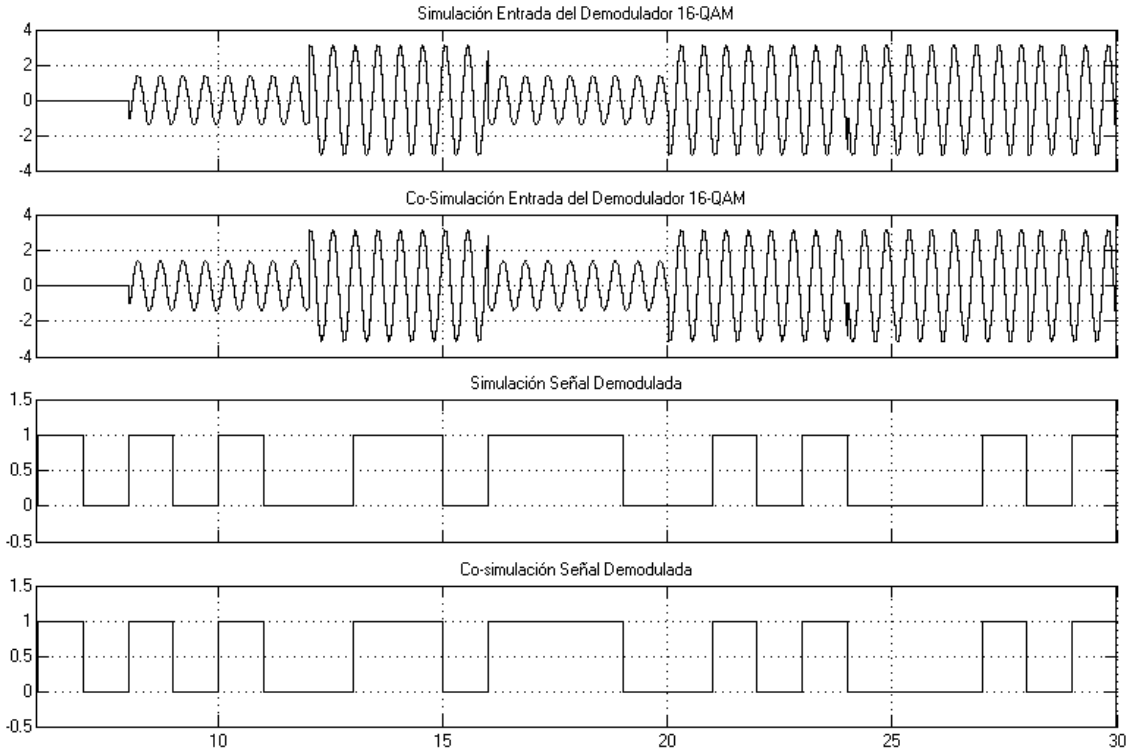


Figura 5.9. Co-simulación del Demodulador 16-QAM

Para 64-QAM:

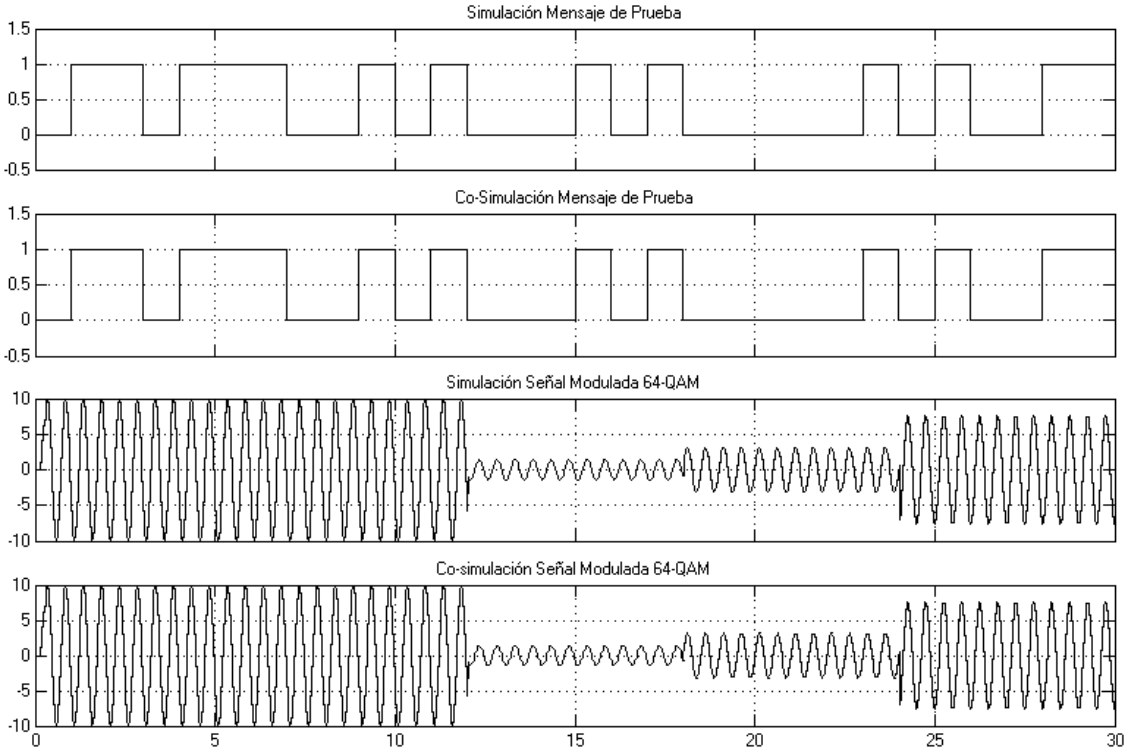


Figura 5.10. Co-simulación del Modulador 64-QAM

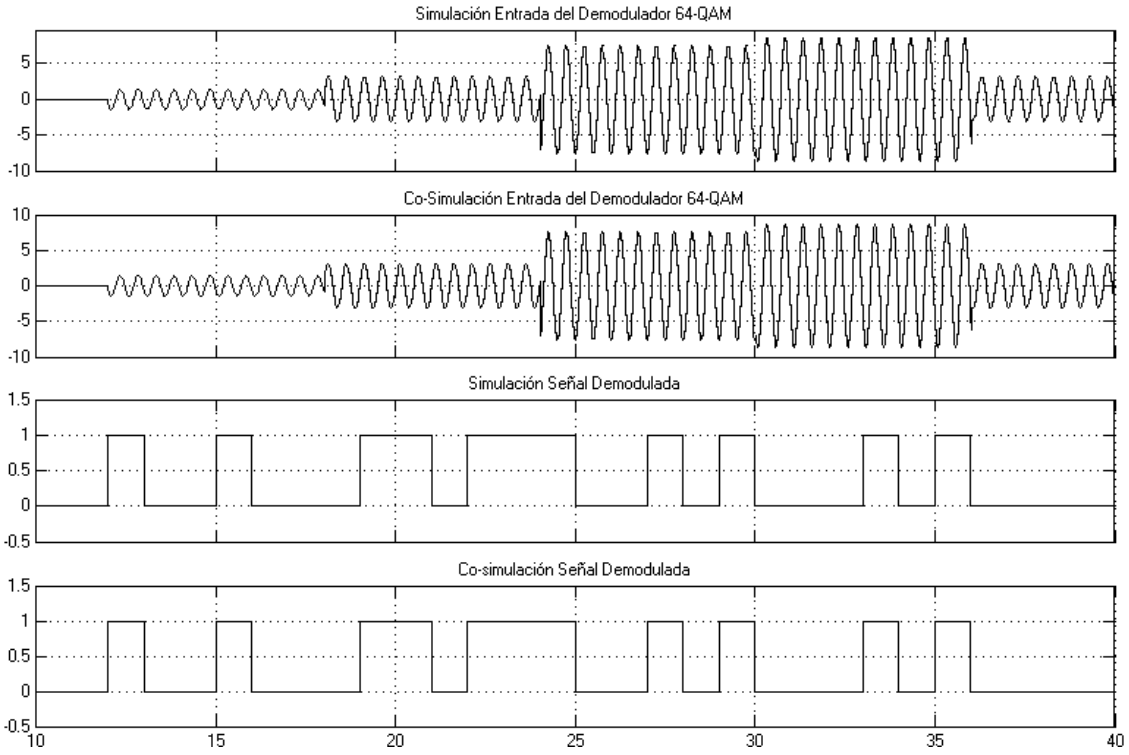


Figura 5.11. Co-simulación del Demodulador 64-QAM

Se observa que los resultados de la co-simulación son idénticos a los obtenidos en la simulación, esto asegura un buen funcionamiento de los sistemas cuando se implementen en hardware.

5.3. GENERACIÓN DEL ARCHIVO BITSTREAM

Como se mencionó en la descripción de bloques del capítulo 3, el Token de System Generator es de vital importancia para la implementación del sistema modelado, porque en éste se configuran principalmente los parámetros de implementación según la tarjeta que se utilice en el proyecto.

La configuración del Token de manera general se muestra en la figura 5.12.

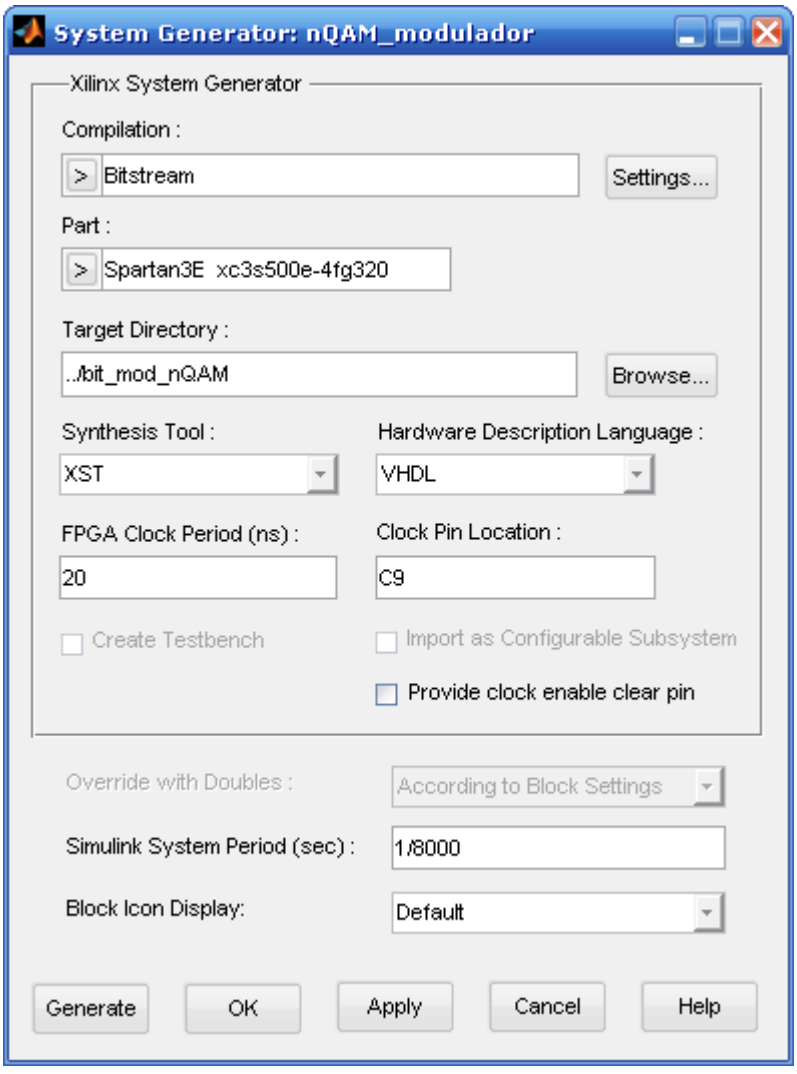


Figura 5.12. Configuración del Token para la generación del archivo bitstream

Los parámetros de configuración del Token: *Part* y *Clock Pin Location* se pueden obtener de la tarjeta directamente o también se los puede buscar en el documento de Xilinx "*Spartan-3E Starter Kit Board User Guide".*

Para la invocación del generador de código se pulsa el botón *Generate,* el cual inicia el proceso de compilación del archivo *bitstream*. El estado de la compilación se muestra en un cuadro de diálogo hasta su culminación, cuando se visualice el mensaje de la figura 5.13 "*Compilation finished successfully*".

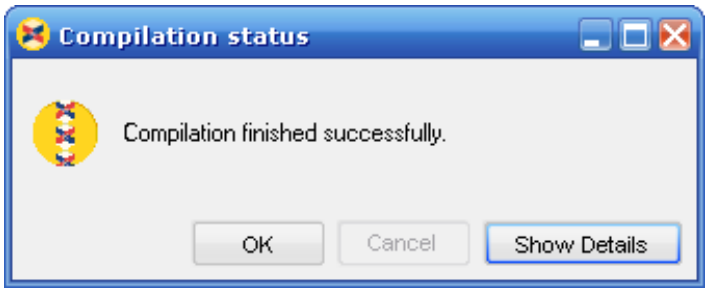


Figura 5.13. Compilación terminada con éxito

El archivo bitstream creado se encuentra en la ubicación especificada en *Target Directory,* con el nombre de 'nqam_modulador_cw.bit' en el caso del modulador y como 'nqam_demodulador_cw.bit' para el demodulador.

5.3.1. DESCARGA

Mediante el programa iMPACT, que es parte del paquete computacional Xilinx ISE 8.1i, se realiza la descarga del archivo *bitstream* creado a la tarjeta.

El proceso inicia con la configuración del dispositivo que utiliza la cadena de programación JTAG (figura 5.14) por parte del programa iMPACT. Esta configuración conectará la tarjeta Spartan-3E con el iMPACT, y además identificará la cadena de programación automáticamente como se muestra en la figura 5.15.

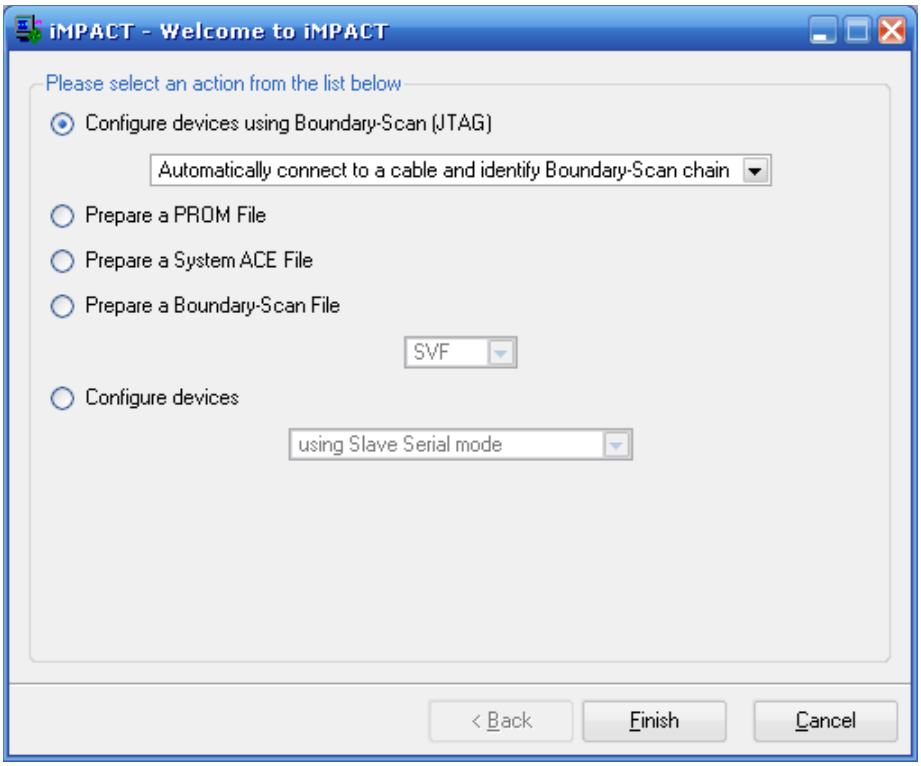


Figura 5.14. Identificación de la tarjeta en iMPACT

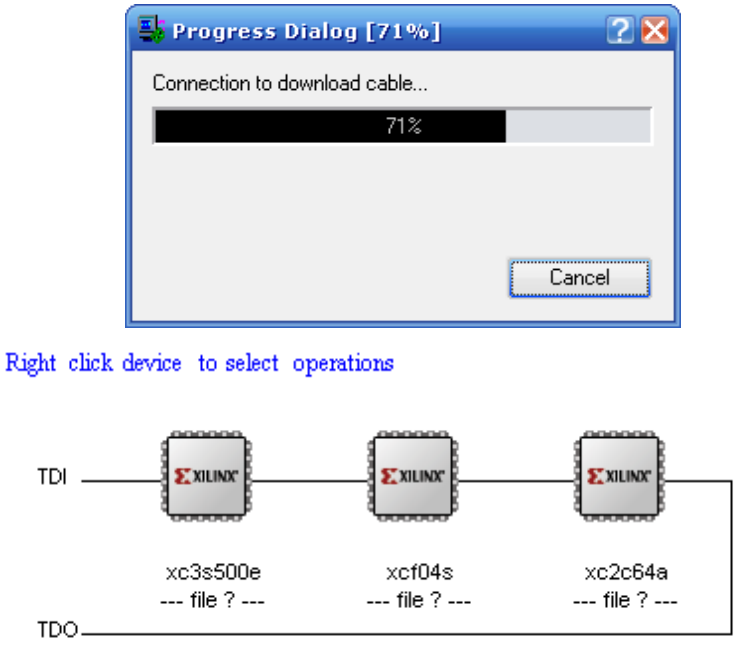


Figura 5.15. Cadena de programación del FPGA

Esta cadena consta de tres partes que son: FPGA, CPLD y memoria ROM. En el FPGA se asigna el archivo *bitstream* de configuración mediante el explorador como se indica en la figura 5.16.

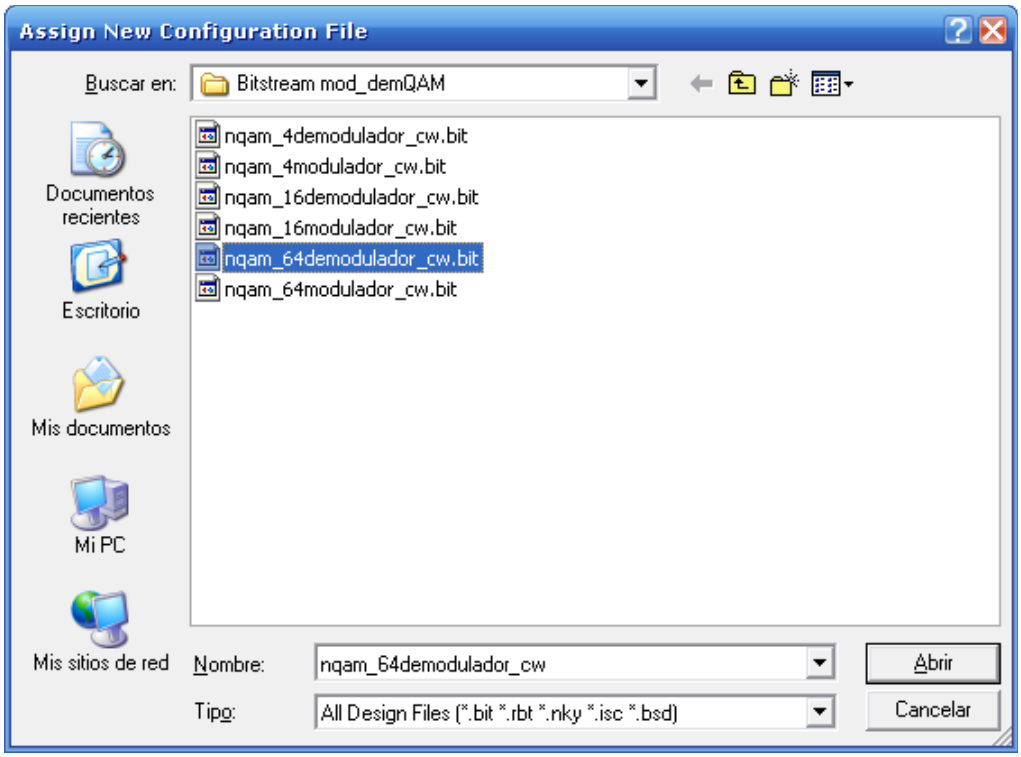


Figura 5.16. Asignación del archivo bitstream

Debido a que no se van a configurar las otras dos partes de la cadena ya que están fuera del alcance de este proyecto, se selecciona *Bypass* en cada una de ellas. Por último, se selecciona *Program* sobre el FPGA para completar la descarga del archivo a la tarjeta.

El mensaje que asegura la descarga del archivo es "*Program Succeeded*" como se indica en la figura 5.17.

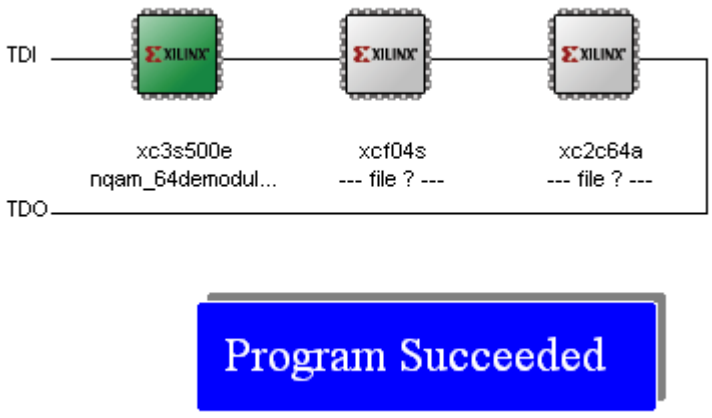


Figura 5.17. Descarga exitosa del archivo bitstream a la Spartan-3E

5.4. PRUEBAS Y ANÁLISIS DE RESULTADOS

Las pruebas de funcionamiento se efectuarán en el modulador y en el demodulador independientemente, en consecuencia la tarjeta de entrenamiento sólo contendrá un sistema a la vez, sin que exista algún tipo de interacción entre los sistemas. Además para realizar las pruebas, se utilizará un circuito externo que permita la conversión de las señales digitales a analógicas, para poder visualizarlas en el osciloscopio y evaluar el funcionamiento de cada sistema.

En la figura 5.18 se muestra el circuito para el DAC y las entradas digitales que se conectarán a la tarjeta.

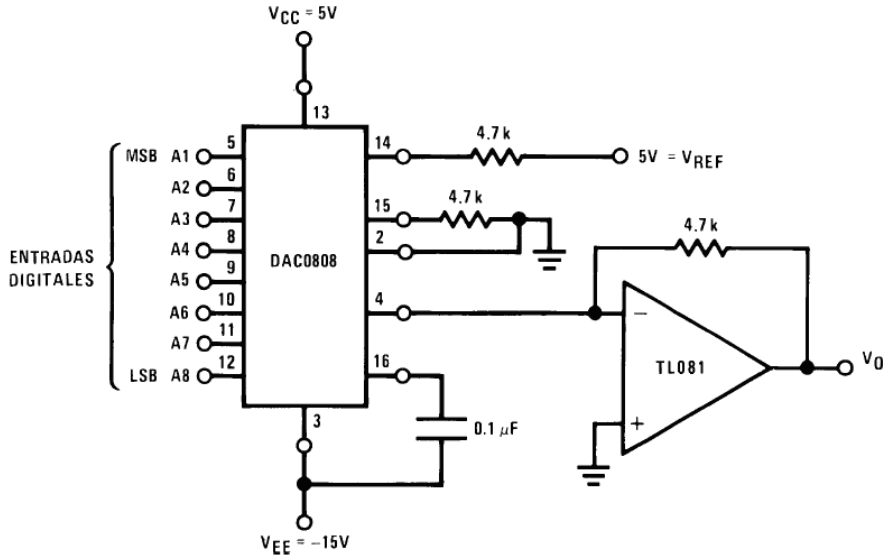


Figura 5.18. Conversor Digital Analógico

Es importante resaltar que los niveles de voltaje obtenidos en la salida del conversor D/A son únicamente ilustrativos y no corresponden con los valores que se indicaron en la simulación.

5.4.1. PRUEBAS DE FUNCIONAMIENTO DEL MODULADOR

5.4.1.1. Fuente de Datos

Para iniciar con las pruebas es necesario asegurarse que el contenido de la fuente sea el correcto debido a que se trata de la secuencia de bits de entrada

para el modulador. La fuente que será utilizada por todos los moduladores QAM, contiene el mensaje ‘nQAM’ en código ASCII de 8 bits por símbolo. Los bits correspondientes al mensaje de prueba se muestran en la figura 5.19.

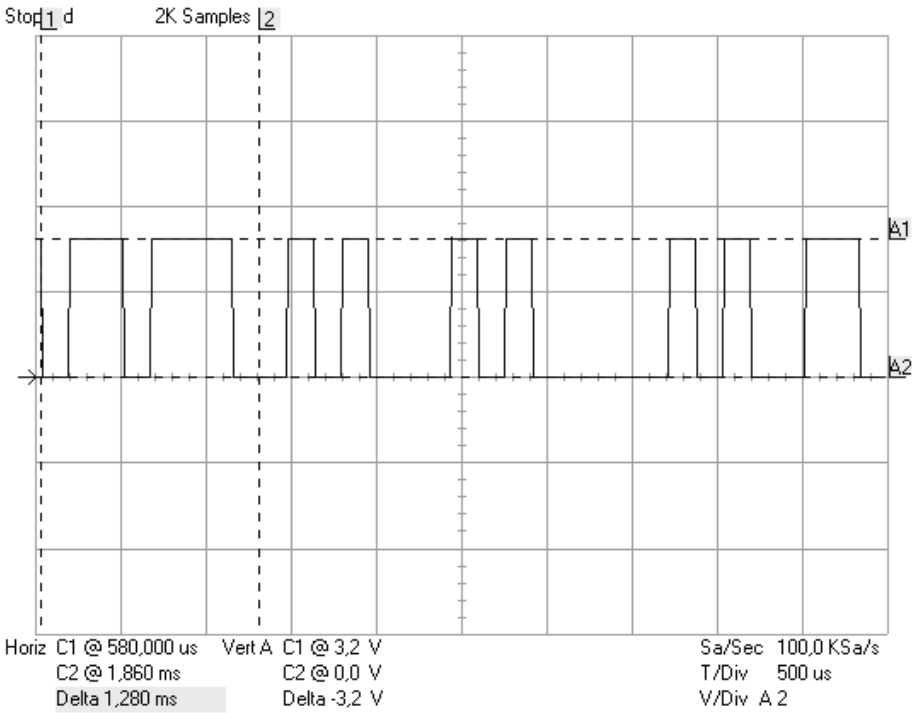


Figura 5.19. Mensaje de Prueba ‘nQAM’ en código ASCII

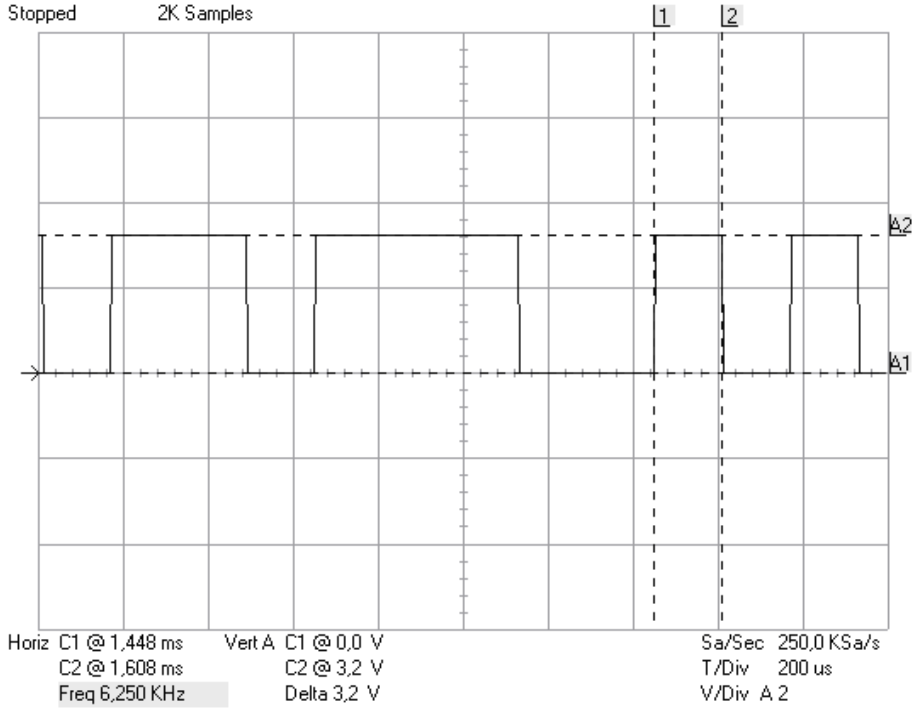


Figura 5.20. Velocidad de los Bits de Entrada

Los marcadores 1 y 2 en la figura 5.19, contienen los primeros 8 bits del mensaje de prueba, lo que corresponde a la letra ‘n’. En cuanto a la figura 5.20, los mismos marcadores limitan a un bit de entrada, para obtener la frecuencia de entrada de datos al modulador que es 6,25 KHz.

5.4.1.2. Divisor de Datos

En este subsistema se separa el flujo serial de bits en los canales paralelos I-Q, este resultado se muestra en las figuras 5.21, 5.22 y 5.23 para cada modulación QAM.

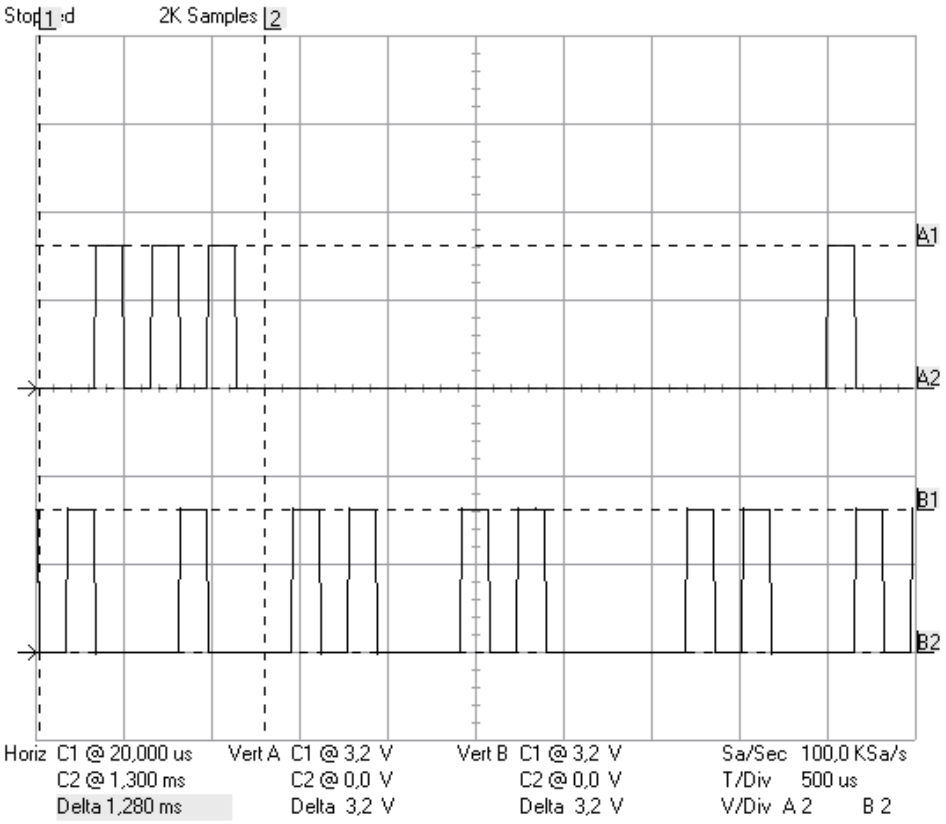


Figura 5.21. Divisor de Bits Canales I-Q, 4-QAM

Los marcadores A1 y A2 contienen los bits del canal I, por otro lado B1 y B2 contienen los del canal Q. Además los marcadores 1 y 2 limitan 8 bits para que se pueda verificar el funcionamiento del divisor mediante el mensaje de prueba.

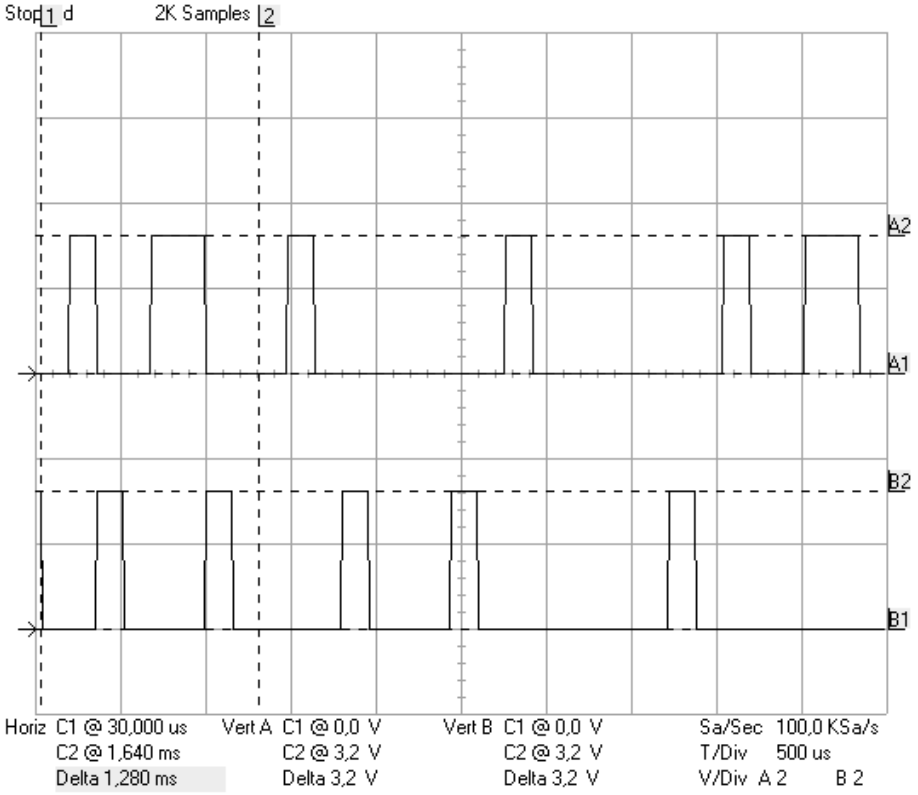


Figura 5.22. Divisor de Bits Canales I-Q, 16-QAM

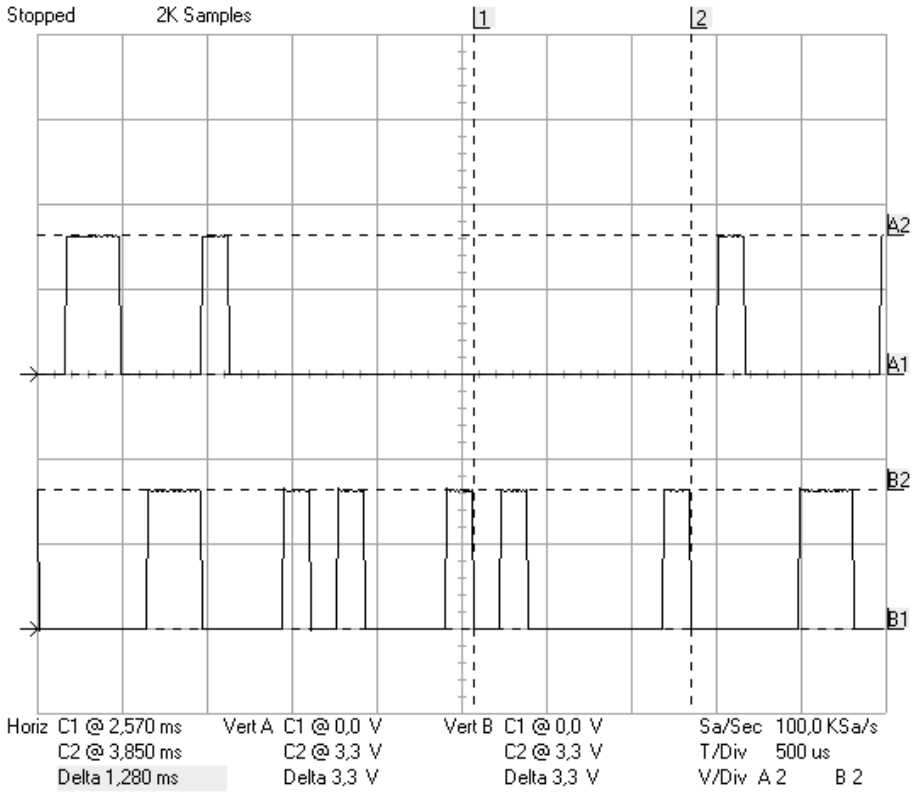


Figura 5.23. Divisor de Bits Canales I-Q, 64-QAM

5.4.1.3. Convertidor de 2 a L Niveles

La señal multinivel que se obtiene por medio del convertidor de niveles se muestra para cada caso de modulación QAM en las figuras 5.24, 5.25 y 5.26.

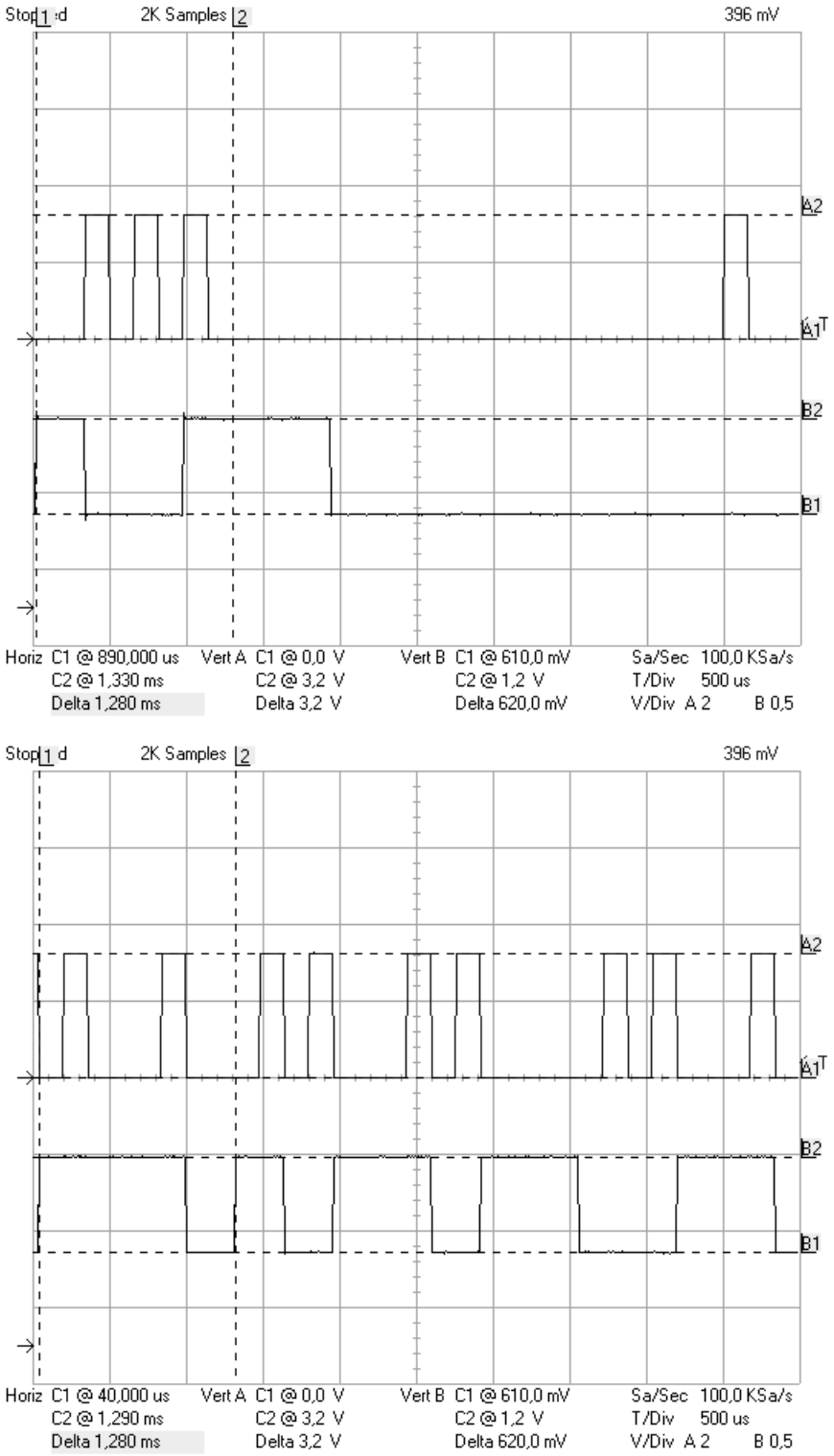


Figura 5.24. Señal Multinivel I-Q, 4-QAM

En la figura 5.24, se muestran las señales del canal I en la parte superior y del canal Q en la inferior. Además, los bits que ingresan al convertidor se encuentran con el marcador A y la señal multinivel con el marcador B para 4-QAM.

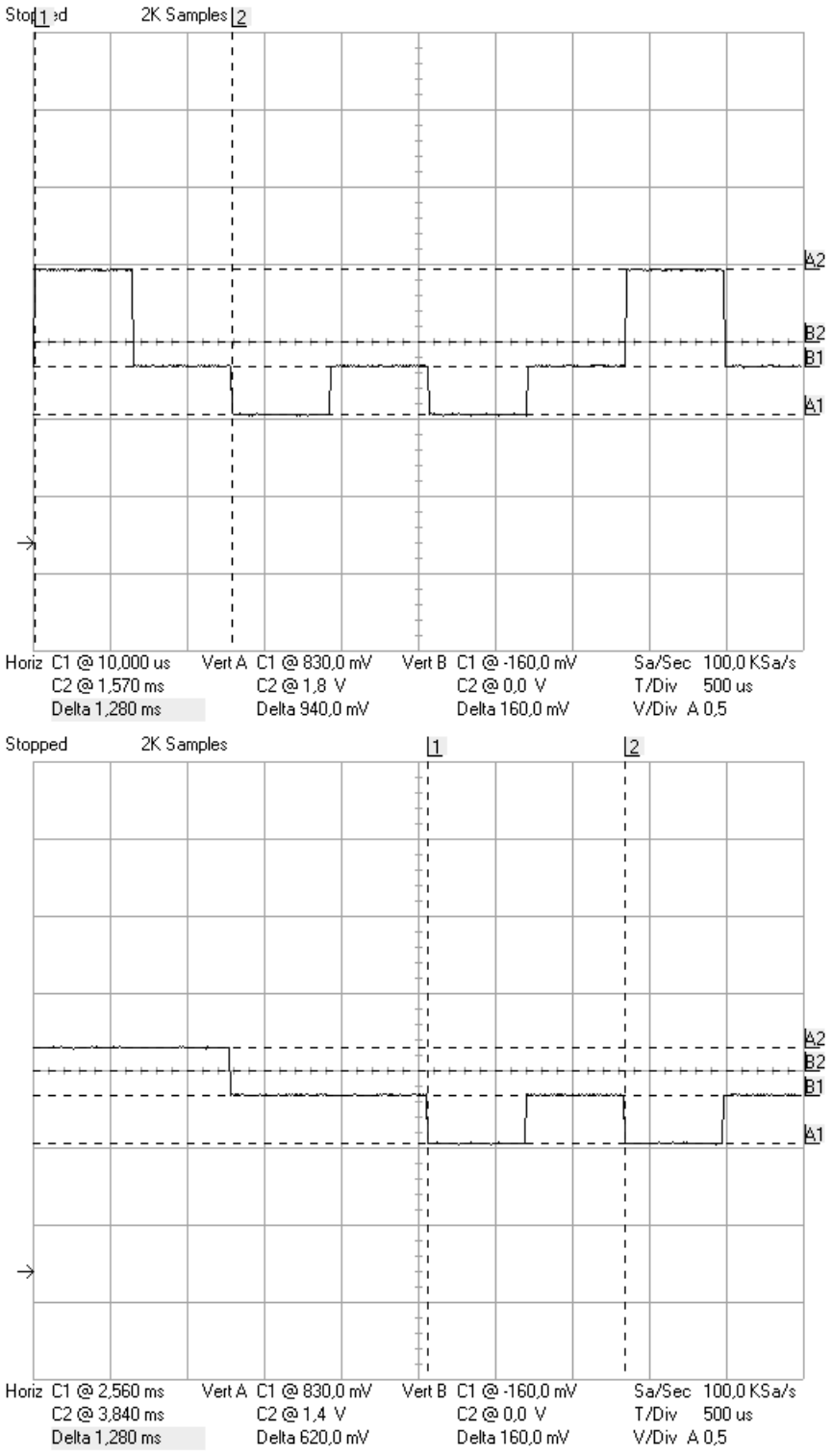


Figura 5.25. Señal Multinivel I-Q,16-QAM

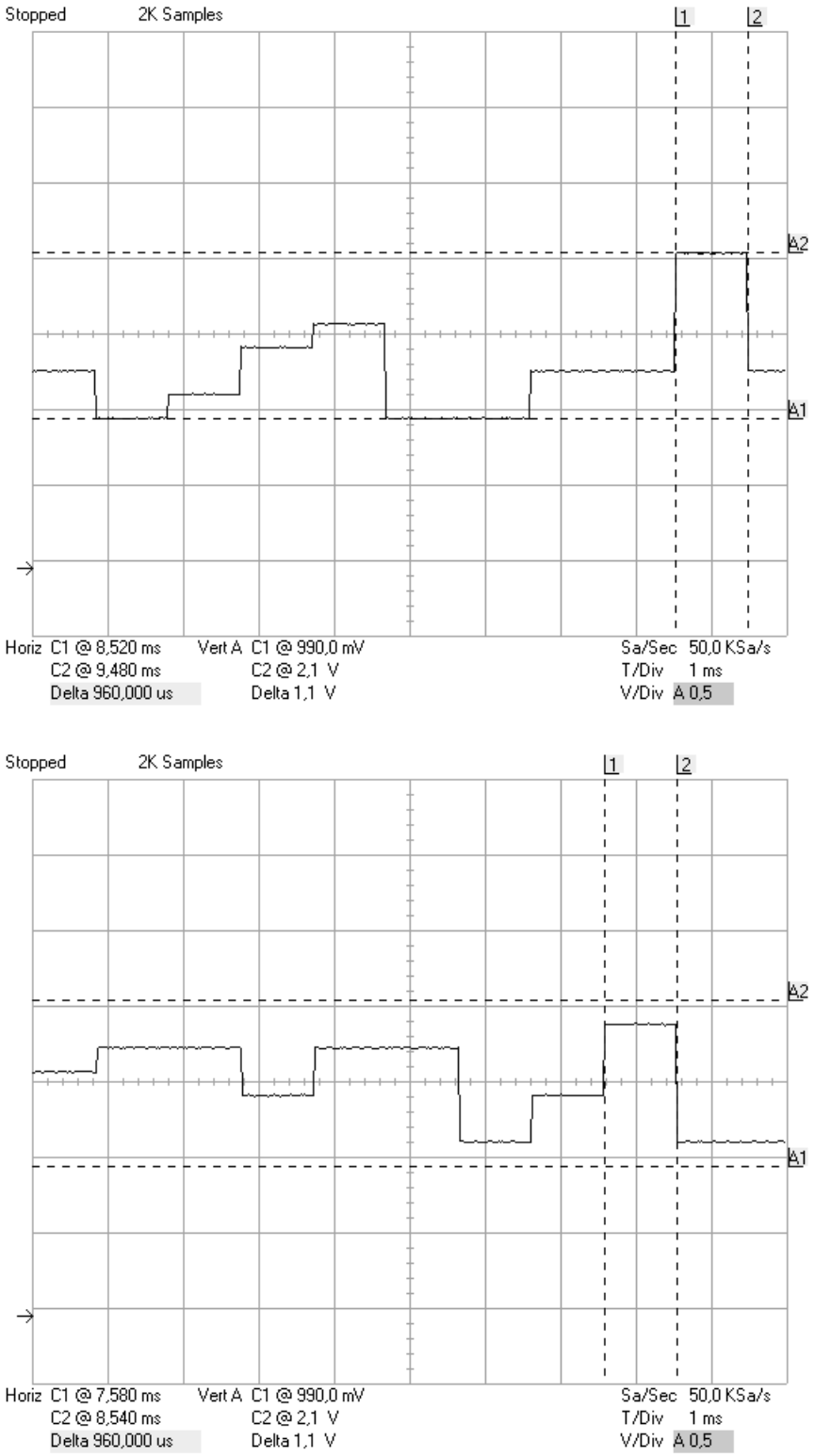


Figura 5.26. Señal Multinivel I-Q, 64-QAM

Las señales multinivel resultantes para 4, 16 y 64-QAM son idénticas a las obtenidas en la simulación del capítulo 4.

5.4.1.4. Oscilador Local

Como sucede con el mensaje de prueba, la portadora generada por el oscilador local es idéntica para todos los casos de modulación QAM, en consecuencia la señal portadora coseno que se muestra en la figura 5.27 se utiliza en todos los tipos de modulación.

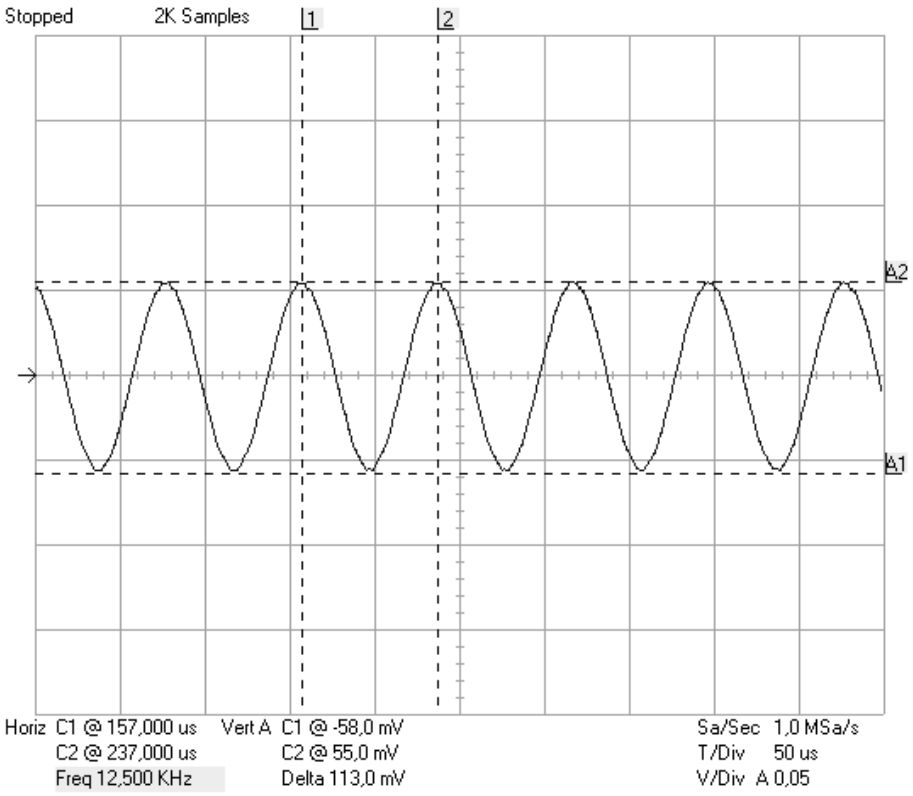


Figura 5.27. Señal Portadora Coseno

El resultado de la frecuencia portadora obtenida en el osciloscopio es 12.5 KHz, coincidiendo con el valor propuesto en el diseño.

5.4.1.5. Multiplicador

El producto entre la señal multinivel y la portadora en fase o en cuadratura, según corresponda al canal I o Q, se muestra en las figuras 5.28, 5.29 y 5.30.

En el resultado de la multiplicación, el cambio entre símbolos se visualiza claramente con los marcadores 1-2, ya sea por el cambio de amplitud, fase o ambos.

En la figura 5.28, que muestra la señal obtenida a la salida del multiplicador para 4-QAM, se observa que el cambio entre símbolos ocurre cada 3,125 KHz, tanto en el canal I (arriba) como en el canal Q (abajo).

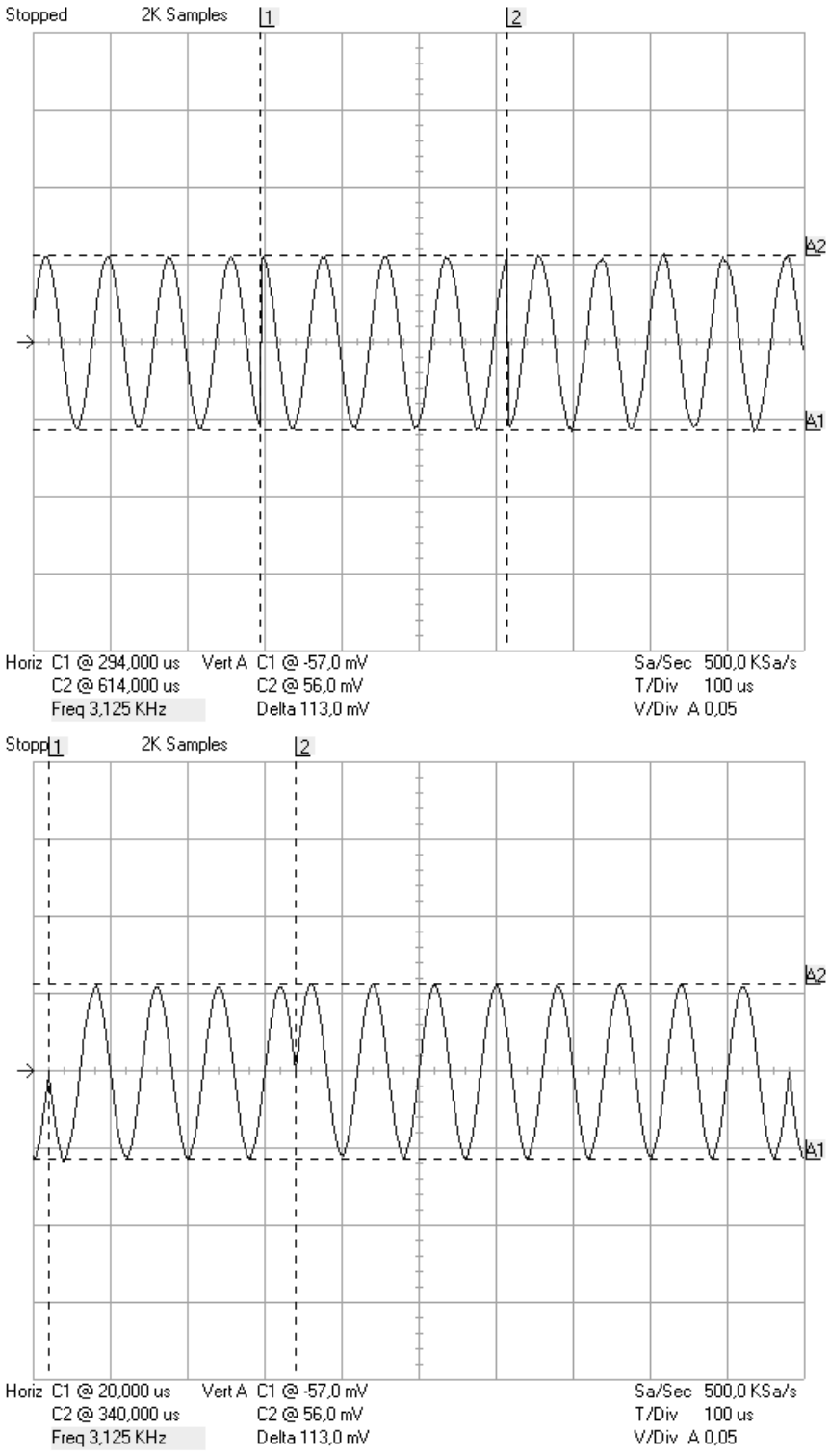


Figura 5.28. Salida del Multiplicador canales I-Q, 4-QAM

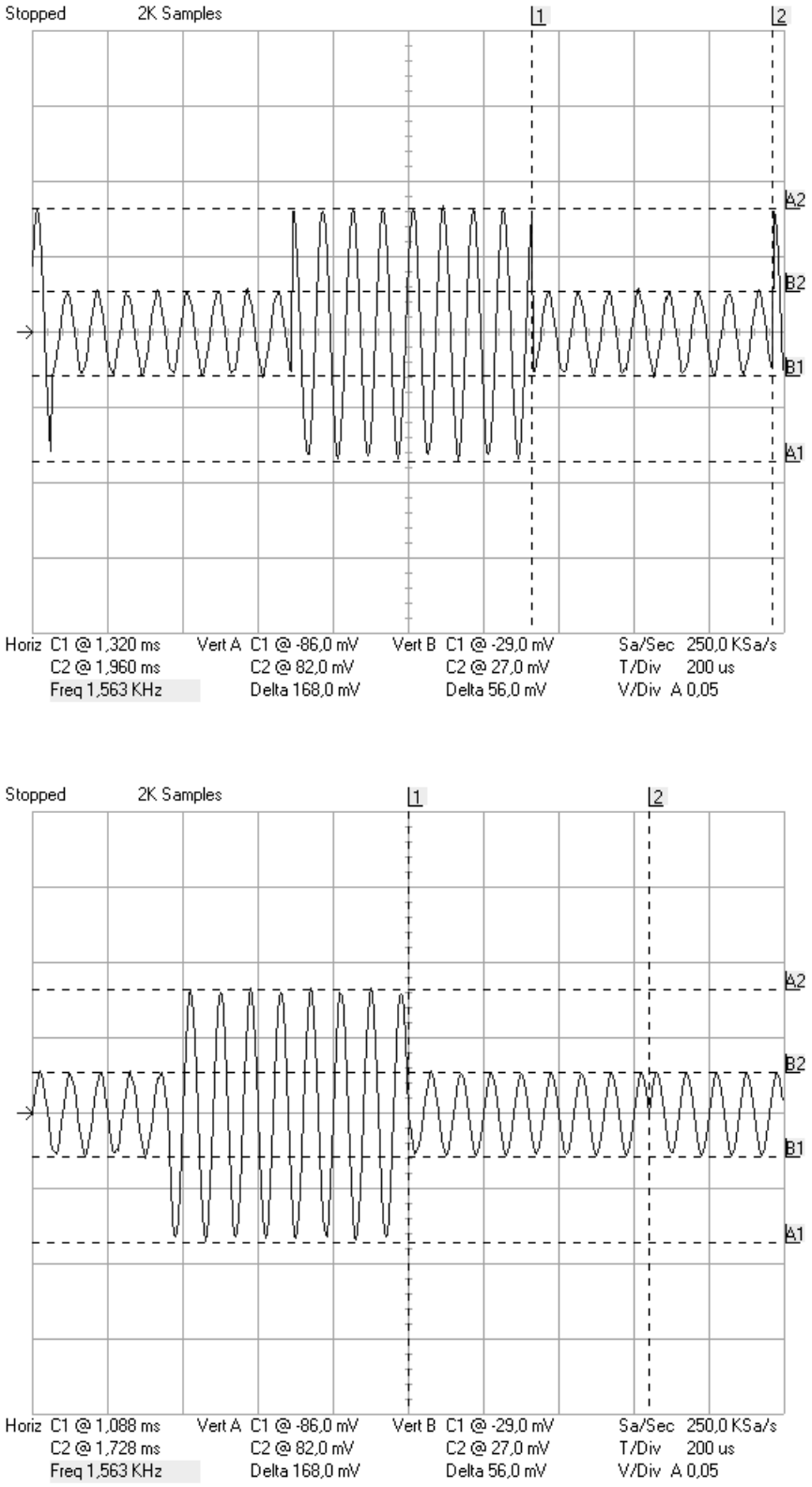


Figura 5.29. Salida del Multiplicador canales I-Q, 16-QAM

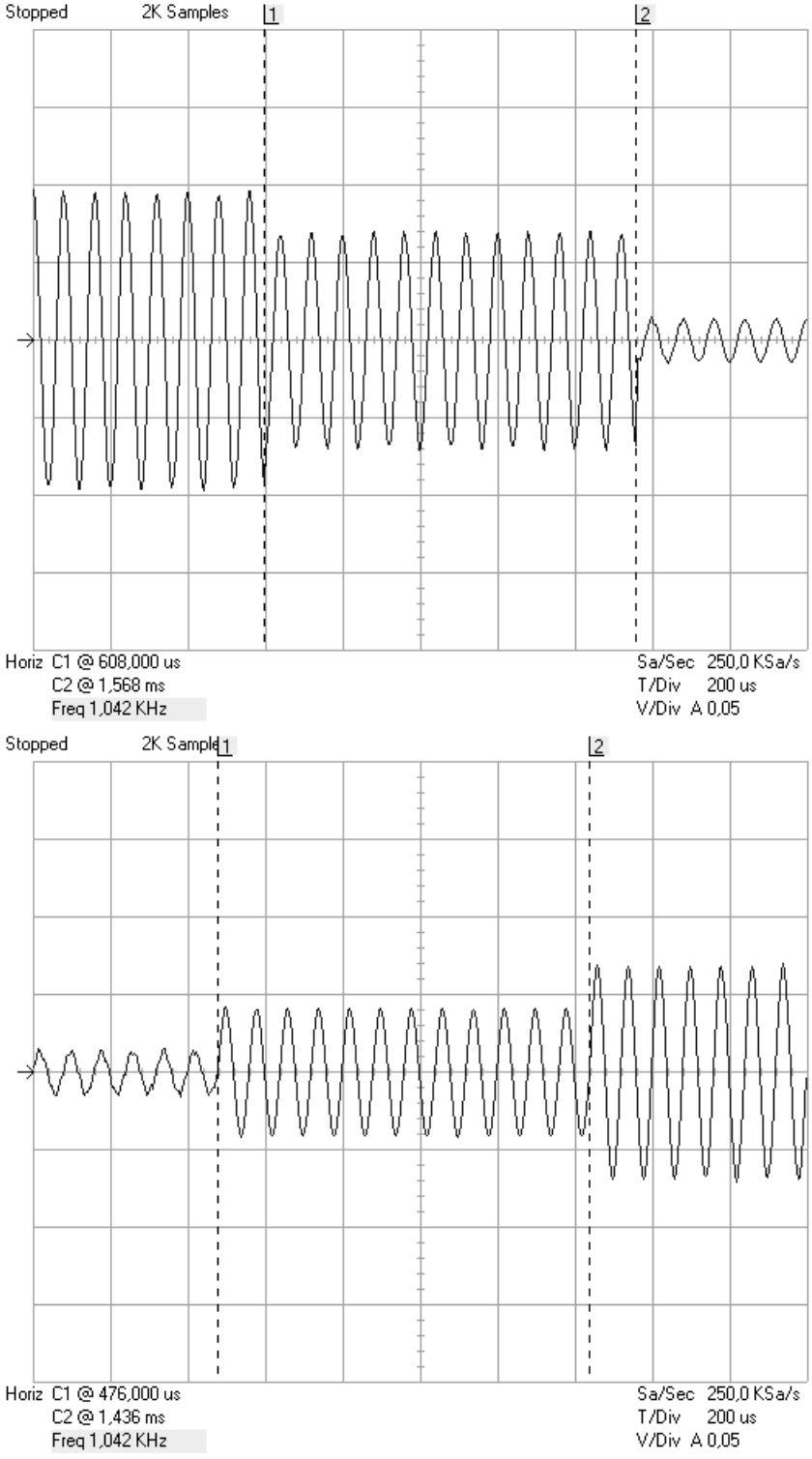


Figura 5.30. Salida del Multiplicador canales I-Q, 64-QAM

Se observa para todos los casos, que existe una relación entre la amplitud obtenida en la implementación con la propuesta en el diseño. Por ejemplo para el caso de 16-QAM la proporción es 3V/1V = 168mV/56mV = 3.

5.4.1.6. Sumador Lineal

La última prueba de funcionamiento básico en este sistema es la señal modulada QAM. El resultado para cada modulación QAM se encuentra en las figuras 5.31, 5.32 y 5.33.

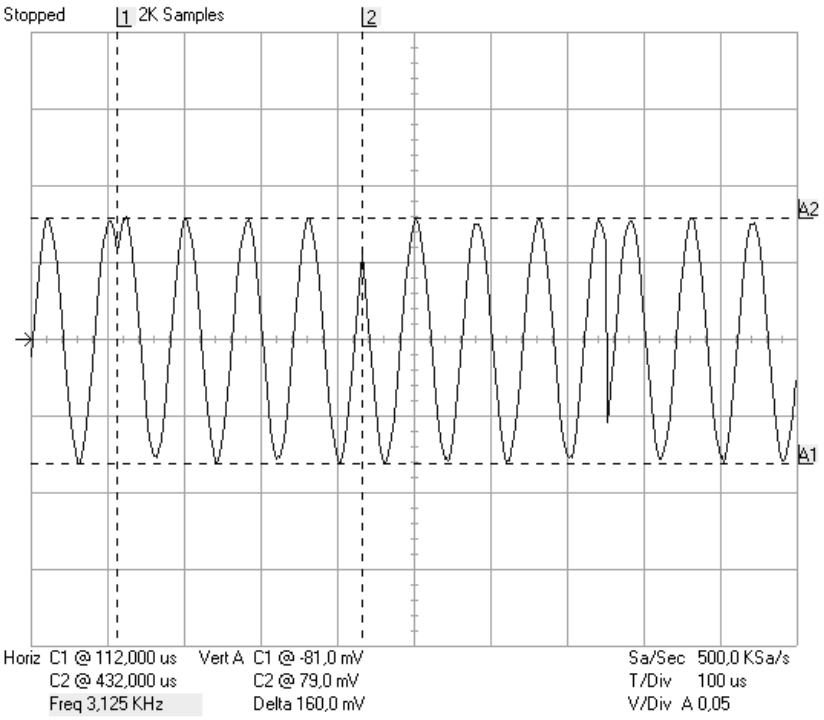


Figura 5.31. Señal Modulada en 4-QAM

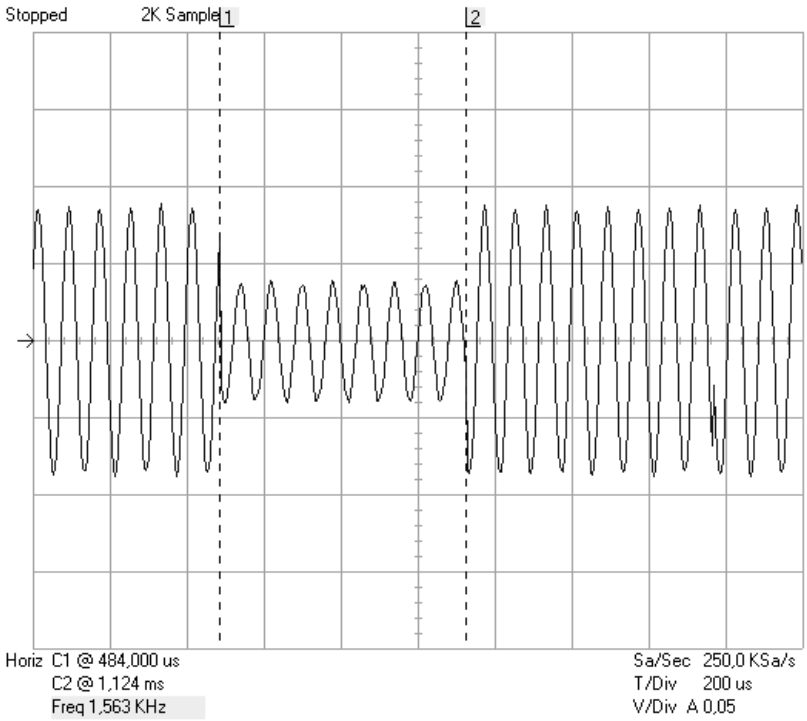


Figura 5.32. Señal Modulada en 16-QAM

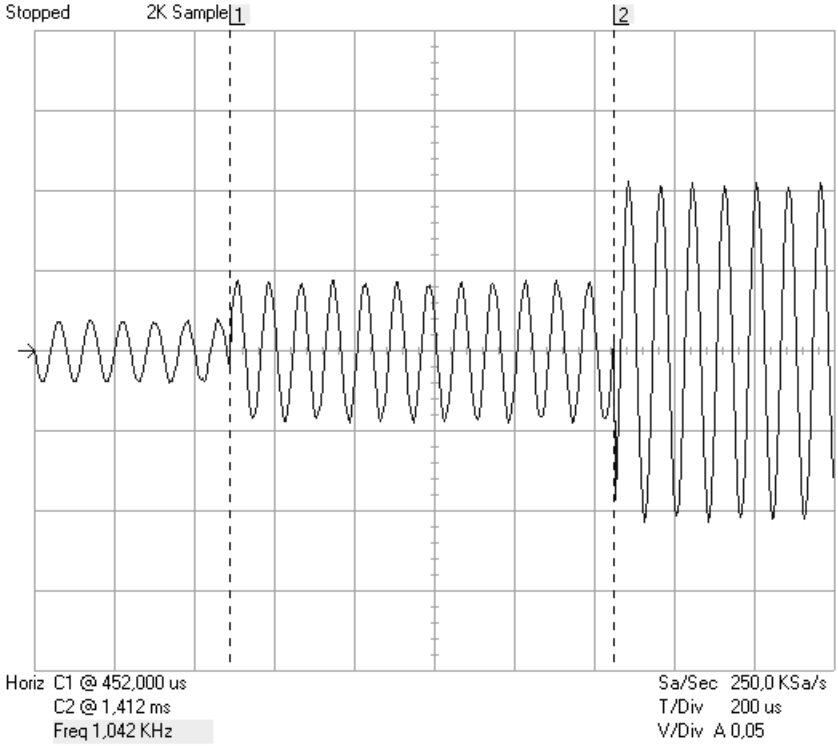


Figura 5.33. Señal Modulada en 64-QAM

Los resultados que se obtienen en el osciloscopio se ajustan exactamente a los conseguidos en la simulación. La velocidad de modulación para cada caso de modulación QAM se resume en la tabla 5.2.

Modulación	V_{MOD} [Kbaudios]	
	Simulación	Práctico
4-QAM	3,125	3,125
16-QAM	1,563	1,563
64-QAM	1,042	1,042

Tabla 5.2. Velocidad de Modulación para n-QAM

5.4.2. PRUEBAS DE FUNCIONAMIENTO DEL DEMODULADOR

5.4.2.1. Entrada del Demodulador

La señal QAM, resultado de la modulación, se almacenó en una memoria ROM para que se realicen las pruebas del demodulador. El resultado para cada modulación QAM se muestra en las figuras 5.34, 5.35 y 5.36.

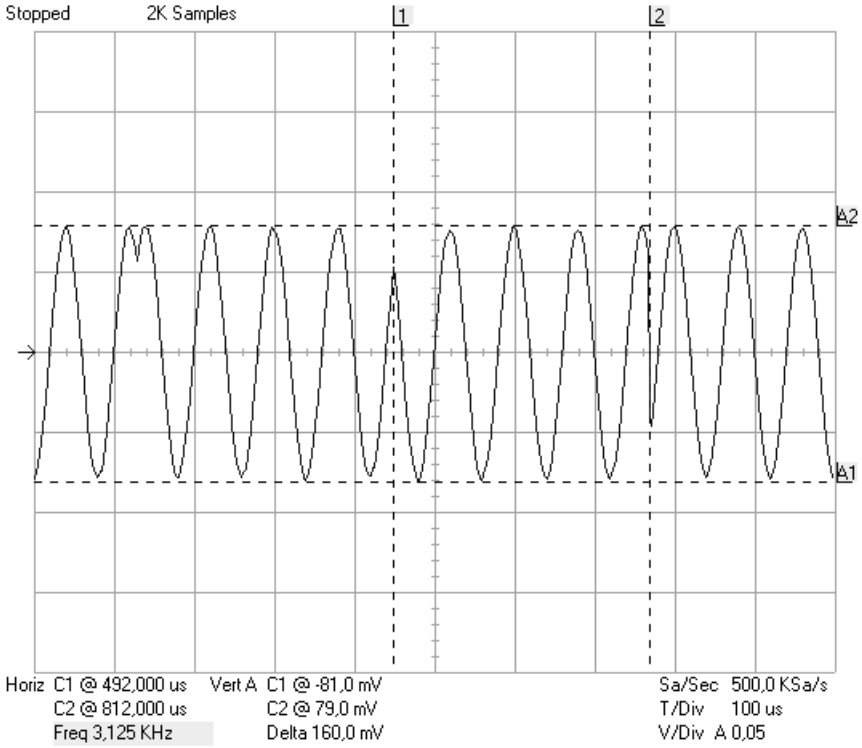


Figura 5.34. Entrada del Demodulador 4-QAM

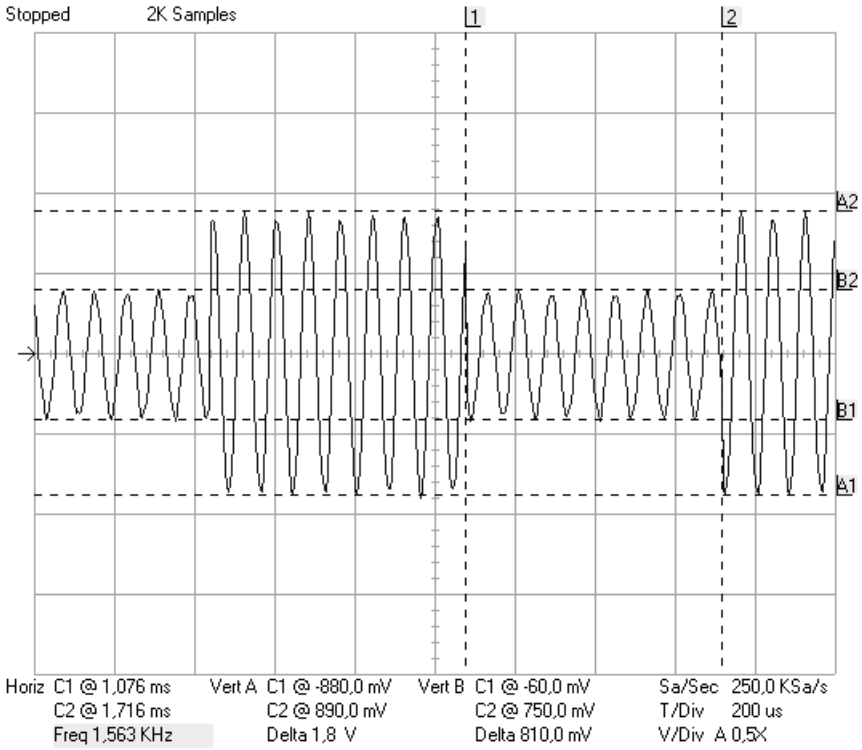


Figura 5.35. Entrada del Demodulador 16-QAM

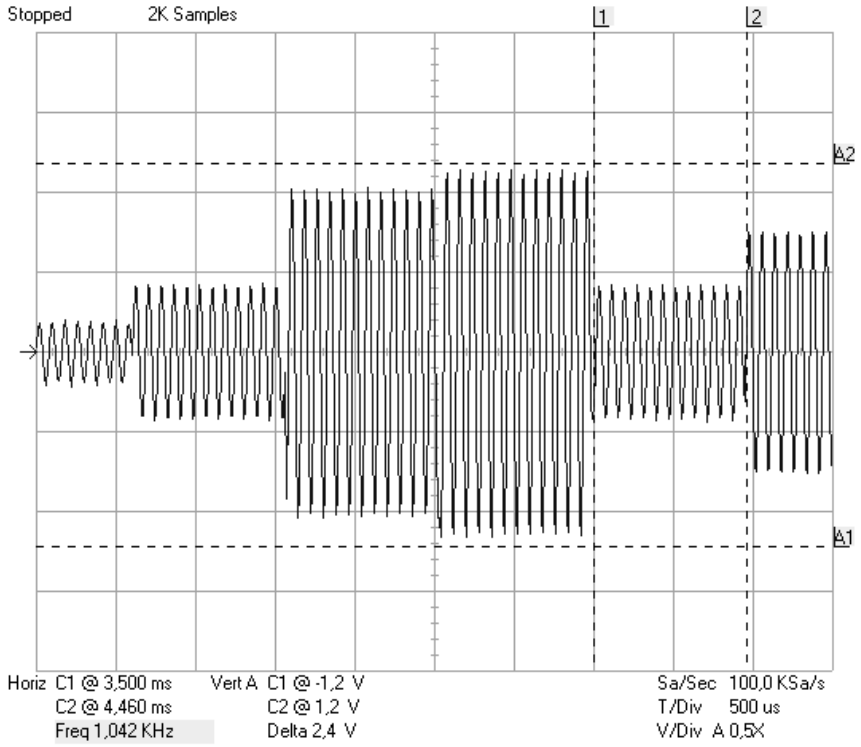


Figura 5.36. Entrada del Demodulador 64-QAM

5.4.2.2. Oscilador Local

Como en el caso del modulador, la portadora generada por el oscilador local es idéntica para todos los casos de modulación QAM, en consecuencia la señal portadora seno que se muestra en la figura 5.37 se utiliza en todos los casos.

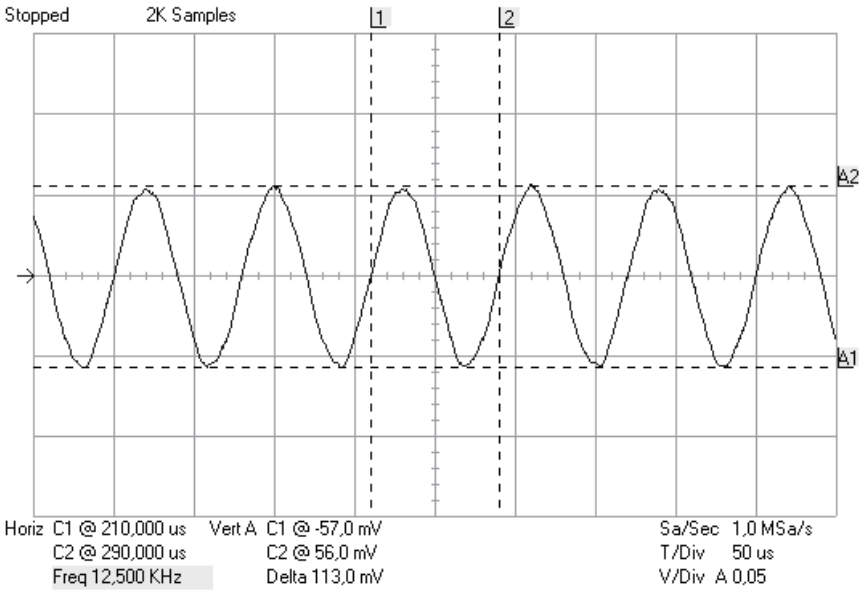


Figura 5.37. Señal Portadora Seno

El resultado de la frecuencia portadora en fase es exactamente igual a la diseñada de 12.5 KHz.

5.4.2.3. Multiplicador

El producto entre la señal modulada QAM y la portadora en fase o en cuadratura según corresponda al canal I o Q, se muestra en las figuras 5.38 a 5.43.

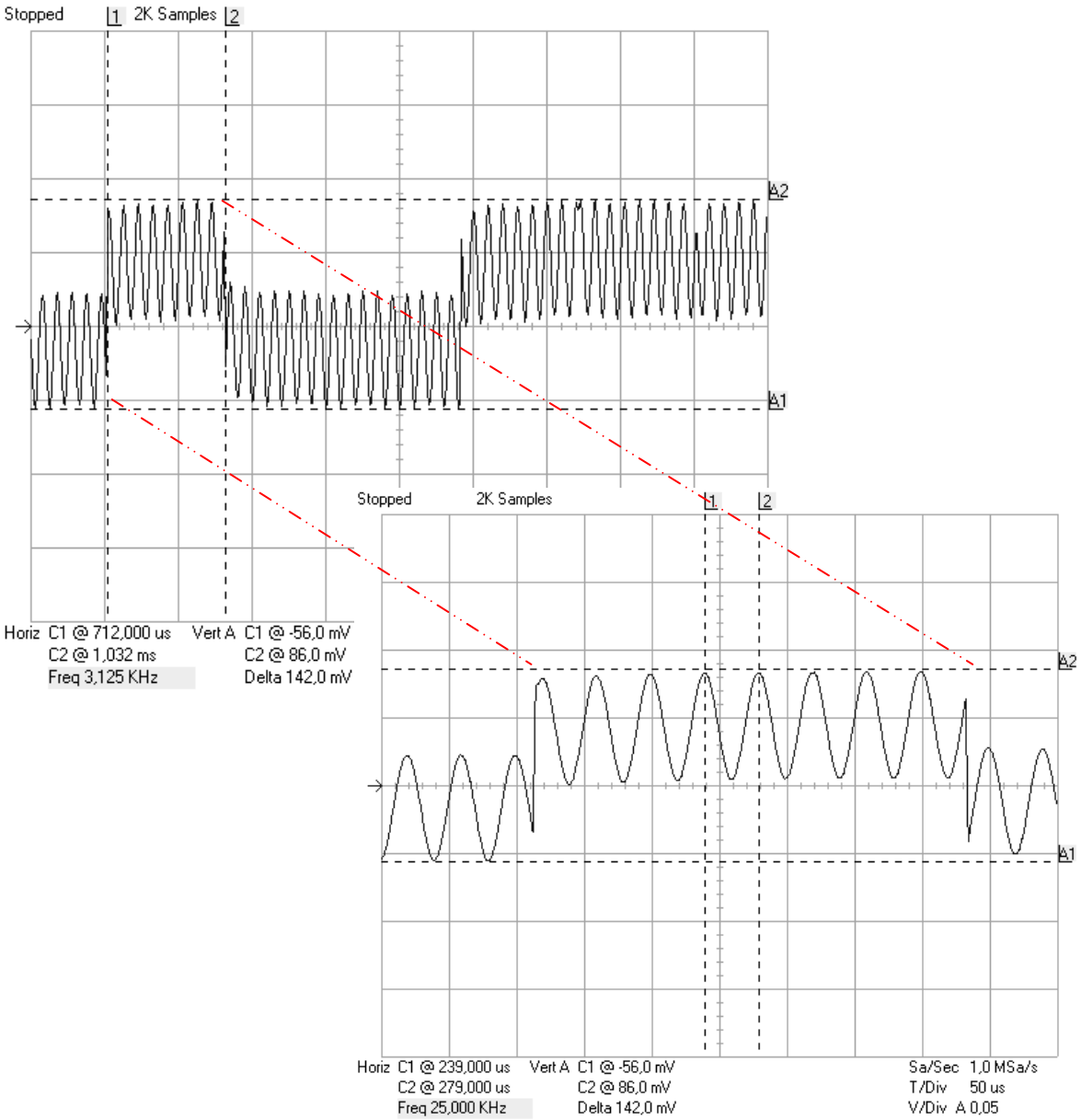


Figura 5.38. Salida del Multiplicador canal I, demodulador 4-QAM

En la figura 5.38 se muestra el resultado de la multiplicación de la señal 4-QAM con la portadora en fase. Además se realiza un acercamiento para observar de

mejor manera la componente continua más una componente de alta frecuencia de 25 KHz, la cual es el doble de la frecuencia de la portadora.

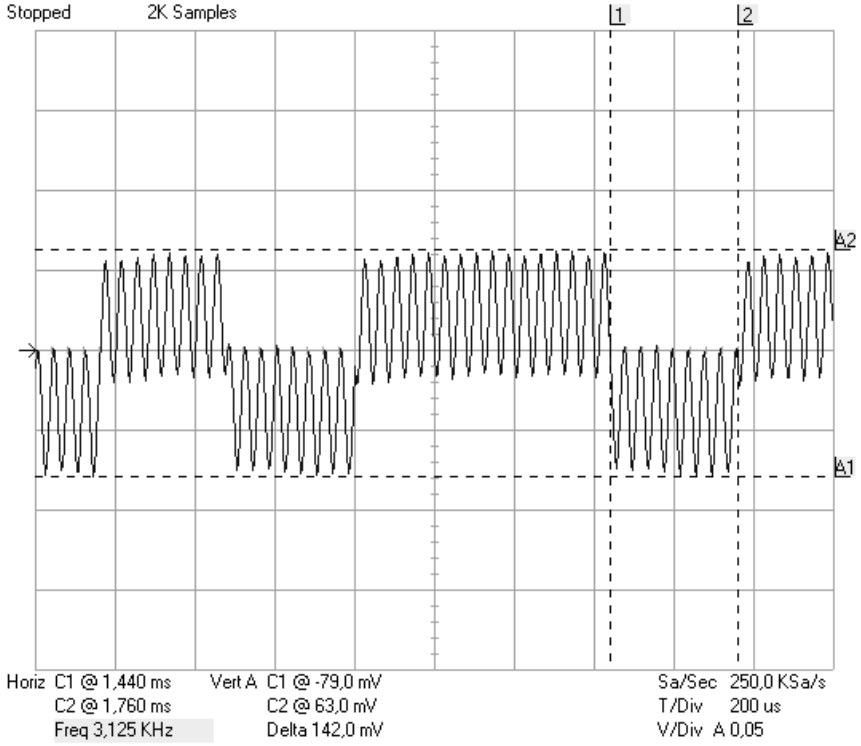


Figura 5.39. Salida del Multiplicador canal Q, demodulador 4-QAM

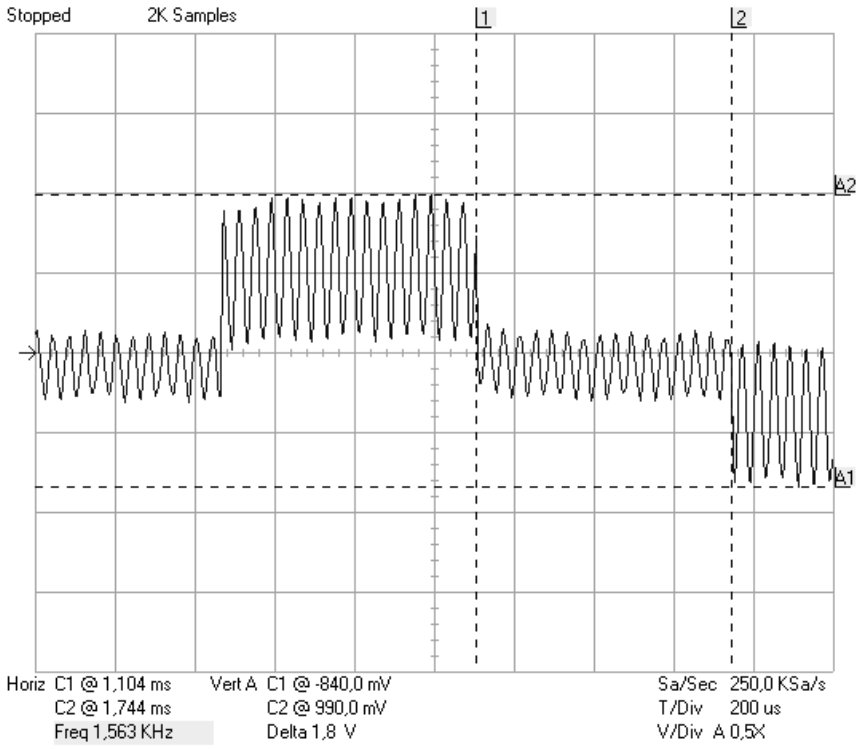


Figura 5.40. Salida del Multiplicador canal I, demodulador 16-QAM

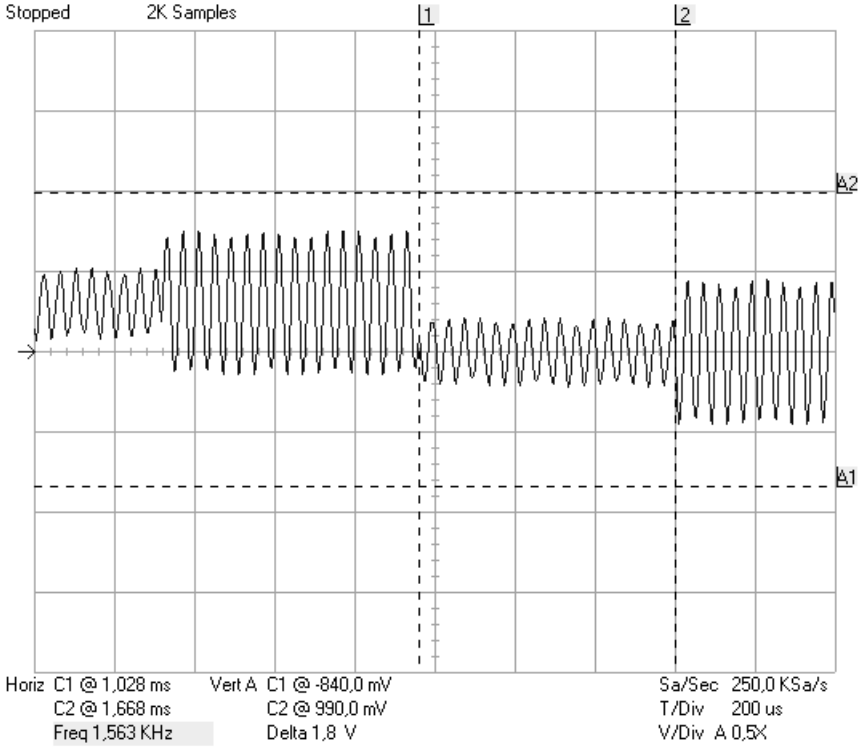


Figura 5.41. Salida del Multiplicador canal Q, demodulador 16-QAM

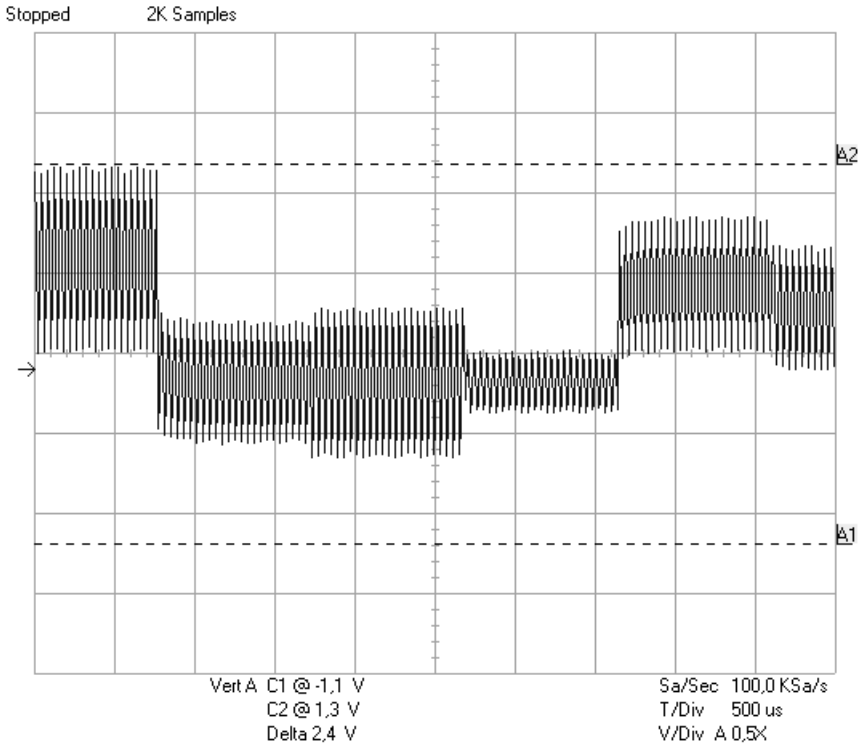


Figura 5.42. Salida del Multiplicador canales I, demodulador 64-QAM

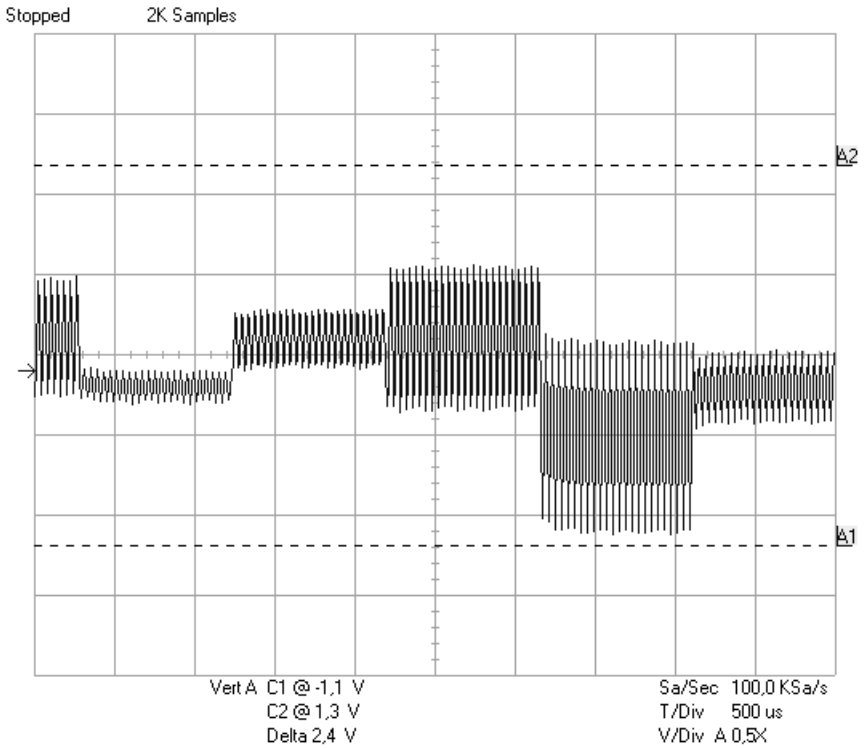


Figura 5.43. Salida del Multiplicador canales I-Q, demodulador 64-QAM

Para 16 y 64-QAM se han utilizado los marcadores A1 y A2 del eje vertical, para indicar el valor mínimo y el máximo que se alcanzaría después de la multiplicación.

5.4.2.4. Filtro LPF

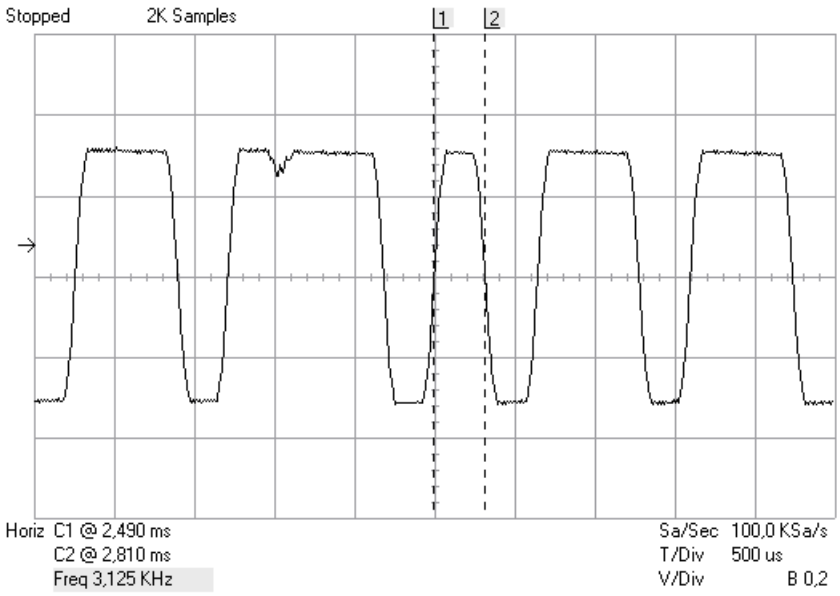


Figura 5.44. Salida del Filtro DAFIR canal Q, 4-QAM

Para eliminar la componente de alta frecuencia se utilizó un filtro pasa bajos DAFIR. El resultado para cada modulación QAM en el canal Q se muestra en las figuras 5.44, 5.45 y 5.46.

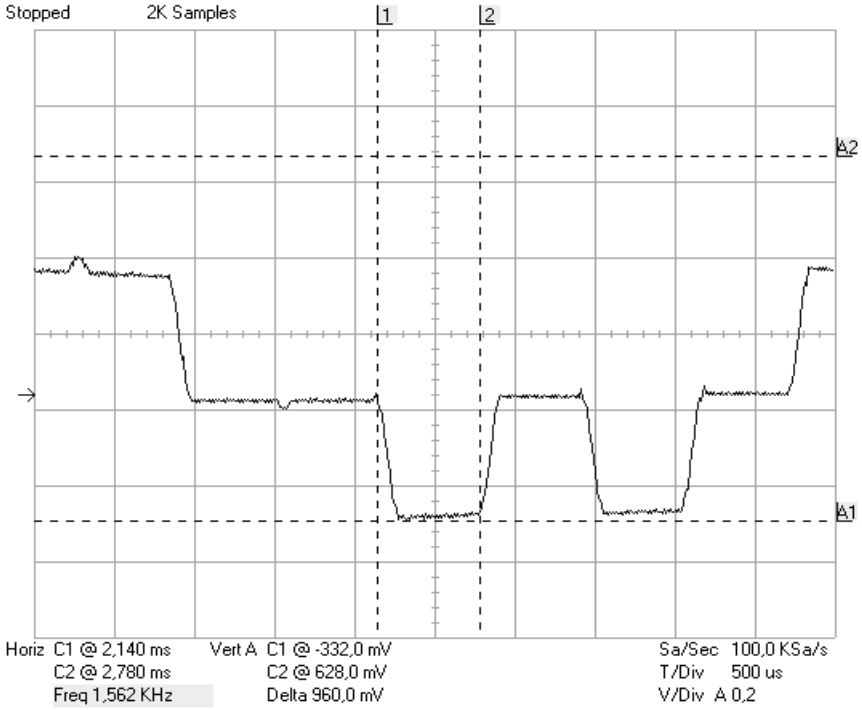


Figura 5.45. Salida del Filtro DAFIR canal Q, 16-QAM

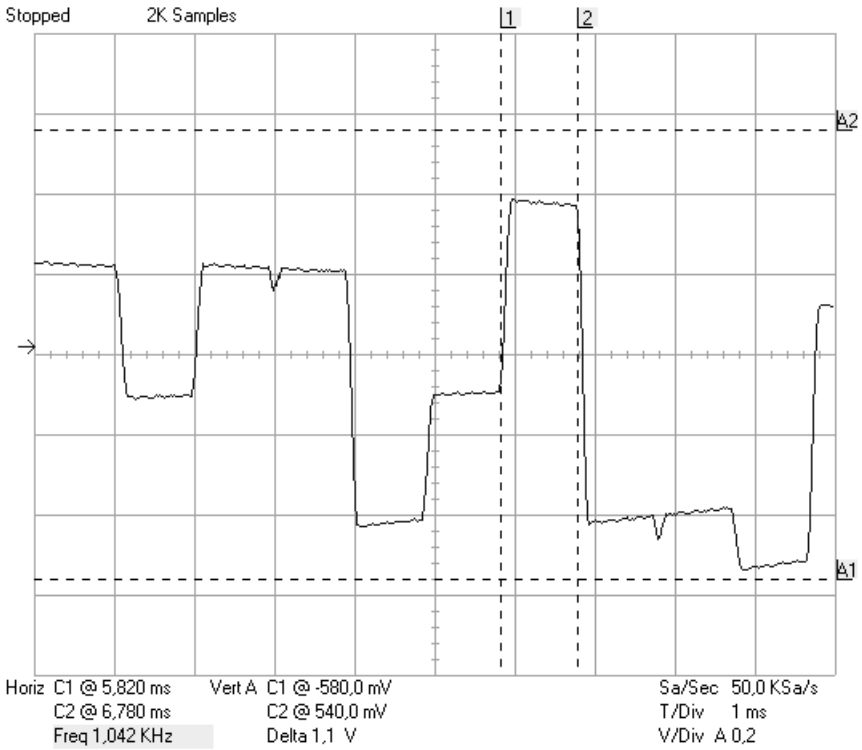


Figura 5.46. Salida del Filtro DAFIR canal Q, 64-QAM

La salida del filtro LPF en los canales I-Q se muestra en las figuras 5.47, 5.48 y 5.49, para 4,16 y 64-QAM. Y únicamente para 4-QAM, en la figura 5.47, se muestra la señal multinivel con el marcador A1-A2 y los bits de cada canal con B1-B2.

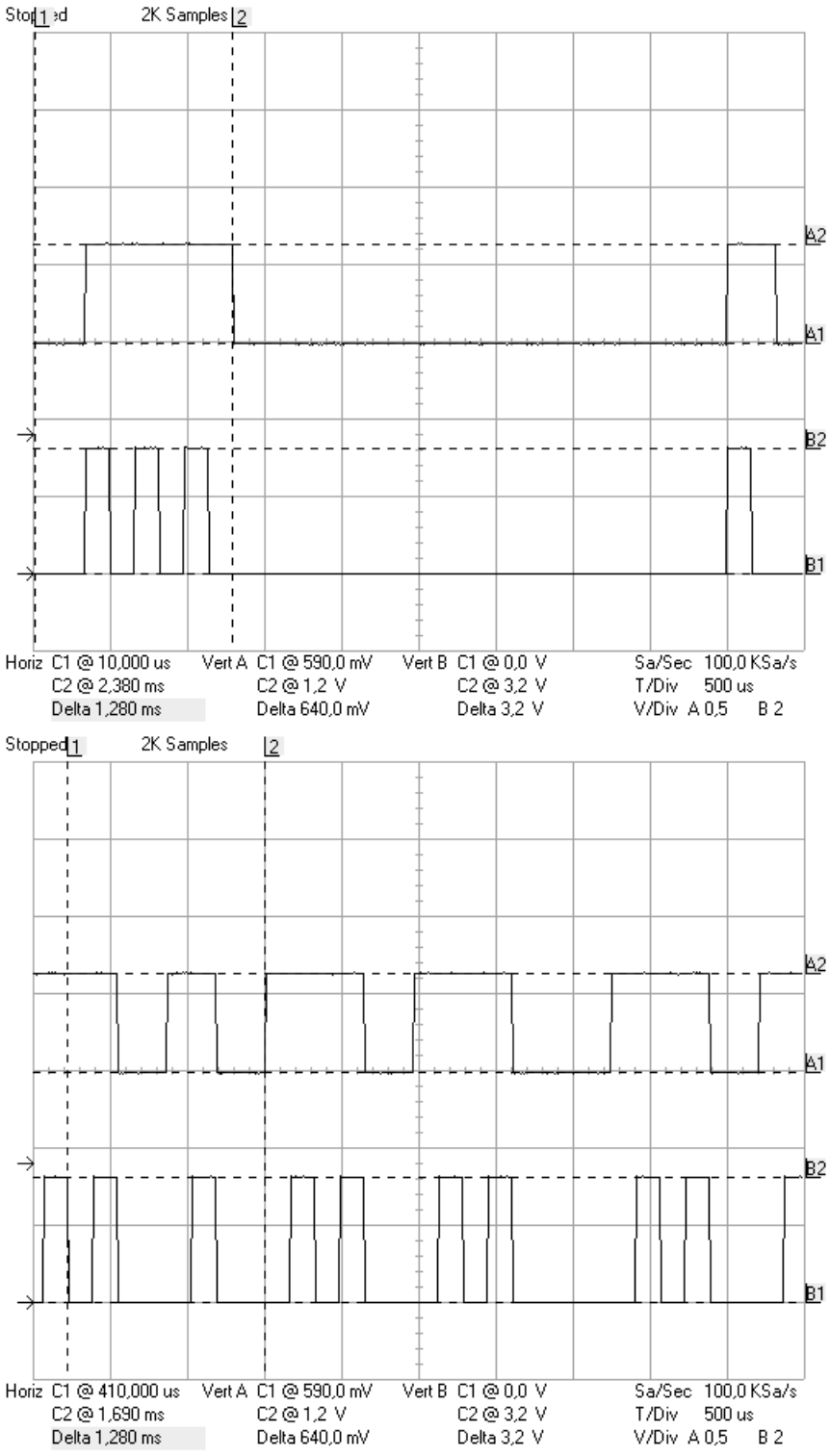


Figura 5.47. Señal Multinivel I-Q, demodulador 4-QAM

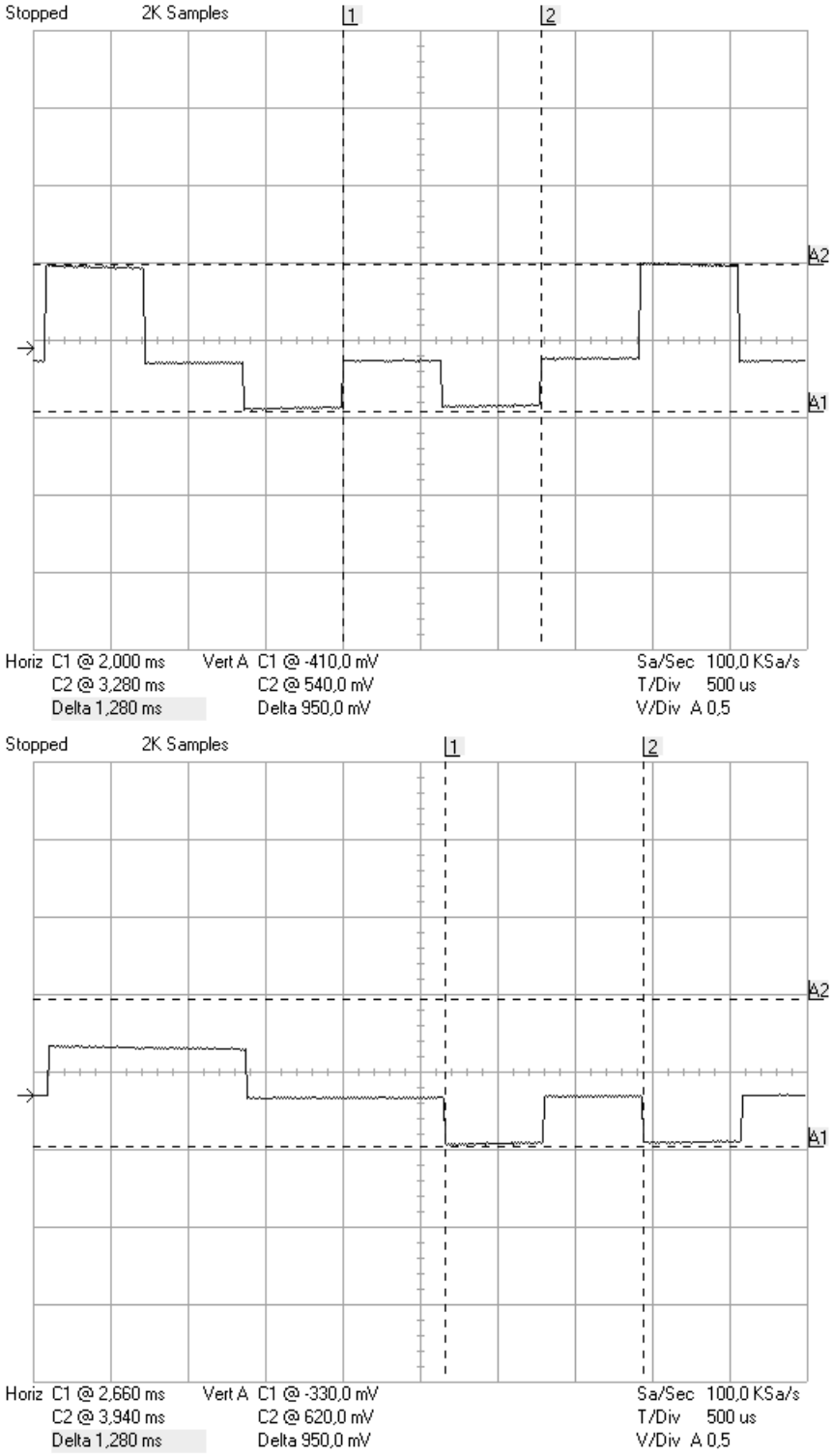


Figura 5.48. Señal Multinivel I-Q, demodulador 16-QAM

Para 16 y 64-QAM se han utilizado los marcadores A1 y A2 del eje vertical para indicar los valores mínimo y máximo que alcanzarían.

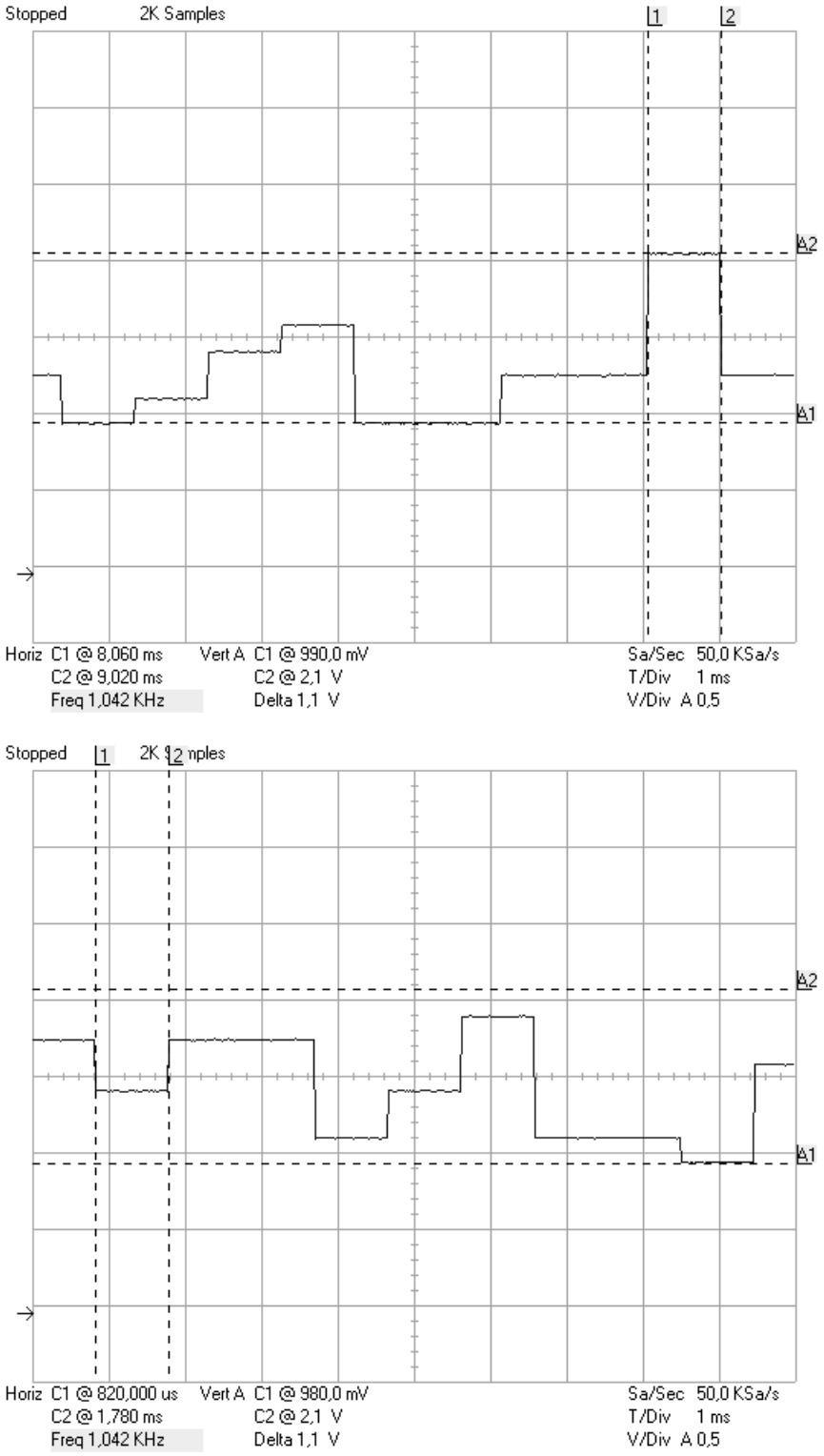


Figura 5.49. Señal Multinivel I-Q, demodulador 64-QAM

5.4.2.5. Convertidor de L a 2 Niveles

En la salida de este subsistema están los bits I-Q convertidos de la señal multinivel y se muestra en las figuras 5.50, 5.51 y 5.52 para cada modulación QAM.

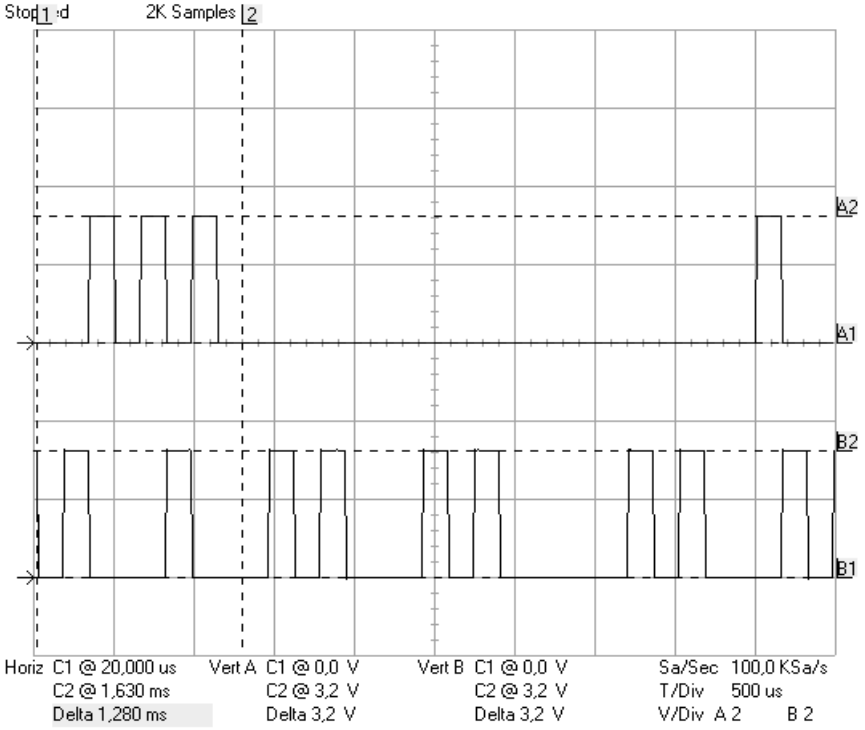


Figura 5.50. Bits de Canales I-Q, demodulador 4-QAM

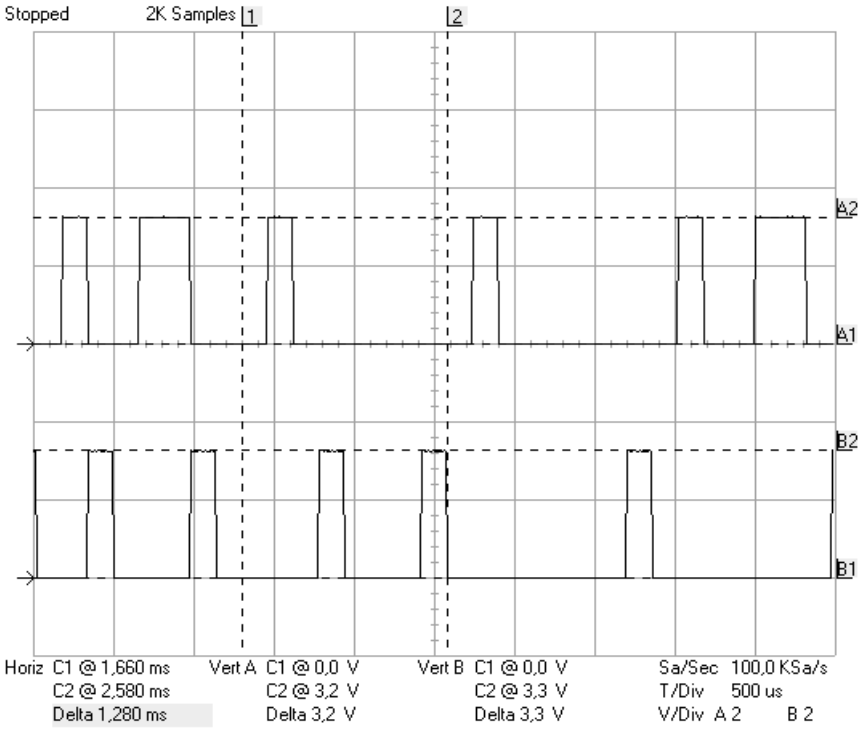


Figura 5.51. Bits de Canales I-Q, demodulador 16-QAM

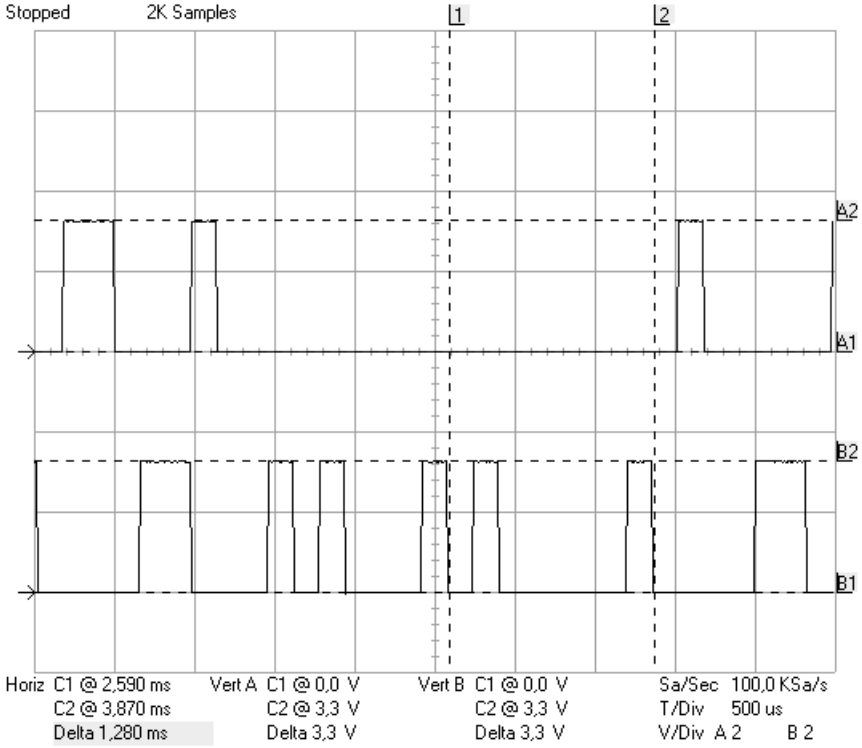


Figura 5.52. Bits de Canales I-Q, demodulador 64-QAM

5.4.2.6. Conversor Paralelo – Serial

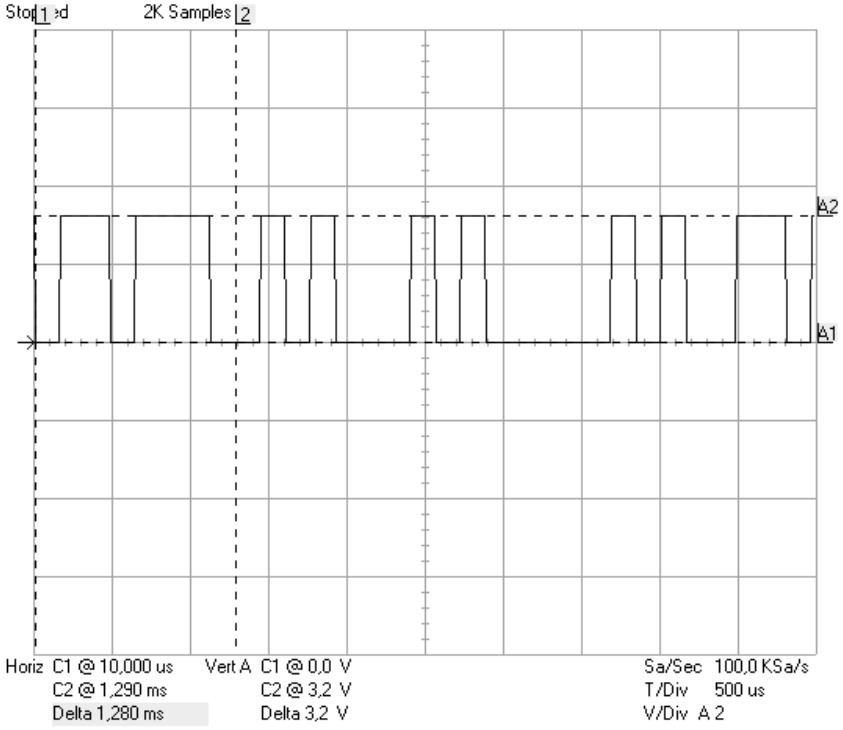


Figura 5.53. Señal Demodulada en 4-QAM

La última prueba de funcionamiento del demodulador es la secuencia de bits recuperada que se extraerá de su salida. El resultado para cada caso de modulación QAM se muestra en las figuras 5.54, 5.55 y 5.56.

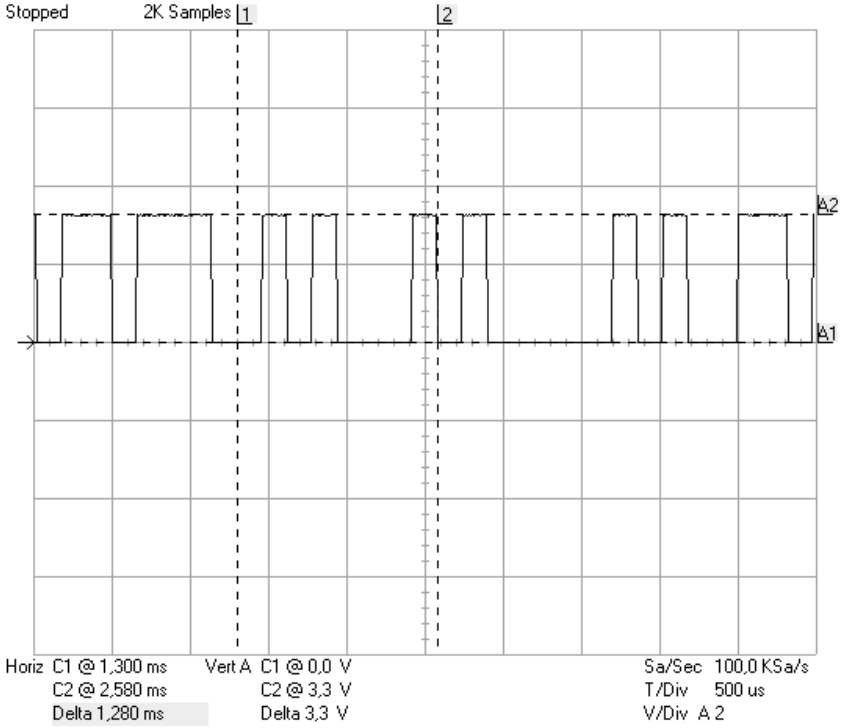


Figura 5.54. Señal Demodulada en 16-QAM

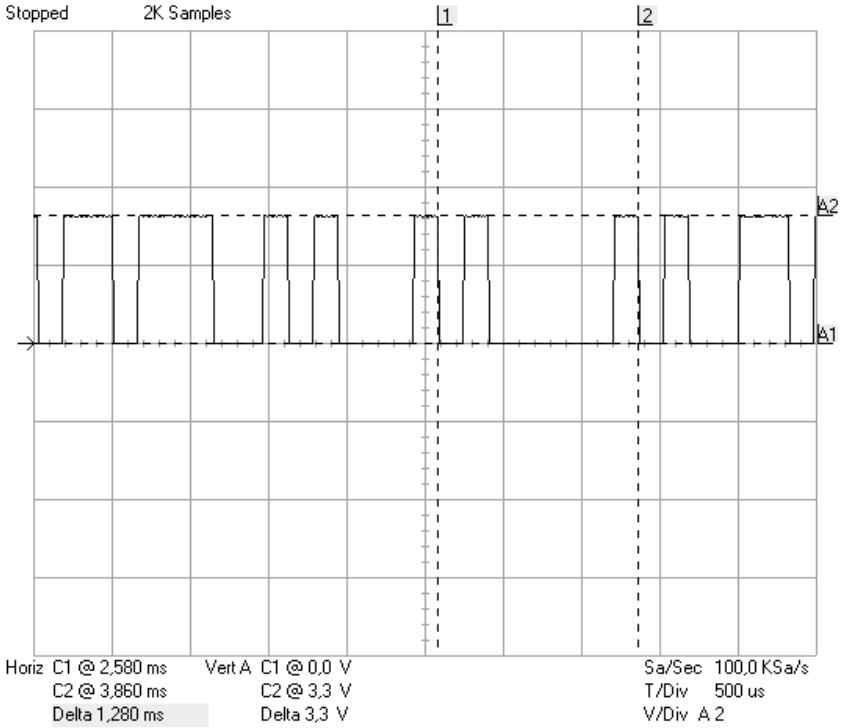


Figura 5.55. Señal Demodulada en 64-QAM

Como se esperaba, la secuencia de bits recuperada es la misma en todos los casos de modulación QAM, debido a que se utilizó el mismo mensaje de prueba, excepto por la diferencia de retardos inherentes al tiempo de procesamiento de la señal en cada sistema.

CAPÍTULO VI

6. CONCLUSIONES Y RECOMENDACIONES

6.1.CONCLUSIONES

- Las estructuras generalizadas que se emplean para el diseño de los sistemas desarrollados en este proyecto, evidencian la gran versatilidad de la descripción de hardware por medio de la programación con Xilinx System Generator, debido a que se realizan los diferentes moduladores QAM sin alterar el diagrama de bloques general.

- La señal de reloj utilizada por el FPGA, limita la velocidad de operación del sistema implementado en la tarjeta. Sin embargo, esto se puede modificar con la especificación de una fuente externa para la señal de reloj. En caso de la Spartan-3E contiene un cristal oscilador de 50MHz.

- La posibilidad de realizar cambios en el diseño, modificando únicamente el software con XSG, demuestra la enorme potencialidad de desarrollo en cualquier tipo de modelo o sistema sobre un FPGA.

- La co-simulación en hardware se convierte en una eficaz herramienta de verificación, gracias a que permite la comprobación directa sobre un FPGA de una porción compilada del sistema para evaluar su comportamiento.

- Las tarjetas de entrenamiento basadas en FPGAs se pueden utilizar para realizar prototipos en distintas áreas como son: sistemas de comunicaciones, redes de información, transmisión de datos, procesamiento digital de señales entre otras, debido a la gran densidad de compuertas y biestables que la componen, además de su variedad de recursos de interconexión para su configuración interna y la gran cantidad de periféricos que permiten su relación con dispositivos externos a la tarjeta de entrenamiento.

- El bloque para la estimación de recursos se convierte en un instrumento de decisión substancial cuando se va a sintetizar el modelo realizado en XSG, porque se conoce de forma rápida el porcentaje de los recursos que se van a utilizar y en algunos casos cambiar las opciones de implementación.

- En cuanto a las limitaciones de diseño en XSG se encuentra el manejo de los periféricos como son: ADC, DAC, LCD, RS232, VGA, entre otros.

6.2. RECOMENDACIONES

- Para un trabajo futuro, se podrían realizar pruebas de propagación en espacio libre agregando un sistema de amplificación y una antena, además de proveer una frecuencia de portadora variable.

- También se recomienda profundizar en el manejo de otros componentes de Xilinx ISE, como el EDK de tal forma que se pueda utilizar todos los periféricos que componen la tarjeta de entrenamiento Spartan-3E de Xilinx.