IMPLEMENTACION DE MODULACIONES DIGITALES CON LA FPGA SPARTAN

NELSON FABIAN MORENO GARCIA

UNIVERSIDAD PONTIFICA BOLIVARIANA
FACULTAD DE INGENIERIA ELECTRONICA
BUCARAMANGA
2016

IMPLEMENTACION DE MODULACIONES DIGITALES CON LA FPGA SPARTAN

INVESTIGADOR:
NELSON FABIAN MORENO GARCIA

DIRECTOR:
ING. ALEX ALBERTO MONCLOU, M.SC

UNIVERSIDAD PONTIFICA BOLIVARIANA
FACULTAD DE INGENIERIA ELECTRONICA
BUCARAMANGA
2016

Nota de aceptación.

_____
_____
_____
_____
_____

_____
Presidente del Jurado

_____
Firma del Jurado

_____
Firma del Jurado

DEDICATORIA

Dedico este proyecto de grado a Dios quien me ha brindado infinitas bendiciones y un apoyo incondicional en los momentos tanto felices como tristes vividos en mi faceta personal, familiar, laboral y estudiantil.

A mis padres que gracias a Dios me han dado fortaleza, me han brindado una armonía familiar por medio de un gran lazo de amor y felicidad que nos une a todos basados en un apoyo incondicional tanto en los momentos difíciles como en las alegrías.

A mi esposa y mi hijo que siempre han estado conmigo en los momentos difíciles y que han sido motivo de inspiración, ya que por ellos he podido sacar adelante mi proyecto de grado con la convicción de poder brindarles un mejor futuro.

A mi director de proyecto de grado Alex Monclou, porque siempre estuvo guiándome y motivándome para poder alcanzar los objetivos planteados durante mi pregrado y quien gracias a Dios compartió sus conocimientos y supo aprovechar lo mejor de mí.

Contenido

Tabla de Imágenes

RESUMEN GENERAL DE TRABAJO DE GRADO

TITULO: IMPLEMENTACÍON DE MODULACIONES DIGITALES EN LA FPGA SPARTAN PARA EL LABORATORIO DE COMUNICACIONES DE LA UNIVERSIDAD PONTIFICIA BOLIVARIANA

AUTOR(ES): NELSON FABIAN MORENO GARCIA

FACULTAD: Facultad de Ingeniería Electrónica

DIRECTOR(A): ALEX ALBERTO MONCLOU, M.Sc.

RESUMEN

El documento muestra la implementación de las modulaciones digitales básicas como son ASK, FSK, PSK, QAM y TDM por medio de la FPGA Spartan XC3S700A programada en el software ISD Desing Suite System Edition 14.7 de XILINX en el lenguaje verilog. Se deja un documento en donde se detalla la programación para cada modulación y su implementación en el laboratorio de comunicaciones en su respectivo computador. De esta manera, el estudiante vera la modulación digital en cada puesto de trabajo, lo cual hará que la observación de cada una de las modulaciones sea dinámica puesto que el grupo de estudiantes rotara para verlas.

PALABRAS CLAVES: FPGA, Modulaciones Digitales, Verilog, ASK, FSK, PSK, QAM, TDM.

V° B° DIRECTOR DE TRABAJO DE GRADO

GENERAL SUMMARY OF WORK OF GRADE

TITLE: IMPLEMENTATION OF DIGITAL MODULATIONS FPGA SPARTAN LABORATORY OF COMMUNICATIONS UNIVERSIDAD PONTIFICIA BOLIVARIANA

AUTHOR(S): NELSON FABIAN MORENO GARCIA

FACULTY: Facultad de Ingeniería Electrónica

DIRECTOR: ALEX ALBERTO MONCLOU, M.Sc.

ABSTRACT

The document shows the implementation of the basic digital modulations such as ASK, FSK , PSK , QAM and TDM are through XC3S700A Spartan FPGA programmed in the ISD Desing System Suite software XILINX Edition 14.7 in Verilog language. a document in which the programming for each modulation and its implementation in the laboratory of computer communications in their respective detailed left . In this way , the student will see the digital modulation in each job , which will make the observation of each of the modulations is dynamic as the rotated group of students to see them.

KEYWORDS:

FPGA, DIGITAL MODULATIONS, VERILOG, ASK, FSK, PSK, QAM, TDM.

V° B° DIRECTOR OF GRADUATE WORK

INTRODUCCION

Gracias a las facilidades que aportan los sistemas en el tratamiento de datos digitales, el uso de sistemas informáticos se ha extendido en campos del desarrollo profesional por sus ventajas al aumentar la velocidad y el volumen de datos que estas pueden manejar. Por ello en el campo de las comunicaciones se ha venido generando un mayor interés en la creación de sistemas orientados a la transmisión de datos digitales [1].

En telecomunicaciones el término modulación engloba el conjunto de técnicas para transportar información sobre una onda portadora, típicamente una onda senoidal. Estas técnicas permiten un mejor aprovechamiento del canal de comunicación lo que permitirá transmitir más información simultánea y proteger la información de posibles interferencias o ruidos [2]. La modulación digital brinda la posibilidad de utilizar mecanismos que detectan y corrigen más fácilmente estas interferencias o ruidos ocurridos durante la transmisión [1].

En este trabajo de grado se utilizan los conocimientos adquiridos durante el pregrado en el campo de las comunicaciones y de las técnicas digitales, para la implementación de las modulaciones propuestas (ASK,FSK,PSK,QAM,TDM) por medio de la FPGA SpartanXC3S700A que se encuentran en el almacén de la Universidad Pontificia Bolivariana.

En el capítulo 1 se habla de los objetivos propuestos que son la transmisión y recepción de cada una de las modulaciones digitales. Pag 5.

En el capítulo 2 se hace un resumen de cada una de las modulaciones en donde se explica cómo funcionan y como varían respecto a las demás ya sea en amplitud, fase y frecuencia; además se habla del hardware utilizado en las FPGAS y el tipo de software ISD Desing Suite System Edition utilizados en el desarrollo de este proyecto de grado. Pag (6-16).

En el capítulo 3 se realiza una introducción en el manejo del ISD Desing Suite System Edition en donde el driver del software permite reconocer el tipo de dispositivo utilizado y el lenguaje con que se programara; además se dejan los pasos a seguir para la creación de los módulos en lenguaje verilog y se da una introducción en cada uno de los módulos utilizados (DAC, comparador, divisores de frecuencia, el generador seno y las roms de las señales) y se dejara los diagramas de bloques de las modulaciones digitales. Pag (17-27).

En el capítulo 4 se deja constancia de los resultados obtenidos de cada una de las modulaciones digitales que son: ASK, FSK, PSK, TDM Y QAM. Pag (28-32).

En el capítulo 5 se deja constancia de las conclusiones obtenidas durante el desarrollo del proyecto de grado, así como las referencias de cada una de las bibliografías utilizadas. Pag (34-35).

1. OBJETIVOS

1.1. Objetivo General:

Implementar diferentes mecanismos de modulación digital por medio de la FPGA SpartanXC3S700A.

1.2. Objetivos Específicos:

- Implementar la transmisión y recepción de la modulación digital ASK por medio de la FPGA SpartanXC3S700A.
- Implementar la transmisión y recepción de la modulación digital FSK por medio de la FPGA SpartanXC3S700A.
- Implementar la transmisión y recepción de la modulación digital BPSK por medio de la FPGA SpartanXC3S700A.
- Implementar la transmisión y recepción de la modulación digital QAM por medio de la FPGA SpartanXC3S700A.
- Implementar la transmisión y recepción de la modulación digital TDM por medio de la FPGA SpartanXC3S700A.
- Documentar el proceso realizado para cada una de las diferentes modulaciones digitales.

2. GENERALIDADES:

La telecomunicación del prefijo griego tele, “distancia” y del latín communicare es una técnica consistente en transmitir un mensaje desde un punto a otro normalmente con el atributo típico adicional de ser bidireccional. Cuando nos comunicamos, estamos compartiendo información [1].

La palabra datos se refiere a hechos, conceptos e instrucciones presentados en cualquier formato acordado entre las partes que crean y utilizan dichos datos.

La transmisión de datos es el intercambio de información entre dos dispositivos a través de algún medio de transmisión, o el espectro radioeléctrico. Para que la transmisión de datos sea posible, los dispositivos de comunicación deben de ser parte de un sistema de comunicación formado por hardware y software. La efectividad del sistema de comunicación de datos depende de cuatro características fundamentales:

- Entrega
- Exactitud
- Puntualidad
- Retardo variable [1].

Desde la década de los 60 las comunicaciones se han orientado hacia los sistemas digitales por las ventajas que estas tienen respecto a las comunicaciones analógicas, por exactitud y menor interferencia de ruido, también de esto el costo de los circuitos digitales como microprocesadores y el manejo de la información bit a bit que provee a las diferentes técnicas de modulación un encriptado (técnica de seguridad) son ventajas que llaman la atención [2].

Las modulaciones digitales al igual que las modulaciones análogas nos ofrecen varios métodos de transmisión que modifican la señal portadora para transmitir la información deseada, la Modulación en Amplitud (AM) equivalente a la Modulación por desplazamiento de Amplitud (ASK) varia la amplitud de su señal portadora proporcional a la señal moduladora. La Modulación de Frecuencia (FM) equivalente a la Modulación por desplazamiento de Frecuencia (FSK) varia la frecuencia de su señal portadora proporcional a la señal moduladora y la Modulación de Fase (PM) equivalente a la Modulación por desplazamiento de Fase (PSK) varia la fase de su señal portadora proporcional a la señal moduladora. Las Modulaciones Digitales además permiten hacer combinaciones entre diferentes métodos de transmisión optimizando así el manejo del ancho de banda y el canal de transmisión y en este proyecto se mezcló la PSK y ASK para obtener la señal QAM.

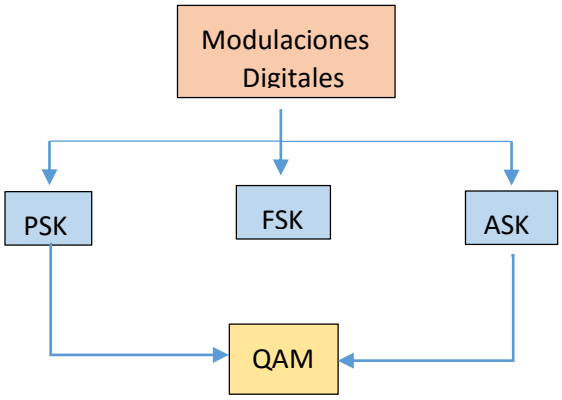

Figura 1. Tipos de Modulaciones Digitales [3]

La fig. 1 muestra el diagrama de bloques de las modulaciones digitales analógicas. Los datos digitales deben ser modulados sobre una señal analógica que ha sido manipulada para aparecer como dos valores distintos correspondientes al 0 y al 1 binario.

2.1. Modulaciones Digitales:

2.1.1. Modulación por desplazamiento de Amplitud (ASK) : Es una modulación en donde la potencia de la señal portadora se cambia para representar el 1 o 0 binario, en este caso la frecuencia y la fase de la portadora se mantiene constantes [3] [4].

La modulación en ASK es variante de la modulación en AM que se adapta perfectamente a las condiciones de los sistemas digitales, además de que le permite trabajar sobre una sola frecuencia de transmisión en vez de tener que utilizar con pulsos cuadrados que contienen componentes en todas las frecuencias del espectro [4].

La modulación ASK en su forma más simple es denominada OOK (*On Off Keying*) que representa los diferentes estados digitales 0 o 1 con la ausencia y presencia de una señal portadora respectivamente para cada estado.

La ecuación que describe la modulación digital de amplitud mediante una señal binaria es:

$$v_{AM}\,(t) = [1 + v_m(t)] * \left[\frac{A}{2} * \cos(w_c t)\right]$$

Donde:

$v_{AM}(t)$: Voltaje de la onda de amplitud modulada.

$v_m(t)$: Señal digital moduladora (volts).

A/2: Amplitud de la señal portadora no modulada (volts).

w_c : Frecuencia de la señal portadora en radianes (radianes por segundo)

Tomando en cuenta esta ecuación, si [$v_m(t)$] es una forma de onda binaria normalizada, en la que +1 V=1 lógico y -1 V=0 lógico. Por consiguiente, para una entrada de 1 lógico, $v_m(t)$=+1, y la ecuación se reduce a:

$$v_{AM}\,(t) = [1 + 1] * \left[\frac{A}{2} * \cos(w_c t)\right]$$

$$= A\cos(w_c t)$$

Para una entrada de 0 lógico, $v_m(t) = -1$, la ecuación se reduce a:

$$v_{AM}\,(t) = [1 - 1] * \left[\frac{A}{2} * \cos(w_c t)\right]$$

$$= 0$$

Así, para 100% de modulación, $v_{AM}\,(t)$ es $A\cos(w_c t)$ o 0. Por consiguiente la portadora está encendida o apagada.

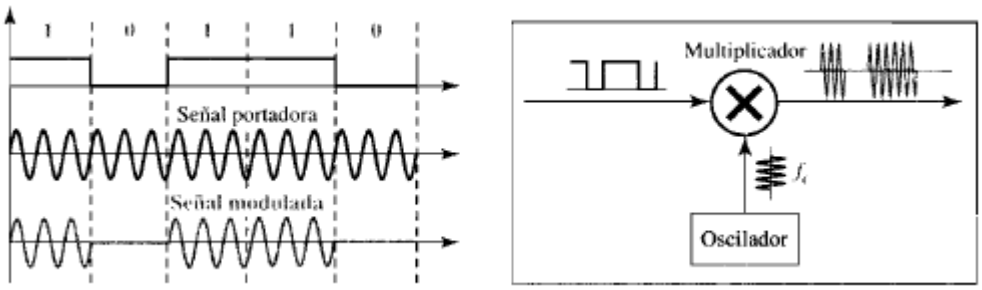

Figura 2 Implementación de ASK Binario (OOK) [3]

En la figura 2 se observa como la señal portadora es multiplicada por el valor binario de la señal a transmitir dando como resultado la señal modulada con una amplitud de 0 para un valor binario de 0 y una señal sinodal para el valor binario 1, como se puede representar en la siguiente tabla.

Valor Binario	Ecuación de la Señal Modulada
0	0
1	A sen (Wc t)

2.1.2. Modulación por desplazamiento de Frecuencia (FSK): Es una modulación en donde la frecuencia de la señal portadora cambia para representar el 1 o 0 binarios. La frecuencia de la señal durante la duración del bit es constante y su valor depende del tipo de bit que se desee trasmitir, tanto la amplitud como la fase de la señal portadora permanecen constantes [3] [5].

FSK evita la mayor parte de los problemas de ruido de la ASK, debido a que el dispositivo en el receptor está buscando cambios específicos de frecuencia en cierto número de periodos [3].

Una forma de explicar FSK de manera sencilla es a través de BFSK ó FSK binaria. Consideremos dos señales portadoras con dos frecuencias diferentes, se utiliza la primera frecuencia portadora si el dato binario es 0; se usa la segunda si el dato binario es 1 teniendo así una señal modulada con dos frecuencias. La señal modulada puede ser no coherente cuando hay discontinuidad en la fase de la señal debido al elemento en la señal que termina y la señal siguiente con diferente frecuencia. En la señal modulada coherente la fase se mantiene entre la frontera de los dos elementos de la señal utilizando un oscilador controlado por voltaje que cambia su frecuencia según el voltaje de entrada.

La ecuación general de la FSK binaria es:

$$v_{fsk}(t) = v_c \cos\{2\pi[f_c + v_m(t)\Delta f]t\}$$

Donde:

$v_{fsk}(t) =$ Forma de onda binaria FSK

$v_c =$ Amplitud de la portadora (volts)

$f_c =$ Frecuencia central de la portadora (hertz)

$\Delta f =$ Desviación máxima de frecuencia (hertz)

$v_m(t) =$ Señal moduladora de entrada binaria (± 1)

De acuerdo con la ecuación, el corrimiento máximo de frecuencia de portadora, Δf, es proporcional a la amplitud y a la polaridad de la señal binaria de entrada. La señal moduladora [$v_m(t)$] es una forma de onda binaria normalizad, en la que el 1 lógico = 1, y el 0 lógico = -1.

Así, para una entrada de 1 lógico, $v_m(t) = +1$, y la ecuación toma la forma siguiente:

$$v_{fsk}(t) = v_c \cos\{2\pi[f_c + \Delta f]t\}$$

Para una entrada de 0 lógico, $v_m(t) = -1$, y la ecuación se transforma en

$$v_{fsk}(t) = v_c \cos\{2\pi[f_c + \Delta f]t\}$$

Con una FSK binaria, la señal binaria de entrada corre (desvía) a la frecuencia de la portadora. Cuando la señal binaria de entrada cambia de un 0 lógico a un 1 lógico y viceversa, la frecuencia de salida se desplaza entre dos frecuencias: una frecuencia de marca, frecuencia de trabajo o frecuencia de 1 lógico (f_m), y una frecuencia de espacio o de 0 lógico (f_s).

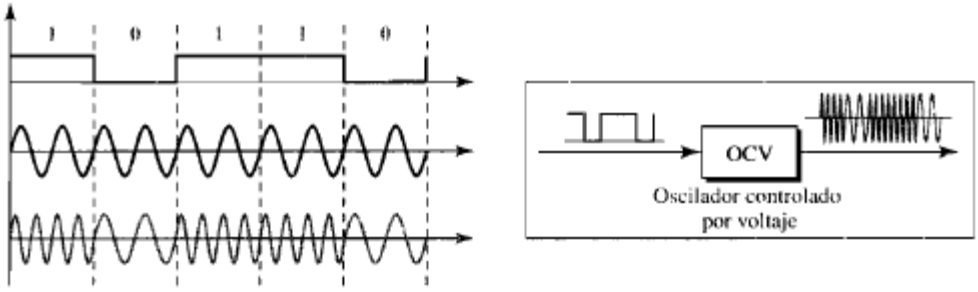

Figura 3 Implementación de BFSK [3]

En la figura 3 se observa como la señal portadora es modificada por el oscilador controlado por voltaje aumentando la frecuencia de la señal modulada cuando el estado de la señal moduladora es 1 y dejando su frecuencia igual cuando es 0, como se puede representar en la siguiente tabla.

Valor Binario	Ecuación de la Señal Modulada
0	A sen (Wc t)
1	A sen ((Wc+Δf) t)

2.1.3. Modulación por desplazamiento de fase (PSK): Es una modulación en donde la fase de la portadora cambia para representar el 1 o 0 binario. Tanto la amplitud de pico como la frecuencia de la portadora permanecen constantes.

Un modulador PSK de 2 fases, pone la fase de la portadora en uno de entre 2 valores dependiendo de la señal moduladora. Un sistema de dos fases se denomina BPSK [6].

PSK no es susceptible a la degradación por ruido que afecta a ASK ni a las limitaciones de banda de FSK. Esto significa que pequeñas variaciones en la señal se pueden detectar fiablemente en el receptor [3].

La señal modulada resultante, responde a la expresión:

$$A_p \cdot cos[2\pi ft + \theta]$$

Donde:

- A_p=amplitud
- f=frecuencia
- t=tiempo
- θ=representa cada uno de los valores posibles de la fase, tantos como estados tenga la señal codificada en banda base multinivel.

  Dependiendo del número de posibles fases a tomar, recibe diferentes denominaciones. Dado que lo más común es codificar un número entero de bits por cada símbolo, el número de fases a tomar es una potencia de dos. Así tendremos BPSK con 2 fases (equivalente a PAM), QPSK con 4 fases (equivalente a QAM), 8-PSK con 8 fases y así sucesivamente.

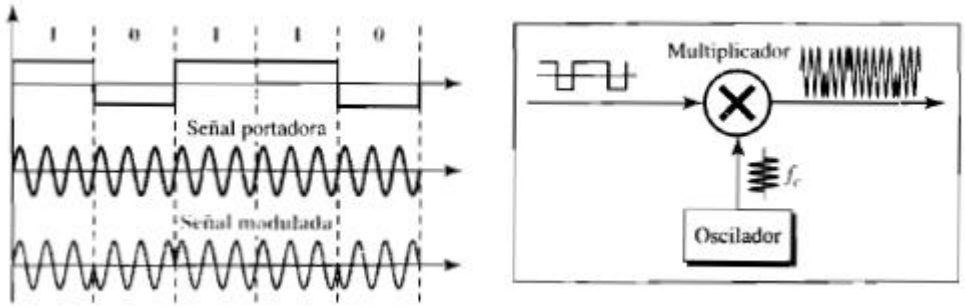

Figura 4 Implementación Modulación BPSK [3]

En la figura 4 se observa como la señal portadora es modificada para cuando el estado binario de la señal moduladora es 0 mostrando el complemento de la señal portadora al dar un desfase de 180°. Cuando el estado binario de la señal moduladora es 1 el desfase en la señal portadora es de 0°. Como se puede representar en la siguiente tabla.

Valor Binario	Ecuación de la Señal Modulada
0	A sen ((Wc t)+θ)
1	A sen (Wc t)

2.1.4. Modulación de Amplitud en Cuadratura (QAM) : La modulación de amplitud en cuadratura significa combinar ASK y PSK de tal forma que haya un contraste máximo entre cada bit, en este caso se pueden tener por variaciones en fase y variaciones en amplitud dándonos X veces y X posibles variaciones y el número correspondiente de bits [3].

Teóricamente hay infinidad de variaciones pero las más usadas son 4-QAM y 8-QAM en ambos casos el número de desplazamiento en amplitud es menor al desplazamiento en fase, debido a que los cambios en amplitud son más susceptibles al ruido.

La modulación QAM permite que dos señales provenientes de dos fuentes independientes, pero con características de ancho de banda similares, ocupen el mismo ancho de banda de transmisión y se puedan separar en el extremo receptor, ahorrando así el uso del ancho de banda disponible. Así, si dos señales $d_1(t)$ y $d_2(t)$, modulan dos señales portadoras de la misma frecuencia, una desfasada en 90° respecto a la otra, mediante el uso de moduladores de producto (que multiplican las señales por la portadora).

La señal QAM se obtiene modulando en DBL (doble banda lateral):

$$s(t) = d_1(t)\cos(2\pi f_c t) + d_2(t)\cos(2\pi f_c t - 90°)$$

$$s(t) = d_1(t)\cos 2\pi f_c t + d_2(t)\sin 2\pi f_c t$$

Donde

$d_1(t)$ =Es la componente en fase de la señal QAM

$d_2(t)$ =La componente en cuadratura

f_c = Frecuencia central de la portadora (hertz)

Las señales de datos $d_1(t)$ y $d_2(t)$ se modulan utilizando DSB-SC (doble banda lateral con portadora suprimida); la señal $d_1(t)$ utilizando la portadora de referencia en fase; y la señal $d_2(t)$ utilizando la portadora de referencia en cuadratura. Las señales moduladas $d_1(t)$ y $d_2(t)$ se suman para formar la señal QAM.

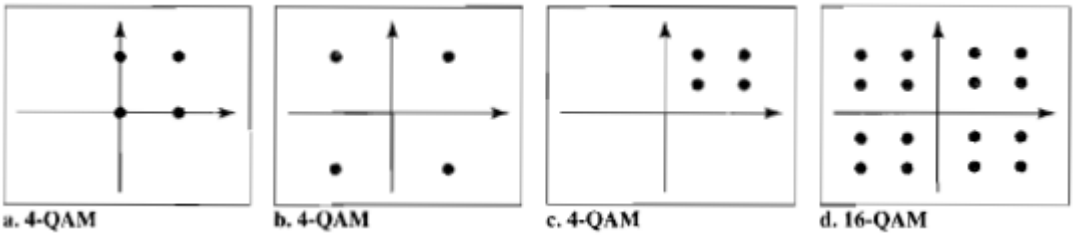

Figura 5 Diagrama de Constelaciones para algunos QAM [3]

En la figura 5 se puede ver algunos tipos de QAM en donde la fase es el eje X y la amplitud el eje Y, mostrando así diferentes diagramas de constelación para un 4QAM.

2.1.5. Multiplexación por división de tiempo (TDM): La multiplexación por división en el tiempo síncrona es posible cuando la velocidad de transmisión alcanzable por el medio excede la velocidad de las señales digitales a transmitir. Se pueden transmitir varias señales digitales (o señales analógicas que transportan datos digitales) a través de una única ruta de transmisión.

La técnica TDM síncrona se denomina síncrona no porque se emplee transmisión síncrona, sino porque las ranuras temporales (almacena un carácter de datos) se preasignan y fijan a las distintas fuentes.

En la TDM síncrona se asigna exactamente siempre la misma ranura de tiempo a cada dispositivo, tanto si el dispositivo tiene o no algo que transmitir, las ranuras de

tiempo se agrupan en tramas. Una trama está formada por un ciclo completo de ranuras de tiempo, incluyendo una o más ranuras dedicadas a cada dispositivo [3] [7].

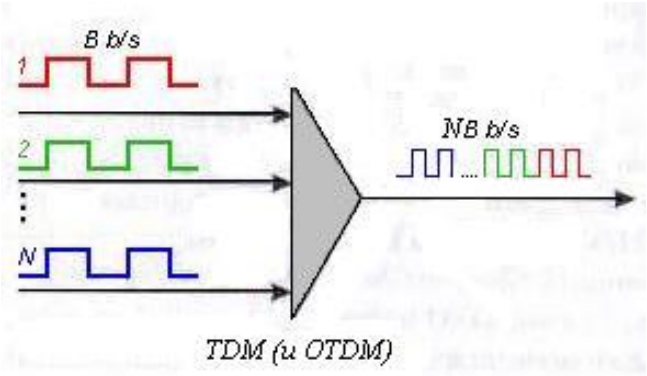

Figura 6 Multipexación por División de Tiempo [3]

En la figura 6 se muestra el esquema de un multiplexor de cuatro canales. En él las señales son muestreadas, de acuerdo a una secuencia de control lógica.

2.2. Hardware:

2.2.1. *Field Programmable Gate Arrays*: *Field Programmable Gate Arrays* (FPGAs) son dispositivos semiconductores que se basan en torno a una matriz de bloques lógicos configurables (CLBs) conectados a través de interconexiones programables. Las FPGAs pueden ser reprogramables para requisitos de las aplicaciones o funcionalidades deseadas después de la fabricación. Esta característica distingue las FPGAs de circuitos integrados de aplicación específica, que se fabrican a medida para las tareas de diseño específico **[8]**.

Los arreglos de las FPGAs son chips de silicio reprogramables. La adopción del chip FPGA en las industrias es guiada por el hecho que las FPGAs combinan lo mejor de los circuitos integrados de aplicación específica (ASICs) y sistemas basados en procesador. Las FPGAs ofrecen velocidades temporizadas por hardware y fiabilidad, pero sin requerir altos volúmenes de recursos para compensar el gran gasto que genera un diseño personalizado de ASIC [9].

A diferencia de los procesadores, las FPGAs son verdaderamente paralelos por naturaleza, así las diferentes operaciones de procesamiento no tienen que competir por los mismos recursos. Cada tarea de procesamiento independiente es asignada a una sección del chip y puede ejecutarse de manera autónoma sin ser afectada por otros bloques de lógica. Como resultado, el rendimiento de una parte de la aplicación no se ve afectado cuando se agregan otros procesos [9].

Figura 7 FPGA SpartanXC3S700A. [13]

En la figura 7 se observa la FPGA SpartanXC3S700A utilizada para el desarrollo e implementación de este trabajo de grado.

2.2.2. FPGA SpartanXC3S700A: La FPGA SpartanXC3S700A pertenece a la familia de FPGA Spartan3A, esta es una mejora a la familia de las Spartan3E y Spartan3, esta familia de FPGAs ofrece densidades de compuertas desde las 50mil hasta 1.4millones. La Spartan XC3S700A tiene 700mil compuertas equivalentes a 13.248 celdas lógicas, los bloquen que conforman esta tarjeta son:

- Bloques lógicos configurables (CLB) flexibles que implementan la lógica programada más elementos de almacenamiento como los flip-flop, ejecutando así funciones lógicas como por ejemplo las de almacenamiento de datos.
- Bloques de entrada salida (IOB) que controlan el flujo de datos y la lógica interna del dispositivo, apoyando el flujo bidireccional de datos.
- Bloque de RAM proporciona almacenamiento de datos en forma de 18 Kbit.
- Bloque multiplicador aceptan dos números de 18 bits binarios como multiplicandos y calcula el producto.
- Bloque Reloj Digital Manager (DCM) proporciona las soluciones completamente digitales de auto-calibración en la difusión, consiste en retrasar, multiplicar, dividir y cambiar de fase el reloj de señales.

La familia Spartan-3A cuenta con una rica red de enrutamiento que interconecta los cinco bloques funcionales, que transmiten señales entre ellos. Cada bloque funcional tiene asociado una matriz de conmutación que permite múltiples conexiones al enrutamiento.

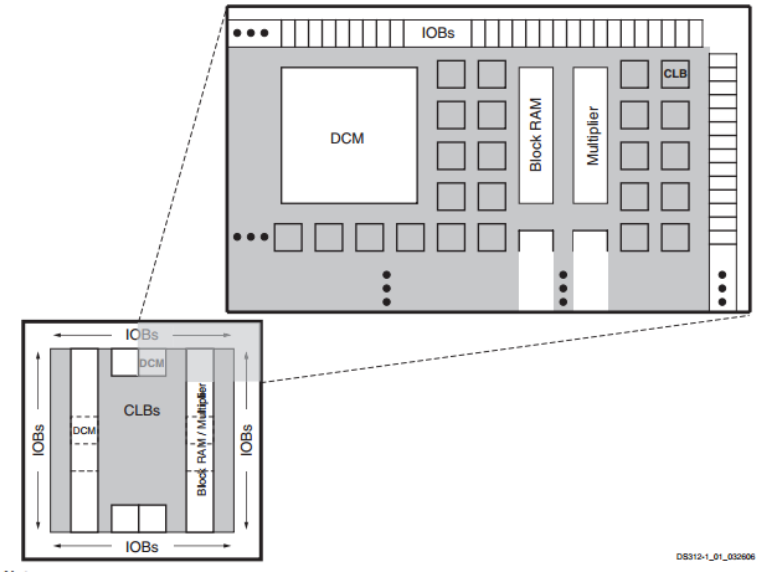

Figura 8 Bloques Funcionales de la FPGA Spartan3A [3]

En la figura 8 muestra cómo están organizados los distintos componentes en una FPGA Spartan 3A. Se puede observar el anillo de bloques de E/S (Input/Output Blocks,IOBs) que rodea el arreglo matricial de los bloques lógicos configurables (Configurable Logic Blocks, CLBs).

2.3.Software:

2.3.1.*ISD Desing Suite System Edition*: Proporciona las herramientas fundamentales, tecnologías y flujo de diseño familiar para lograr resultados óptimos de diseño. Estos incluyen compuerta reloj inteligente para la reducción de potencia dinámica, diseño de equipo para los equipos de diseño de sitios múltiples, la preservación de diseño para la repetitividad de tiempo, y una opción de reconfiguración parcial de una mayor flexibilidad del sistema, el tamaño, la potencia y la reducción de costos **[10]**.

El ISE Design Suite ofrece herramientas para mejorar la productividad del diseñador y para proporcionar configuraciones flexibles de las ediciones Design Suite. Acelera la creación de algoritmos en lenguaje C, C++ y especificaciones del sistema C para ser dirigidos directamente en Xilinx en todos los dispositivos programables sin la necesidad de crear manualmente un RTL (registro de transferencia de nivel). La reconfiguración parcial es una tecnología de reconfiguración que permite a los

diseñadores cambiar la funcionalidad sobre la marcha, lo que elimina la necesidad de reconfigurar completamente y volver a establecer vínculos, mejorando dramáticamente la flexibilidad que ofrecen los FPGAs [10].

2.3.2. Verilog: Verilog HDL es un lenguaje de descripción de hardware que se utiliza para diseñar y documentar los sistemas electrónicos y permite a los diseñadores el diseño en diferentes niveles de abstracción. Verilog es el HDL más utilizado con una comunidad de usuarios de más de 50.000 diseñadores activos [11] [12]**.**

Verilog nació en 1985 como un lenguaje propietario, pero en 1990 dicho lenguaje se convirtió de dominio público, permitiendo a las empresas hacer uso del mismo, con el fin de aumentar la difusión del lenguaje [12].

Verilog es utilizado para describir sistemas digitales tales como procesadores, memorias o elementos más simples como son los flip-flops. Esto significa que un lenguaje de este tipo puede utilizarse para describir cualquier hardware (digital) a cualquier nivel. Estos sistemas se pueden describir de las siguientes formas: [12] [13].

- Nivel estructural: Se utilizan elementos previamente creados, por el propio desarrollador o por terceros, donde cada elemento se interconecta con otro. conectan varios bloques cuya funcionalidad es pequeña, para que comunicándolos entre si realicen una tarea mayor.

- Nivel de comportamiento: El diseñador describe la transferencia de información entre registros [12].

Estos dos niveles de descripción pueden mezclarse, dando diseños mixtos. Aunque existen multitud de lenguajes HDL, tan sólo en la actualidad se utilizan dos de ellos: Verilog y VHDL, los cuales son estándares de la IEEE [12].

3. METODOLOGIA Y PROCEDIMIENTO:

3.1. Descripción del Problema: Actualmente es de vital importancia tener conceptos claros sobre las modulaciones digitales y el manejo de estas para implementarlas en los dispositivos programables como las FPGA, ya que no todos los estudiantes de Ingeniería Electrónica durante el pregrado se familiarizan los mecanismos de modulación desarrollados en tecnologías digitales avanzadas. Por lo cual nace la necesidad de implementar un módulo que permita acercarnos al manejo de las modulaciones digitales en donde el objeto de estudio es la transmisión y recepción de la señal digital además de utilizar recursos de la universidad como lo son las FPGA SpartanXC3S700A, tecnología que se usa con menor intensidad dado el avance que se tiene con las nuevas FPGAS. El proyecto de grado las utiliza y las deja como parte del laboratorio de comunicaciones para dar a conocer las modulaciones digitales básicas.

El módulo permitirá conocer algunos mecanismos de modulación digital que se implementara en la FPGA SpartanXC3S700A y en la documentación entregada, se podrá seguir paso a paso la aplicación de la FPGA en cada modulación desarrollada. Esto proporcionara al estudiante las habilidades en el campo de la comunicación digital para que en su futuro profesional como ingeniero electrónico cuente con bases para la comprensión y análisis de las diversas modulaciones.

3.2.**Configuración de *ISD Desing Suite System Edition* :** La configuración de *ISD Desing Suite System Edition* permite que el *driver* del software reconozca el tipo de dispositivo que se desea utilizar y el lenguaje con el que se programara y posteriormente agregar o crear los módulos en verilog .

Para configurar el software *ISD Desing Suite System Edition* se ejecuta el software y en la pestaña *file* seleccionamos *New Project* como se observa en la figura 9. Esto abre una ventana de dialogo.

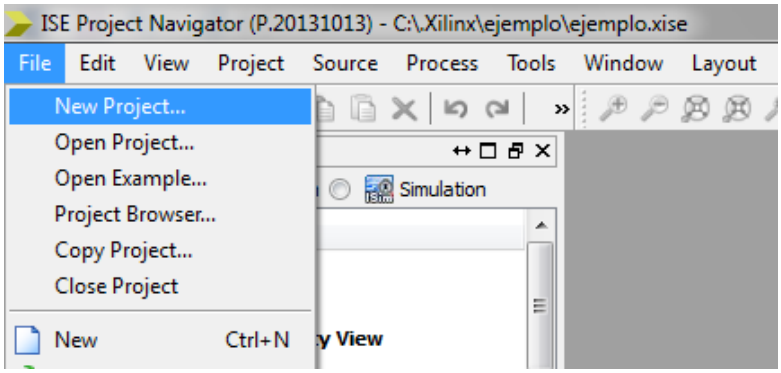

Figura 9. Crear un Nuevo Proyecto en ISD Desing Suite System Edition

La ventana de dialogo mostrada en la figura 10 indica en donde queremos guardar el proyecto y como se llama la carpeta en la que estará guardado, además de permitir dar una descripción breve del mismo.

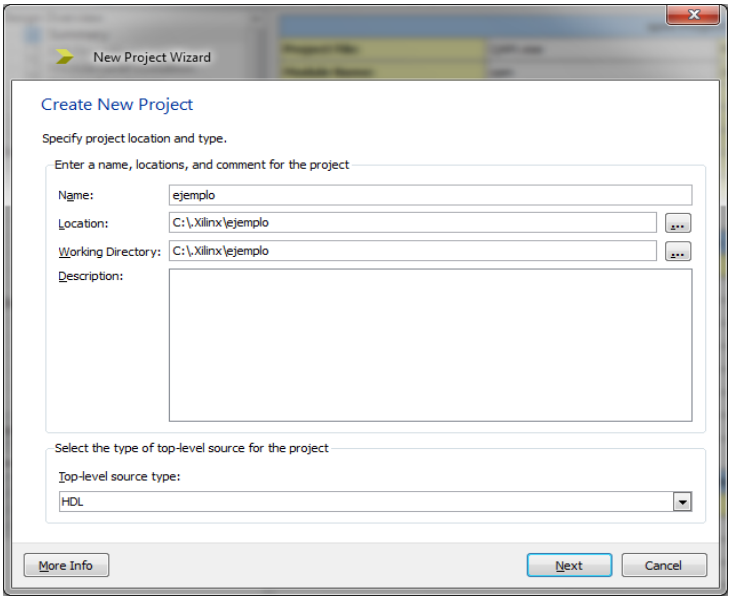

Figura 10. Ventana de dialogo#1 para Crear un Nuevo Proyecto en ISD Desing Suite System Edytion

Al hacer clik en el boton *next* la ventana de dialogo muestra una segunda ventada en donde se configura la familia de la FPGA , el dispositivo y el lenguaje de programación que se muestra en la figura 11.

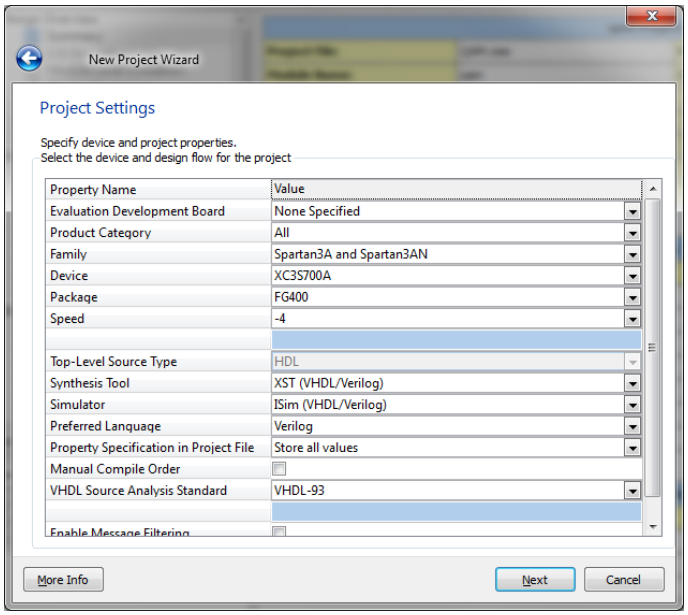

Figura 11. Ventana de dialogo#2 para Crear un Nuevo Proyecto en ISD Desing Suite System Edytion

Por último se hace click en el botón *next* y la configuración muestra una ventana en donde se encuentran todas los arreglos ya implementados. En la ventana de trabajo del software se muestra un árbol del proyecto en donde se configuran los diferentes módulos de programación en verilog para cada modulación digital.

3.3. Creación de un nuevo módulo de Verilog: Una vez configurado el *ISD Desing Suite System Edition,* en la ventana de trabajo se muestra el árbol de proyecto que se creó anteriormente mostrado en la figura 12, en este árbol es donde se agregan los diferentes módulos base de verilog diseñados para la implementación de cada modulación digital, para crear un nuevo módulo se hace clik izquierdo en el árbol del proyecto y seleccionamos *New Source.*

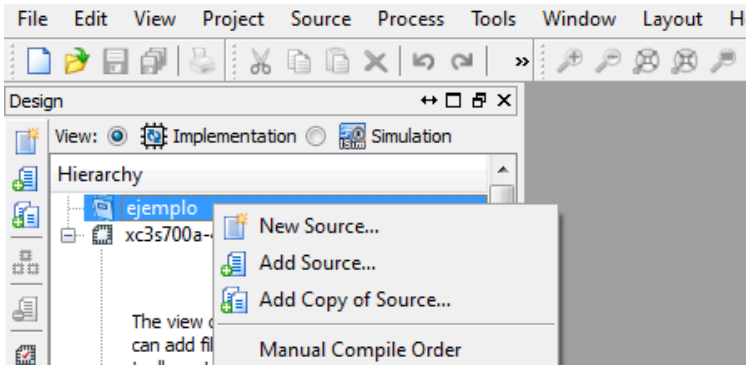

Figura 12. Crear un Nuevo Módulo en Verilog

Al crear el nuevo módulo de verilog, se abre una ventana de dialogo en donde seleccionamos el tipo de modulo deseado en nuestro caso verilog module, el nombre del módulo y el lugar donde vamos a guardar este módulo como archivo, finalizamos haciendo click en next como se observa en la figura 13. Una nueva ventana del módulo nos pregunta por las entradas y salidas de esté como se observa en la figura 14, si en el momento no las tenemos definidas podemos omitir esta ventana y hacer click en next, automáticamente se abre la ventana de trabajo figura 15, en donde programamos y diseñamos los modulos en *ISD Desing Suite System Edition.*

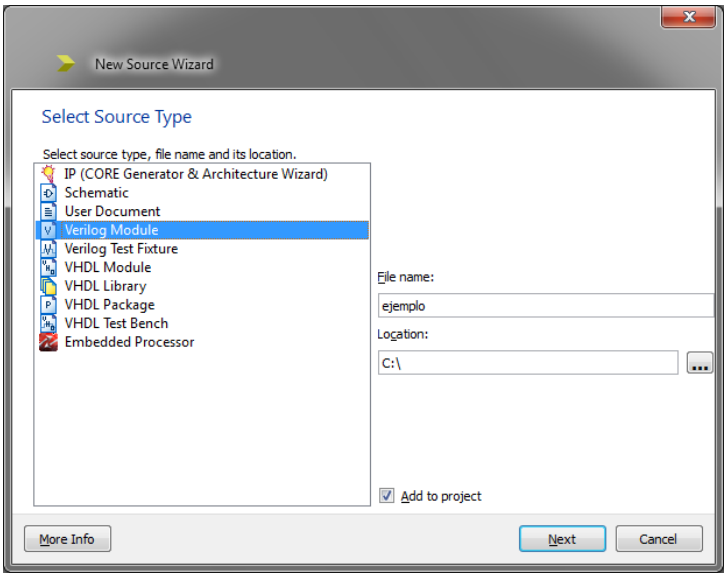

Figura 13. Configuración del Modúlo de Verilog

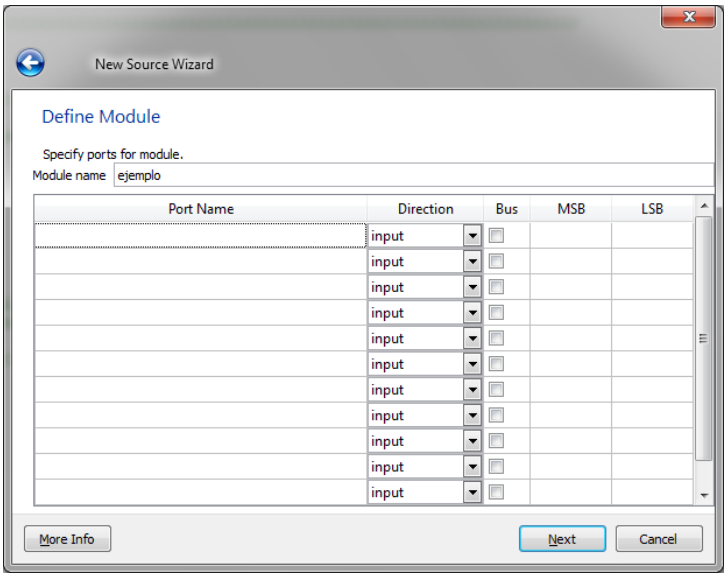

Figura 14. Asignación de entradas y salidas del Módulo de Verilog

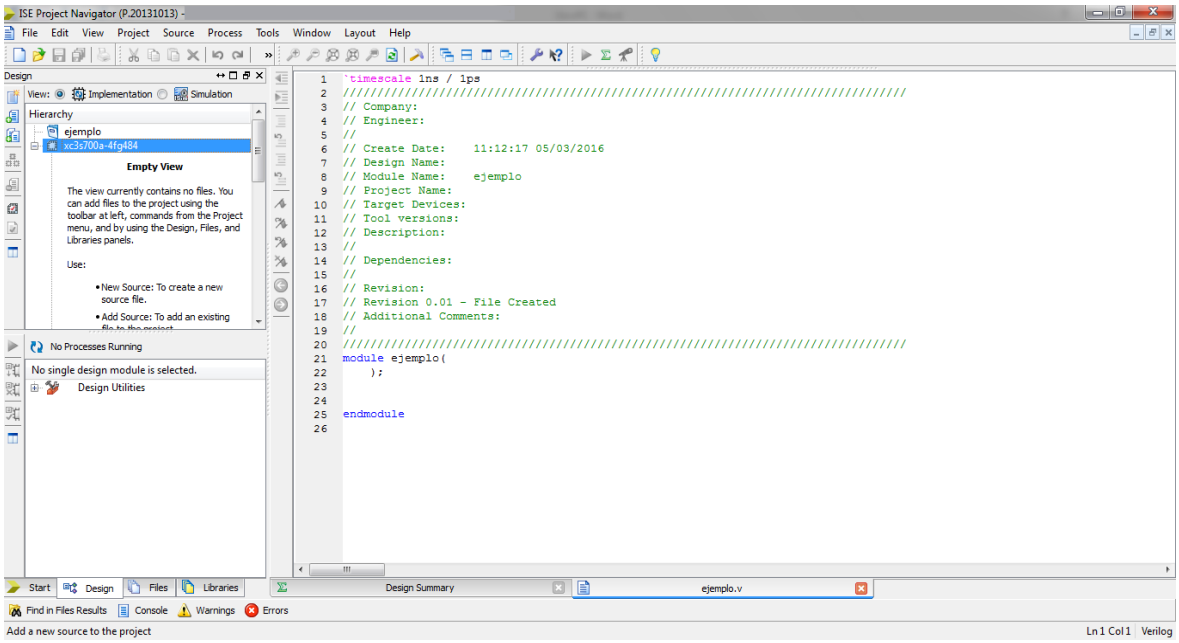

Figura 15. Ventana de Trabajo en ISD Desing Suite System Edition

3.3. Módulos Base en Verilog para la implementación de las diferentes Modulaciones Digitales: Para la implementación de las diferentes modulaciones digitales en la FPGA SpartanXC3S700A, se crean diferentes módulos base en verilog con funciones específicas que al ejecutarlos simultáneamente y agregando algunas condiciones se visualiza el comportamiento de cada una de las modulaciones digitales observando diferencias en sus características como amplitud, frecuencia y fase. Los módulos utilizados son:

3.3.1. Divfreq (Divisor de frecuencia): El divisor de frecuencia utiliza el reloj interno de la FPGA SpartanXC3S700A de 50MHz para crear un nuevo reloj con una nueva frecuencia, esta señal de reloj es una señal cuadrada utilizada como señal moduladora debido a su tren de pulso 0 y 1. El divisor de frecuencia funciona con un contador que aumenta su valor con cada ciclo de reloj interno aumentando así el periodo en el estado de la salida conmutando esta entre valores de 0 y 1 permitiendo bajar la frecuencia de la señal de reloj a nuestras necesidades.

El siguiente diagrama de bloque muestra el funcionamiento de este módulo base y en el Anexo 1 se puede estudiar más detalladamente el algoritmo de programación.

El divisor de frecuencia permite bajar la frecuencia de la señal de salida y la frecuencia de la señal de clock y al final se visualiza en el osciloscopio donde se obtiene frecuencias de 512 (herz), 1200 (herz) y 2200 (herz).

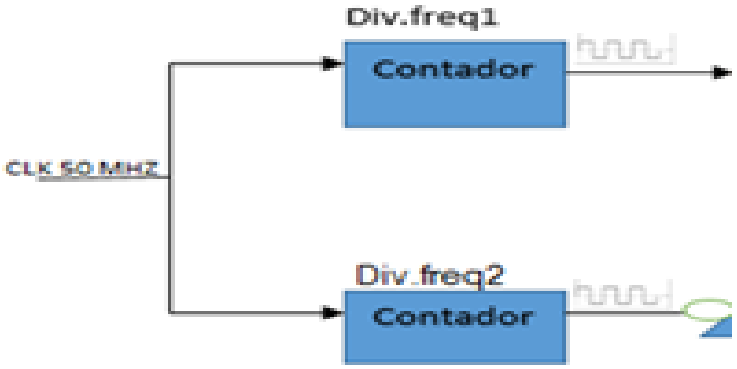

Figura 16 Diagrama de bloque Modulo divfreq

3.3.2. Rom_sen (Memoria): Es una memoria direccionada con 127 posiciones en donde cada posición tiene un valor binario de 8 bits expresado de forma decimal, este valor representa la magnitud de la señal portadora que se utilizara para las modulaciones. El siguiente diagrama de bloque muestra el funcionamiento de este módulo base y en el Anexo 2 se puede estudiar más detalladamente el algoritmo de programación.

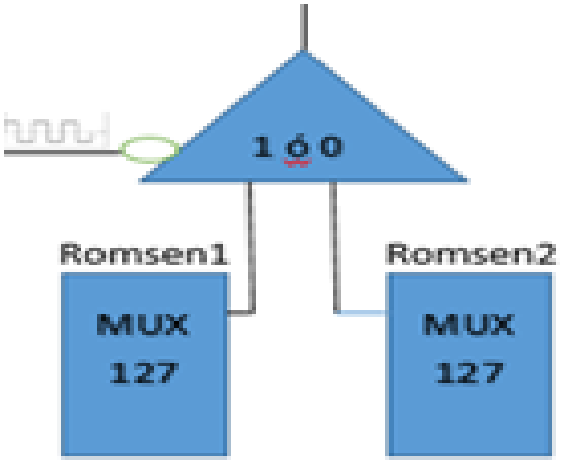

Figura 17 Diagrama de bloque Modulo rom_sen

3.3.3. Gen_sen (Generador de Señales): El generador de señales utiliza un reloj de entrada y un contador que permite hacer un barrido en todas las direcciones del módulo rom_sen, la salida de este módulo es el valor de la salida asignado por el módulo rom_seno. El siguiente diagrama de flujo muestra el funcionamiento de este módulo base y en el Anexo 3 se puede se puede estudiar más detalladamente el algoritmo de programación.

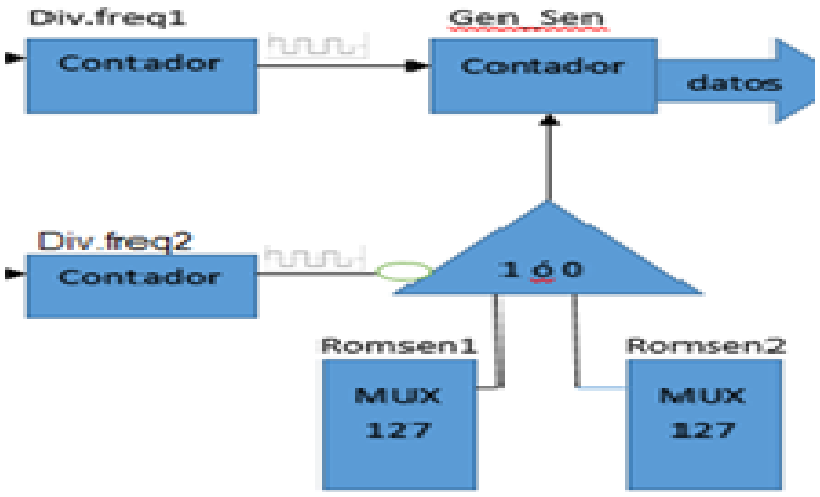

Figura 18 Diagrama de bloque Modulo gen_sen

3.3.4. Dac (Conversor Digital/Analogo): Este módulo es implementado de un módulo creado por el ingeniero Holguer Becerra en el Semillero ADT [13]. Este módulo permite convertir los valores de cada dirección de la memoria en una salida análoga continua de FPGA. Para poder visualizar la señal modulada es necesario aplicar un filtro pasabajos en la salida. En el Anexo 4 se puede estudiar más detalladamente el algoritmo de programación.

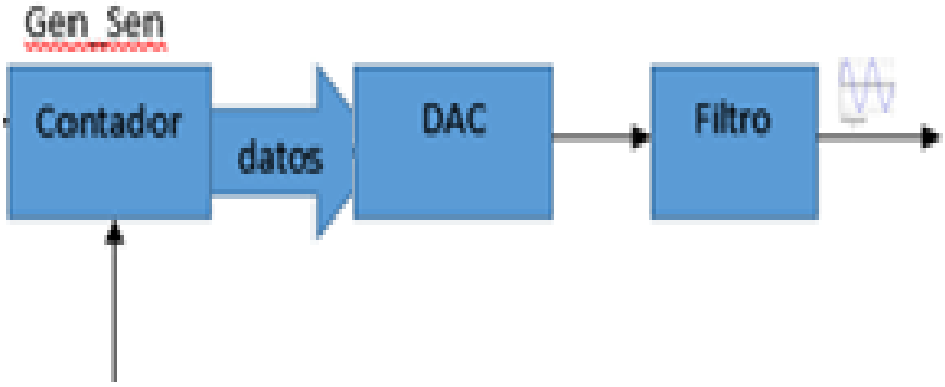

Figura 19 Diafragma de bloque Modulo DAC

3.3.5. Comp (Comparado recepción): En el comparador tenemos una recepción de la señal transmitida, comparando el valor de la señal de entrada del DAC con un valor binario de una memoria de referencia. En el Anexo 5 se puede se puede estudiar más detalladamente el algoritmo de programación.

3.4. Modulaciones digitales en la FPGA: Teniendo claro la función de los diferentes módulos base en verilog para la implementación de las modulaciones digitales, explicaremos cada una de ella en un diagrama de bloques.

3.4.1. ASK (Modulación por desplazamiento de Amplitud)

En este módulo se cuenta con dos módulos ram_sen, cada una tiene en sus direcciones el dato binario para las diferentes amplitudes de la señal trasmitida que serán generadas por el módulo DAC, estos datos son conectados al módulo DAC por medio de las variables bus_duty1 y bus_duty2 respectivamente a cada memoria en el módulo gen_sen, dependiendo del estado de la señal del módulo divftreq divisor2, una de las dos variables es conectada al módulo DAC teniendo así la generación de la señal modulada de transmisión con dos amplitudes diferentes. En el Anexo 6 se puede estudiar más detalladamente el algoritmo de programación.

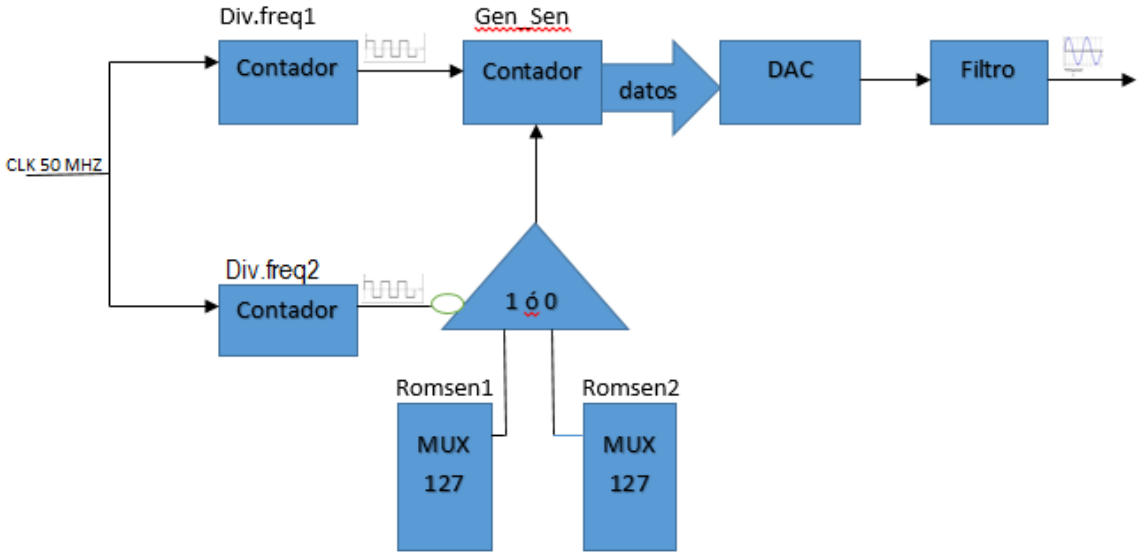

Figura 20 Diagrama de bloques ASK

3.4.2. FSK (Modulación por desplazamiento de Frecuencia)

En este módulo se cuenta con una memoria para la reconstrucción de la señal seno a trasmitir, el cambio con el que se generan las diferentes frecuencias es por medio

del módulo divfreq divisor1 que cambia la frecuencia con la que el dato de la memoria es llamado en el módulo gen_sen y posteriormente generado en el módulo DAC, el modulo principal tiene una condición en donde la variable frec cambia su valor dependiendo del estado del módulo divfrq divisor2, esta variable es la entrada del módulo divisor1 con el cual cambia su frecuencia. . En el Anexo 7 se puede estudiar más detalladamente el algoritmo de programación.

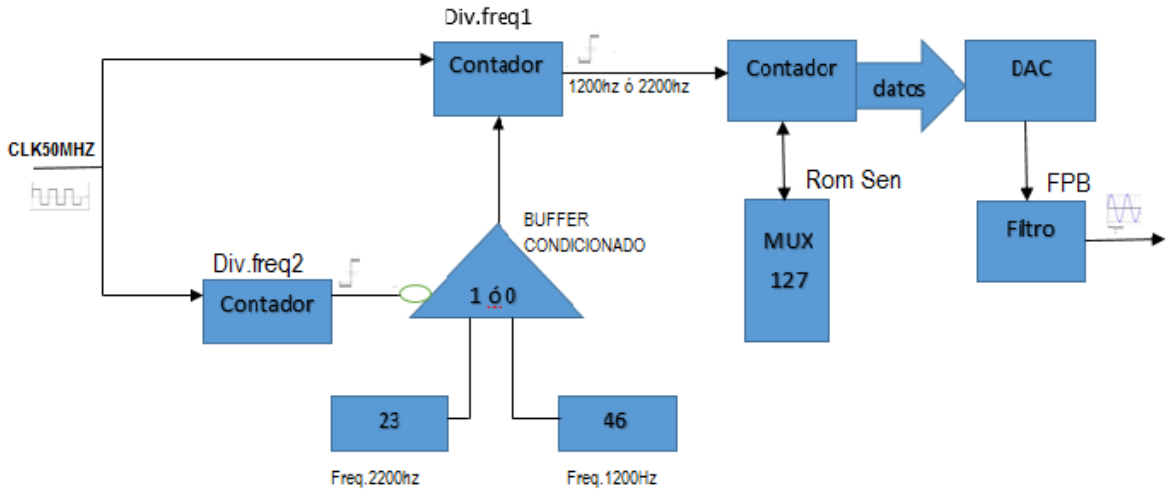

Figura 21 Diagrama de bloques FSK

3.4.3. PSK (Modulación por desplazamiento de fase)

Este módulo cuenta con un módulo ram_seno de memoria para la reconstrucción de la señal seno a trasmitir, el cambio con el que se generan las diferentes fases es por medio de dos contadores en el módulo gen_seno, direccionando el dato muestreado de la memoria, un periodo de señal tiene 127 muestras, los contadores en este módulo inician en 0 o 64 teniendo así la posibilidad de manejar un desfase de 0° y 90° respectivamente. En el módulo principal la variable cond define cuál de los dos contadores utilizar dependiendo del estado del módulo divfreq divisor2 y el módulo DAC reconstruye la señal transmitida dando el comportamiento deseado en la salida. En el Anexo 8 se puede estudiar más detalladamente el algoritmo de programación.

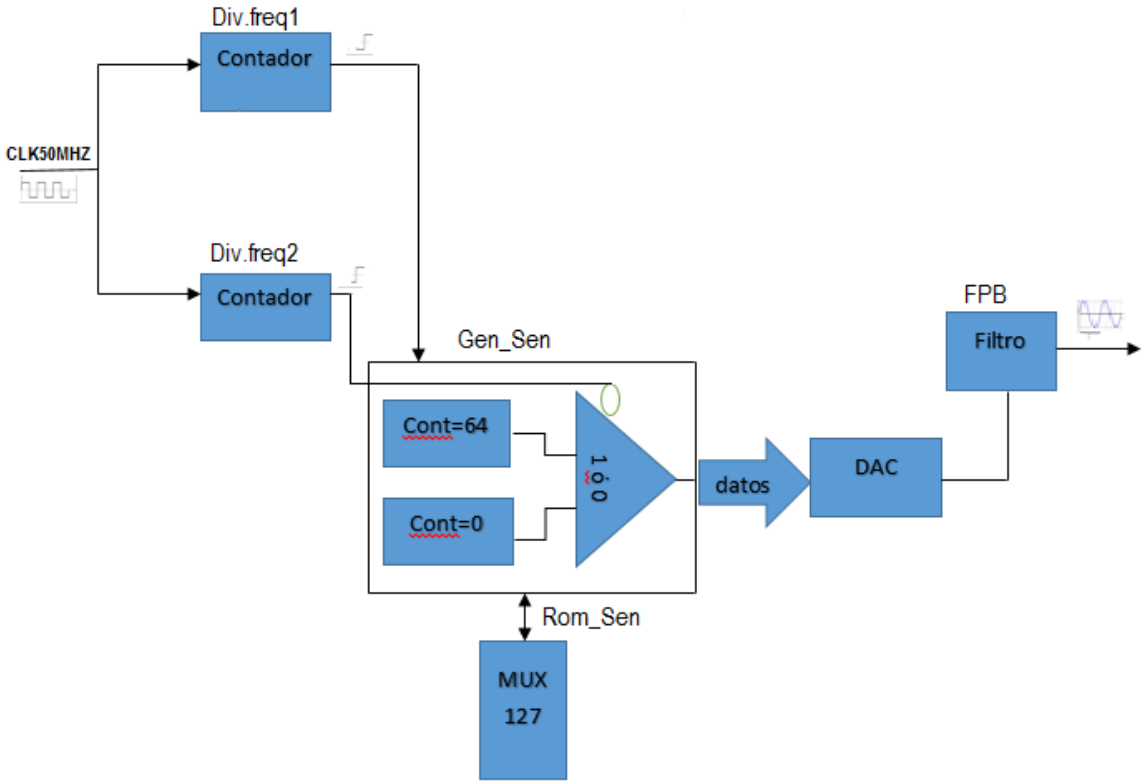

Figura 22 Diagrama de bloques PSK

3.4.4.TDM (Multiplexacion por división de Tiempo)

Este módulo cuenta dos módulos de ram en el módulo gen_sen y dos módulos DAC para cada una de las señales a multiplexar, en nuestro caso tenemos una memoria para una señal sinusoidal y una señal triangular, en el módulo gen_sen se reconstruyen las dos señales simultáneamente, y cada módulo DAC genera la señal correspondiente, el módulo divfreq divisor2 maneja la salida multiplexando las dos señales dependiendo del estado en que se encuentre, además de esto podemos ver las señales separadas multiplexadas en el tiempo con la variable de entrada SW. En el Anexo 9 se puede estudiar más detalladamente el algoritmo de programación.

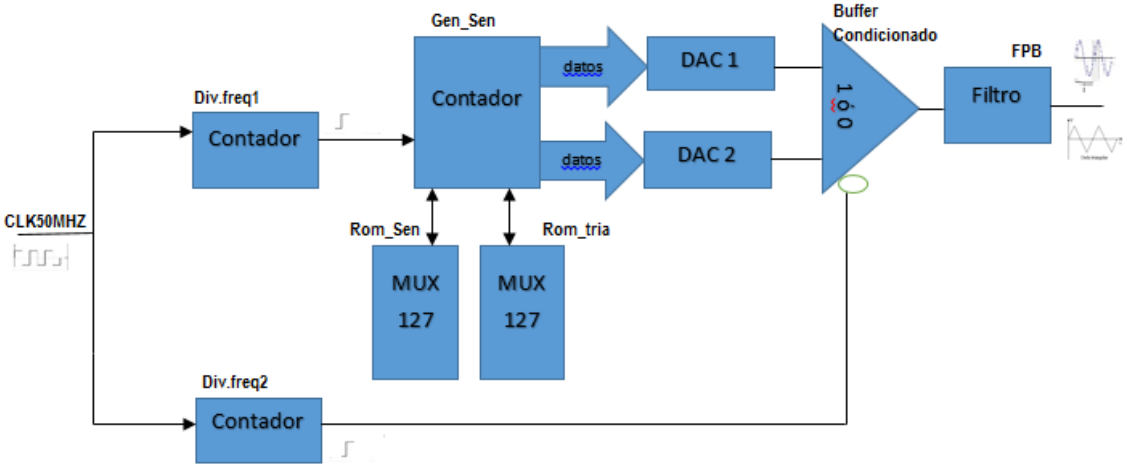

Figura 23 Diagrama de bloques TDM

3.4.5. QAM (Modulación de Amplitud en Cuadratura)

El comportamiento de esta modulación digital es la combinación de la modulación ASK y PSK, para combinar las amplitudes y las fases se hizo necesario tener dos módulos rom_sen para cada una de las diferentes amplitudes y dos contadores en el módulo gen_sen para manejar los desfases de 0° y 180°. En este módulo a diferencia de los demás se cuenta con un generador de estados que es un contador de 0 a 3 teniendo así las 4 posibles combinaciones de las señales en cuadratura, la variable est es un vector de dos posiciones que cambia con el valor del generador de estados, la posición [0] está asociada a la variable sel que indica cuál de los dos contadores en el módulo gen_sen se utiliza, modificando así la fase de la señal de salida y la posición [1] está asociada a la variable bus_sen quien define cuál de las dos memorias se utiliza, modificando así la amplitud de la señal de salida. En el Anexo 10 se puede estudiar más detalladamente el algoritmo de programación.

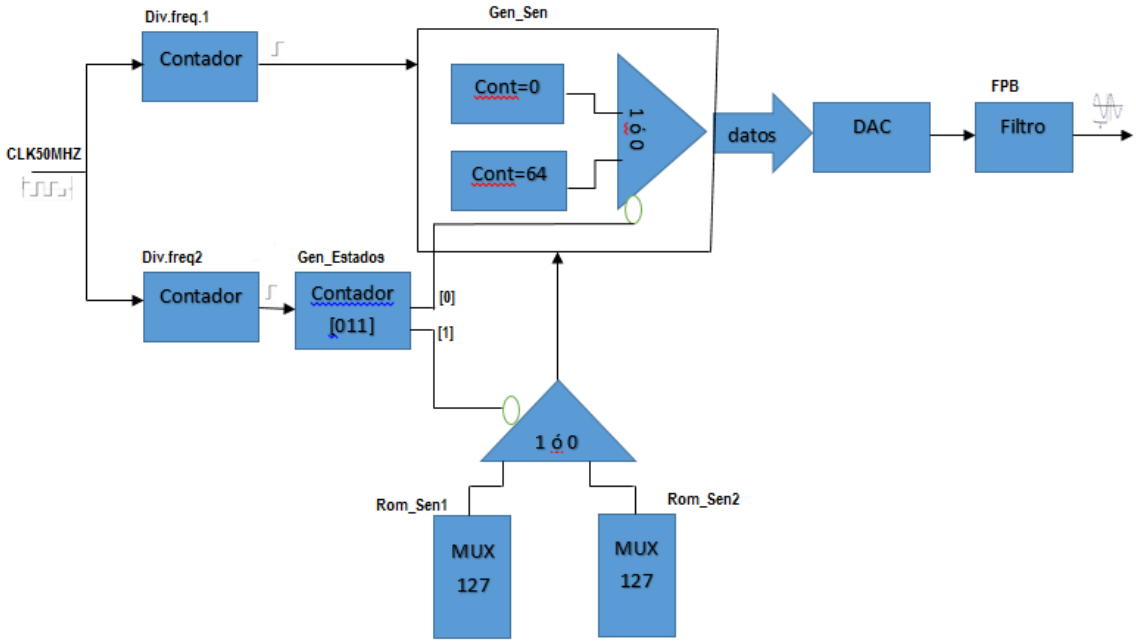

Figura 24 Diagrama de bloques QAM

4. RESULTADOS

4.1 Resultados Modulación ASK

Para la modulación ASK en la figura 25 se observa la señal moduladora con una frecuencia de 512Hz y su señal modulada de 1200Hz variando su amplitud según el estado lógico de la señal de moduladora. En la figura 26 se observa la señal de recepción con la misma frecuencia que la señal moduladora y sus mismos estados lógicos.

Figura 22. Resultados Modulación ASK Transmisión

Figura 23 resultados Modulación ASK Recepción

4.2. Resultados Modulación FSK

En la modulación FSK en la figura 27 se observa la señal moduladora con una frecuencia de 512Hz y la señal modulada en la transmisión con una frecuencia de 1200Hz para un estado lógico [0] y 2200Hz para un estado lógico [1] de la señal moduladora. En la figura 28 se observa la señal de recepción con la misma frecuencia que la señal moduladora y sus mismos estados lógicos.

Figura 24.Respuesta Modulación FSK Transmisión

Figura 25. Respuesta Modulación FSK Recepción

4.3. Resultados modulación PSK

En la modulación PSK en la figura 29 se observa la señal moduladora con una frecuencia de 512Hz y la señal modulada en la transmisión con una frecuencia de 1200Hz. La señal modulada de transmisión cambia su fase 180° para un estado lógico [1] de la señal moduladora, en la figura 30 se observa la señal de recepción con la misma frecuencia que la señal moduladora y sus mismos estados lógicos.

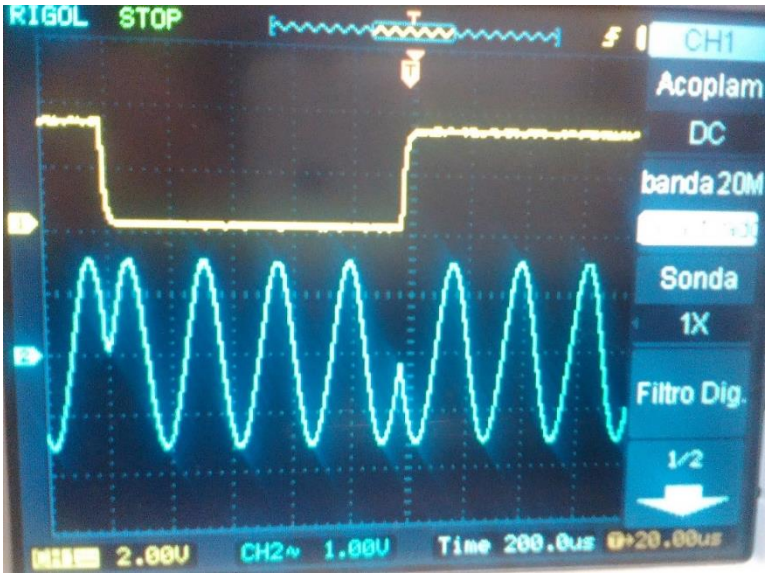

Figura 26 Resultados Modulación PSK Transmisión

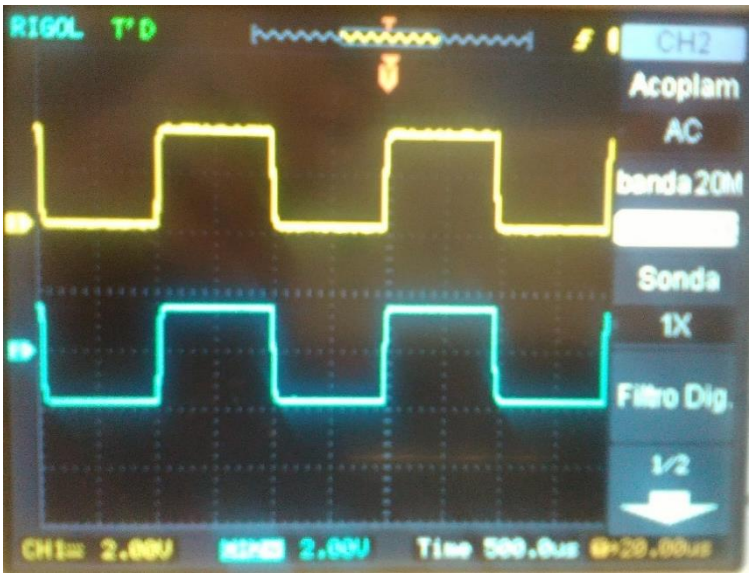

Figura 27 Resultados Modulación PSK Recepción

4.4. Resultados TDM

En la multiplexacion por tiempo en la figura 31 se pueden observar las dos señales de entrada, una señal sinosoidal y una señal triangular con una frecuencia de 512Hz cada una. En la figura 32 se observa la multiplexación de tiempo la cual se hace con una frecuencia de 8000Hz y en la figura 33 se observa las dos señales por separado multiplexadas en el tiempo.

Figura 28. Señales de entrada para la Multiplexacion de tiempo

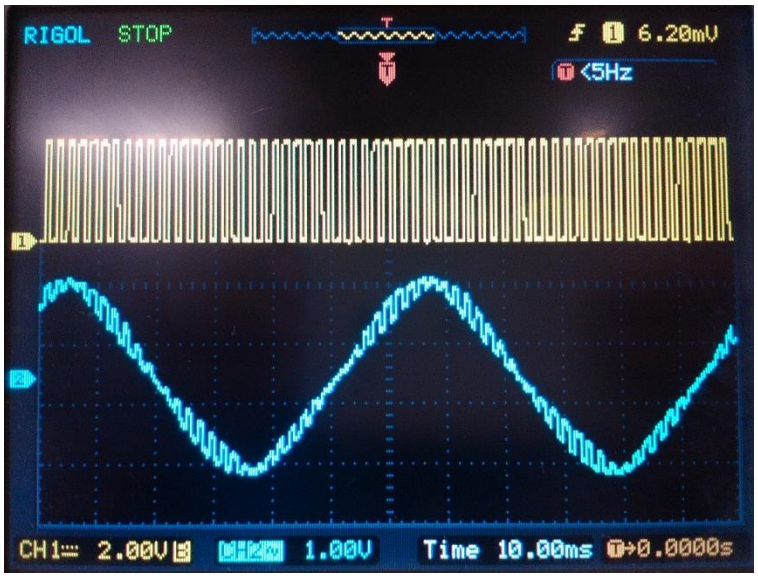

Figura 29 Respuesta de la Multiplexacion de Tiempo

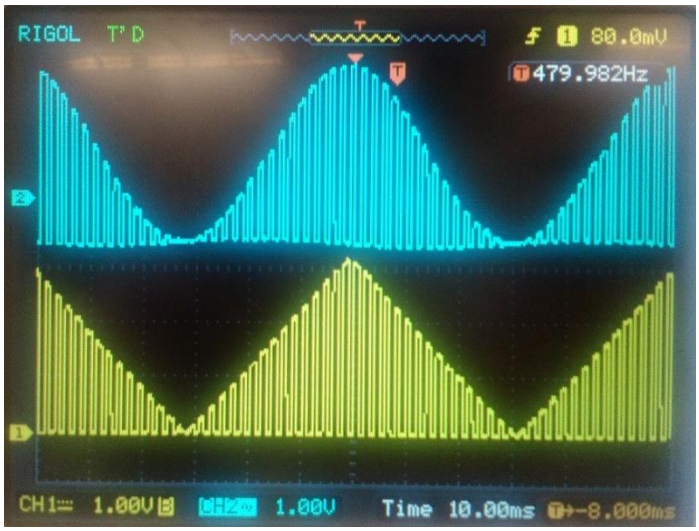

Figura 30. Señales de Salida de la Multiplexacion de Tiempo

4.5. Resultados Modulación QAM

En la modulación QAM en la figura 34 se observan los estados con una frecuencia de 512Hz del generador de estados y para cada uno de ellos su respectiva señal en cuadratura:

ESTADO (BINARIO)	**SEÑAL MODULADA GENERADA**
[00]	Se observa la señal con amplitud mayor y un desfase de 0°.
[01]	Se observa la señal con amplitud mayor y un desfase de 180°
[10]	Se observa la señal con amplitud menor y un desfase de 0°.
[11]	Se observa la señal con amplitud menor y un desfase de 180°.

La señal de modulada de trasmisión tiene una frecuencia de1200Hz en todas sus cuadraturas.

Figura 31 Respuesta Modulación QAM

5. CONCLUSIONES

Las FPGAS han demostrado que son dispositivos programables con un alto rendimiento ya que permite implementar grandes sistemas digitales esta facilita al programador tener una gran versatilidad al momento de desarrollar cualquier sistema digital.

La implementación de las modulaciones digitales en la FPGA SpartanXC3S700A nos indica la factibilidad técnica y operativa para la realización de las prácticas demostrativas referente a las modulaciones digitales.

Al desarrollar la modulación ASK se observó la respuesta de acuerdo a la teoría donde el cero (0) binario nos da una amplitud diferente respecto al uno (1) binario. En uno de los dígitos binarios representa la señal portadora. Para este caso frecuencia y fase se mantienen constante.

En las pruebas realizadas se observó que la modulación FSK cumple lo que está estipulado en la teoría la amplitud no se afecta siempre y cuando se mantenga un parámetro de portadora estable; donde la información digital está contenida en la variable frecuencia de la señal senoidal.

Al realizar la modulación PSK se observó la respuesta de acuerdo a lo que nos brinda la teoría donde quien transmite solo debe cambiar la fase de la señal para representar cada bit, donde la señal de entrada es una señal de clock y la señal modulada de transmisión cambia su fase 90 grados para un estado lógico (1) de la señal de entrada.

En las pruebas realizadas en la modulación TDM se observó la respuesta de acuerdo a la teoría en donde por un mismo canal podemos enviar varias señales. En este caso se envía una señal senoidal y una señal triangular las que son multiplexadas en el tiempo.

Al desarrollar la modulación QAM se observó que para cada estado se obtiene su respectiva señal en cuadratura donde para el estado [00] se observa la amplitud mayor con un desfase de 0 grados, para el estado [01] se observa la amplitud mayor y un desfase de 90 grados, para el estado [10] se observa la amplitud menor y un desfase de 0 grados y para el estado [11] se observa la amplitud menor y un desfase de 90 grados.

Es importante resaltar la buena utilización de las tarjetas Spartan XC3S700A en este proyecto dado que ya no son usadas ni pedidas en los laboratorios de digitales o de arquitectura. Debido a que la facultad de electrónica cuenta con versiones nuevas de las FPGAS. En este momento la facultad cuenta con las tarjetas DEO, DEO-Nano y las SOC, FPGAS más modernas y con software también actualizado.

BIBLIOGRAFÍA

[1] «Introducción a las telecomunicaciones,» [En línea]. Available: https://teoriadelastelecomunicaciones.wordpress.com/unidad1/. [Último acceso: 21 02 2016].

[2] «COMUNICACIONES DIGITALES,» [En línea]. Available: www.fi-b.unam.mx/Profesores/AbelHerrera/.../descargas/capitulo1.doc. [Último acceso: 29 04 2016].

[3] B. A. Forouzan, Transmicion de datos y Redes de Comunicacion, Madrid: mc graw hill, 2002.

[4] «Modulacion ASK,» [En línea]. Available: http://www.ecured.cu/Modulaci%C3%B3n_ASK. [Último acceso: 21 02 2016].

[5] «MODULACIÓN DIGITAL :FSK – PSK - QAM,» [En línea]. Available: http://www.electronicafacil.net/tutoriales/MODULACION-DIGITAL-FSK-PSK-QAM.php. [Último acceso: 21 02 2016].

[6] u. p. d. Madrid, «Comunicaciones Digitales,» [En línea]. Available: http://www.gr.ssr.upm.es/docencia/grado/csat/material/CSA08-5-ModulacionesDigitales_2p. [Último acceso: 22 02 2016].

[7] «telecomunicaciones,» [En línea]. Available: http://yuricodelaotelecomunicaciones.blogspot.com.co/2012/03/2-multiplexacion-por-division-en-tiempo.html. [Último acceso: 22 02 2016].

[8] XILINX, «XILINX,» [En línea]. Available: http://www.xilinx.com/training/fpga/fpga-field-programmable-gate-array.htm. [Último acceso: 26 04 2016].

[9] «National Instruments,» [En línea]. Available: http://www.ni.com/white-paper/6983/es/. [Último acceso: 29 04 2016].

[10] «XILINX,» [En línea]. Available: http://www.xilinx.com/products/design-tools/ise-design-suite.html. [Último acceso: 03 05 2016].

[11] «Verilog Resources,» [En línea]. Available: http://www.verilog.com/. [Último acceso: 03 05 2016].

[12] J. WILEY, FPGA PROTOTYPING BY VERILOG EXAMPLES, Cleveland State University: INC, 2008.

[13] «Semillero ADT,» [En línea]. Available: https://sites.google.com/site/semilleroadt/. [Último acceso: 12 03 2016].

ANEXO 1: Divfreq (Divisor de frecuencia) algoritmo de programación.

```
module divfreq(clk,in,out); //entradas y salidas

input clk;
input [31:0]in;
output out;

reg [31:0]contador=32'b0;  // registros
reg toggle=1'b0;

assign out=toggle; // asignacion de salida

always@(posedge clk)
begin
if(contador[31:0]>=in[31:0])contador[31:0]<=32'd0;
else contador[31:0]<=contador[31:0]+1'b1;         // contador aumentando de 1 en 1

end

always@(posedge clk)
begin
if(contador[31:0]>=in[31:0]) toggle<=~toggle; // cambio de la salida
else toggle<=toggle;

end
```

ANEXO 2: Rom_sen (Memoria) algoritmo de programación.

```
module rom_sen(address,out
    );

input [6:0]address;
output [7:0]out;

reg [7:0]datos;

assign out[7:0]=datos[7:0];

always@(*)
begin
case(address[6:0])

//valores del seno para cada dirección (127 muestreos)

7'd 0 :datos = 8'd127 ;
7'd 1 :datos = 8'd133 ;
7'd 2 :datos = 8'd140 ;
7'd 3 :datos = 8'd146 ;
7'd 4 :datos = 8'd152 ;
7'd 5 :datos = 8'd158 ;
7'd 6 :datos = 8'd164 ;
7'd 7 :datos = 8'd170 ;
7'd 8 :datos = 8'd176 ;
7'd 9 :datos = 8'd182 ;
7'd 10:datos = 8'd187 ;
7'd 11:datos = 8'd193 ;
7'd 12:datos = 8'd198 ;
7'd 13:datos = 8'd203 ;
7'd 14:datos = 8'd208 ;
7'd 15:datos = 8'd213 ;
7'd 16:datos = 8'd217 ;
7'd 17:datos = 8'd222 ;
7'd 18:datos = 8'd226 ;
7'd 19:datos = 8'd230 ;
7'd 20:datos = 8'd233 ;
7'd 21:datos = 8'd236 ;
7'd 22:datos = 8'd240 ;
```

```
7'd 23:datos = 8'd242 ;
7'd 24:datos = 8'd245 ;
7'd 25:datos = 8'd247 ;
7'd 26:datos = 8'd249 ;
7'd 27:datos = 8'd251 ;
7'd 28:datos = 8'd252 ;
7'd 29:datos = 8'd253 ;
7'd 30:datos = 8'd254 ;
7'd 31:datos = 8'd254 ;
7'd 32:datos = 8'd254 ;
7'd 33:datos = 8'd254 ;
7'd 34:datos = 8'd253 ;
7'd 35:datos = 8'd252 ;
7'd 36:datos = 8'd251 ;
7'd 37:datos = 8'd250 ;
7'd 38:datos = 8'd248 ;
7'd 39:datos = 8'd246 ;
7'd 40:datos = 8'd244 ;
7'd 41:datos = 8'd241 ;
7'd 42:datos = 8'd238 ;
7'd 43:datos = 8'd235 ;
7'd 44:datos = 8'd231 ;
7'd 45:datos = 8'd228 ;
7'd 46:datos = 8'd224 ;
7'd 47:datos = 8'd220 ;
7'd 48:datos = 8'd215 ;
7'd 49:datos = 8'd210 ;
7'd 50:datos = 8'd206 ;
7'd 51:datos = 8'd201 ;
7'd 52:datos = 8'd195 ;
7'd 53:datos = 8'd190 ;
7'd 54:datos = 8'd185 ;
7'd 55:datos = 8'd179 ;
7'd 56:datos = 8'd173 ;
7'd 57:datos = 8'd167 ;
7'd 58:datos = 8'd161 ;
7'd 59:datos = 8'd155 ;
7'd 60:datos = 8'd149 ;
7'd 61:datos = 8'd143 ;
7'd 62:datos = 8'd136 ;
7'd 63:datos = 8'd130 ;
```

```
7'd 64:datos = 8'd124 ;
7'd 65:datos = 8'd118 ;
7'd 66:datos = 8'd111 ;
7'd 67:datos = 8'd105 ;
7'd 68:datos = 8'd99;
7'd 69:datos = 8'd93;
7'd 70:datos = 8'd87;
7'd 71:datos = 8'd81;
7'd 72:datos = 8'd75;
7'd 73:datos = 8'd69;
7'd 74:datos = 8'd64;
7'd 75:datos = 8'd59;
7'd 76:datos = 8'd53;
7'd 77:datos = 8'd48;
7'd 78:datos = 8'd44;
7'd 79:datos = 8'd39;
7'd 80:datos = 8'd34;
7'd 81:datos = 8'd30;
7'd 82:datos = 8'd26;
7'd 83:datos = 8'd23;
7'd 84:datos = 8'd19;
7'd 85:datos = 8'd16;
7'd 86:datos = 8'd13;
7'd 87:datos = 8'd10;
7'd 88:datos = 8'd8 ;
7'd 89:datos = 8'd6 ;
7'd 90:datos = 8'd4 ;
7'd 91:datos = 8'd3 ;
7'd 92:datos = 8'd2 ;
7'd 93:datos = 8'd1 ;
7'd 94:datos = 8'd0 ;
7'd 95:datos = 8'd0 ;
7'd 96:datos = 8'd0 ;
7'd 97:datos = 8'd0 ;
7'd 98:datos = 8'd1 ;
7'd 99:datos = 8'd2 ;
7'd 100 :datos = 8'd3 ;
7'd 101 :datos = 8'd5 ;
7'd 102 :datos = 8'd7 ;
7'd 103 :datos = 8'd9 ;
7'd 104 :datos = 8'd12;
```

```
7'd 105 :datos = 8'd14;
7'd 106 :datos = 8'd18;
7'd 107 :datos = 8'd21;
7'd 108 :datos = 8'd24;
7'd 109 :datos = 8'd28;
7'd 110 :datos = 8'd32;
7'd 111 :datos = 8'd37;
7'd 112 :datos = 8'd41;
7'd 113 :datos = 8'd46;
7'd 114 :datos = 8'd51;
7'd 115 :datos = 8'd56;
7'd 116 :datos = 8'd61;
7'd 117 :datos = 8'd67;
7'd 118 :datos = 8'd72;
7'd 119 :datos = 8'd78;
7'd 120 :datos = 8'd84;
7'd 121 :datos = 8'd90;
7'd 122 :datos = 8'd96;
7'd 123 :datos = 8'd102 ;
7'd 124 :datos = 8'd108 ;
7'd 125 :datos = 8'd114 ;
7'd 126 :datos = 8'd121 ;
7'd 127 :datos = 8'd127 ;
endcase
end
endmodule
```

ANEXO 3: Gen_sen (Generador de Señales) algoritmo de programación.

```
module gen_sen(clk,out
    );

input clk;
output [7:0]out;

reg [6:0]contador=7'd0;

always@(posedge clk)
begin
contador[6:0]<=contador[6:0]+1'b1;
end
rom_sen seno(.address(contador[6:0]),.out(out[7:0]));

endmodule
```

ANEXO 4: Dac (conversor Digital/Analogo) algoritmo de programación.

```
define MSBI 7

module dac(DACout, DACin, clk, Reset
    );

output DACout; // This is the average output that feeds low pass filter
reg DACout; // for optimum performance, ensure that this ff is in IOB

input [`MSBI:0] DACin; // DAC input (excess 2**MSBI)
input clk;
input Reset;

reg [`MSBI+2:0] DeltaAdder; // Output of Delta adder
reg [`MSBI+2:0] SigmaAdder; // Output of Sigma adder
reg [`MSBI+2:0] SigmaLatch; // Latches output of Sigma adder
reg [`MSBI+2:0] DeltaB; // B input of Delta adder

always @(SigmaLatch)
begin
 DeltaB = {SigmaLatch[`MSBI+2], SigmaLatch[`MSBI+2]} << (`MSBI+1);
end

always @(DACin or DeltaB)
begin
 DeltaAdder = DACin + DeltaB;
end

always @(DeltaAdder or SigmaLatch)
begin
 SigmaAdder = DeltaAdder + SigmaLatch;
end

always @(posedge clk or posedge Reset)
begin
if(Reset)
begin
 SigmaLatch <= #1 1'b1 << (`MSBI+1);
 DACout <= #1 1'b0;
end
```

```
else
begin
 SigmaLatch <= #1 SigmaAdder;
 DACout <= #1 SigmaLatch[`MSBI+2];
end
end
endmodule
```

ANEXO 5: Comp (comparado recepción) algoritmo de programación.

```
module comp(clk,in_1,out
    );

input clk;
input [7:0]in_1;
output out;

wire [7:0]in_2;
reg [6:0]contador1=8'd0;

always@(posedge clk)
begin
contador1[6:0]<=contador1[6:0]+1'b1;
end

rom_sen seno_com(.address(contador1[6:0]),.out(in_2[7:0]));

assign out=(in_1==in_2)?1'd1:1'd0;

endmodule
```

ANEXO 6: ASK (Modulación por desplazamiento de Amplitud) algoritmo de programación.

```
module ask(CLK_50MHZ,LED,J18_IO
    );
input CLK_50MHZ;
output [3:0]J18_IO;
output [7:0]LED;

wire clk_div1;
wire clk_div2;

wire [7:0]bus_duty1;
wire [7:0]bus_duty2;
wire out_pwm=clk_div2;

wire [7:0]bus_sen;
wire out_seno;
wire out_rx;

divfreq divisor1(.clk(CLK_50MHZ),.in(32'd45),.out(clk_div1));
divfreq divisor2(.clk(CLK_50MHZ),.in(32'd48828),.out(clk_div2));

gen_sen seno(.clk(clk_div1),.out1(bus_duty1[7:0]),.out2(bus_duty2[7:0]));
dac
dac_seno(.clk(CLK_50MHZ),.DACin(bus_sen[7:0]),.DACout(out_seno),.Reset(1'b0
));
comp rx(.clk(clk_div1),.in_1(bus_sen[7:0]),.out(out_rx));

assign LED[0]=1'd0;
assign LED[1]=1'd0;
assign LED[7:2]=6'd0;
assign J18_IO[0]=out_pwm;
assign J18_IO[1]=out_seno;
assign J18_IO[2]=out_rx;
assign J18_IO[3]=1'd0;
assign
bus_sen[7:0]=(out_pwm==1)?{bus_duty1[7],bus_duty1[6:0]}:{bus_duty2[7],bus_dut
y2[6:0]};
endmodule
```

Anexo 7: FSK (Modulación por desplazamiento de frecuencia) algoritmo de programación.

```
module fsk(CLK_50MHZ,LED,J18_IO
   );
input CLK_50MHZ;
output [3:0]J18_IO;
output [7:0]LED;


wire clk_div1;
wire clk_div2;

wire [7:0]bus_duty;
wire [7:0]bus_sen={bus_duty[7],bus_duty[6:0]};
wire out_seno;

wire out_pwm=clk_div2;
wire [31:0]frec;


divfreq divisor1(.clk(CLK_50MHZ),.in(frec[31:0]),.out(clk_div1));
divfreq divisor2(.clk(CLK_50MHZ),.in(32'd48828),.out(clk_div2));

gen_sen seno(.clk(clk_div1),.out(bus_duty[7:0]));

dacdac_seno(.clk(CLK_50MHZ),.DACin(bus_sen[7:0]),.DACout(out_seno),.Reset(
1'b0));

comp rx(.clk1(clk_div1),.clk2(clk_div2),.in(bus_sen[7:0]),.out(out_rx));

assign LED[0]=out_rx;
assign LED[7:1]=7'd0;
assign J18_IO[0]=out_pwm;
assign J18_IO[1]=out_seno;
assign J18_IO[2]=out_rx;
assign J18_IO[3]=1'd0;

assign frec[31:0]=(out_pwm==1)?32'd23:32'd46;
endmodule
```

ANEXO 8: PSK (Modulación por desplazamiento de fase) algoritmo de programación.

```
module psk(CLK_50MHZ,LED,J18_IO
   );

input CLK_50MHZ;
output [3:0]J18_IO;
output [7:0]LED;

wire clk_div1;
wire clk_div2;

wire [7:0]bus_duty;
wire out_pwm=clk_div2;

wire [7:0]bus_sen={bus_duty[7],bus_duty[6:0]};
wire out_seno;
wire cond;

divfreq divisor1(.clk(CLK_50MHZ),.in(32'd46),.out(clk_div1));
divfreq divisor2(.clk(CLK_50MHZ),.in(32'd48828),.out(clk_div2));

gen_sen seno(.clk(clk_div1),.sel(cond),.out(bus_duty[7:0]));

dac
 dac_seno(.clk(CLK_50MHZ),.DACin(bus_sen[7:0]),.DACout(out_seno),.Reset(1'b
 0));

comp rx(.clk(clk_div1),.in_1(bus_sen[7:0]),.out(out_rx));

assign LED[0]=1'd0;
assign LED[1]=1'd0;
assign LED[7:2]=6'd0;
assign J18_IO[0]=out_pwm;
assign J18_IO[1]=out_seno;
assign J18_IO[2]=out_rx;
assign J18_IO[3]=1'd0;
assign cond=(out_pwm==1)?1'd1:1'd0;
endmodule
```

ANEXO 9: TDM (Multiplexación por División de Tiempo) algoritmo de programación.

```
module tdm(CLK_50MHZ,LED,J18_IO,SW
   );

input CLK_50MHZ;
input [3:0]SW;
output [3:0]J18_IO;
output [7:0]LED;

wire clk_div1;
wire clk_div2;

wire [7:0]bus_duty1;
wire [7:0]bus_duty2;

wire out_pwm=clk_div2;

wire out_seno;
wire out_tria;

reg led_2;
reg out_seno_reg;
reg out_tria_reg;

divfreq divisor1(.clk(CLK_50MHZ),.in(32'd12207),.out(clk_div1));
divfreq divisor2(.clk(CLK_50MHZ),.in(32'd48828),.out(clk_div2));

gen_sen seno(.clk(clk_div1),.out1(bus_duty1[7:0]),.out2(bus_duty2[7:0]));

dac
dac_seno(.clk(CLK_50MHZ),.DACin(bus_duty1[7:0]),.DACout(out_seno),.Reset(1'
b0));
dac
dac_tri(.clk(CLK_50MHZ),.DACin(bus_duty2[7:0]),.DACout(out_tria),.Reset(1'b0));

always@(SW[0] or out_pwm)
```

```
if(SW[0]==1'b1)
begin
  led_2<=1'd1;
  if(out_pwm==1'b1)
  begin
  out_seno_reg<=out_seno;
  out_tria_reg<=1'b0;
  end
  else
  begin
  out_seno_reg<=1'b0;
  out_tria_reg<=out_tria;
  end
end
else
begin
 led_2<=1'd0;
 out_seno_reg<=out_seno;
 out_tria_reg<=out_tria;
end


assign LED[0]=led_2;
assign LED[1]=1'd0;
assign LED[7:2]=6'd0;
assign J18_IO[0]=out_seno_reg;
assign J18_IO[1]=out_tria_reg;
assign J18_IO[2]=(out_pwm==1)?out_seno:out_tria;
assign J18_IO[3]=1'd0;

endmodule
```

ANEXO 10: QAM (Modulación de Amplitud en cuadratura) algoritmo de programación.

```
module qam(CLK_50MHZ,LED,J18_IO
   );
input CLK_50MHZ;
output [3:0]J18_IO;
output [7:0]LED;

wire clk_div1;
wire clk_div2;

wire [7:0]bus_duty1;
wire [7:0]bus_duty2;

wire out_seno;
reg [7:0]bus_sen;

wire [1:0]est;
reg sel;

wire in;
wire [1:0]out_rx;
wire out;

divfreq divisor1(.clk(CLK_50MHZ),.in(32'd46),.out(clk_div1));
divfreq divisor2(.clk(CLK_50MHZ),.in(32'd48828),.out(clk_div2));
divfreq divisor3(.clk(CLK_50MHZ),.in(32'd48828),.out(clk_div3));

est gen_est(.clk(clk_div2),.out(est[1:0]));

gen_sen seno(.clk(clk_div1),.sel(sel),.out1(bus_duty1[7:0]),.out2(bus_duty2[7:0]));

dac
 dac_seno(.clk(CLK_50MHZ),.DACin(bus_sen[7:0]),.DACout(out_seno),.Reset(1'b
 0));

vec_out senal_in(.clk(clk_div3),.in(est[1:0]),.out(in));
vec_out senal_out(.clk(clk_div3),.in(out_rx[1:0]),.out(out));
```

```
always@(*)
begin
 case(est[0])

 1'd0: sel=1'd0 ;
 1'd1:sel=1'd1 ;

 endcase
end

always@(*)
begin
 case(est[1])
 1'd0:  bus_sen=bus_duty1;
 1'd1: bus_sen=bus_duty2;
 endcase
end

comp rx(.clk(clk_div1),.in_1(bus_sen[7:0]),.out(out_rx));

assign LED[1:0]=est[1:0];
assign LED[3:2]=out_rx[1:0];
assign LED[7:4]=4'd0;

assign J18_IO[0]=in;
assign J18_IO[1]=out_seno;
assign J18_IO[2]=out;
assign J18_IO[3]=1'd0;

endmodule
```