UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DEPARTAMENTO DE INGENIERÍA ELECTRÓNICA

DISEÑO E IMPLEMENTACIÓN DE UN ENTORNO PARA LA EXPERIMENTACIÓN EN LA TRANSMISIÓN DE DATOS MEDIANTE LEDS DE ILUMINACIÓN

Autor: Pau Salvador Llàcer

Tutores: Javier Valls Coquillat y Maria José Canet Subiela

Trabajo Fin de Máster presentado en el Departamento de Ingeniería Electrónica de la Universitat Politècnica de València para la obtención del Título de Máster Universitario en Ingeniería de Sistemas Electrónicos

Curso 2019-20

Valencia, septiembre de 2020

Resumen

En el presente trabajo se realiza un entorno para la experimentación que permite transmitir y capturar señales con un ancho de banda de hasta 30 MHz a través de LEDs de iluminación comerciales. El sistema se implementa en un dispositivo FPGA que dispone de DACs y ADCs de alta velocidad y se controla desde Matlab. Se implementa una librería de rutinas en Matlab para excitar las bombillas LED con diferentes modulaciones y para sincronizar y demodular la señal recibida por el fotodiodo. Se realizan experimentos de transmisión de diferentes modulaciones.

Resum

En el present treball es realitza un entorn per a l'experimentació que permet transmetre i capturar senyals amb un ample de banda de fins a 30MHz a través de LEDs d'il·luminació comercials. El sistema s'implementa en un dispositiu FPGA que disposa de DACs i ADCs d'alta velocitat i es controla des de Matlab. S'implementa una llibreria de rutines en Matlab per excitar les bombetes LED amb diferents modulacions i per a sincronitzar i desmodular el senyal rebut pel fotodíode. Es realitzen experiments de transmissió de diferents modulacions.

Abstract

In the present work an environment for experimentation that allows transmitting and capturing signals with a bandwidth of up to 30 MHz through commercial lighting LEDs is developed. The system is implemented in an FPGA device that has high-speed DACs and ADCs and is controlled from Matlab. A library of routines is implemented in Matlab to excite the LED bulbs with different modulations and to synchronize and demodulate the signal received by the photodiode. Transmission experiments of different modulations are performed.

Índice

Índice de figuras

Índice de tablas

Capítulo 1. Introducción y objetivos

1.1 Introducción

En las últimas décadas ha crecido exponencialmente la demanda de comunicaciones inalámbricas. Se prevé que en los próximos años, la mitad del tráfico se genere desde y para dispositivos situados en el interior de edificios ([1]). Debido a esto, las redes inalámbricas de interiores pueden llegar a quedar congestionadas dado que disponen de un ancho de banda limitado.

Como alternativa a estas tecnologías inalámbricas, basadas comúnmente en las normas del IEEE 802.11, han aparecido las comunicaciones ópticas inalámbricas (OWC) que ofrecen la posibilidad de la creación de pequeñas celdas de comunicación en las redes. Las OWC ofrecen ventajas respecto a las redes inalámbricas habituales, ya que son inmunes a interferencias electromagnéticas. Por el contrario, al basarse en luz, las OWC requieren de visión directa entre el transmisor y el receptor, lo que obliga a tener un transmisor en cada habitación en la que se desee tener conexión.

1.2 Objetivos

El objetivo de este TFM es el diseño e implementación de un entorno para la experimentación que permita transmitir y capturar señales con un ancho de banda de hasta 30 MHz a través de LEDs de iluminación comerciales.

Los objetivos concretos se enumeran a continuación:

- Codificar con HDL, verificar e implementar en un dispositivo FPGA los bloques de comunicación PC-FPGA, de gestión de memoria y de generación y captura con los DACs y ADCs.
- Desarrollo de una librería de rutinas de Matlab para el control del sistema desde el PC.
- Puesta en marcha del modulador del LED y del fotodiodo receptor.
- Experimentación con modulaciones digitales QPSK y M-QAM de diferentes tasas de bit.

1.3 Antecedentes

Dentro de las OWC se pueden distinguir dos categorías diferentes: las basadas en rayos directivos (habitualmente en la banda de infrarrojos) y las basadas en luz visible generada por LEDs de iluminación (VLC). Las tecnologías basadas en rayos directivos requieren de una infraestructura independiente y se basan en tecnologías maduras ([1]), mientras que las segundas pueden ser implementadas junto con el sistema de iluminación y en la actualidad están siendo desarrolladas por numerosos grupos de investigación y han aparecido, también, algunas soluciones comerciales ([2]).

Basado en la tecnología VLC se plantea crear lo que se ha denominado Li-Fi ([2]). Dicha tecnología crea una red de comunicaciones sobre la infraestructura de iluminación basadas en LED, lo que permite un ahorro energético y, además, facilita el despliegue. En la mayoría de los trabajos publicados se usa una red VLC para el enlace descendiente, mientras que para el enlace ascendiente se utiliza típicamente tecnología Wi-Fi.

Los LED de iluminación suelen implementarse con un chip LED azul con un recubrimiento de fósforo que genera luz blanca. El fósforo tiene una respuesta lenta lo que limita el ancho de banda a unos pocos MHz. A pesar de esto, en [3] se alcanzan 262 Mbit/s a una distancia de 5 m utilizando la modulación NRZ-OOK combinada con una compensación de la respuesta del LED y un filtro azul para aumentar el ancho de banda. En [4], por otro lado, se consiguen mayores tasas de

transmisión, llegando a los 2Gbit/s a 1.5 m utilizando la modulación OFDM con 256 portadoras. Otras alternativas proponen usar multiplexación por longitud de onda utilizando un LED RGB como en [5], donde se alcanzan 2.2Gbit/s a una distancia de 30 cm.

1.4 Estructura del documento

El documento está estructurado de la siguiente forma: En el punto 2 se detallan las herramientas utilizadas y la metodología de trabajo utilizadas durante el trabajo. En el punto 3, se desarrolla el diseño y la implementación del entorno desarrollado. En el punto 4, se presentan los resultados obtenidos con el entorno de desarrollo. En el punto 5, se describen las líneas futuras de trabajo. Por último, en el punto 6 se presentan las conclusiones.

Capítulo 2. Herramientas y metodología de trabajo

2.1 Diseño de sistemas basados en FPGA

Las denominadas FPGA (Field-Programmable Gate Array) son dispositivos programables que contienen bloques de lógica cuya interconexión y funcionalidad puede ser configurada en el momento, mediante lenguajes de descripción hardware (HDL). La lógica programable puede implementar funciones sencillas como las llevadas a cabo por puertas lógicas o sistemas complejos de procesado de señal.

Las FPGAs se utilizan en aplicaciones similares a los ASIC, sin embargo, las FPGAs son más lentas y tienen un mayor consumo de energía. A pesar de esto, las FPGAs tienen las ventajas de ser reprogramables, lo que permite mucha flexibilidad frente a los ASIC que tienen un diseño fijo. En FPGA los costes y el tiempo de desarrollo son mucho menores.

A las FPGAs se les ha ido añadiendo elementos hardware como memorias, multiplicadores o bloques DSP que permiten la implementación de ciertos algoritmos con mayor facilidad y velocidad.

A la hora de plantear el diseño de un sistema basado en FPGA, se debe tener en cuenta los elementos hardware con los que cuenta el dispositivo con el que se está trabajando, para adaptar el diseño a estos elementos y poder conseguir el mayor rendimiento.

Otro elemento importante a la hora de afrontar un diseño en este tipo de sistemas son los formatos de datos que pueden ser utilizados en cada uno de los componentes y transportados por los diferentes buses. En los dispositivos FPGA se suelen utilizar los formatos de precisión finita con coma fija, lo que requiere un meticuloso análisis de los parámetros y señales utilizadas durante el procesado. Se debe ser muy cuidadoso a la hora de elegir el formato de datos s, ya que de esto dependerá la calidad final de la señal. No escoger adecuadamente el formato puede provocar diversos errores. Por un lado, puede producirse saturación (o wrap) si no se cubre todo el rango dinámico de la señal con el número de bits asignados o puede producir que se pierda la precisión necesaria para representar valores muy bajos o cambios pequeños entre diferentes valores de amplitud (conocido como ruido de cuantificación). Además, si se utilizan muchos bits para la representación de la señal provoca un aumento en el coste hardware de las operaciones (número de recursos hardware) y aumenta también el tiempo de computación de los operadores, disminuyendo la frecuencia máxima de operación del sistema.

2.2 Placa de desarrollo utilizada

Para la realización del prototipo se utiliza la placa de desarrollo DE2-115 de Terasic ([6]) que cuenta con la FPGA Cyclone IV EP4CE115F29C7. Esta placa de desarrollo cuenta con variedad de periféricos (Figura 1), como dos conectores Gigabit Ethernet (GbE), un conector HSMC (High-Speed Mezzanine Card), una pantalla LCD de 16x2 caracteres, diversos pulsadores, interruptores, leds y displays 7 segmentos, entre muchos otros.

De los periféricos disponibles, en el diseño presentado en este documento se utilizan los siguientes:

- Pantalla LCD 16x2, para mostrar información de control.
- Una de las conexiones RJ45, para la comunicación de la placa con el PC.
- Pulsadores e interruptores, como señales de control.
- Conector HSMC, para la comunicación con los conversores ADC y DAC.

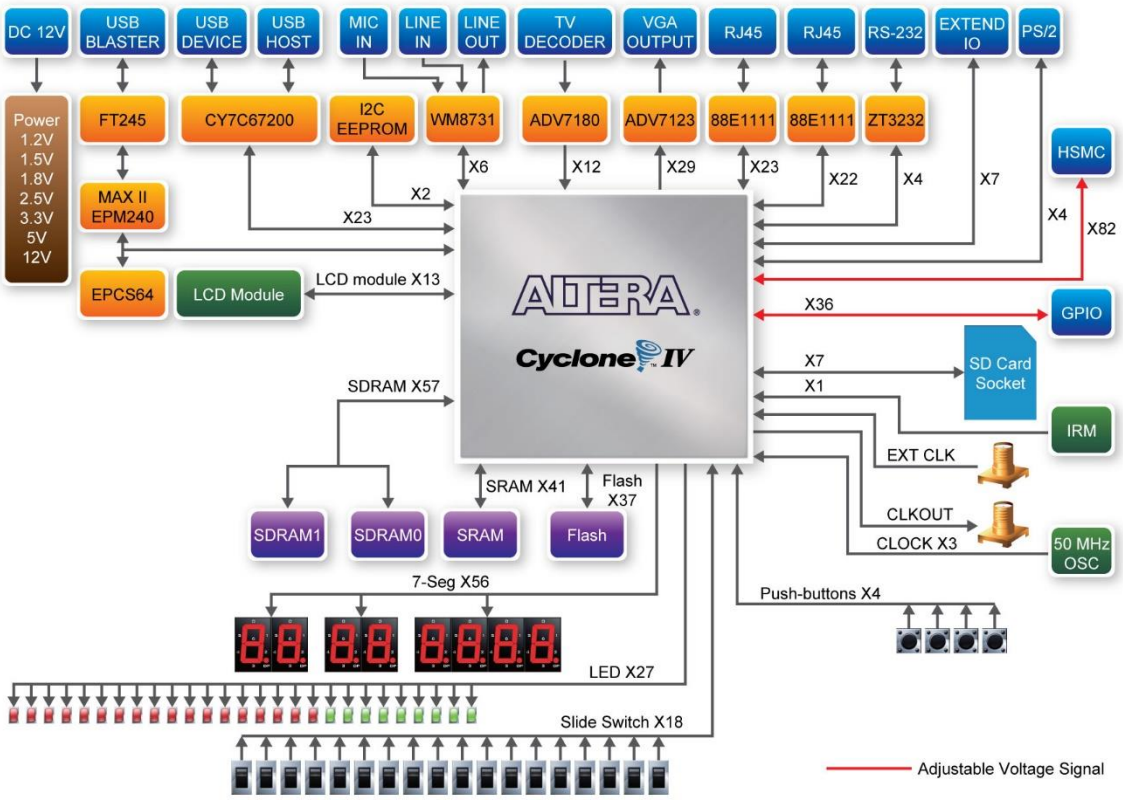

Figura 1. Periféricos DE2-115. Fuente: www.intel.com

La FPGA contenida en la placa (Cyclone IV EP4CE115F29C7) cuenta con 114 480 elementos lógicos (LEs), 3.8Mbits de memoria RAM, 266 multiplicadores de 18x18 bits, 4 PLLs de propósito general y 518 puertos de E/S.

2.2.1 Conversores AD y DA

Para la implementación del prototipo es necesario la utilización de conversores analógico-digital (ADC) y digital-analógico (DAC) para poder alimentar al led que ha de transmitir la información y poder recuperar la información recibida del fotodiodo.

Los conversores utilizados son los disponibles en la placa THDB ADA también de Terasic ([7]), dicha placa cuenta con dos conversores ADC AD9248 y dos conversores DAC DA9767. Dichos conversores trabajan con 2V de pico a pico y tienen una resolución de 14 bits. Los DAC pueden funcionar hasta 125MHz y los ADC hasta 65MHz. En el prototipo implementado se utilizan los DAC a 120MHz y los ADC a 60MHz para simplificar la implementación.

En la Figura 2 se puede ver la placa de desarrollo utilizada junto a los conversores.

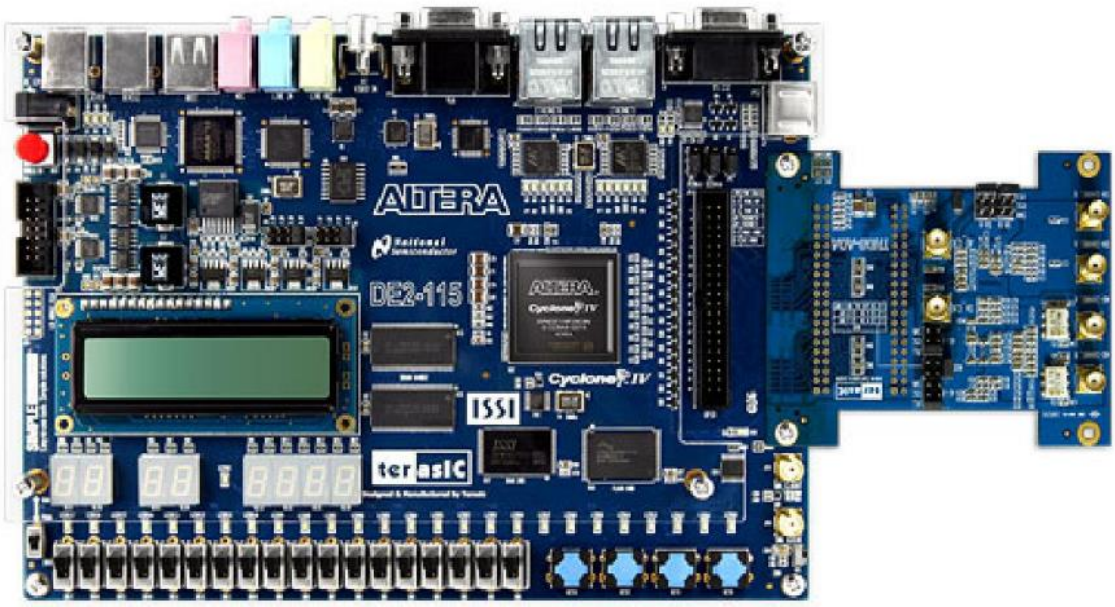

Figura 2. Tarjeta DE2-115 con los conversores THDB ADA

2.3 Software utilizado

Durante el proceso de diseño e implementación del prototipo presentado, se ha utilizado distinto software:

- **Quartus Prime**. Es un software propietario de Intel FPGA que permite la programación de sus dispositivos FPGA. Además del propio software se han utilizado distintas herramientas incluidas en el paquete de Quartus Prime, como el analizador lógico Signal Tab Logic Analyzer.
- **ModelSim**. Software de simulación de HDL utilizado para la depuración y verificación del correcto funcionamiento del procesado aplicado en el prototipo.
- **Matlab**. Software matemático utilizado para el control del sistema e intercambio de datos mediante una red TCP/IP. Además, se ha utilizado para realizar la experimentación y obtener los resultados. Del paquete de Matlab también se utiliza Simulink, utilizado en la simulación del procesado de señal, generando modelos para comparar los resultados con los simulados.

2.4 Metodología de trabajo

El desarrollo de este trabajo se puede dividir en tres grandes bloques, por una parte, la implementación hardware, que incluye el diseño de circuitos de control para gestionar la recepción y envío de datos, las memorias y los conversores y el procesado de señal. Por otra parte, el desarrollo de una librería de funciones de Matlab para implementar diferentes modulaciones y comunicarse con el hardware, y, por último, la experimentación con el entorno desarrollado.

En la parte de procesado de señal, la metodología seguida ha sido la siguiente:

1. Diseño del procesado en Simulink con formato de datos en coma flotante. Denominado Golden Model.
2. Conversión del diseño a un modelo en precisión finita en Simulink y validación comparándolo con el Golden Model.
3. Implementación del diseño en HDL y comprobación mediante herramientas de test que el resultado es el mismo al obtenido en el punto 2.

4. Programación del dispositivo FPGA y comprobación visual del correcto funcionamiento mediante analizadores lógicos u osciloscopios (mediante un DAC).

Capítulo 3. Diseño del prototipo

3.1 Especificaciones técnicas del prototipo

En el presente trabajo se desea realizar un prototipo capaz de transmitir una modulación MQAM con un ancho de banda máximo de 30 MHz a través de la luz visible controlado desde un PC con Matlab. Para ello se necesita algún tipo de conexión entre el PC y el sistema de transmisión. Esta conexión se desea realizar mediante una red TCP/IP, ya que posibilita una alta velocidad de transmisión.

Por lo que respecta a la parte hardware, de desea poder transmitir un ancho de manda de, al menos, 30MHz, por tanto, se necesita una frecuencia de transmisión y recepción de 60MHz. Debido a la alta frecuencia de la señal a transmitir y recibir se ha decidido realizar la conexión con los conversores AD y DA mediante FPGA.

Para la comunicación entre el PC y la FPGA se desean utilizar dos puertos UDP, uno de ellos para la transmisión de datos y otro para el envío de instrucciones del PC hacia la FPGA.

El puerto UDP por el que se enviarán las instrucciones, deberá seguir el siguiente protocolo:

- El primer conjunto de 64 bits indicará en número de muestras a procesar
- El segundo conjunto de 64 bits será la instrucción que se desea enviar que puede ser:
    - Envío de datos del PC a la memoria de la FPGA
    - Envío de datos de la memoria de la FPGA al PC
    - Iniciar transmisión de los datos de la memoria de la FPGA por el DAC
    - Recepción de datos por el ADC y guardarlos en la memoria de la FPGA
    - Iniciar transmisión de datos por el DAC y recepción de datos por el ADC en el mismo instante

En la Figura 3 se `puede ver el prototipo a diseñar.

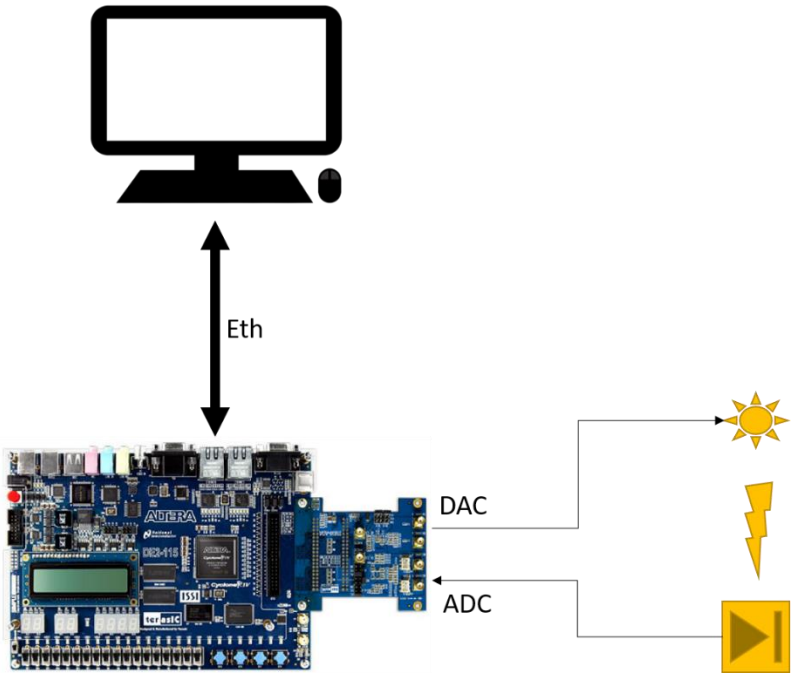

Figura 3. Prototipo a diseñar

3.2 Implementación del prototipo

El prototipo diseñado contiene dos grandes bloques de implementación, la parte hardware implementada sobre una FPGA, y la parte software implementada sobre Matlab.

3.2.1 Implementación hardware en FPGA

La implementación del hardware se divide en dos grandes bloques, la parte de control, dominada en gran parte por máquinas de estados, y la parte de procesado de señal.

En la Figura 4 se puede observar un esquema del circuito implementado en la FPGA. En dicho esquema se pueden observar las distintas máquinas de estado implementadas, la interface Ethernet, las dos memorias y el filtro interpolador. Además, se pueden observar los distintos dominios de frecuencia utilizados en el diseño.

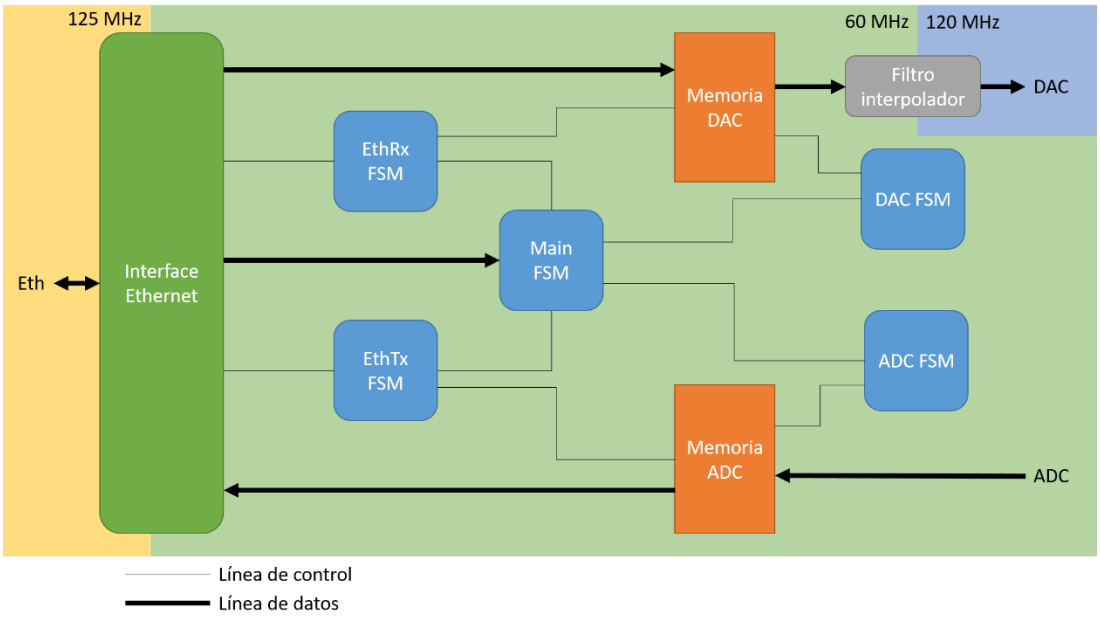

Figura 4. Esquema del circuito implementado en la FPGA

3.2.1.1 Interface Ethernet

Para la implementación de la interface Ethernet se ha utilizado una implementación ya realizada anteriormente en el grupo de investigación en el que se ha realizado el actual trabajo. Dicha interface implementa las capas de red (MAC, Ethernet), de internet (IP, ARP) y de transporte (UDP). La interface ofrece una conexión AXI con un bus de datos de 64 bits.

Internamente la interface está dividida en dos partes, ya que se utilizan dos puertos UDP, uno para la transmisión de datos, y otro para el control de la FPGA. Cabe destacar que la parte de recepción de órdenes solo tiene implementada la parte de recepción y no la de transmisión, ya que la FPGA en ningún momento ha de enviar ningún mensaje al PC.

3.2.1.2 Máquina de estados de control

La máquina de estados de control es una máquina de Moore y es la encargada de recibir órdenes del PC y ejecutarlas en la FPGA. El funcionamiento de la máquina de estados se describe a continuación. Su diagrama de estados se muestra en la Figura 5.

Al inicio la máquina está esperando a que llegue un valor válido (*rx_tvalid*), en el momento en que llega un dato válido, se registra dicho dato (*regN*) ya que es el número de muestras a procesar, a continuación se recibe la instrucción a realizar, por tanto, según la instrucción recibida se deberá

cambiar al estado correspondiente para activar las salidas correspondientes (*dac*, *adc*, *txEth*), una vez activadas las salidas se ha de ir a un último estado de espera, ya que, el paquete transmitido tiene un tamaño mínimo superior al utilizado y por tanto, se ha de esperar a que se termine de recibir el paquete (*rx_tlast*). Una vez finalizada la recepción del paquete se vuelve a esperar un nuevo paquete.

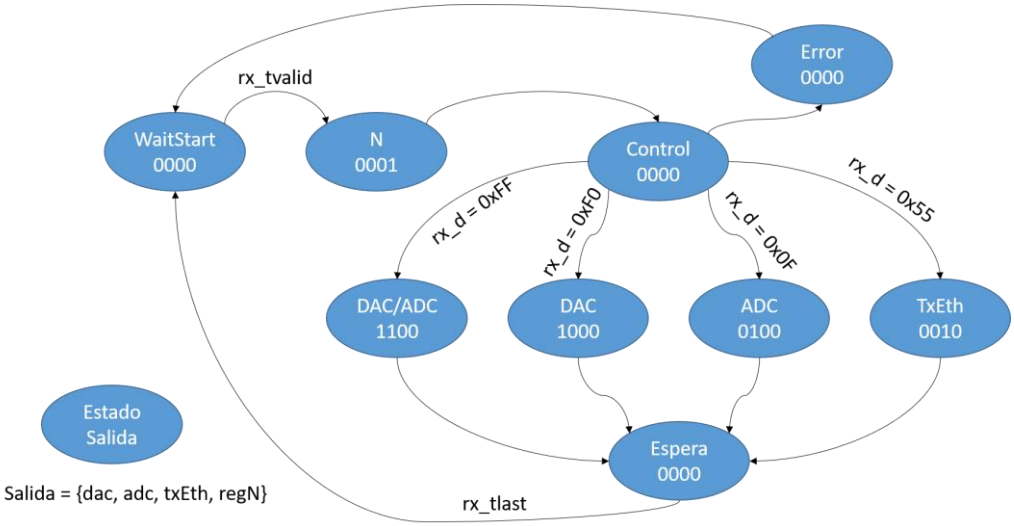

Figura 5. Diagrama de estados de control

3.2.1.3 Máquina de estado de recepción Ethernet

La máquina de estado de recepción de Ethernet es una máquina de Moore y es la encargada de recibir los datos a través del puerto de Ethernet. Su diagrama de estados se muestra en la Figura 6. La máquina de estados se activa cuando recibe la señal de dato válido (*tvalid*) y activa la señal para guardar en la memoria los datos recibidos (*wr_en*) hasta llegar al último dato (*tlast*).

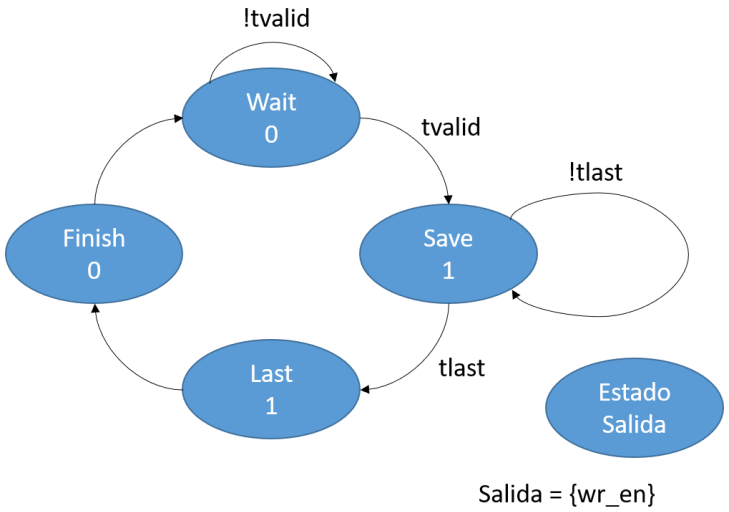

Figura 6. Diagrama de estados recepción Ethernet

3.2.1.4 Máquina de estado de transmisión Ethernet

La máquina de estados de la transmisión de Ethernet (Figura 7) también es una máquina de Moore. La máquina de estados empieza con estado de espera, hasta que llega la señal para iniciar la transmisión (*initTx*), después de éste hay un estado de espera para poder leer el primer dato de la memoria, a continuación, está es estado de lectura y envío de datos que activa la señal *tvalid* para cumplir el estándar AXI, este estado se repite hasta que se ha de enviar el último dato (*isLast*).

Esta máquina de estados necesita una ruta de datos (Figura 8) el cual ha de contar desde 0 hasta *N*. Para ello, se han creado las señales *ld_addr*, y *cl_addr* para sumar 1 a la cuenta o reiniciar la cuenta, respectivamente. Estas señales son controladas por la máquina de estados como se puede ver en la Figura 7.

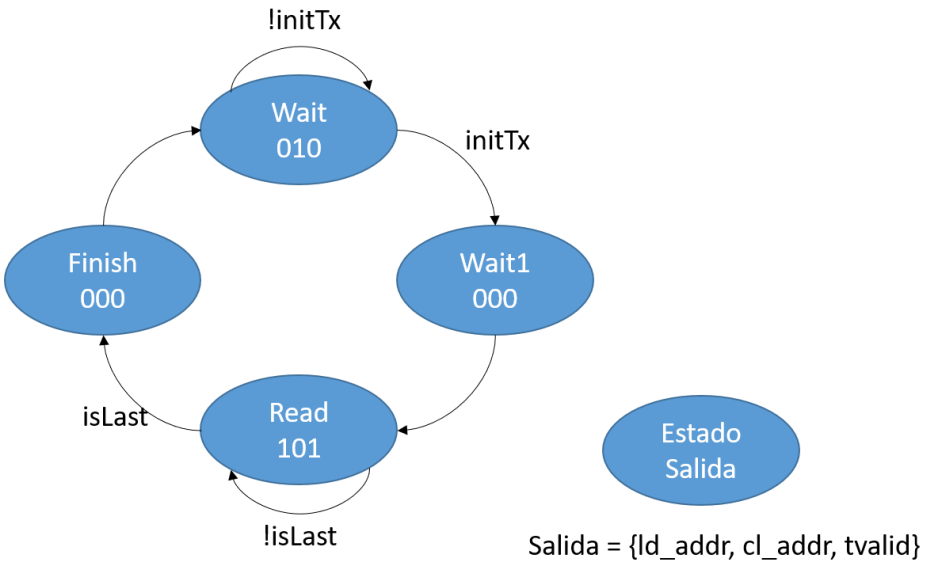

Figura 7. Diagrama de estados transmisión Ethernet

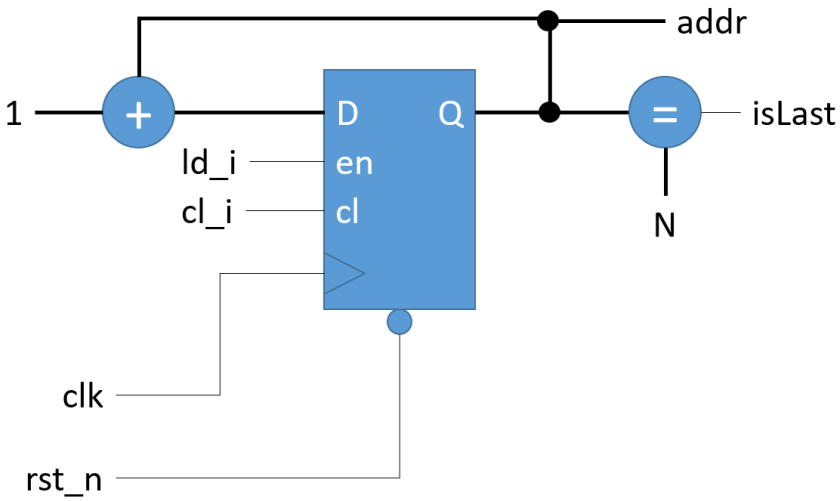

Figura 8. Datapath máquina de estados transmisión Ethernet

3.2.1.5 Máquina de estado de transmisión DAC

La máquina de estados de transmisión del DAC (Figura 9) es muy parecida a la máquina de estados anterior de transmisión de Eth. El primer estado es el de inicio en el que se espera la señal para iniciar la transmisión (*start*), después, está el estado de transmisión que se va a repetir hasta que se active la señal *isLast*, que se pasará al estado final. Esta máquina de estados también tiene una ruta de datos que es la misma que la de la máquina de estados anterior (Figura 8).

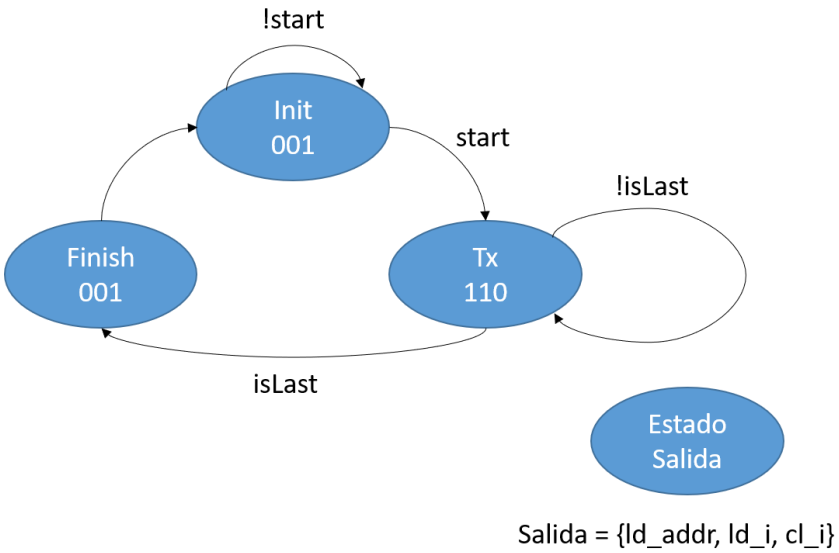

Figura 9. Diagrama de estados transmisión DAC

3.2.1.6 Máquina de estado de recepción ADC

La máquina de estados de la recepción del ADC también es muy simple (Figura 10). Tiene dos estados, el primero de espera y cuando llega la señal de activación para empezar a recibir datos del ADC (*start*), se pasa al segundo estado en el que se van recibiendo todos los datos hasta llegar al final (*end*).

Esta máquina de estados también necesita una ruta de datos (Figura 11) en la que simplemente hay un contador para parar de recibir datos del ADC.

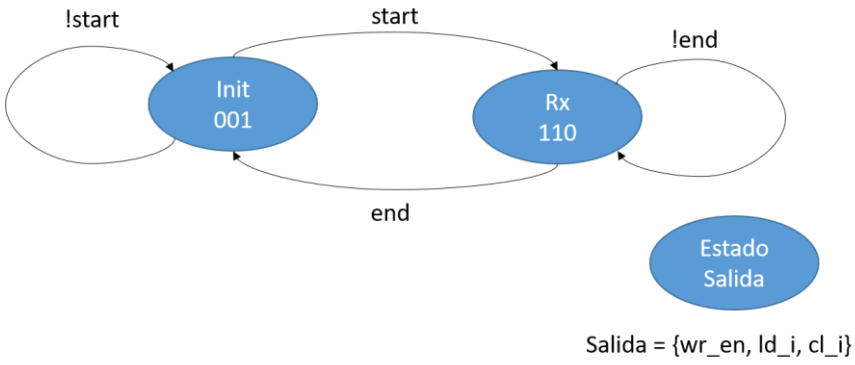

Figura 10. Diagrama de estados recepción ADC

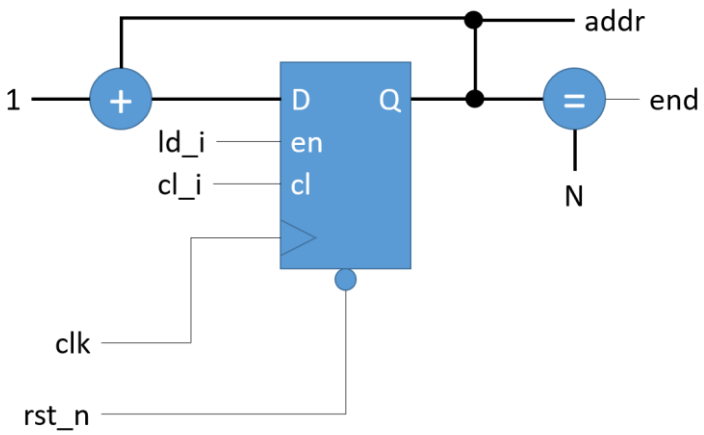

Figura 11. Datapath máquina de estados recepción ADC

3.2.1.7 Memorias RAM

En este trabajo se utilizan dos memorias RAM de doble puerto, una para los datos recibidos por el puerto Ethernet y que van a ser enviados a través del DAC y la otra memoria para los datos recibidos a través del ADC y que van a ser enviados a través del puerto Ethernet.

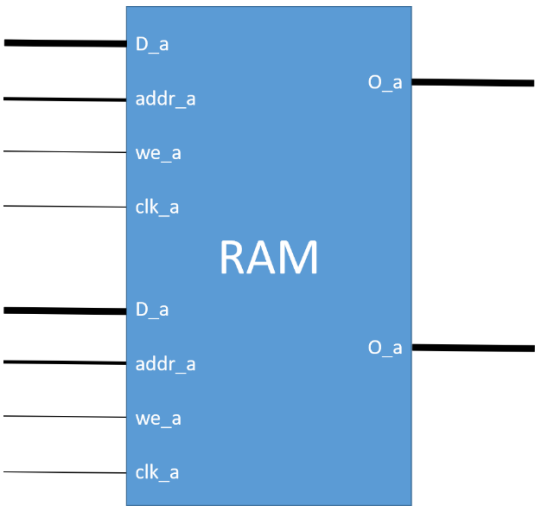

Figura 12. Memoria RAM de doble puerto

En la Figura 12 se puede observar el diagrama de una memoria RAM de doble puerto. Se puede observar que tiene un puerto de entrada de datos (*D_a* y *D_b*), dirección en la que se quiere escribir y/o leer (*addr_a* y *addr_b*), una señal de habilitación de la escritura (*we_a* y *we_b*), una salida de datos (*O_a* y *O_b*) y una señal de reloj (*clk_a* y *clk_b*). Como se puede observar todas las señales están duplicadas para conseguir que los dos puertos de la memoria RAM sean totalmente independientes.

3.2.1.8 Filtro interpolador

En la Figura 4 se puede observar que el DAC funciona a 120MHz, mientras que el resto del circuito (incluidas las memorias) funcionan a 60Mhz lo que hace necesario un filtro interpolador por 2 para poder convertir la frecuencia de muestreo de los datos de 60MHz a 120MHz.

Para la implementación del filtro interpolador se ha decidido utilizar un filtro polifásico, de esta forma se consigue que el filtro funcione a 60MHz, con lo que su implementación es más sencilla.

Se desea que el filtro sea bastante abrupto, por lo que se ha decidido utilizar un filtro FIR de orden 32, el cual se puede ver su transformada Z en la ecuación (1).

$$\begin{aligned} H(z) = {} & -0.00377z^{-1} + 0.00771z^{-3} - 0.01646z^{-5} + 0.03185z^{-7} \\ & - 0.05727z^{-9} + 0.10129z^{-11} - 0.19576z^{-13} + 0.63099z^{-15} \\ & + z^{-16} + 0.63099z^{-17} - 0.19576z^{-19} + 0.10129z^{-21} \\ & - 0.05727z^{-23} + 0.03185z^{-25} - 0.01646z^{-27} + 0.00771z^{-29} \\ & - 0.00377z^{-31} \end{aligned} \quad (1)$$

En la Figura 13 se puede observar la respuesta en frecuencia del filtro interpolador. Se puede observar que la frecuencia de corte es 0.5 que es la frecuencia de corte correspondiente para hacer un filtro interpolador por 2. Además, se puede comprobar que el filtro tiene fase lineal lo que asegura que el filtro no va a distorsionar la señal.

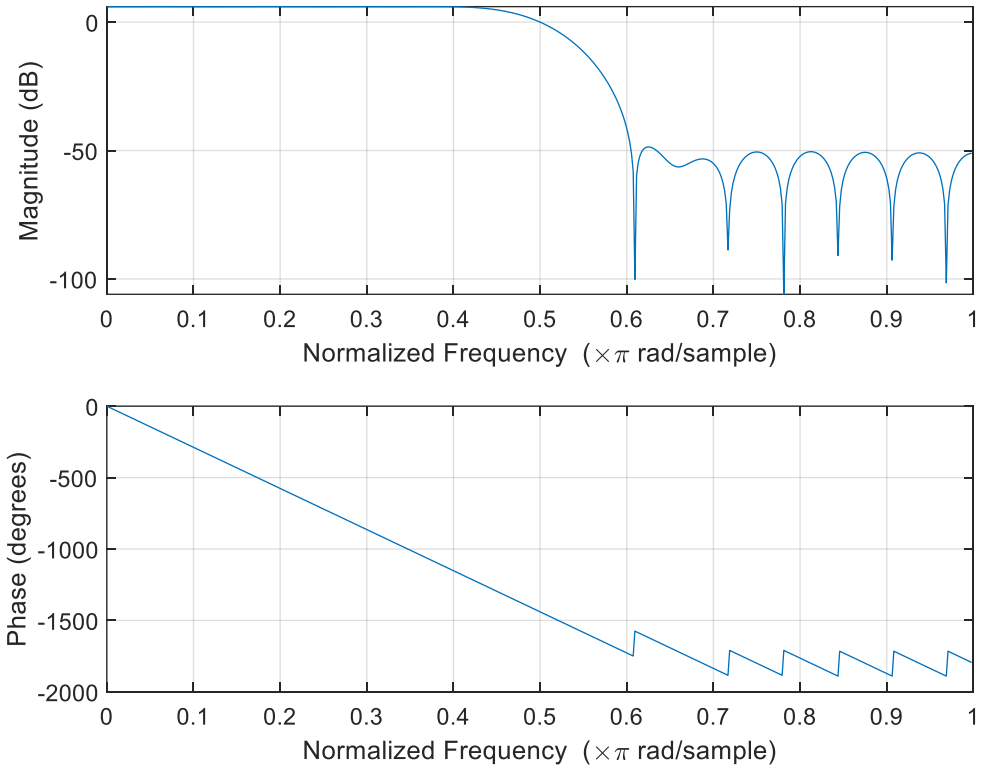

Figura 13. Respuesta en frecuencia del filtro interpolador

Este filtro cumple todos los requisitos para el filtro que se desea implementar, pero tiene un problema y es que si se amplía la respuesta en frecuencia en la zona de paso del filtro (Figura 14) se puede observar que la respuesta no es plana totalmente, sino que tiene ciertos lóbulos con ganancia mayor a 6dB lo que puede provocar problemas en la posterior cuantificación del filtro. Es por esto por lo que se ha decidido añadir una ganancia de 0.95 al filtro y la transformada Z del filtro resultante se puede ver en la ecuación (2).

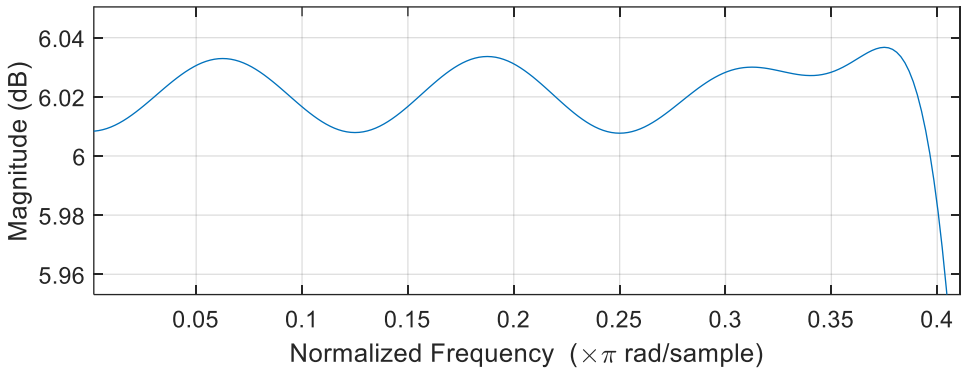

Figura 14. Respuesta en frecuencia de la zona de paso del filtro interpolador

$$\begin{aligned} H(z) = {} & -0.00358z^{-1} + 0.00732z^{-3} - 0.01564z^{-5} + 0.03025z^{-7} \\ & - 0.05441z^{-9} + 0.09623z^{-11} - 0.18597z^{-13} + 0.59944z^{-15} \\ & + 0.95z^{-16} + 0.59944z^{-17} - 0.18597z^{-19} + 0.09623z^{-21} \\ & - 0.05441z^{-23} + 0.03025z^{-25} - 0.01564z^{-27} + 0.00732z^{-29} \\ & - 0.00358z^{-31} \end{aligned} \tag{2}$$

Como se puede observar en la transformada Z del filtro (ecuación (2)), el filtro es de media banda, de forma que se puede reducir su implementación variando el orden de los operadores. En la Figura 15 se puede observar la implementación del filtro interpolador.

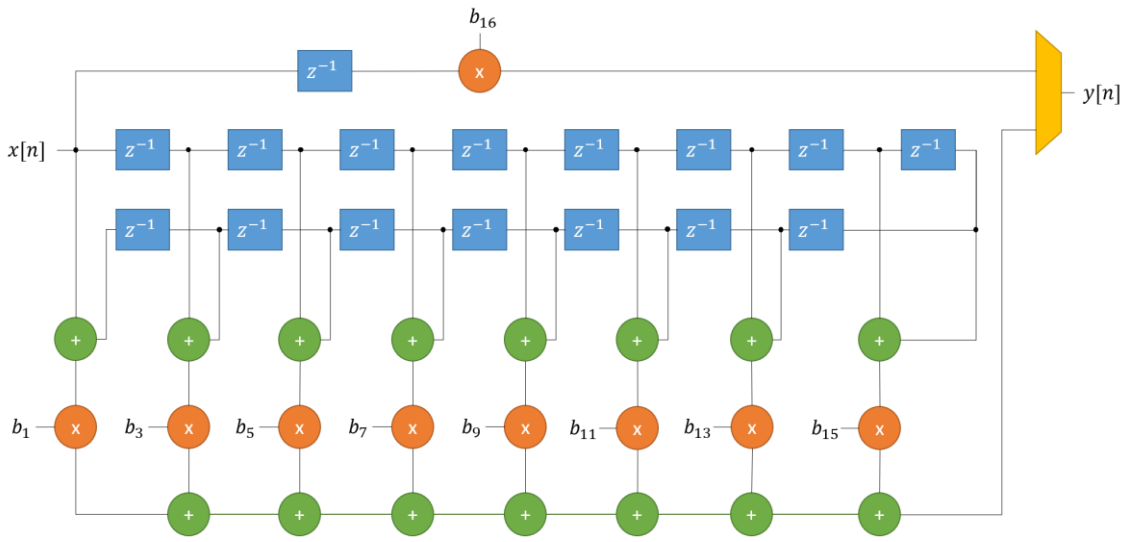

Figura 15. Esquema del filtro interpolador

3.2.1.9 Utilización de recursos hardware

El sistema ha sido codificado y verificado en Verilog excepto el interface Ethernet que está escrito en VHDL.

El sistema ha sido implementado en el dispositivo Cyclone IV. En la Tabla I se pueden ver los recursos utilizados por cada uno de los bloques implementados y por el total del circuito.

Tabla I. Recursos hardware utilizados

	Elementos Lógicos	Registros	Multiplicadores 18x18	Memorias M9K
Ethernet	4136	3240	0	184
FSM Control	50	10	0	0
FSM Rx Eth	30	30	0	0
FSM Tx Eth	31	20	0	0
FSM DAC	27	17	0	0
FSM ADC	39	16	0	0
Memorias RAM	0	0	0	120
Filtro interpolador	829	776	8	0
Resto del circuito	238	206	0	0
Total	**5380**	**4315**	**8**	**304**

El hardware implementado utiliza un 4.7% de los elementos lógicos, un 3.01% de los multiplicadores y un 70.37% de los bloques M9K de memoria disponibles en el dispositivo utilizado. Esto indica que el diseño implementado es compatible con otras implementaciones dentro del mismo dispositivo. El recurso más utilizado es la memoria, pero cabe destacar que es fácilmente ampliable utilizando memorias externas, es más, la misma placa utilizada contiene distintas memorias externas.

3.2.2 Implementación software en Matlab

La parte software de este trabajo es el desarrollo de una librería de funciones para implementar diferentes modulaciones en Matlab. Las modulaciones implementadas han sido las M-QAM (es decir, 4QAM, 16QAM, …). La implementación software se divide en cuatro bloques, la modulación, la demodulación, la sincronización y la comunicación con el hardware.

3.2.2.1 Modulación

Como ya se ha comentado, se van a implementar las modulaciones M-QAM. La función implementada tiene las siguientes entradas:

- Señal digital a modular
- Tamaño de la modulación (4QAM, 16QAM, …)
- Número de muestras por símbolo
- Longitud de span
- Frecuencia de modulación

El primer paso es asignar a cada conjunto de bits un símbolo de la constelación, a continuación, se ha de sobremuestrear añadiendo ceros, tantos como indique el número de muestras por símbolo. El siguiente paso es separar la parte real de la parte imaginaria y a cada parte aplicarle el filtro de coseno alzado para posteriormente multiplicar la parte real por el coseno de la frecuencia de modulación y la parte imaginaria por el seno de dicha frecuencia. Por último, se han de sumar las dos ramas. El esquema del modulador se puede ver en la Figura 16.

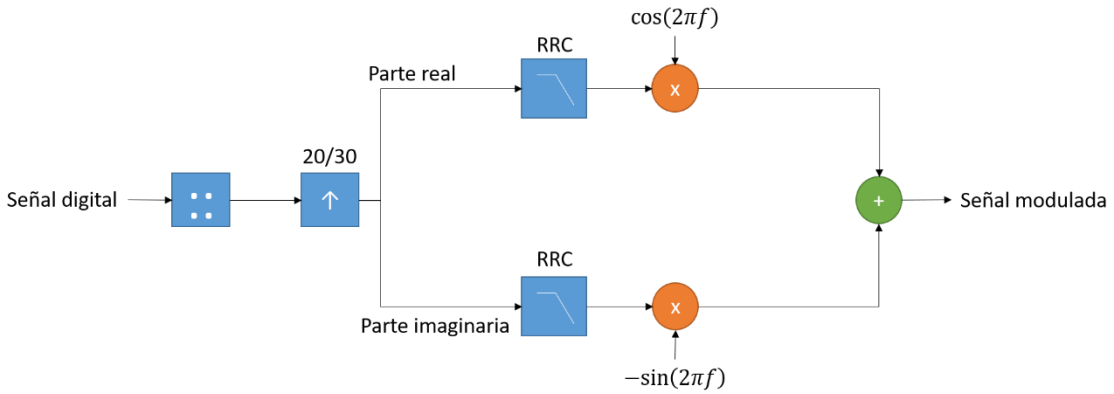

Figura 16. Modulador MQAM

3.2.2.2 Demodulación

La demodulación de las señales M-QAM es muy parecida a la modulación. La función implementada tiene las siguientes entradas:

- Señal a demodular
- Tamaño de la modulación (4QAM, 16QAM, …)
- Número de muestras por símbolo
- Longitud de span
- Frecuencia de modulación
- Desfase de la señal recibida

El primer paso para demodular la señal es multiplicarla por un lado por el coseno de la frecuencia de modulación y por otro por el seno y de esta manera conseguir la parte real e imaginaria de la señal modulada. A continuación, se ha de filtrar con un filtro de coseno alzado para posteriormente bajar la tasa de muestreo. En este momento ya se tienen los símbolos de la constelación y solo falta asignar a cada símbolo el conjunto de bits correspondientes para obtener la señal digital transmitida. El esquema del demodulador se puede ver en la Figura 17.

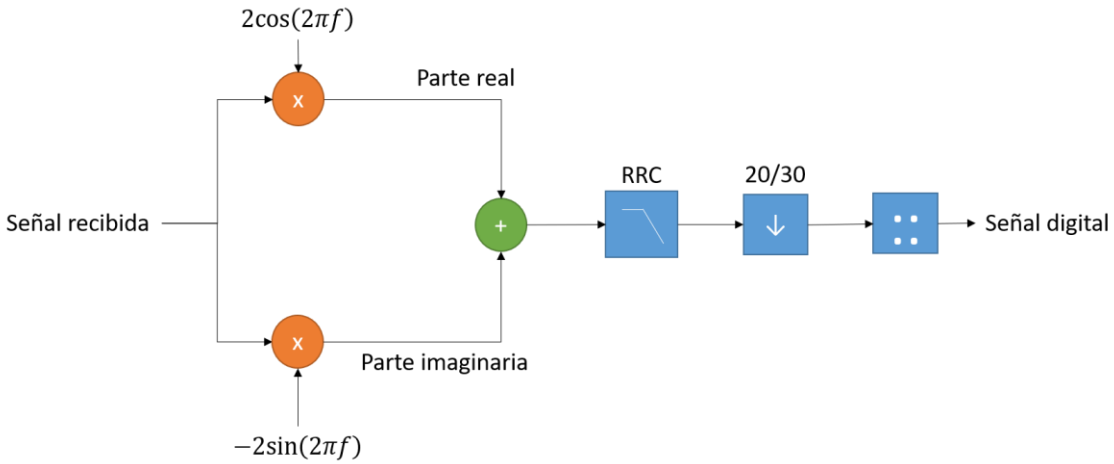

Figura 17. Demodulador MQAM

Al modulador le falta la parte de sincronización para poder obtener la señal digital correctamente. La sincronización se verá en el siguiente punto.

3.2.2.3 Sincronización

Para conseguir demodular la señal recibida correctamente es necesario sincronizar la señal recibida, para ello se ha de utilizar una señal conocida por el modulador y el demodulador.

3.2.2.3.1 Sincronización en la modulación

La sincronización en la modulación consiste en añadir al inicio de la transmisión la señal conocida entre el modulador y el demodulador. Esta señal va a estar modulada en QPSK (o 4QAM) independientemente de la modulación posterior que tenga la señal a transmitir, ya que, de esta forma es más fácil de sincronizar.

3.2.2.3.2 Sincronización en la demodulación

La sincronización en la demodulación consiste en encontrar la señal conocida modulada en QPSK utilizando la correlación, una vez obtenida esta señal se demodula utilizando el esquema de la Figura 17 pero el último paso de obtener la secuencia binaria no se realiza, sino que se obtiene la constelación de los símbolos recibidos.

Estos símbolos recibidos se han de mover al primer cuadrante todos, para de esta manera, obtener el desfase entre la señal ideal (todos los símbolos a 45º) y la señal recibida.

El desfase obtenido en esta etapa se ha de tener en cuenta una vez se ha obtenido la constelación para corregir los símbolos y una vez corregidos obtener la secuencia binaria correctamente.

3.2.2.4 Comunicación con el hardware

La comunicación con el hardware se hace a través de una red TCP/IP utilizando dos puertos UDP, uno para la transmisión de datos y otro puerto para que el PC mande órdenes al hardware.

Para la comunicación entre el PC y la FPGA se han creado varias funciones básicas:

- Enviar datos del PC a la memoria de la FPGA
- Enviar datos de la memoria de la FPGA al PC
- Enviar datos guardados en la memoria de la FPGA a través del DAC
- Recibir datos del ADC y guardarlos en la memoria de la FPGA
- Enviar datos a través del DAC y recibir datos del ADC al mismo instante

Además de estas 5 funciones básicas se han creado funciones con combinaciones de las anteriores:

- Enviar datos del PC a la memoria de la FPGA y a continuación enviar estos datos a través del DAC
- Recibir datos del ADC, guardarlos en memoria y posteriormente enviar estos datos al PC
- Enviar datos del PC a la memoria de la FPGA, a continuación, enviar estos datos a través del DAC al mismo tiempo que se reciben datos por el ADC y finalmente, enviar los datos recibidos al PC

Capítulo 4. Resultados obtenidos

4.1 Montaje de experimentación

Para la experimentación del software y hardware implementado se ha diseñado y montado el banco de pruebas que se muestra en la Figura 18.

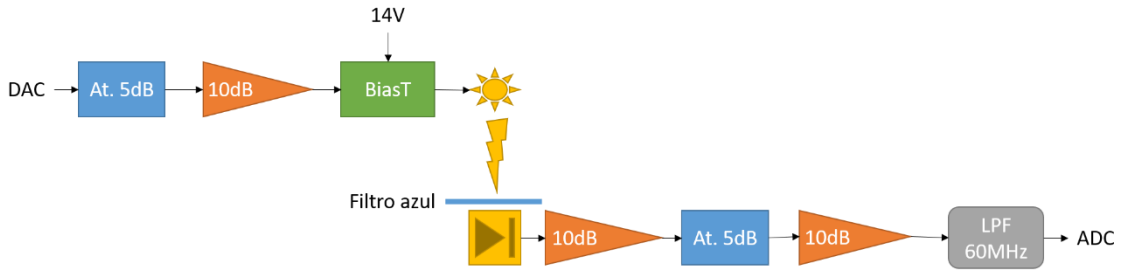

Figura 18. Montaje de experimentación

El montaje está formado por atenuadores y amplificadores de señal, un filtro paso bajo, un LED, un fotodiodo receptor, un filtro azul y un BiasT.

La salida del DAC necesita ser amplificada 5dB para ajustarse a la entrada del LED, por esto va conectada a un atenuador de 5dB y, a continuación, un amplificador de 10dB. A continuación, la señal va conectada a un BiasT que alimenta el LED con 14V. Después, va conectado el LED que emite luz hacia un fotodiodo que está a una distancia de 1.5m.

Entre el LED y el fotodiodo se ha colocado un filtro azul para aumentar el ancho de banda del LED. Para enviar la señal al ADC es necesario amplificar la señal 15dB, por esto se han conectado dos amplificadores de 10dB y un atenuador de 5dB. Por último, es necesario colocar un filtro paso bajo para evitar el aliasing en el ADC.

En la Figura 19 se puede ver el montaje realizado en el laboratorio.

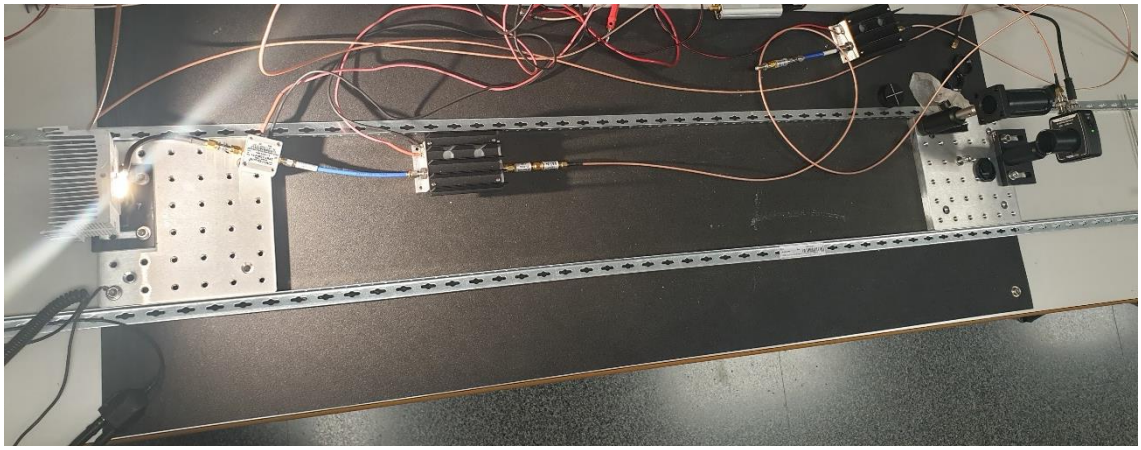

Figura 19. Montaje realizado

4.2 Resultados

Para comprobar el correcto funcionamiento del montaje diseñado se han realizado dos pruebas distintas. Por un lado, se ha medido la respuesta en frecuencia del montaje completo y, por otro lado, se ha medido el BER (Tasa de error binario) de las modulaciones 4QAM, 16QAM y 64QAM.

4.2.1 Respuesta en frecuencia del sistema

Para medir la respuesta en frecuencia se han enviado de manera secuencial senoidales desde 1kHz hasta 25MHz. Estas frecuencias han sido seleccionadas de manera escalonada de teniendo en cuenta que en las bajas frecuencias hay menos muestras en la respuesta en frecuencia que en las altas frecuencias. Desde 1kHz a 15kHz se ha realizado en pasos de 1kHz, desde 15kHz hasta 90kHz se ha realizado en pasos de 5kHz y, por último, desde 90kHz hasta 25MHz se ha realizado en pasos de 50kHz. Se ha repetido el proceso cinco veces para cada frecuencia. En la Figura 20 se puede observar la respuesta en frecuencia del sistema. Como se puede observar en la respuesta en frecuencia el ancho de banda del sistema es de15kHz hasta 8MHz.

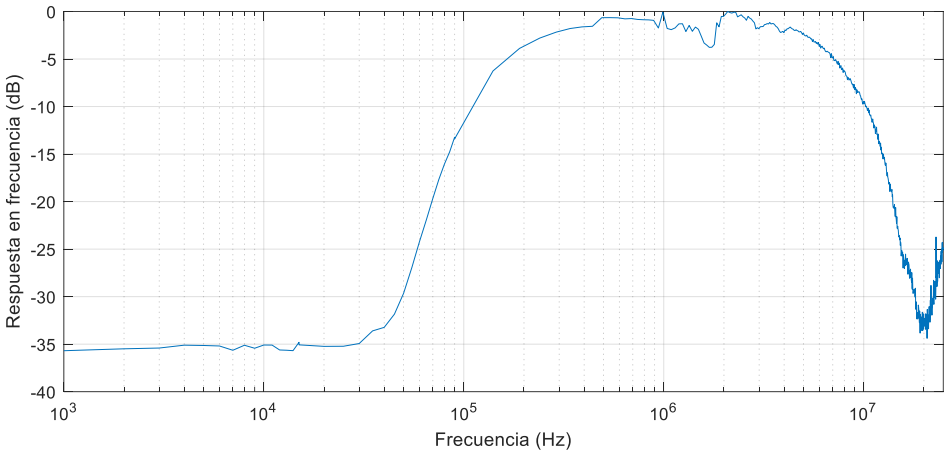

Figura 20. Respuesta en frecuencia del sistema

4.2.2 BER de las modulaciones

El sistema se ha comprobado con las modulaciones 4QAM, 16QAM y 64QAM. Además de estas tres modulaciones también se ha comprobado el sistema con 20 y 30 muestras por símbolo. Todas las pruebas se han realizado con una frecuencia central de 5MHz y 1000 símbolos por trama, aunque por el retardo que tiene el circuito solo se contabilizan 989 símbolos.

Para cada modulación se han enviado 40 tramas con 20 muestras por símbolo y 40 con 30 muestras por símbolo. En la recepción se han descartado las tramas en las que ha habido errores de sincronización.

Para el cálculo del BER es necesario saber en número de bits recibidos, esto se puede obtener con la ecuación (3), una vez obtenidos el número de bits recibidos se puede obtener el BER con la ecuación (4).

$$bits\ recibidos = \frac{símbolos\ recibidos}{trama} \cdot \frac{nº\ bits}{símbolo} \cdot nº\ tramas \tag{3}$$

$$BER = \frac{bits\ erróneos}{bits\ recibidos} \tag{4}$$

Sobre el cálculo del BER cabe destacar que para las modulaciones M-QAM el BER mínimo para poder considerar la transmisión como buena es de un 0.38% si se utiliza en la transmisión un sistema de corrección de errores con *hard-decision* o 2.4% si se utiliza *soft-decision*.

Para la modulación 4QAM con 20 muestras por símbolo el resultado ha sido de 0 errores, por lo que la transmisión ha sido perfecta en todas las pruebas. En la Figura 21 se puede ver una de las constelaciones recibidas y se puede observar cómo están bien definidos los cuatro símbolos del 4QAM.

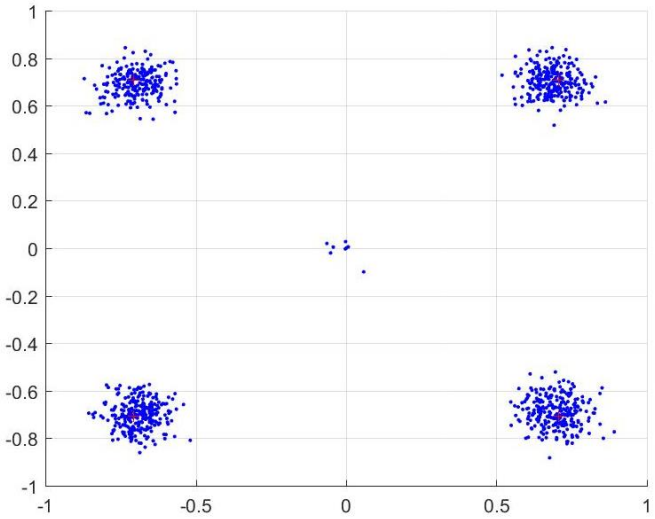

Figura 21. Constelación 4QAM, 20 muestras por símbolo

En la Figura 22 se puede observar el ancho de banda utilizado para la transmisión de la modulación 4QAM con 20 muestras por símbolo. Se puede ver como el ancho de banda está centrado en 5MHz y tiene un ancho de banda de 3MHz.

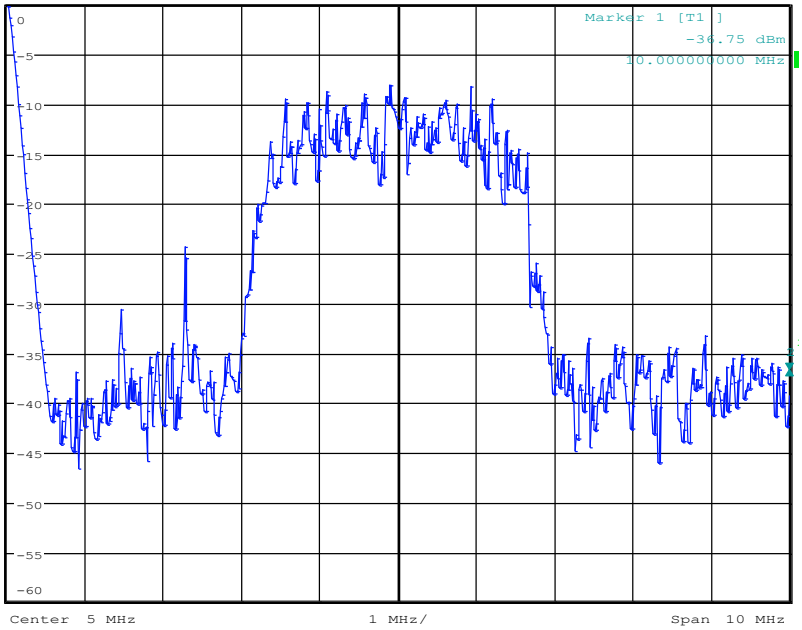

Figura 22. Ancho de banda de 4QAM con 20 muestras por símbolo

Para la modulación 4QAM con 30 muestras por símbolo el resultado también ha sido de 0 errores al igual que en las anteriores pruebas. En la Figura 23 se puede observar que los 4 símbolos están bien diferenciados. Además, se puede destacar que la dispersión de los símbolos es menor en este caso que en el caso anterior. En esta prueba cabe destacar que se ha eliminado una trama, ya que, la sincronización inicial no ha funcionado correctamente.

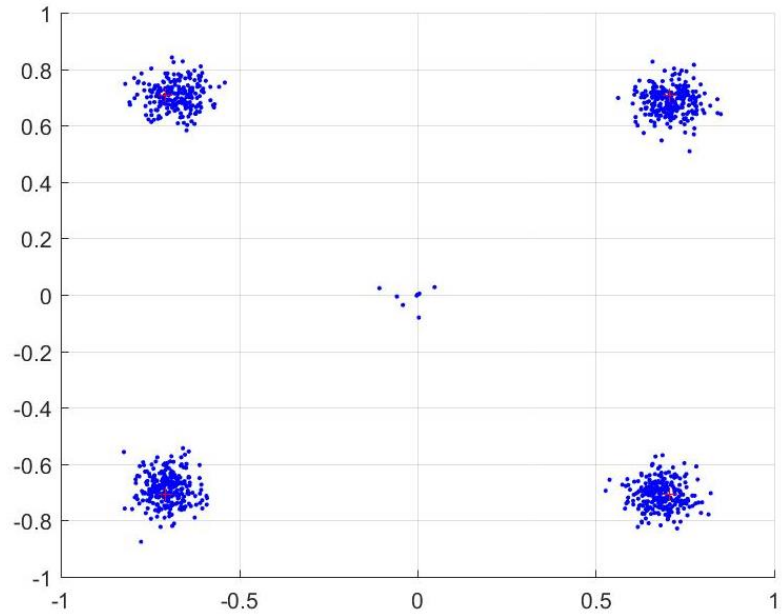

Figura 23. Constelación 4QAM, 30 muestras por símbolo

En el caso de 16QAM con 20 muestras por símbolo, en las 40 tramas enviadas solo ha habido 2 bits erróneos, lo que supone un BER de un 0.00137%. En la Figura 24 se puede observar una de las constelaciones recibidas. Se puede ver que los símbolos siguen estando diferenciados. En esta prueba cabe destacar que se han eliminado tres tramas, ya que, la sincronización inicial no ha funcionado correctamente (se han considerado 37 tramas recibidas para el cálculo del BER).

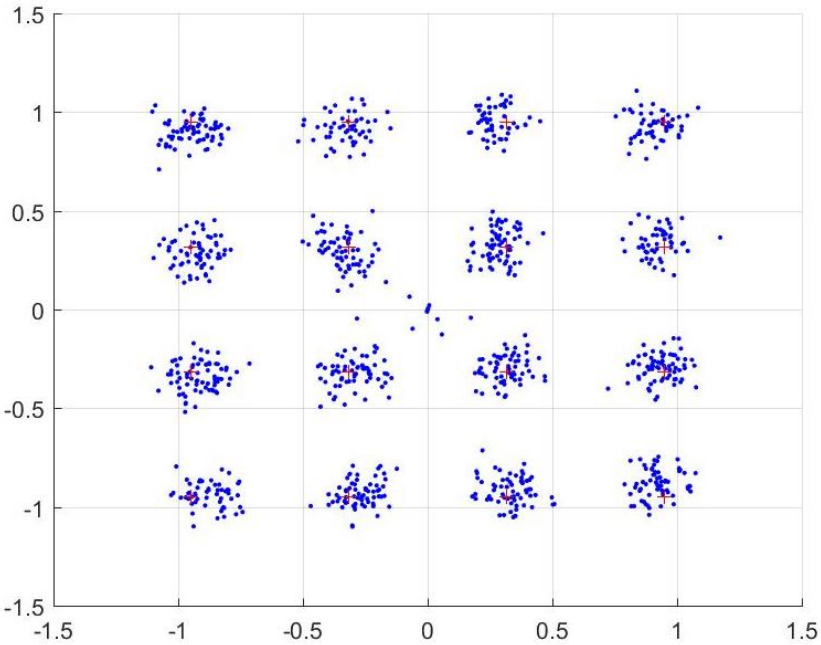

Figura 24. Constelación 16QAM, 20 muestras por símbolo

En la Figura 25 se puede ver el ancho de banda utilizado para la transmisión de 16QAM con 20 muestras por símbolo. Se puede observar que está centrado en 5MHz y tiene un ancho de banda de 3MHz.

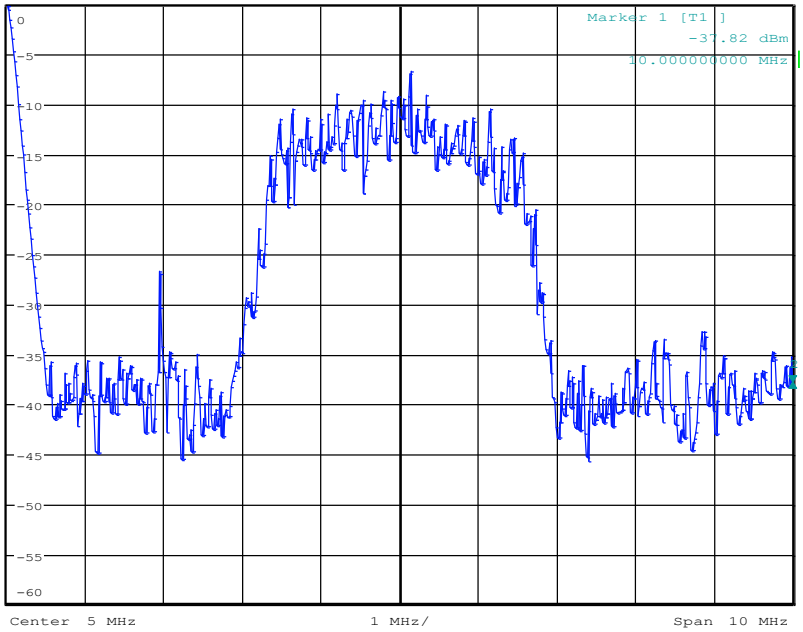

Figura 25. Ancho de banda de 16QAM con 20 muestras por símbolo

Para 16QAM con 30 muestras por símbolo ha habido 0 errores en todas las transmisiones, lo que supone un BER de un 0%. En la Figura 26 se puede observar la constelación de una de las transmisiones realizadas. Al igual que en el caso de 4QAM, se puede observar que ha habido menos dispersión que en el caso de 20 muestras por símbolo, lo que se ve también en los bits erróneos de la transmisión.

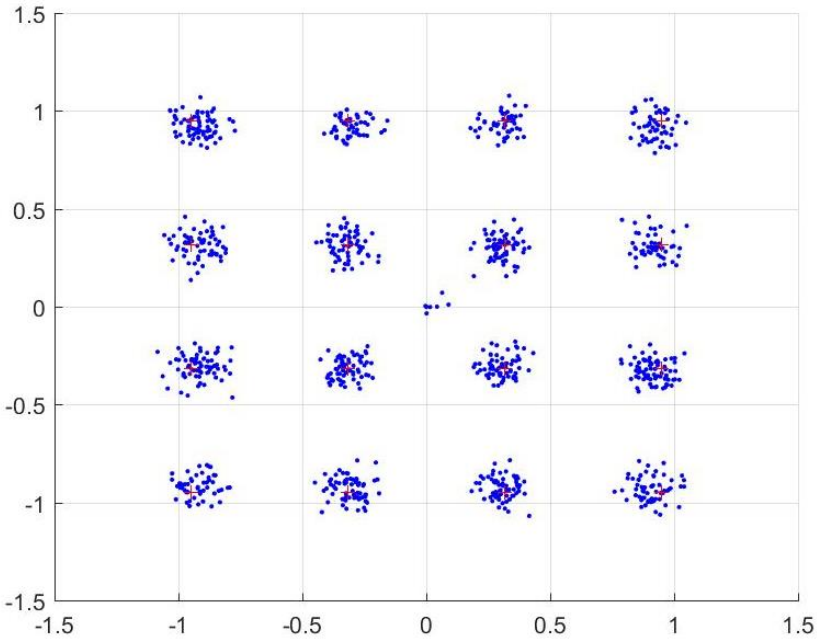

Figura 26. Constelación 16QAM, 30 muestras por símbolo

En la Figura 27 se puede observar como el ancho de banda ha variado con respecto al caso de 20 muestras por símbolo, ya que, en este caso el ancho de banda es de 2MHz.

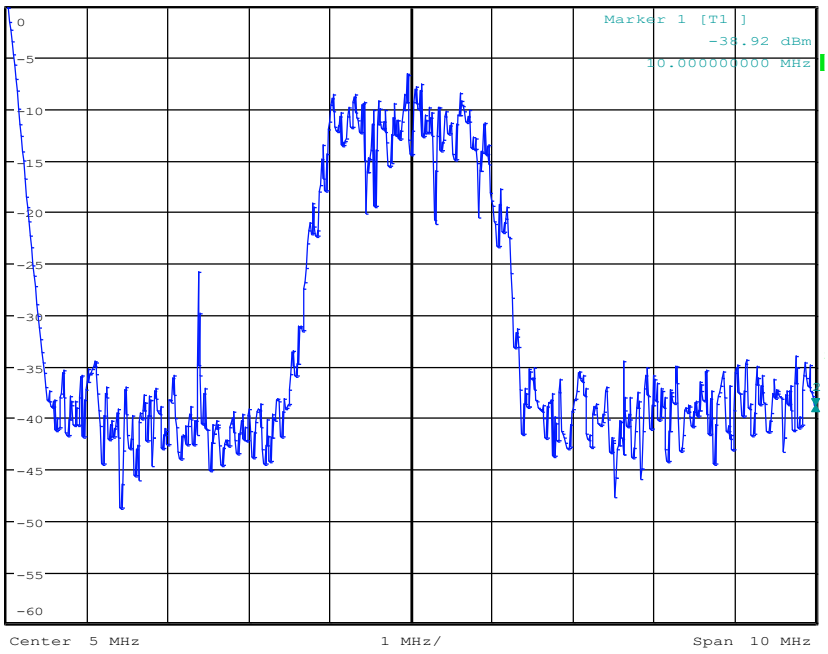

Figura 27. Ancho de banda de 16QAM con 30 muestras por símbolo

En el caso de 64QAM con 20 muestras por símbolo ha habido 2250 bits erróneos en la transmisión lo que supone un BER del 0.97%. En la Figura 28 se puede observar una de las constelaciones recibidas, se puede observar cómo los símbolos son muy poco diferenciables entre ellos provocando esto el alto BER. En esta prueba cabe destacar que se ha eliminado una trama, ya que, la sincronización inicial no ha funcionado correctamente (cálculo de BER realizado con 39 tramas).

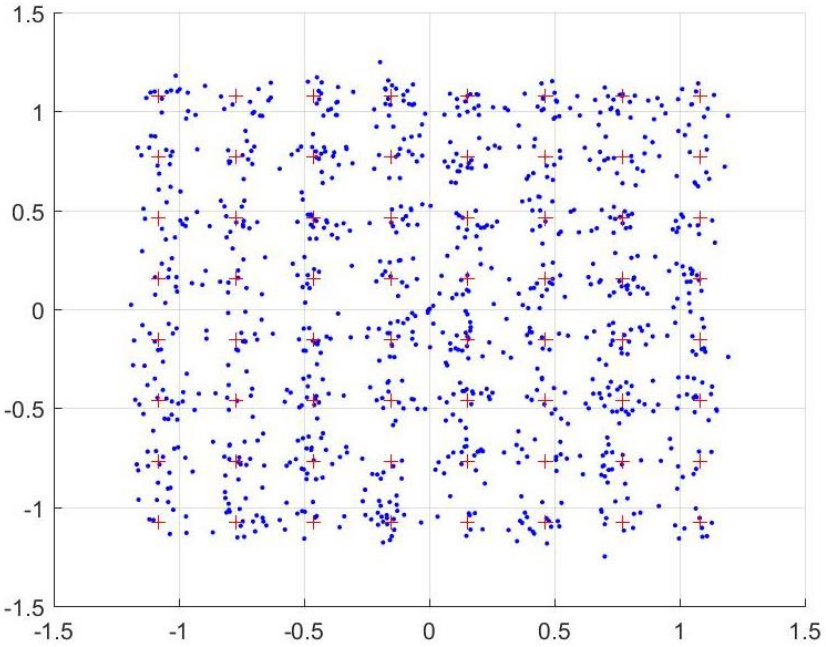

Figura 28. Constelación 64QAM, 20 muestras por símbolo

Por último, para el caso de 64QAM con 30 muestras por símbolo ha habido 646 bits erróneos en las transmisiones, lo que supone un BER del 0.28%. En la Figura 29 se puede ver una de las constelaciones recibidas. Se puede observar que hay menos distorsión que en el caso anterior,

pero los símbolos no son claramente diferenciables entre ellos. En esta prueba cabe destacar que se ha eliminado una trama, ya que, la sincronización inicial no ha funcionado correctamente.

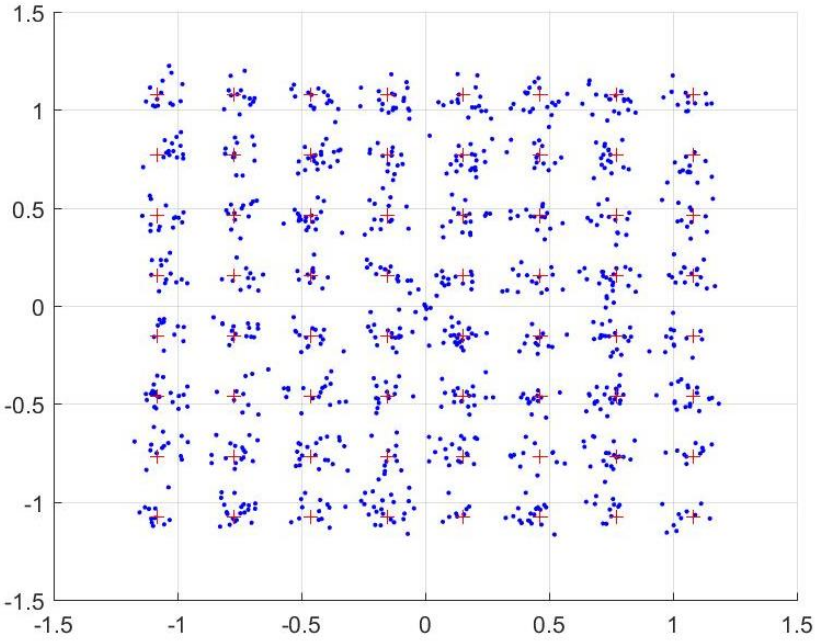

Figura 29. Constelación 64QAM, 30 muestras por símbolo

Como resumen, en la Tabla II se puede observar los BER de todos los casos citados anteriormente.

Tabla II. Resumen BER

	20 muestras por símbolo	**30 muestras por símbolo**
4QAM	0%	0%
16QAM	0.00137%	0%
64QAM	0.97%	0.28%

4.3 Discusión de los resultados

De los resultados mostrados en el punto anterior, se puede ver que el sistema contiene un ancho de banda más que suficiente para ser utilizado como transmisor de datos a alta velocidad.

En los resultados obtenidos se puede ver que en el caso de las modulaciones 4QAM y 16QAM los BER obtenidos son muy buenos En los resultados obtenidos se puede ver que en el caso de las modulaciones 4QAM y 16QAM los BER obtenidos son muy buenos, mientras que el BER obtenido para 64QAM con 30 muestras por símbolo es aceptable con corrección de errores *hard-decision* (BER inferior a 0.38%). En el caso de 64QAM con 20 muestras por símbolo debería usarse corrección de errores *soft-decision*, ya que en este caso se considera aceptable un BER inferior a 2.4%. Además de las distintas modulaciones probadas, se puede observar que al aumentar el tamaño de la modulación los resultados son peores. También cabe destacar la mejoría en la transmisión al aumentar el número de muestras por símbolo, lo que, por otro lado, provoca

una reducción en el ancho de banda de transmisión, pero, al mismo tiempo, baja la velocidad de transmisión.

Capítulo 5. Futuras líneas de trabajo

Como líneas futuras del presente trabajo se dejan:

- Caracterización completa del sistema diseñado para conocer las no linealidades y el posible dimensionado del sistema.
- Mejora del sistema implementado con la introducción de un microprocesador.
- Testeo del sistema a distintas frecuencias de modulación
- Optimización del sistema mediante el uso de otras modulaciones, como, por ejemplo, modulaciones multiportadora.
- Diseño de un sistema con diferentes LEDs y diferentes receptores y análisis de las interferencias producidas entre ellos.

Capítulo 6. Conclusiones

Como conclusiones, se considera que se han logrado los objetivos del trabajo, ya que, se ha diseñado e implementado un sistema para la experimentación de diferentes modulaciones mediante iluminación LED. Se han diseñado unas librerías en Matlab para la modulación y demodulación M-QAM, para sincronización y para la comunicación con el hardware de control. Estas librerías son capaces de comunicarse con la FPGA enviando y recibiendo las modulaciones para posteriormente obtener estadísticas sobre el buen funcionamiento del entorno de experimentación.

Además, se ha implementado el hardware para posibilitar la comunicación con los DAC y los ADC y poder llevar a cabo la comunicación mediante tecnología LED. El hardware implementado utiliza menos de un 30% de los recursos disponibles en el dispositivo utilizado, lo que hace que el diseño implementado sea compatible con otras implementaciones dentro del mismo dispositivo.

También se ha realizado un montaje con un LED y un fotodiodo para la experimentación del software y hardware implementado. Se ha estudiado el correcto funcionamiento del entorno de experimentación para distintas modulaciones M-QAM obteniendo muy buenos resultados para las modulaciones 4QAM y 16QAM y resultados aceptables para 64QAM, aunque en este último caso el número de muestras por símbolo condiciona el sistema de corrección de errores a utilizar *soft-decision*.

Capítulo 7. Referencias

[1] T. Koonen, "Indoor Optical Wireless Systems: Technology, Trends, and Applications," *J. Light. Technol.*, vol. 36, no. 8, pp. 1459–1467, Apr. 2018, doi: 10.1109/JLT.2017.2787614.

[2] C. Chen, H. Haas, L. Yin, and Y. Wang, "What is LiFi?," *J. Light. Technol. Vol. 34, Issue 6, pp. 1533-1544*, vol. 34, no. 6, pp. 1533–1544, Mar. 2016.

[3] W. Xu, M. Zhang, D. Han, Z. Ghassemlooy, P. Luo, and Y. Zhang, "Real-Time 262-Mb/s Visible Light Communication with Digital Predistortion Waveform Shaping," *IEEE Photonics J.*, vol. 10, no. 3, Jun. 2018, doi: 10.1109/JPHOT.2018.2829905.

[4] X. Huang *et al.*, "2.0-Gb/s Visible Light Link Based on Adaptive Bit Allocation OFDM of a Single Phosphorescent White LED," *IEEE Photonics J.*, vol. 7, no. 5, Oct. 2015, doi: 10.1109/JPHOT.2015.2480541.

[5] G. Cossu, A. M. Khalid, P. Choudhury, R. Corsini, and E. Ciaramella, "34 Gbit/s visible optical wireless transmission based on RGB LED," *Opt. Express*, vol. 20, no. 26, 2012, doi: 10.1364/oe.20.00b501.

[6] "Terasic - DE Main Boards - Cyclone - Altera DE2-115 Development and Education Board." [Online]. Available: https://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&No=502. [Accessed: 18-Mar-2020].

[7] "Terasic - Daughter Cards - AD/DA - Highspeed AD/DA Card." [Online]. Available: https://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=73&No=278&PartNo=1. [Accessed: 18-Mar-2020].