PONTIFICIA UNIVERSIDAD CATOLICA DE CHILE
ESCUELA DE INGENIERIA

DISEÑO E IMPLEMENTACIÓN EN FPGA DE UN MÓDEM DE BANDA BASE PARA REDES INALÁMBRICAS DE SENSORES

HÉCTOR PÉREZ DÍAZ

Tesis presentada a la Dirección de Investigación y Postgrado
como parte de los requisitos para optar al grado de
Magister en Ciencias de la Ingeniería

Profesor Supervisor:
CHRISTIAN OBERLI
MARCELO GUARINI

Santiago de Chile, Agosto 2012

PONTIFICIA UNIVERSIDAD CATOLICA DE CHILE
ESCUELA DE INGENIERIA

DISEÑO E IMPLEMENTACIÓN EN FPGA DE UN MÓDEM DE BANDA BASE PARA REDES INALÁMBRICAS DE SENSORES

HÉCTOR PÉREZ DÍAZ

Miembros del Comité:
CHRISTIAN OBERLI
MARCELO GUARINI
DIEGO LÓPEZ-GARCÍA
ANGEL ABUSLEME
PABLO ZEGERS

Tesis presentada a la Dirección de Investigación y Postgrado
como parte de los requisitos para optar al grado de
Magister en Ciencias de la Ingeniería

Santiago de Chile, Agosto 2012

Al lector

AGRADECIMIENTOS.

Quisiera agradecer a mis padres, profesores y amigos por el apoyo incondicional que me han dado durante mis años de estudio.

Índice general

Índice de tablas

Índice de figuras

RESUMEN

Diversos avances tecnológicos han permitido el desarrollo de dispositivos microelectrónicos altamente eficientes y de muy bajo costo. En los últimos años, esta tecnología ha sido utilizada para desarrollar pequeños microcontroladores, transceptores de radio y sensores para medir variables físicas, cada uno de ellos en un solo circuito integrado. La combinación de estos componentes en un nodo ha establecido un área de desarrollo en comunicaciones que emerge rápidamente, conocida como redes inalámbricas de sensores.

Las aplicaciones de redes de sensores requieren que los nodos operen aisladamente y para extender su vida útil es importante minimizar el consumo energético de las comunicaciones entre nodos. La incorporación de tecnología de múltiples antenas en transmisión y recepción permitiría disminuir el consumo energético de un enlace manteniendo su desempeño, solución que hasta ahora no ha sido explorada.

El diseño de un nodo con múltiples antenas requiere, entre otros, de un módem capaz de interactuar con múltiples radios bajo restricciones de consumo energético y tamaño.

Esta tesis describe el diseño e implementación de un módem en banda base que opera en conjunto con un microcontrolador y un módulo de radiofrecuencia. El diseño permite al módem integrarse con sistemas de transmisión y recepción con múltiples antenas, utilizar distintos esquemas de modulación y operar a tasa de datos variable.

Un conjunto de simulaciones computacionales permitieron verificar el correcto funcionamiento del módem, y adicionalmente se corroboró su operación implementando el *hardware* en una FPGA de desarrollo.

Entre los trabajos futuros se propone conectar el módem implementado en FPGA a un microcontrolador y uno o más módulos de radiofrecuencia.

Palabras Claves: redes inalámbricas de sensores, diseño en FPGA,
módem banda base

ABSTRACT

Several technological advances have allowed the development of both highly-efficient and low-cost microelectronic devices. In recent years, this technology has been used to develop microcontrollers, radio transceivers and sensors to measure physical variables, each one of them in a single integrated circuit. The combination of these components has established a fast-growing development area on communications, known as wireless sensor networks.

Sensor network applications require nodes to operate in isolation, so it is important to minimize the node energy consumption to extend its life span. This can be achieved using multiple antenna technology for both transmission and reception. So far, no solution like this has been explored.

The design of a node with multiple antennas requires to have a modem capable of interacting with multiple radios and comply with power consumption and size restrictions, among other requirements.

This thesis describes the design and implementation of a base-band modem that operates with a microcontroller and an RF module. The design allows the integration with multiple-antenna systems, different modulation schemes and variable data rates.

A group of computer simulations was used to verify the correct operation of the modem, and further confirmed its operation by implementing the hardware on an FPGA.

Future work in this project consists of connecting the modem in FPGA with a microcontroller and one or more RF modules.

Keywords: wireless sensor networks, FPGA design, base-band modem

Capítulo 1. INTRODUCCIÓN

El continuo avance tecnológico de las últimas décadas ha permitido el desarrollo de dispositivos electrónicos de comunicaciones pequeños, altamente capaces y de bajo consumo energético, tales como sensores, microcontroladores y transceptores de radio.

La combinación de estos componentes en un solo dispositivo llamado nodo, ha permitido el desarrollo de un área de investigación conocida como redes inalámbricas de sensores (en adelante WSN, *Wireless Sensor Networks*).

Las WSN constituyen un campo de las comunicaciones inalámbricas donde se pronostica el mayor crecimiento de la industria de las telecomunicaciones (Hać, 2003). Sus aplicaciones son diversas, siendo la más común el monitoreo de eventos o variables del entorno, tales como la distribución de temperatura de un bosque, la oscilación de un edificio durante un sismo, los signos vitales de un paciente hospitalizado o el movimiento de un glaciar.

Una WSN se compone de varios nodos capaces de comunicarse entre sí para transmitir la información medida por cada uno de ellos a un dispositivo de almacenamiento central. Cada uno de estos nodos contiene uno o más sensores, un microcontrolador que maneja las funciones del nodo, y un transceptor que permite la comunicación inalámbrica entre nodos. El transceptor típicamente integra un módem y un circuito de radiofrecuencia (en adelante, RF). Todos los componentes son alimentados por una batería que limita su tiempo de funcionamiento.

Debido a las aplicaciones de WSN en ambientes hostiles o de difícil acceso, es necesario que los nodos operen con poca o ninguna intervención humana durante el mayor tiempo posible. Por lo tanto, es deseable que operen con mínimo consumo energético.

Se han investigado diversas técnicas para reducir el consumo energético de un nodo en una WSN. Considerando que la transmisión y recepción de datos representa una gran parte del consumo energético del nodo, la utilización de tecnología de múltiples antenas

ha aparecido como una herramienta que permite aumentar la eficiencia energética, disminuyendo el consumo de energía requerido para transmitir un dato correctamente (Rosas y Oberli, 2012).

La utilización de tecnologías MIMO (*Multiple-Input Multiple-Output*) permitiría obtener un enlace de mayor alcance sin aumentar el consumo energético (para obtener una red con mayor área de cobertura) o reducir el consumo energético de un enlace manteniendo su alcance constante, logrando una red más eficiente que una red SISO (*Single-Input Single-Output*) (Goldsmith, 2005).

Actualmente, no existen WSN cuyos nodos utilicen tecnologías MIMO y tampoco existen transceptores MIMO de bajo consumo a nivel comercial. Asimismo, no es posible modificar un transceptor SISO para que opere con múltiples módulos de RF, debido a que los diseños son cerrados y no permiten adaptar el módem para la operación con múltiples antenas. Por consiguiente, el diseño de un módem compatible con tecnologías MIMO representa un paso fundamental en el diseño de WSN con múltiples antenas.

1.1. Objetivos y resultados esperados

El objetivo del presente trabajo es diseñar e implementar un módem en un lenguaje de descripción de *hardware* que incluya una unidad de control y una interfaz compatible con microcontroladores y módulos de RF disponibles en el mercado.

Una vez finalizado y depurado el diseño del módem, se espera poder verificar su funcionalidad sobre una FPGA (*Field-Programmable Gate Array*) de desarrollo que realice transmisiones y recepciones en banda base validando el trabajo realizado.

El diseño debe ser eficiente en términos energéticos, flexible para su uso en comunicaciones MIMO y compatible con un microcontrolador estándar. Esto permite realizar pruebas comparativas entre distintas tecnologías de transmisión, en términos de alcance, consumo energético y/o tasa de datos. El módem debe soportar tasas de datos variables y múltiples modulaciones, y permitir la inclusión de bloques codificadores.

Asimismo, el diseño sebe ser abierto para permitir modificaciones de funcionalidad acordes a futuros cambios de requerimientos.

1.2. Organización del documento

El presente documento está organizado de la siguiente forma: en el Capítulo II se presenta el estado del arte de los microcontroladores y transceptores utilizados en WSN, y de *testbeds* MIMO disponibles. En el Capítulo III se describen los componentes del módem diseñado y la forma en que estos componentes interactúan entre sí. En el Capítulo IV se describe la implementación de cada uno de estos componentes. En el Capítulo V, se presentan los resultados de las simulaciones y la verificación del *hardware*. Finalmente en el Capítulo VI se presentan las conclusiones y el trabajo futuro.

Capítulo 2. ESTADO DEL ARTE

A continuación se describe la arquitectura de un nodo típico utilizado en WSN. También se describen plataformas de prueba MIMO en FPGA, algunas de las cuales implementan parte del transceptor MIMO sobre una FPGA.

2.1. Arquitectura de un nodo

En la Figura 2.1 se muestra la arquitectura de un nodo utilizado en una WSN típica. El microcontrolador está a cargo de controlar las funciones del nodo y sus componentes. Cuenta con una memoria y un controlador USB para almacenar y modificar su configuración.

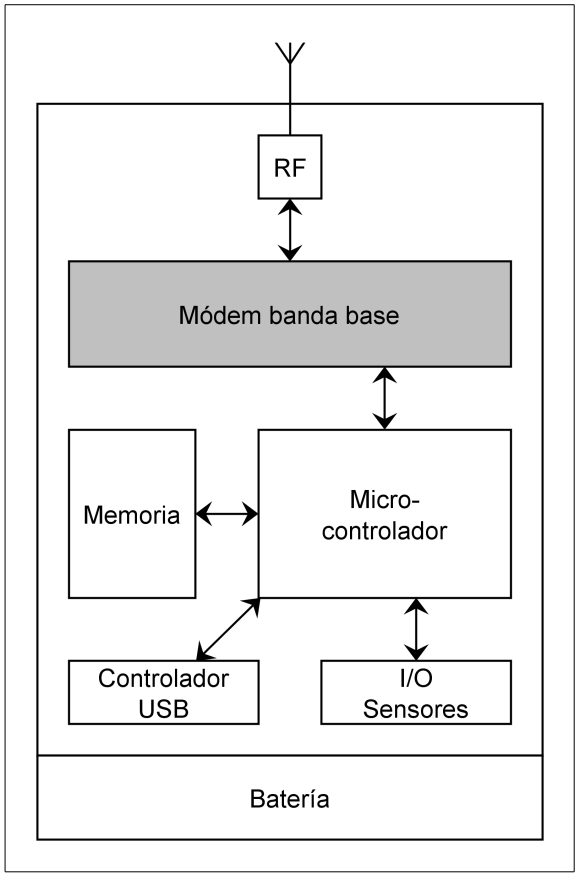

FIGURA 2.1. Arquitectura de un nodo utilizado en una WSN típica.

Los sensores miden variables del ambiente, y éstas son procesadas para ser enviadas a un nodo vecino a través de un enlace inalámbrico, que se logra gracias al módem de banda base y a el módulo de RF. El par módem banda base y módulo de RF es conocido como transceptor.

Dado que el objetivo de esta tesis es el diseño de un módem en banda base, es de particular interés conocer el funcionamiento de varios transceptores disponibles en el mercado y cómo éstos interactúan con un microcontrolador estándar para WSN.

Entre los microcontroladores más utilizados en WSN están MSP430 de Texas Instruments (TI) y ATmega128 de Atmel. Ambos microcontroladores ofrecen la funcionalidad necesaria para operar en una WSN con un bajo consumo energético e interfaces estándar para la conexión con otros dispositivos (Texas Instruments Inc., 2012; Atmel, 2011).

El estándar IEEE-802.15.4 define las funcionalidades de un dispositivo de bajo consumo en las capas física y de acceso al medio (IEEE, 2006). Este estándar es el más utilizado en WSN y algunos transceptores que cumplen con este estándar son el XE1205 de Semtech, AT86RF230 de Atmel, y los CC2420 y CC2520, ambos de TI (Semtech, 2005; Atmel, 2009; Texas Instruments Inc., 2010b; TIK WSN Research Group ETH Zurich, 2012).

Existen diversos nodos comerciales que utilizan una combinación de alguno de estos microcontroladores y transceptores. Algunos nodos que usan la combinación ATmega128-AT86RF230 son Raven de Atmel e Iris Mote de Crossbow. La combinación MSP430-CC2420 es más utilizada que el par anterior, algunos ejemplos son: shimmer de Shimmer Research, PowWow de INRIA CAIRN, SenseNode de GeneTLab y Ubi-Mote2 de C-DAC (TIK WSN Research Group ETH Zurich, 2012). Por este motivo, se considerará esta combinación en el desarrollo del módem de banda base.

A continuación se analiza el nodo Z1 de Zolertia, que utiliza el microcontrolador MSP430 y el transceptor CC2420, ambos de TI.

Nodo Z1 de Zolertia

El nodo Z1 de Zolertia es un módulo inalámbrico de propósito general para el desarrollo e investigación de WSN. Es un nodo flexible y expansible que soporta una gran cantidad de sistemas operativos de código abierto. En la Figura 2.2 se puede ver una fotografía del nodo indicando sus principales componentes (Zolertia, 2011).

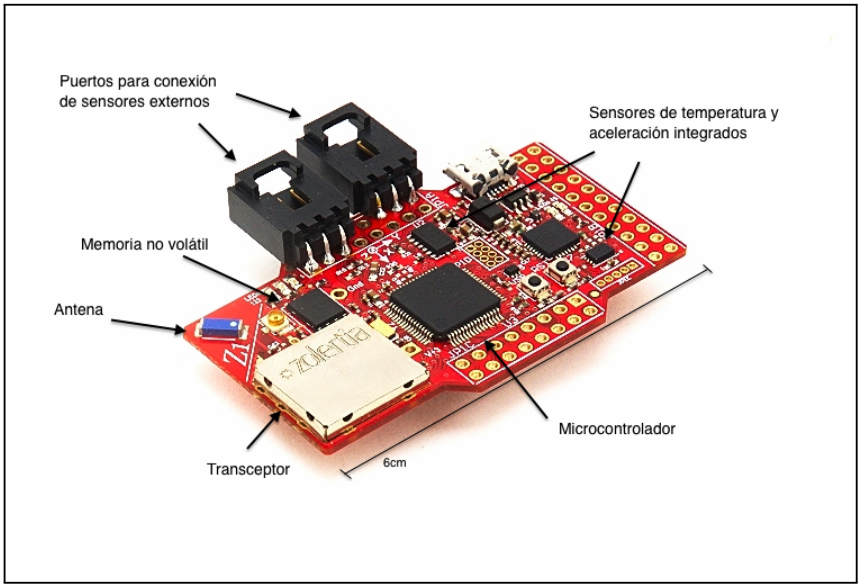

FIGURA 2.2. Nodo Zolertia Z1 (Zolertia, 2011).

Se utilizó el nodo Z1 como referencia debido a su flexibilidad para diseñar y desarrollar aplicaciones y protocolos.

Transceptor Texas Instruments CC2420

El transceptor CC2420 integra un módem en banda base y un módulo de RF compatibles con el estándar IEEE-802.15.4 (Texas Instruments Inc., 2010a) y opera en la banda ISM no licenciada de 2.4 GHz (IEEE, 2006; Subtel MTT, Gobierno de Chile, 2005). Este producto está diseñado para operar en una WSN cumpliendo requerimientos de bajo costo y bajo consumo energético.

En la Figura 2.3 se muestra el diagrama de bloques del transceptor CC2420, donde se puede ver el detalle del módulo de RF, el módem digital, la unidad de control y la interfaz con el microcontrolador. El diseño del módem se basa en el diseño del CC2420,

implementando módulos con funcionalidad similar para todos los bloques que muestra la Figura 2.3 excepto para el módulo de RF.

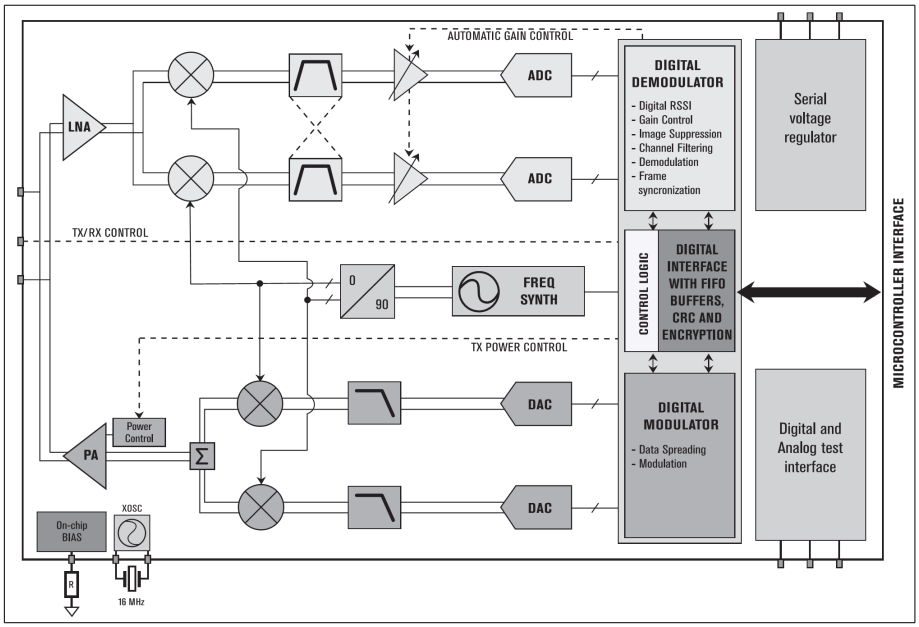

FIGURA 2.3. Diagrama de bloques del transceptor CC2420 (Texas Instruments Inc., 2010a), con el detalle del módulo de RF (izquierda) y el módem banda base con su unidad de control (derecha).

El transceptor CC2420 se comunica con el microcontrolador a través de una interfaz SPI que se describe a continuación.

Serial Parallel Interface Bus

La interfaz *Serial Parallel Interface Bus* (en adelante SPI) fue desarrollada por Motorola y permite comunicar datos en forma serial entre dos o más componentes (Motorola Inc., 2003).

Una conexión SPI cuenta con un dispositivo maestro y uno o más esclavos. La comunicación es *full-duplex* (bi-direccional y simultánea) sin necesidad de que cada dispositivo tenga una dirección o identificación única, lo que facilita agregar o eliminar periféricos sin necesidad de intervenir todos los componentes. En la Figura 2.4 se muestra la conexión entre un maestro y dos esclavos, cuyas señales de comunicación son descritas en la Tabla 2.1.

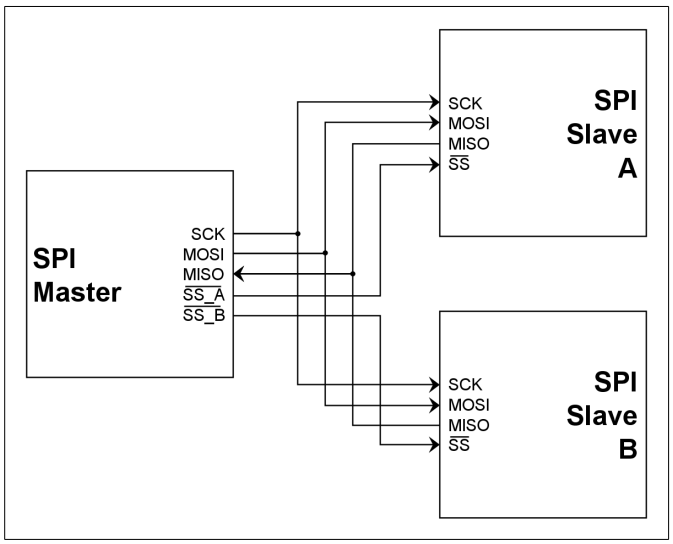

FIGURA 2.4. Diagrama de conexión SPI entre un maestro (izquierda) y dos esclavos (derecha).

La interfaz SPI se utiliza en una gran variedad de productos electrónicos debido a sus bajos requisitos de *hardware* y a la flexibilidad para conectarse a múltiples dispositivos a distintas tasas de datos. Otras alternativas, como I^2C, son más simples, pero no ofrecen transmisiones *full duplex* ni la velocidad y flexibilidad de SPI (NXP Semiconductors, 2012).

Conexión	Descripción
SCK	Determina la velocidad de la comunicación, depende del maestro.
MOSI	Transmite en forma serial los bits desde el maestro hacia el esclavo.
MISO	Recibe en forma serial los bits desde el esclavo hacia el maestro.
SS	Habilita la comunicación entre el maestro y algún esclavo.

TABLA 2.1. Conexiones de la interfaz SPI.

2.2. Plataformas experimentales MIMO en FPGA

Una plataforma experimental, o *testbed*, se utiliza para medir el rendimiento de un sistema de comunicaciones bajo distintos escenarios. Los *testbed* sobre FPGA permiten realizar pruebas sobre *hardware* real, comunicando distintos dispositivos a través de canales inalámbricos reales o simulados. Estas plataformas de prueba son buenos referentes para el diseño de módems sobre FPGA.

El trabajo de (Azami, Ghorssi, Hemesi, Mohammadi, y Abdipour, 2008) muestra un módem MIMO controlado por un computador, donde la FPGA se reserva al procesamiento de datos en banda base y control de módulos de RF. Este *testbed* opera sobre la FPGA de desarrollo Spartan-3, muy similar a la utilizada en este proyecto, y se construyó con el objetivo de evaluar el rendimiento de un enlace MIMO 4×4 en un ambiente real. El estudio de (Kim y Torlak, 2008) describe un trabajo similar, enfocado en comparar esquemas de transmisiones MIMO realizando análisis en línea, utilizando FPGAs más rápidas y con más compuertas.

El *testbed* diseñado en (Bialkowski y Uthansakul, 2011) incluye un emulador de canal MIMO 2×2 con el fin de medir el rendimiento de esquemas de trasmisión Alamouti, realizando un análisis fuera de línea.

La Tabla 2.2 muestra un resumen sobre las FPGA utilizadas en los distintos *testbeds*

Autor(es)	FPGA utilizada	Cantidad	Canal	Procesamiento
(Azami y cols., 2008)	Xilinx Spartan-3	2	Real	fuera de línea
(Kim y Torlak, 2008)	Xilinx Virtex-II Pro	5	Real	en línea
(Bialkowski y Uthansakul, 2011)	Altera Stratix II	1	Emulado	fuera de línea

TABLA 2.2. Comparación entre las plataformas experimentales investigadas.

Los *testbeds* investigados ofrecen funcionalidades que abarcan áreas que exceden los objetivos de esta tesis, y logran muchos de sus objetivos con la ayuda de múltiples FPGA de desarrollo operando en paralelo, manejadas por un computador a cargo del control y el

procesamiento de la información. Sin embargo, sirven como referencia para determinar la capacidad de la FPGA que se debe utilizar para implementar este proyecto. En particular, (Azami y cols., 2008) utiliza una FPGA similar a la elegida para este trabajo.

Capítulo 3. DESCRIPCIÓN GENERAL DEL MÓDEM

Este capítulo describe la estructura del módem y los componentes requeridos para su funcionamiento. También se describe la operación individual de cada uno de estos componentes y el flujo de información entre ellos.

3.1. Descripción de los componentes del sistema

El módem está compuesto por cuatro componentes básicos que se muestran en la Figura 3.1: un módulo de interfaz con el microcontrolador, una unidad de control, un conjunto de memorias y un bloque modulador/demodulador.

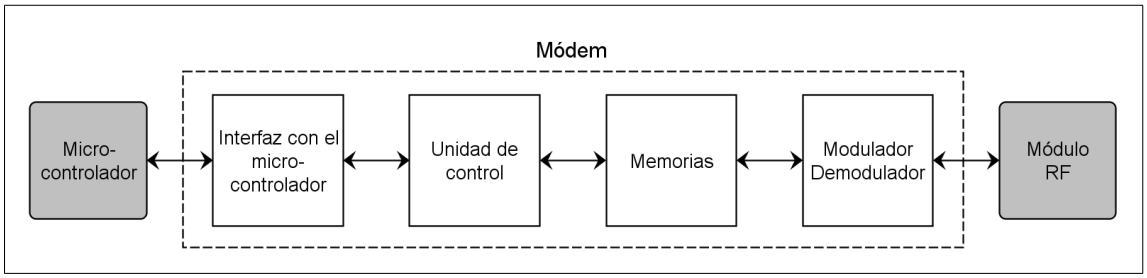

FIGURA 3.1. Diagrama de bloques del módem. Se detallan los bloques principales y su interfaz con el microcontrolador y el módulo de RF.

A través de la interfaz con el microcontrolador, se reciben las instrucciones y datos que serán entregados a la unidad de control, quien controla las funciones internas del módem y los módulos de RF. Esta unidad además controla las memorias del módem que almacenan la configuración y los paquetes de datos recibidos y por transmitir. Finalmente un bloque modulador/demodulador se encarga de procesar estos paquetes de datos y convertirlos en señales de banda base digital para ser transmitidos por el o los módulos de RF.

A continuación se detallan las características y funcionalidades de cada uno de los componentes del módem.

Interfaz con el microcontrolador

La comunicación entre el módem y el microcontrolador se realiza a través del estándar SPI. Se utiliza esta interfaz porque, como ya se mencionó antes, es un puerto estándar en diversos microcontroladores disponibles actualmente en el mercado.

En el nodo, el microcontrolador actúa como SPI maestro para comunicarse con distintos periféricos. Por lo tanto, en el módem corresponde un módulo SPI esclavo.

Unidad de control

La unidad de control, denominada *dispatcher*, recibe instrucciones desde el módulo SPI esclavo para procesarlas y ejecutarlas. El *dispatcher* debe ser compatible con la interfaz mencionada anteriormente y con las memorias que reciben la información y datos de las instrucciones.

Memorias

Al interior del módem existen dos bloques de memoria. Un bloque está basado en registros y el otro es una memoria de acceso aleatorio (en adelante RAM, *Random Access Memory*). Ambos bloques cumplen distintas funciones que se describen a continuación y comparten la estructura típica de una memoria como se muestra en la Figura 3.2. Las conexiones se describen en la Tabla 3.1.

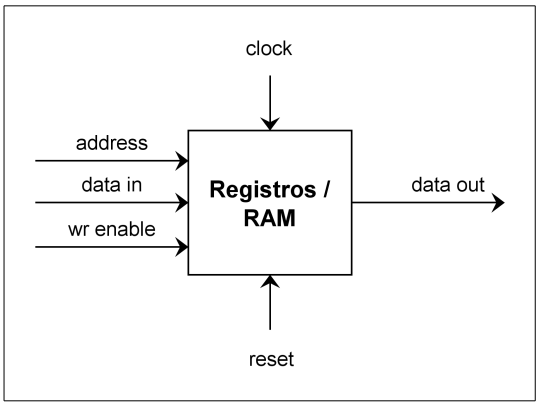

FIGURA 3.2. Diagrama de bloques de una memoria típica.

La memoria basada en registros se utiliza para almacenar la configuración del módem y otros periféricos. Los registros pueden ser leídos en cualquier momento por los componentes del módem. Así, al asignar variables a los valores presentes dentro de estos registros, es posible almacenar configuraciones dinámicas para el módem y para los módulos de RF.

Las RAM son muy utilizadas debido a que permiten almacenamiento temporal de alta velocidad. En el módem, una porción de la RAM se organiza con una estructura de lectura/escritura FIFO (*First In, First Out*) para el procesamiento de paquetes enviados y recibidos.

Conexión	Descripción
reset	Reinicia la memoria, incluyendo sus *outputs*.
clock	Determina la velocidad de lectura y escritura en la memoria.
address	Indica la dirección de memoria que se desea leer o escribir.
data in	Es el puerto por donde se escriben los datos en la memoria.
data out	Es el puerto por donde se leen los datos desde la memoria.
wr enable	Habilita la escritura en memoria. La lectura se habilita al desactivarse la escritura.

TABLA 3.1. Conexiones de una memoria típica, tanto para una RAM como para un arreglo de registros.

En una memoria con estructura FIFO la primera información en ser escrita es la primera en ser leída. Debido a que en una memoria con estructura FIFO los datos pueden tener distintas velocidades de entrada y salida, son útiles como *buffer* de datos. En el módem, se utilizan dos memorias FIFO como *buffer* para almacenar temporalmente paquetes de datos por enviar (FIFO de transmisión) y paquetes de datos recibidos (FIFO de recepción).

La implementación más utilizada de memorias FIFO es la cola circular. En la Figura 3.3 se muestra la representación gráfica de una FIFO de cola circular que almacena hasta

8 elementos, y se detalla la posición de los punteros de lectura y escritura (rd ptr y wr ptr, respectivamente) después de realizar operaciones básicas.

La Figura 3.4 muestra el diagrama de bloques de la FIFO utilizada en el diseño del módem. Sus señales de comunicación son distintas a las de una memoria típica debido a que no hay un puerto de dirección, ya que éste se maneja automáticamente a través de los punteros de lectura y escritura. Además, en una FIFO existen dos *clocks*, uno para los entrada de datos y otro para la salida de datos. También existen dos puertos distintos para habilitar la escritura y lectura. Las señales de comunicación de una FIFO típica se escriben en la Tabla 3.2.

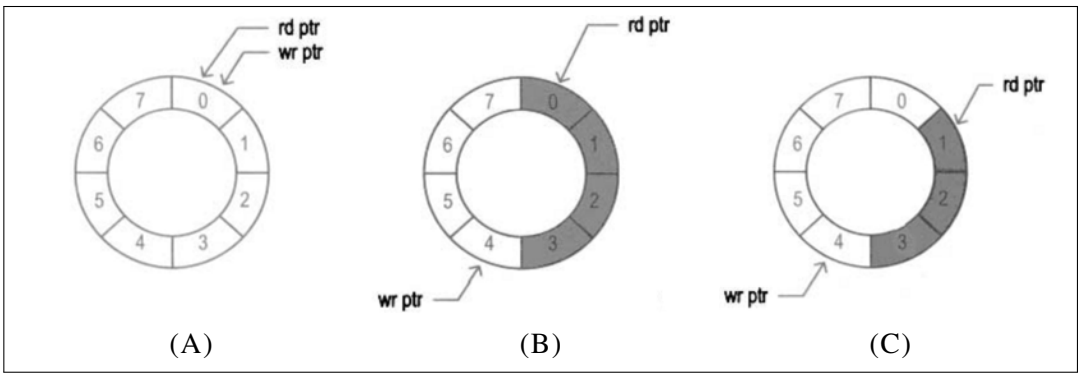

FIGURA 3.3. Estados de una FIFO circular: inicial (A), tras escribir 4 elementos (B) y después de leer 1 elemento (C) (Chu, 2011).

.

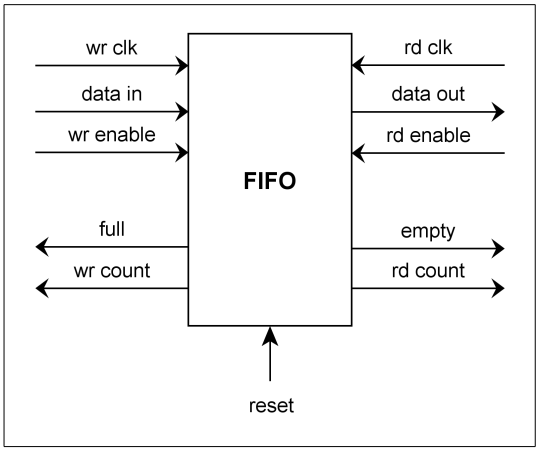

FIGURA 3.4. Diagrama de bloques de una FIFO. Las señales de la izquierda dependen de wr clk y las señales de la derecha de rd clk.

Conexión	Descripción
reset	Reinicia la FIFO: vacía la memoria y mueve los punteros de lectura y escritura a cero.
wr clk	Determina la velocidad de escritura.
rd clk	Determina la velocidad de lectura.
data in	Es el puerto por donde ingresan los datos.
data out	Es el puerto por donde egresan los datos.
wr enable	Habilita la escritura de la FIFO.
rd enable	Habilita la lectura de la FIFO.
full	Indica que la memoria de la FIFO está llena.
empty	Indica que la memoria de la FIFO esta vacía.
wr count	Reporta la cantidad de elementos escritos en la memoria de la FIFO.
rd count	Reporta la cantidad de elementos disponibles para leer desde la FIFO.

TABLA 3.2. Conexiones de una memoria con estructura FIFO típica.

Las FIFOs permiten puertos de escritura (Data IN) y lectura (Data OUT) de distintos tamaños (esto se denomina FIFO asimétrica). Por ejemplo, en una FIFO se pueden escribir *bytes* en forma serial y leerlos en forma paralela. La FIFO de recepción del módem es asimétrica porque el módem recibe símbolos en forma serial desde el módulo de RF y los envía en forma paralela a la unidad de control del módem.

También es posible utilizar distintos *clocks* para obtener distintas velocidades de lectura y escritura (rd clk y wr clk en la Figura 3.4). La FIFO de transmisión del módem utiliza dos *clocks* distintos porque la velocidad de procesamiento del módem puede ser distinta a la velocidad de transmisión requerida (Xilinx Inc., 2012b, 2012c).

Modulador y demodulador

El modulador entrega señales digitales en fase y cuadratura al módulo de RF. Por su parte, el módulo de RF se encarga de convertir estas señales digitales en señales analógicas y transmitirlas mediante ondas electromagnéticas.

De manera similar, en la recepción el demodulador toma secuencias de símbolos en fase y cuadratura, y los convierte en mensajes binarios. Los mensajes binarios son almacenados en la FIFO de recepción para luego ser leídos por el microcontrolador a través de la unidad de control del módem.

Los esquemas de modulación y demodulación incorporados al módem son QPSK, 16-QAM y 64-QAM, con constelaciones especificadas por el estándar IEEE-802.11-2007 (IEEE, 2003, 2009) (Ver anexo B).

3.2. Flujo de información entre componentes

En esta sección se describe la interfaz entre los distintos componentes del módem, detallando las señales de datos, configuración y control que estos componentes intercambian. En la Figura 3.5 se detallan las conexiones desde el microcontrolador hasta las memorias del módem, incluyendo todos los módulos intermedios. Al final de esta sección se presenta el detalle de las conexiones desde las memorias hasta los módulos de RF.

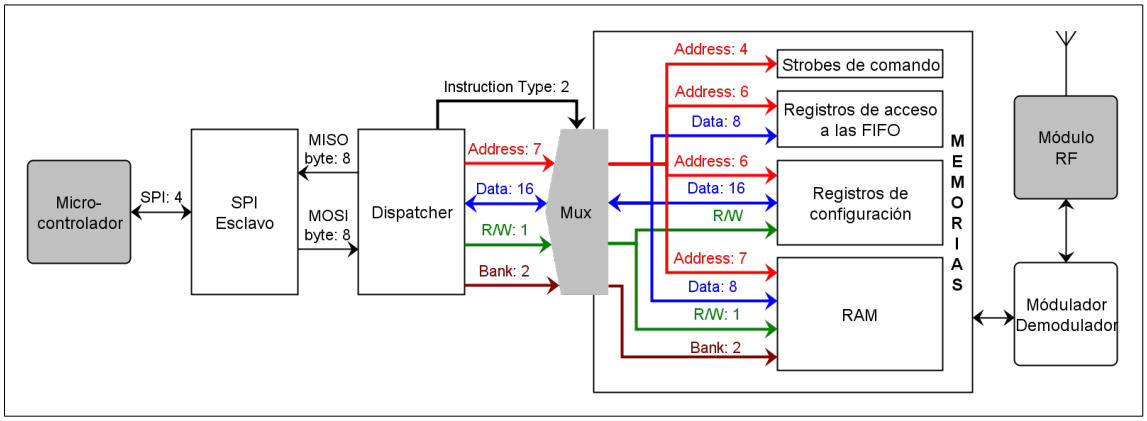

FIGURA 3.5. Diagrama de alto nivel del módem. Los números a la derecha de cada etiqueta indican el tamaño de cada uno de los buses.

Interfaz entre microcontrolador y SPI esclavo

Como se explicó en la sección 3.1, el microcontrolador interactúa con el módem a través de una interfaz SPI.

El microcontrolador envía instrucciones en forma de *bytes* hacia el módem a través del módulo SPI. La funcionalidad *full duplex* de SPI permite que el microcontrolador pueda recibir *bytes* desde el módem para conocer el estado y configuración del módem y el módulo de RF cada vez que el microcontrolador lo solicite.

Interfaz entre SPI esclavo y *dispatcher*

Los *bytes* recibidos desde el módulo SPI esclavo son analizados y clasificados en el *dispatcher* según cuatro tipos de instrucciones:

***Command Strobe*:** Permite activar los registros de control que activan y desactivan diversas funciones del módem, tales como recepción y transmisión, entre otras.

***FIFO Access*:** Permite escribir en la FIFO de transmisión o leer desde la FIFO de recepción del módem.

***Register Access*:** Permite leer o escribir en los registros de configuración.

***RAM Access*:** De manera similar a *Register Access*, permite leer o escribir en la RAM del módem, excluyendo la porción dedicada a la FIFO.

Una vez que el *dispatcher* clasifica las instrucciones, procede a analizar bit a bit cada uno de los *bytes* recibidos, para extraer la información correspondiente (ver Tabla 3.3). De acuerdo al tipo de instrucción, esta información puede contener:

***Address*:** Indica la dirección de memoria que se desea leer o escribir (según corresponda).

R/W: Determina si la instrucción es una operación de lectura o escritura.

***Data*:** Corresponde a los datos leídos o escritos. Dependiendo del tipo de instrucción pueden ser uno o dos *bytes*.

***Bank*:** Define el banco de la RAM al que se desea acceder.

El valor de *address* determina el tipo de instrucción. En el caso de *FIFO Access*, también determina si se trata de una operación de lectura o escritura.

Tipo de instrucción	Address	R/W	Data	Bank	Memoria de destino
Command Strobe	✓				Strobes de comando
FIFO Access	✓		✓		Registros de acceso a las FIFO
Register Access	✓	✓	✓		Registros de configuración
RAM Access	✓	✓	✓	✓	*Random Access Memory*

TABLA 3.3. Información contenida en cada tipo de instrucción junto con la memoria de destino de dicha información. En el caso de *Command Strobe* y *FIFO Access*, la información de R/W está incluida en el valor de *address*.

Interfaz entre *dispatcher* y memorias

Dependiendo del tipo de instrucción, los datos decodificados por el *dispatcher* son enviados a alguna de las memorias del módem según corresponda.

Existen cuatro tipos de memoria. Los primeros tres son memorias basadas en registros y se describen a continuación.

***Strobes* de comando:** Almacena temporalmente instrucciones para ejecutar operaciones tales como leer el estado de algún componente del módem, encender o apagar módulos de RF, reiniciar memorias, vaciar las FIFO, comenzar una transmisión, entre otras.

Registros de acceso a las FIFOs: Almacenan temporalmente el *byte* por escribir en la FIFO de transmisión y el *byte* por leer desde la FIFO de recepción.

Registros de configuración: Almacenan información asociada a la operación del módem y los módulos de RF, que incluye la modulación a utilizar, la cantidad de antenas de transmisión y recepción, entre otros. Estos registros se pueden leer directamente en todo momento.

El cuarto tipo de memoria corresponde a la RAM, dividida en tres bancos que se acceden según el valor de la variable *Bank*. Estos bancos son:

FIFO de transmisión: Almacena los *bytes* pendientes antes de ser modulados para luego ser transmitidos.

FIFO de recepción: Almacena los *bytes* ya demodulados, que están pendientes por ser leídos por el microcontrolador.

RAM *Security*: Corresponde al espacio restante de la RAM. Se utiliza para almacenar variables útiles para la transmisión y capas superiores, como identificación del nodo, llaves de encriptación, etc.

Interfaz entre memorias, modulador-demodulador y RF

La Figura 3.6 muestra el detalle de la interfaz entre las memorias y los bloques moduladores y demoduladores, junto con su respectiva conexión con el o los módulos de RF.

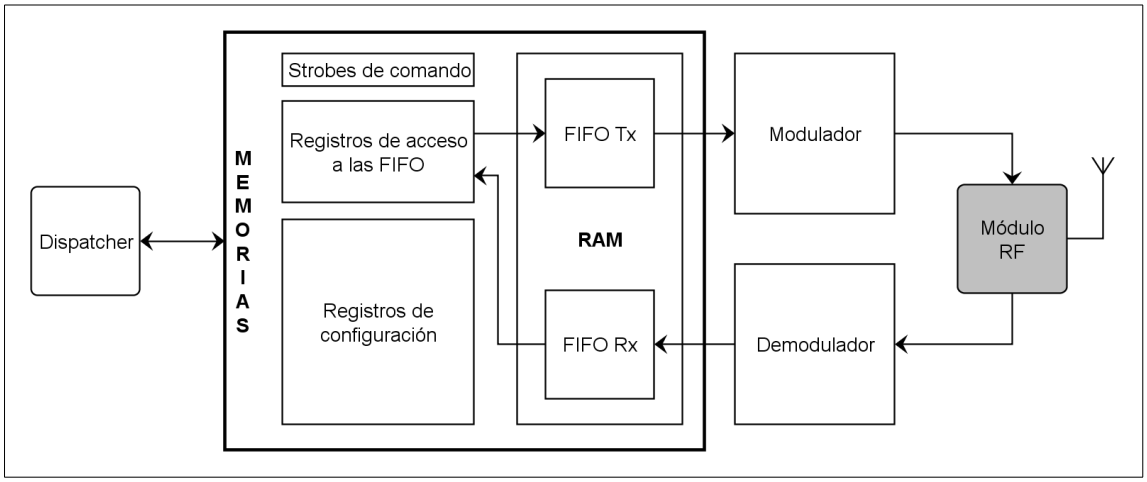

FIGURA 3.6. Detalle de la conexión entre las memorias, el modulador y demodulador, y el módulo de RF.

Como se explicó anteriormente, el módem cuenta con dos FIFOs que actúan como *buffers* para almacenar el paquete de datos por enviar y el paquete de datos recibido. Estos *buffers* se conectan directamente al modulador y demodulador en forma serial, debido a que la cantidad de bits por símbolo enviado o recibido depende del tipo de modulación utilizada. De este modo se asegura compatibilidad con una gran cantidad de modulaciones.

La conexión entre el modulador-demodulador y el módulo de RF es una conexión paralela de ancho variable dentro de una resolución arbitraria, ajustable según las necesidades y restricciones de la implementación. Si el módulo de RF tiene una interfaz de entrada distinta, el diseño modular del módem permite reemplazar esta interfaz de salida sin necesidad de cambiar la estructura interna del módem.

Capítulo 4. IMPLEMENTACIÓN DEL MÓDEM

Este capítulo describe la implementación de los componentes del sistema y muestra algunos ejemplos de su funcionamiento.

La implementación se realizó utilizando el lenguaje de descripción de *hardware* Verilog (IEEE, 2005) y se compiló utilizando el *software* ISE Project Navigator, parte de la colección ISE Design Suite: System Edition versión 14.1 de Xilinx Inc. (Xilinx Inc., 2012a). Se optó por utilizar esta colección de *software* debido a la compatibilidad garantizada con las FPGA de desarrollo utilizadas.

Se utilizaron las FPGA Nexys 2 Spartan-3E 500 (Spartan-3E en adelante) de Digilent Inc. (Xilinx Inc, 2011).

Debido a los diversos requerimientos que los componentes del módem poseen, se utilizaron distintas referencias como material de apoyo para su implementación. La sintaxis básica de Verilog (Palnitkar, 2003; Williams, 2008), implementación de las memorias (Navabi, 2005) y finalmente, la implementación en FPGA (Chu, 2011).

4.1. SPI esclavo

El módulo *SPI Slave* implementado tiene los 4 puertos definidos por el estándar (SCK, MOSI, MISO y SS) conectados al *SPI Master* del microcontrolador, y dos buses de 8 bits conectados al *dispatcher* que comunican el *byte* enviado por el microcontrolador hacia el *dispatcher* (*Byte MOSI*) y el *byte* enviado por el *dispatcher* hacia el microcontrolador (*Byte MISO*). Adicionalmente, se consideró incluir un *flag* de *byte* recibido que le indica al *dispatcher* cuando se ha recibido un mensaje (*Byte rec*, ver Figura 4.1).

Al interior del *SPI Slave* existe un módulo serie-paralelo que convierte la entrada serial MOSI a la salida paralela *Byte MOSI*, y de forma inversa, un módulo paralelo-serie que convierte la entrada *Byte MISO* a la salida serial MISO. Ambos módulos son controlados por un detector de flancos de subida que depende de SCK (*clock* de la comunicación).

El *flag Byte rec* está asociado a un contador que aumenta con cada flanco de subida de SCK. Cuando este contador es igual a 8, el *flag Byte rec* le indica al resto del sistema que se ha recibido un *byte* de instrucción por parte del microcontrolador. El contador y el *flag* se reinician en cero inmediatamente, como se muestra en el diagrama de tiempos de la Figura 4.2.

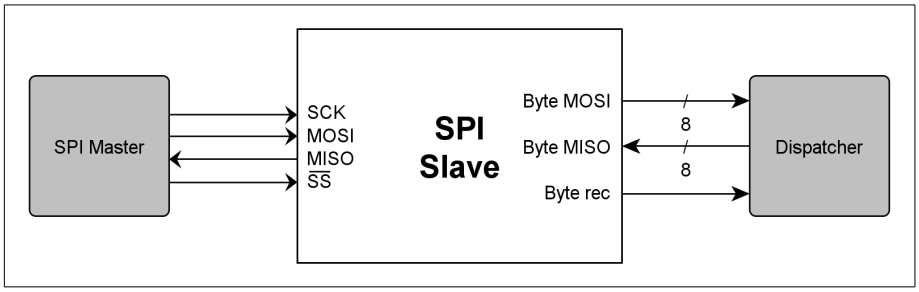

FIGURA 4.1. Conexiones del módulo SPI esclavo, a la izquierda la conexión con el microcontrolador y a la derecha la conexión con el *dispatcher*.

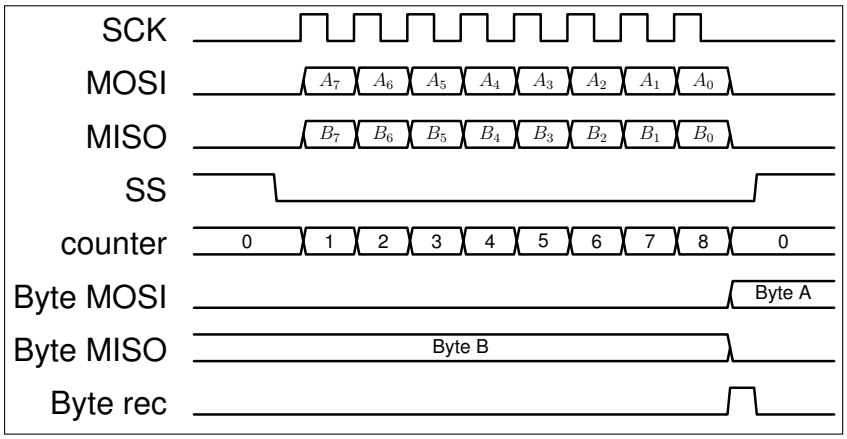

FIGURA 4.2. Detalle de la comunicación SPI. Durante la comunicación, el *byte* B es serializado y enviado hacia el microcontrolador. Después de la comunicación, el *byte* A (recibido desde el microcontrolador) se puede leer en forma paralela.

4.2. *Dispatcher*

El *dispatcher* recibe los *bytes* leídos por el módulo SPI, y los decodifica de acuerdo al tipo de instrucción recibida como se detalló en la sección 3.2. el control del *dispatcher* utiliza la máquina de estados de Mealy que se muestra en la Figura 4.3. Para implementar

la máquina de estados de manera eficiente se siguió la metodología *"Two always block style"* descrita en (Cummings, 2002).

La máquina de estados consta de cuatro estados: un estado IDLE y tres estados de *bytes* recibidos denominados: 1^{ST}, 2^{ND} y 3^{RD} (ST, ND y RD en adelante). Cada uno de estos estados extrae desde los *bytes* de instrucciones distintos datos dependiendo del tipo de instrucción como se indica en la Tabla 4.1. Luego, las instrucciones y datos decodificados se envían a los componentes del módem que corresponda.

Tipo de Instrucción	*bytes*	1^{er} *byte*	2^{do} *byte*	3^{er} *byte*
Strobe Command	1	*Address*	—	—
FIFO Access	2	*Address* + R/W	Data	—
Register Access	3	*Address* + R/W	Data	Data
RAM Access	3	*Address*	*Bank* + R/W	Data

TABLA 4.1. Tipos de instrucciones junto con la información contenida por *byte* recibido.

Por ejemplo, si se quiere escribir un registro, el microcontrolador debe enviar una secuencia de 3 *bytes*. El primer *byte* debe contener la dirección del registro que se desea escribir junto con una orden de escritura, el segundo y tercer *byte* se utilizan para transmitir los 16 bits por escribir. En este caso la máquina de estados comienza en IDLE, pasa por ST, ND y RD. Una vez finalizada la transmisión vuelve al estado IDLE.

El detalle de la información correspondiente a cada bit de las instrucciones se describe en el Anexo A.

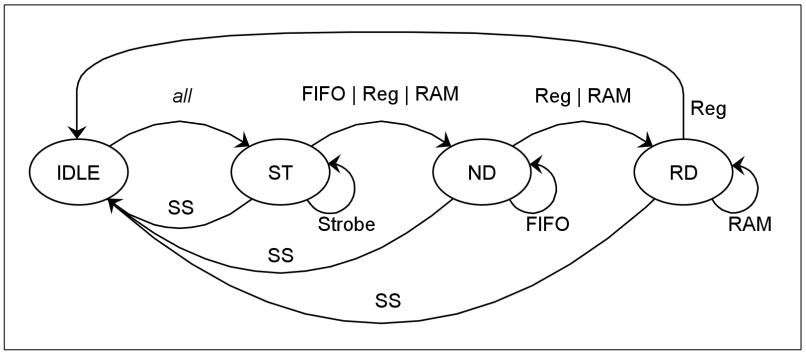

FIGURA 4.3. Máquina de estados para el control del *dispatcher*.

4.2.1. Ejemplos de instrucciones

En la Tabla 4.2 se listan y describen algunos *strobes* de comando y en la Tabla 4.3 se muestran algunos ejemplos para cada uno de los tipos de instrucciones. El diseño abierto del módem permite modificar las funciones de los *strobes* e incluso incluir nuevas funciones utilizando los *strobes* que no son mencionados en la Tabla 4.2.

La Tabla 4.3 muestra los *bytes* de las instrucciones de acceso a las memorias. Los ejemplos para acceso a registros y RAM son genéricos y muestran el rango de direcciones que es posible leer o escribir. Los *bytes* de datos pueden tener cualquier valor. El primer *byte* determina el tipo de instrucción y es único para cada tipo de instrucción.

Instrucción	dec	bin	Descripción
No operation	0	00000000	Lee el *byte* de estado sin realizar ninguna otra operación al interior del módem
Demod enable	3	00000011	Activa la recepción en el módem
Mod enable	4	00000100	Activa la transmisión en el módem
RF enable	5	00000101	Activa los módulos de RF. Ésta se desactiva al finalizar una transmisión o al desactivar la recepción
Demod disable	6	00000110	Desactiva la recepción
Mod disable	7	00000111	Finaliza manualmente la transmisión
Reset Rx	8	00001000	Reinicia la rama de recepción
Reset Tx	9	00001001	Reinicia la rama de transmisión

TABLA 4.2. *Strobes* de comando del módem y su representación en binario y decimal. Las direcciones omitidas están disponibles para incluir funciones nuevas.

4.2.2. Concatenación de instrucciones

El diseño del *dispatcher* permite que el módem pueda recibir múltiples instrucciones en forma consecutiva para ahorrar tiempo y simplificar la comunicación con el microcontrolador, contribuyendo a disminuir el consumo energético. La Figura 4.4 muestra las combinaciones de instrucciones posibles de concatenar. En la Tabla 4.4 se detalla el comportamiento de la máquina de estados del *dispatcher* cuando se concatena una misma instrucción varias veces.

Tipo	Instrucción	dec	bin
FIFO Access	FIFO Tx Write	62 Datos	00111110 xxxxxxxx
	FIFO Rx Read	127 Datos	01111111 xxxxxxxx
Register Access	Register Write	16 – 61 Datos Datos	00010000 – 00111101 xxxxxxxx xxxxxxxx
	Register Read	80 – 125 Datos Datos	01010000 – 01111101 xxxxxxxx xxxxxxxx
RAM Access	RAM Write	128 – 240 64 Datos	10000000 – 11110000 01000000 xxxxxxxx
	RAM Read	128 – 240 96 Datos	10000000 – 11110000 01100000 xxxxxxxx

TABLA 4.3. Instrucciones de lectura y escritura de memorias, y su representación en binario y decimal. Los datos pueden tener cualquier valor entre 0 y 255.

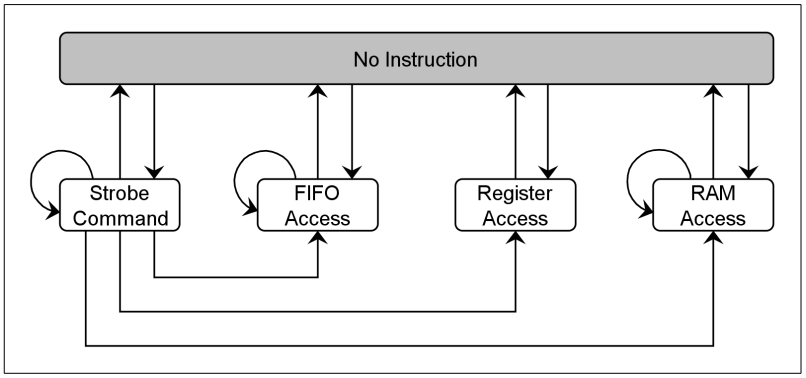

FIGURA 4.4. Diagrama de concatenación de instrucciones, tanto para instrucciones iguales como diferentes.

Es importante notar que hay ciertas combinaciones de instrucciones que no es posible concatenar. Por ejemplo, no se puede enviar un *Strobe Access* después de un *FIFO Access* ya que el *dispatcher* no puede diferenciar un *byte* de datos de un *byte* de instrucciones en esta situación, por lo tanto, el *byte* de instrucción *Command Strobe* sería grabado en

Instrucciones consecutivas	Secuencia de estados del *dispatcher*
Command Strobe	IDLE $\rightarrow$ ST $\rightarrow$ ST $\rightarrow \ldots \rightarrow$ ST $\rightarrow$ IDLE
FIFO Access	IDLE $\rightarrow$ ST $\rightarrow$ ND $\rightarrow$ ND $\rightarrow \ldots \rightarrow$ ND $\rightarrow$ IDLE
Register Access	IDLE $\rightarrow$ ST $\rightarrow$ ND $\rightarrow$ RD $\rightarrow$ IDLE
RAM Access	IDLE $\rightarrow$ ST $\rightarrow$ ND $\rightarrow$ RD $\rightarrow$ RD $\rightarrow \ldots \rightarrow$ RD $\rightarrow$ IDLE

TABLA 4.4. Comportamiento de la máquina de estados del *dispatcher* cuando se concatenan instrucciones del mismo tipo.

la FIFO de transmisión y la instrucción no sería ejecutada. Para conseguir que la combinación anterior se realice correctamente habría que finalizar la transmisión SPI entre las instrucciones para que el control del *dispatcher* pase al menos una vez por el estado IDLE.

A continuación se presentan tres ejemplos de concatenación de instrucciones: una concatenación de *Strobes* seguida de un acceso simple a una FIFO, una concatenación de accesos a una FIFO y a una RAM, y finalmente un acceso no concatenado a un registro. En todos los ejemplos se muestran las señales del módulo SPI, junto con el estado actual del *dispatcher* (FSM State).

1. Para concatenar *Strobes* basta enviarlos de forma consecutiva. De este modo la máquina de estados del *dispatcher* se mantendrá en el estado ST hasta que finalice la transmisión SPI del microcontrolador o se envíe una instrucción distinta. En la Figura 4.5 se muestra la concatenación de dos *Strobes* seguidos de un acceso simple a una FIFO.

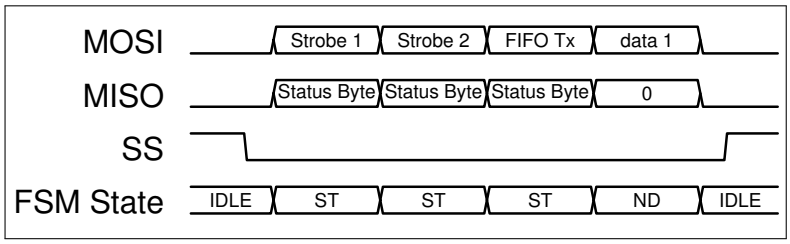

FIGURA 4.5. Dos accesos a un *Strobe* seguidos de un acceso de escritura simple a la FIFO de transmisión.

2. Para concatenar accesos a una FIFO, el *dispatcher* recibe el primer *byte* de instrucción, que indica la FIFO a la que se requiere acceder. Los siguientes *bytes* recibidos son considerados datos por almacenar o leer, quedando el *dispatcher* en el estado ND. Este acceso termina cuando el microcontrolador finaliza la comunicación a través del módulo SPI (ver Figura 4.6).

   El acceso múltiple a la memoria RAM opera de forma similar. Los dos primeros *bytes* indican la dirección de memoria a acceder. Cada *byte* recibido a continuación se lee/escribe en la dirección siguiente (ver señal *RAM Address* de la Figura 4.7).

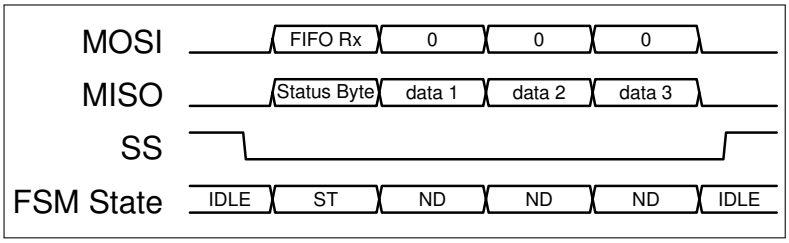

FIGURA 4.6. Tres accesos de lectura a la FIFO de recepción. Esta operación requiere de 4 *bytes*. Si no se concatenan instrucciones, se requieren 6 *bytes*.

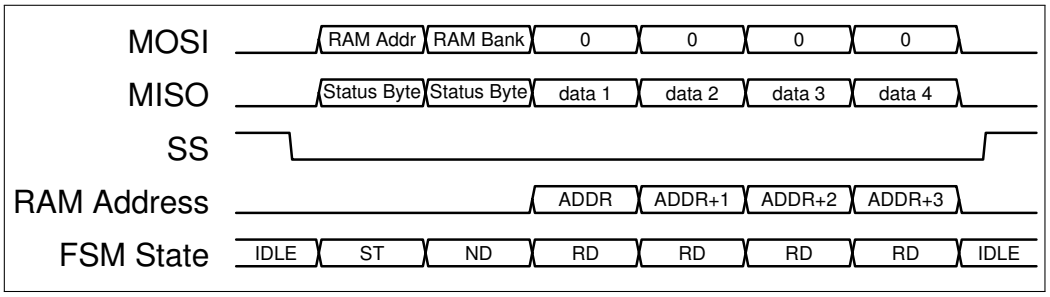

FIGURA 4.7. Cuatro accesos de lectura a la RAM. Esta operación requiere de 6 *bytes*. Si no se concatenan instrucciones, se requieren 12 *bytes*.

3. El último ejemplo corresponde a un acceso no concatenado a un registro. Como estas operaciones no se pueden concatenar, es necesario emitir las instrucciones de acceso por completo cada vez, como se muestra en la Figura 4.8.

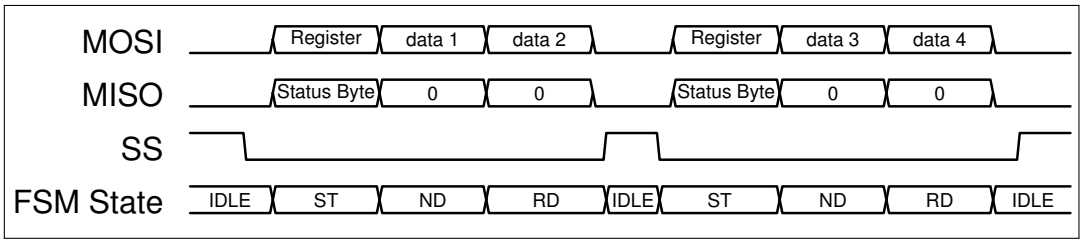

FIGURA 4.8. Múltiples accesos de escritura a un registro. Es necesario pasar por el estado IDLE antes de procesar la segunda instrucción.

4.3. Memorias

Como se explicó en la sección 3.1, existen dos tipos de memoria, aquellas basadas en registros y la RAM. Cada uno de estos tipos contiene tres grupos de direcciones que cumplen distintas funciones. Cada uno de estos seis grupos almacena un número distinto de palabras, y estas palabras a su vez son de distinto tamaño. El detalle de las dimensiones de ambos tipos de memoria se muestra en las Tablas 4.5 y 4.6.

Rango de direcciones	Nombre del grupo	Largo de palabra
0 – 15	*Strobes* de comando	1 bit
16 – 61	Registros de configuración	16 bits
62 – 63	Registros de acceso a las FIFOs	8 bits

TABLA 4.5. Estructura de memoria basada en registros. El rango de direcciones puede ser cubierto con 6 bits.

Rango de direcciones	Nombre del banco	Largo de palabra
0 – 127	FIFO de transmisión	8 bits
128 – 255	FIFO de recepción	
256 – 367	RAM de seguridad	

TABLA 4.6. Estructura de memoria RAM. El rango de direcciones puede ser cubierto con 9 bits.

En particular los grupos de la RAM se denominan bancos, con el propósito de no confundirlos con los grupos de registros.

La RAM siempre almacena palabras de 8 bits (1 *byte*) pues su propósito es almacenar datos. Por otra parte, la memoria basada en registros tiene palabras de distinto tamaño dependiendo de la tarea que ejecute cada grupo de memoria. Los *strobes* son de 1 bit porque representan dos estados posibles, si el comando se ejecuta o no. Los registros de configuración necesitan almacenar distintas configuraciones y algunas requieren más bits que otras (por ejemplo: el número M-ario de modulación). En cambio, los registros de acceso a las FIFOs son de 8 bits, que corresponde al tamaño de las palabras almacenadas en la FIFO de transmisión y recepción, que se almacenan en la RAM.

En operación normal del módem no es posible acceder directamente a los bancos de memoria RAM correspondientes a las FIFO de transmisión y recepción (direcciones 0 a 127 y 128 a 255) debido a que su estructura limita la dirección que se puede leer/escribir en un instante dado. La manera apropiada de acceder a las FIFOs es a través de los registros 62 y 63 (Registros de acceso a las FIFOs) que corresponden a los puertos Data IN y Data OUT de las FIFO de transmisión y recepción respectivamente. Esto permite asegurar que los punteros de escritura y lectura de la FIFO operen correctamente.

El tercer banco (RAM de seguridad) se accede directamente a través de instrucciones del tipo *RAM Access*.

4.4. Modulador y demodulador

El módem soporta múltiples esquemas de modulación, utilizando un sub-bloque distinto por cada esquema de modulación. Esto implica que el módem debe ser capaz de seleccionar el sub-bloque correspondiente de acuerdo a la configuración establecida en los registros de configuración.

Se implementó un módulo M-QAM (IEEE, 2009) genérico parametrizado para distintos valores de M (4, 16 y 64). Una señal *Chip Select* (CS), proveniente de los registros de configuración, asegura que tan solo un sub-bloque esté activo a la vez, evitando conflictos

a la salidas del modulador. La Figura 4.9 muestra la conexión de la FIFO de transmisión con los moduladores y la señal CS que los activa, junto con el *Strobe* de comando (Tx enable) que controla el inicio y fin de las transmisiones.

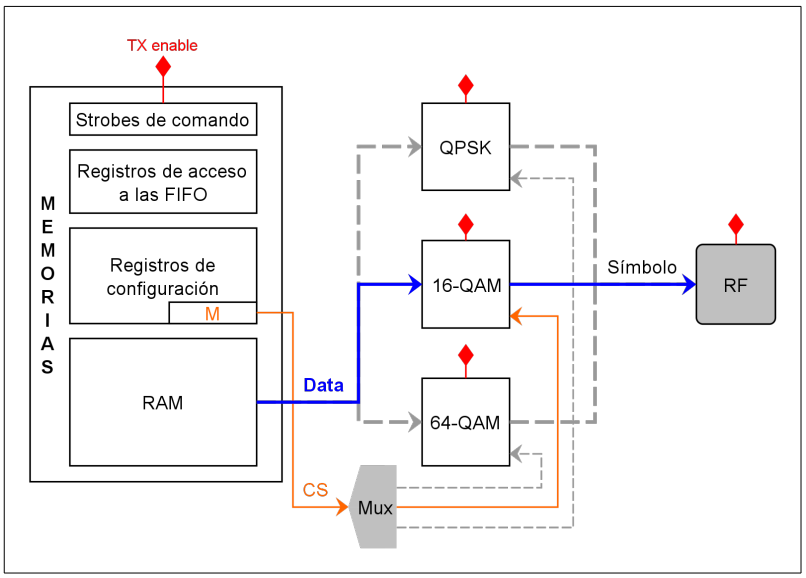

FIGURA 4.9. Conexión entre la FIFO de transmisión, los tres bloques moduladores y el módulo de RF. El acceso de lectura a la FIFO es controlado por el *Strobe* Tx enable que también se conecta con los moduladores y el módulo de RF. En este caso la señal CS sólo activa el modulador 16-QAM mientras que los demás bloques permanecen desactivados.

El demodulador opera en forma análoga al modulador. También se implementó un módulo M-QAM parametrizado para distintos valores de M y se utiliza la misma señal CS para evitar conflictos en la entrada a la FIFO de recepción (ver en la Figura 4.10).

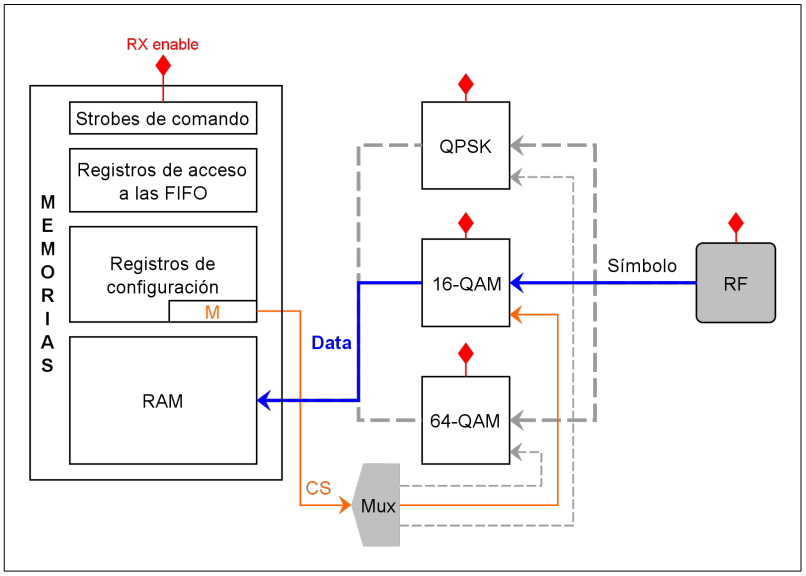

FIGURA 4.10. Conexión entre el módulo de RF, los bloques demoduladores y la FIFO de recepción. El *Strobe* Rx enable se conecta con los demoduladores y el módulo de RF. En este caso la señal CS sólo activa el demodulador 16-QAM mientras que los demás bloques permanecen desactivados.

Capítulo 5. RESULTADOS

En este capítulo se detallan los experimentos de transmisión y recepción de paquetes de datos. Los experimentos se dividen en dos: aquéllos obtenidos por simulaciones y aquéllos obtenidos a partir de la verificación del código implementado en una FPGA de desarrollo.

5.1. Experimentos de simulación

Las simulaciones computacionales se realizaron utilizando el software ISim (Xilinx Inc., 2012a). Las simulaciones requieren de dos archivos principales: el *testbench*, que se utiliza para generar los estímulos para los módulos del módem, y la configuración de onda que permite leer cualquier variable interna de los módulos simulados.

La simulación se realizó en dos etapas, la de transmisión y la de recepción, ya que cada módem necesita de un *testbench* distinto, uno para la configuración de transmisión y otro para la configuración de recepción. Ambos módems virtuales utilizan la misma configuración de onda, ya que en ambas instancias se desea medir las mismas variables internas.

La Figura 5.1 muestra el diagrama de bloques de la simulación, donde se pueden ver las instancias del módem (“Módem A” y “Módem B”, respectivamente), ambas estimuladas por su respectivo *testbench* y medidas por la misma configuración de onda.

La transmisión en banda base digital utiliza una resolución de 4 bits para la fase y 4 bits para la cuadratura, resultando en un total de 8 bits, que corresponde a la resolución suficiente para transmisiones hasta 64-QAM, utilizando números en complemento de dos sin normalizar.

A través de la configuración de onda es posible medir el estado actual del *dispatcher* y el tipo de instrucción que recibe, codificados según lo indica la Tabla 5.1. Por ejemplo, si el módem recibe una instrucción del tipo *RAM Access* entonces la configuración de onda indicará que se ha recibido una instrucción de tipo 3. Y si la máquina de estados del

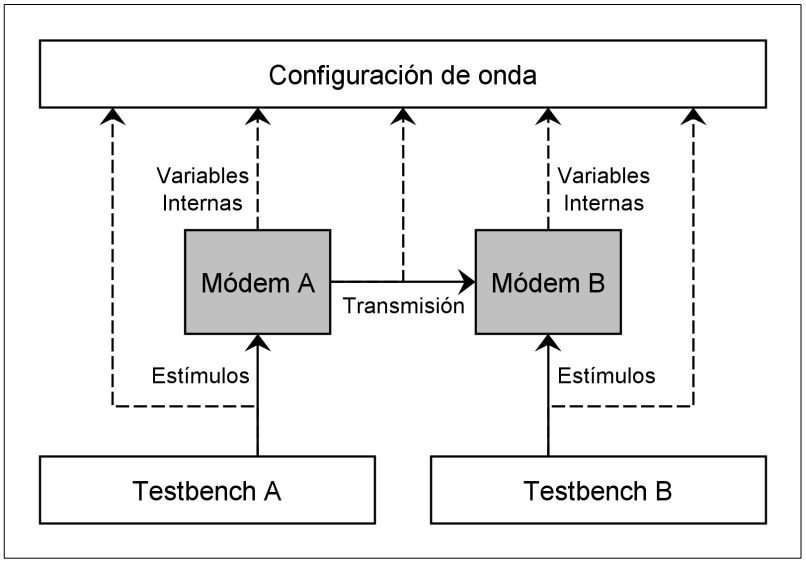

FIGURA 5.1. Diagrama de bloques de la simulación. Las líneas punteadas corresponden a las variables medidas por la configuración de onda.

dispatcher se encuentra en el estado ND, entonces la configuración de onda indicará que el estado actual es el estado 2.

Tipo	decimal
Strobe Command	0
FIFO Access	1
Register Access	2
RAM Access	3

Estado	decimal
IDLE	0
ST	1
ND	2
RD	3

TABLA 5.1. Codificación de los estados del *dispatcher* (a la izquierda) y de los tipos de instrucción recibidos en la configuración de onda (a la derecha).

Para mantener la simulación breve, se simuló la transmisión de la secuencia de números del 1 al 8 (64 bits de información) modulados en 16-QAM (equivalente a 16 símbolos).

A continuación se indican las instrucciones que se enviaron al Módem A para que este opere como transmisor, junto con las instrucciones que se enviaron al Módem B para que este opere como receptor.

Módem A (transmisor):

A1. Reset Tx

A2. Register write (modulación a utilizar)

A3. FIFO Tx write (datos a transmitir)

A4. Mod enable

A5. RF on

A6. Transmisión

Módem B (receptor):

B1. Reset Rx

B2. Register write (modulación a utilizar)

B3. Demod enable

B4. Recepción

B5. Demod disable (una vez finalizada la recepción)

B6. FIFO Rx read (datos recibidos)

A continuación se detalla cada uno de estos pasos, mostrando las formas de onda de interés en cada etapa de la simulación.

A1. Reset Tx

Se comienza por reiniciar la rama de transmisión. En la Figura 5.2 se muestra la recepción del único *byte* de instrucción correspondiente al número 9. La máquina de estados del *dispatcher* pasa por los estados IDLE y ST (0 y 1), el tipo es 0 (*Strobe Command*) y la salida *address* es igual a 9 durante la operación.

A2. Register write

En la Figura 5.3 se muestran los 3 *bytes* de instrucción (16, 0 y 4) requeridos para definir la modulación a utilizar. El dispatcher pasa por los estados ST, ND y RD (1, 2 y 3), el tipo es 2 (*Register Access*) y las salidas *address* y *data* son 16 y 04, respectivamente. Al ser una operación de escritura, la variable "rw" es igual a 0 durante la operación.

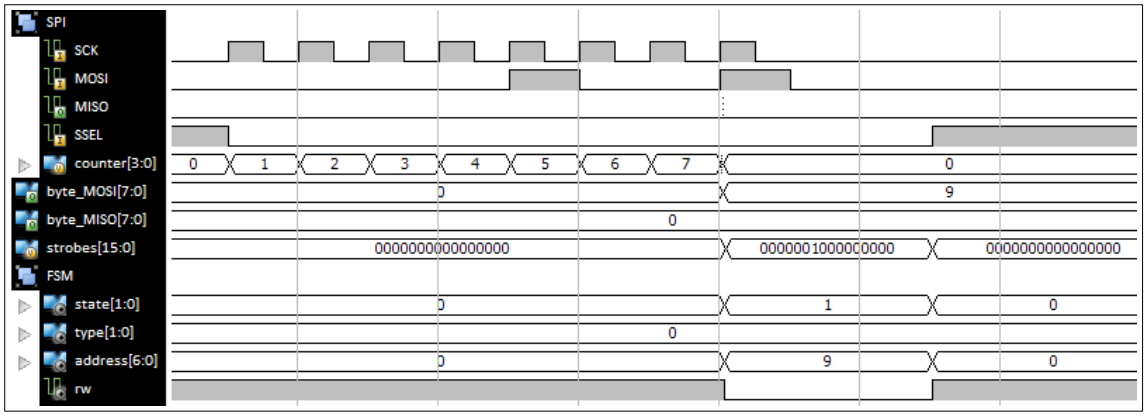

FIGURA 5.2. Formas de onda en el módulo SPI esclavo y *dispatcher* cuando se recibe el *Strobe* número 9, correspondiente a Reset Tx.

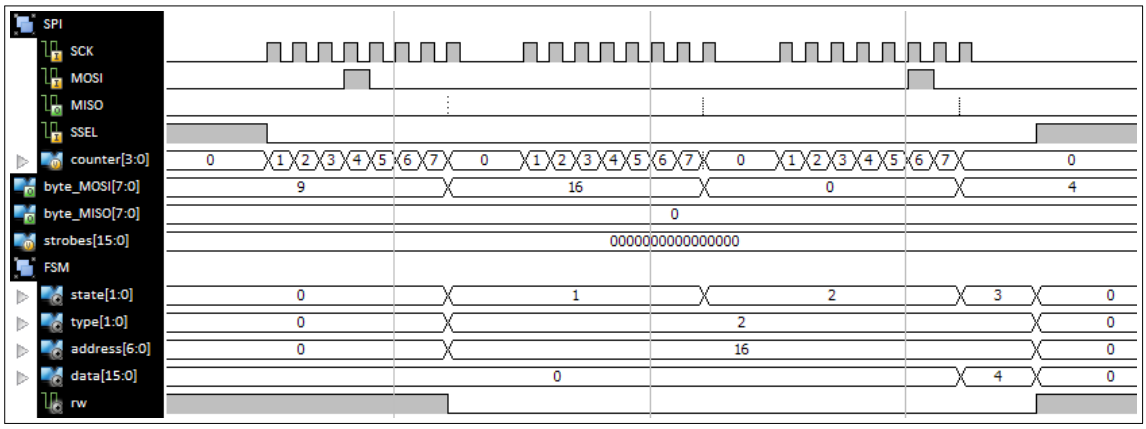

FIGURA 5.3. Formas de onda en el SPI esclavo y el *dispatcher* al configurar la modulación a utilizar. Mediante el uso de la instrucción Register write se escribe el valor 04 en el registro 16.

A3. FIFO Tx write

Se comienza enviando un *byte* igual a 62 para acceder a la FIFO de transmisión. Luego se envían los datos por transmitir (en el ejemplo, los números del 1 al 8). El *dispatcher* pasa por los estados IDLE, ST y ND (0, 1 y 2), el tipo es 1 (*FIFO Access*) y la salida *address* se mantiene constante en 62 (ver Figura 5.4). La Figura 5.4 también muestra el detalle de la FIFO de transmisión, la variable "data_count" indica la cantidad de bits almacenados, llegando a un total de 64 bits (equivalentes a 8 *bytes*).

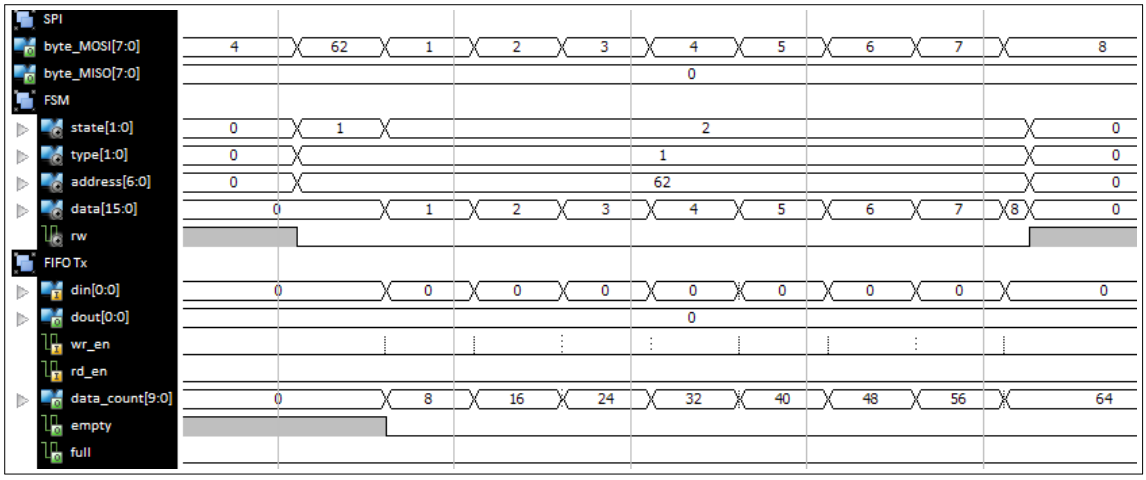

FIGURA 5.4. Formas de onda en el *dispatcher* y la FIFO de transmisión cuando se accede a ésta (FIFO Tx write). En el ejemplo se almacena la secuencia del 1 al 8 en la FIFO de transmisión (correspondiente al *address* 62).

A4. Mod enable / A5. RF on

Los *strobes* de las direcciones 4 y 5 se utilizan para activar la modulación e iniciar la transmisión. La Figura 5.5 muestra el comportamiento del *dispatcher* al recibir estas instrucciones concatenadas.

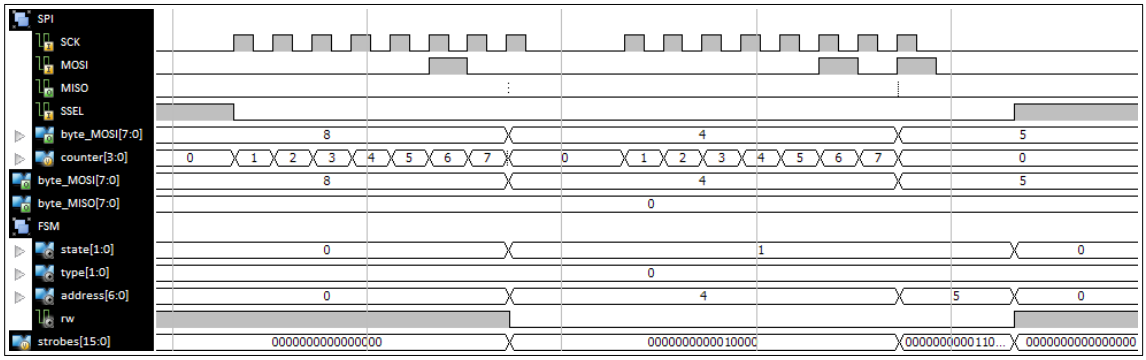

FIGURA 5.5. Formas de onda en el módulo SPI y el *dispatcher* cuando se reciben los *strobes* 4 y 5, que corresponden a Mod enable y RF on.

A6. Transmisión

La Figura 5.6 muestra los resultados finales de esta simulación. Un símbolo dura 20.5 μs y cada símbolo contiene 4 bits de información, por lo tanto, la tasa de datos equivale a

191.5 kb/s. El bloque modulador se desactiva automáticamente al finalizar la transmisión y no es necesario hacerlo manualmente usando el *strobe* 7.

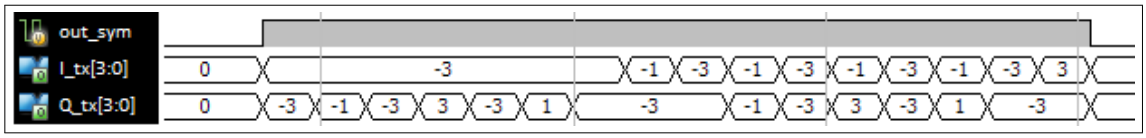

FIGURA 5.6. Salida del modulador durante la transmisión. Los valores de "I_tx" y "Q_tx" corresponden a la representación en fase y cuadratura de símbolos digitales en banda base sin normalizar.

Una vez verificado que los símbolos transmitidos corresponden a la secuencia de *bits* enviada, se procede a simular la recepción en el Módem B.

B1. Reset Rx / B2. Register write

Al igual que en la transmisión, se debe reiniciar la rama de recepción antes de comenzar la operación. De manera inmediata se configura el módem para recibir un mensaje 16-QAM. En la Figura 5.7 se muestra el *strobe* de comando número 8 (Reset Rx) seguido de los tres *bytes* de la instrucción Register write (16, 0 y 4) que indica el esquema de modulación a utilizar. El *dispatcher* pasa por los estados IDLE, ST, ST, ND y RD para procesar estas instrucciones concatenadas.

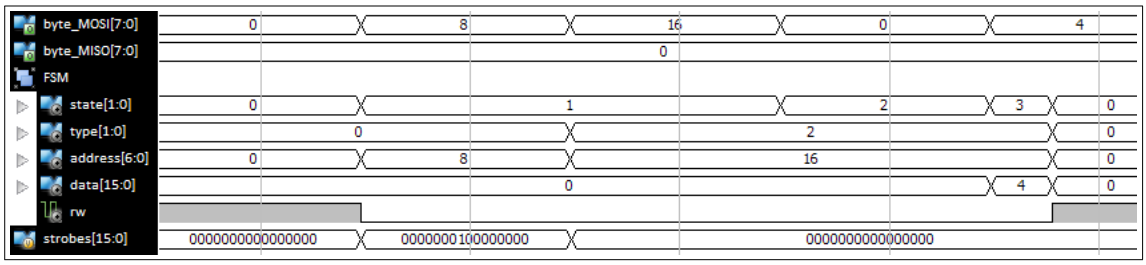

FIGURA 5.7. Formas de onda en el *dispatcher* al recibir el *strobe* número 8 correspondiente a Reset Rx. Se concatena la escritura del valor 04 en el registro 16 (Register write), que permite configurar la demodulación en QAM-16.

B3. Demod enable

El *strobe* 3 se utiliza para activar el demodulador y la recepción. En la Figura 5.8 se muestra el comportamiento de los *strobes* al recibir esta instrucción.

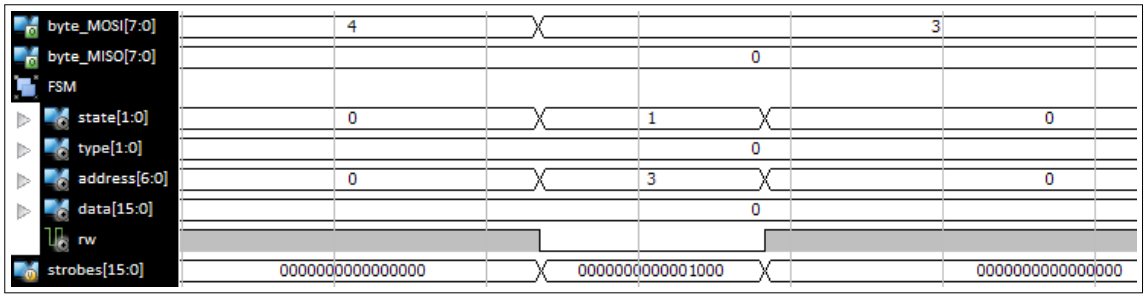

FIGURA 5.8. Formas de onda en el *dispatcher* al recibir el *strobe* número 3, correspondiente a Demod enable.

B4. Recepción

El detalle de la FIFO de recepción durante la recepción se muestra en la Figura 5.9. Los símbolos recibidos son procesados por el demodulador y los bits resultantes son almacenados en la FIFO de recepción en forma serial a través de la variable "din". La variable "rx_count" indica la cantidad de *bytes* recibidos. mientras que "full" y "empty" indican si la FIFO está llena o vacía. Estas variables son útiles para que el microcontrolador sepa cuando se ha terminado de recibir un paquete.

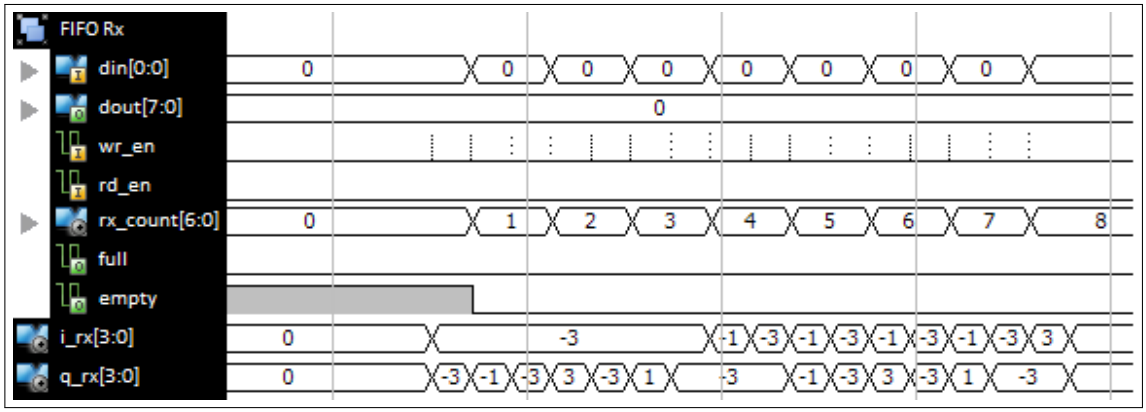

FIGURA 5.9. Formas de onda en la FIFO de recepción durante la recepción en banda base digital. La FIFO de recepción almacena los bits procesados por el demodulador.

B5. Demod disable

Una vez finalizada la recepción, el microcontrolador desactiva el demodulador a través del *strobe* número 6 antes de leer la FIFO de recepción a través del puerto SPI.

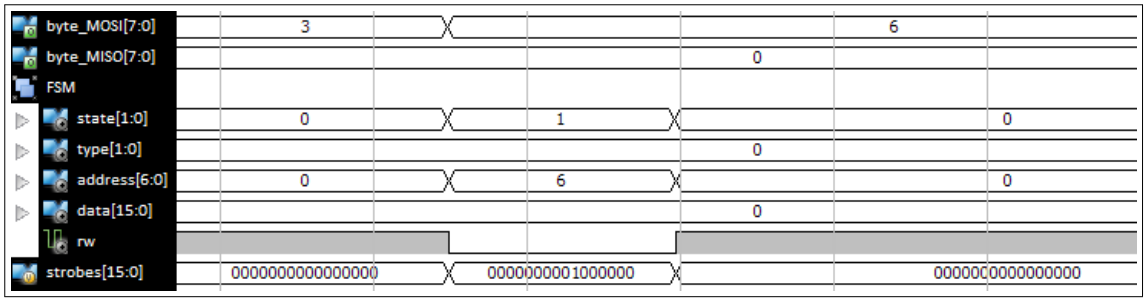

FIGURA 5.10. Formas de onda en el *dispatcher* al recibir el *strobe* número 9, correspondiente a Demod disable.

B6. FIFO Rx read

En la Figura 5.11 se muestra el acceso de lectura a la FIFO de recepción, correspondiente a la instrucción 127 seguida de ocho ceros. Cada cero indica que se desea leer un *byte* desde la FIFO de recepción. La variable "rw" se mantiene en 1 al ser una operación de lectura. Los *bytes* leídos corresponden a la secuencia de números del 1 al 8.

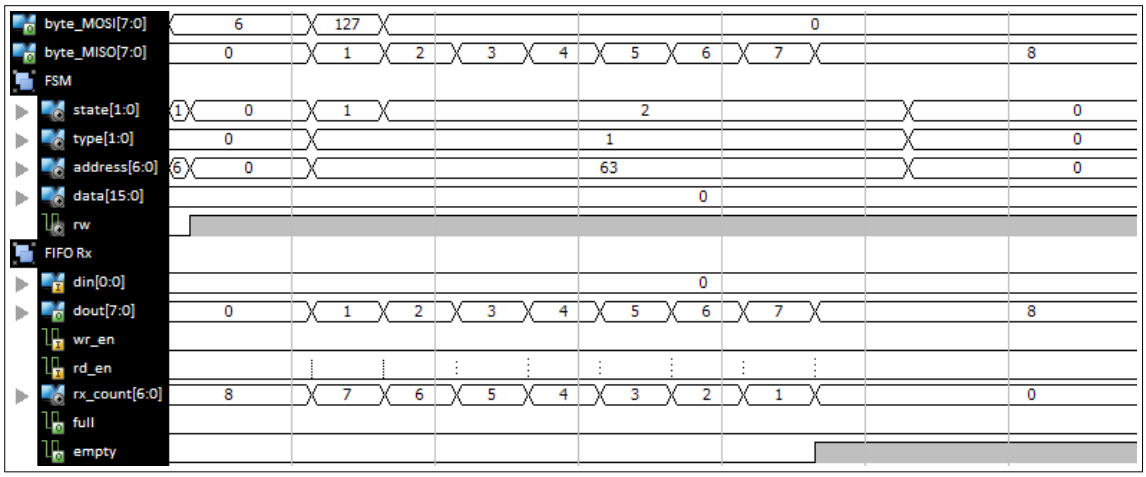

FIGURA 5.11. Formas de onda en el *dispatcher* y la FIFO de recepción cuando se accede a esta (FIFO Rx read). En el ejemplo se lee la secuencia del 1 al 8 en la FIFO de recepción (correspondiente al *address* 127). Los *bytes* leídos desde la FIFO de recepción son enviados al microcontrolador a través de "byte_MISO".

Los resultados de la simulación indican que el diseño funciona correctamente. También es posible modificar el *testbench* para probar módulos por separado o simular condiciones extremas, como *overflow* de memoria en alguna de las FIFO o parámetros de

configuración inválidos, con tal de conocer de antemano el comportamiento del módem ante errores típicos.

5.2. Implementación utilizando FPGA

Se utilizaron dos tarjetas FPGA de desarrollo idénticas (Spartan-3E), ambas programadas con la misma implementación del módem y conectadas a un computador utilizando un emulador de SPI maestro a través un puerto USB. El emulador utilizado es el Bus Pirate v3 de Dangerous Prototypes (Dangerous Prototypes, 2012). Así, el computador hace las veces el microcontrolador.

Para programar las FPGA se utilizó el *software* Adept 2 System. (Digilent Inc., 2010). Para la conexión entre el PC y el Bus Pirate se utilizó el *software* CoolTerm (Meier, 2011) y se emuló un terminal SPI maestro con un *clock* de 1 MHz (equivalente a 128 kb/s).

Como es virtualmente imposible medir todas las variables internas del módem directamente desde la FPGA, se seleccionaron las variables de interés y se programó un módulo adicional para observarlas en tiempo real a través del visualizador de 7 segmentos incluido en la FPGA de desarrollo.

La Figura 5.12 muestra un diagrama de bloques de la implementación en FPGA. La combinación del PC con el Bus Pirate reemplaza al microcontrolador con un puerto SPI maestro. Esta combinación se comunica con una FPGA que contiene el módem configurado en modo de transmisión. De manera análoga, la segunda FPGA contiene el módem configurado en modo de recepción, que se conecta al mismo computador a través de otro Bus Pirate.

La FPGA Spartan-3E cuenta con cuatro puertos de *input*/*output* digitales (ver detalle en la Figura 5.13). El puerto JA1 se utilizó para la interfaz SPI esclavo, los puertos JB1 y JC1 se utilizan para la transmisión y recepción respectivamente, y el puerto JD1 queda libre.

La conexión entre ambas FPGA se hace a través de un cable Ethernet de par trenzado (adaptado para conectarse a los puertos de la FPGA). Cada cable Ethernet contiene cuatro

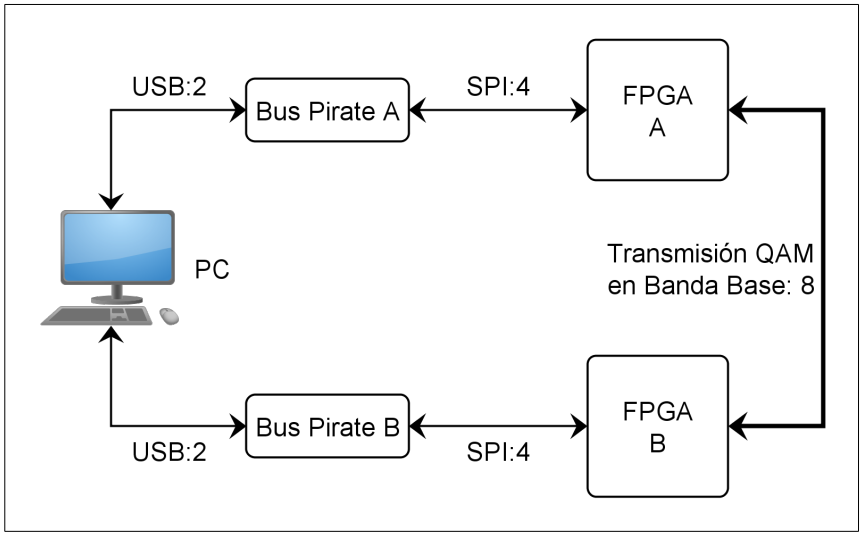

FIGURA 5.12. Diagrama de conexiones correspondiente a la implementación del módem en FPGA. El número a la derecha de cada etiqueta indica el ancho de los buses (no considera conexiones de alimentación y tierra).

pares de cables, suficientes para conectar la fase y la cuadratura de la transmisión en banda base.

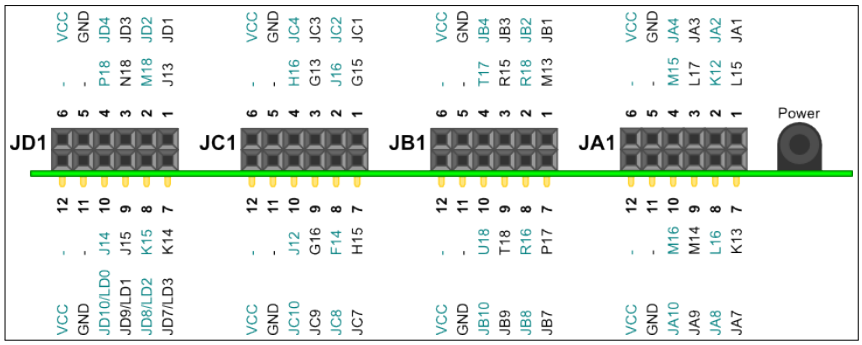

FIGURA 5.13. Diagrama de conexión de los cuatro puertos PMOD de la FPGA de desarrollo utilizada. Cada puerto contiene 12 conexiones, 4 de alimentación y 8 de datos.

La Figura 5.14 muestra una fotografía de ambas FPGAs conectadas a través de dos cables Ethernet entre sí. Cada FPGA está conectada a un emulador Bus Pirate y ambos emuladores se conectan al mismo computador.

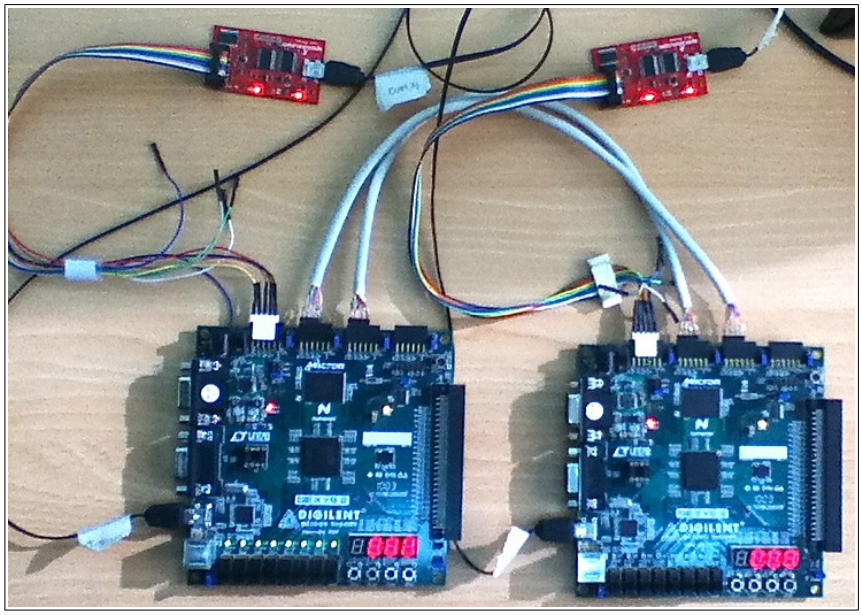

FIGURA 5.14. Fotografía de la implementación en FPGA. Cada FPGA contiene la implementación del módem y se conecta a un computador a través de un Bus Pirate que emula un puerto SPI maestro. Ver Figura 5.12.

El objetivo de esta prueba es replicar la simulación de la sección 5.1, es decir, transmitir los números del 1 al 8 desde el PC A utilizando la FPGA A como dispositivo de comunicación, hasta el PC B a través de la FPGA B.

La Figura 5.15 muestra las instrucciones enviadas desde el computador hacia ambas FPGA. A la izquierda se muestra la comunicación con la FPGA A transmisora (ver Figura 5.15A) y a la derecha la comunicación con la FPGA B receptora. Esta última está dividida en dos partes: 5.15B ocurre antes de la transmisión y 5.15C ocurre después de la transmisión.

En la Figura 5.15A se muestra la configuración de la FPGA A en modo de transmisión y el mensaje que se va a enviar hacia la FPGA B (la secuencia de números del 1 al 8). En la Figura 5.15B se muestra la configuración de la FPGA B en modo de recepción. Una vez ocurrida la comunicación, se accede a la FIFO de recepción de la FPGA B (ver Figura 5.15C) y se lee la misma secuencia de números que se introdujo en la FPGA A.

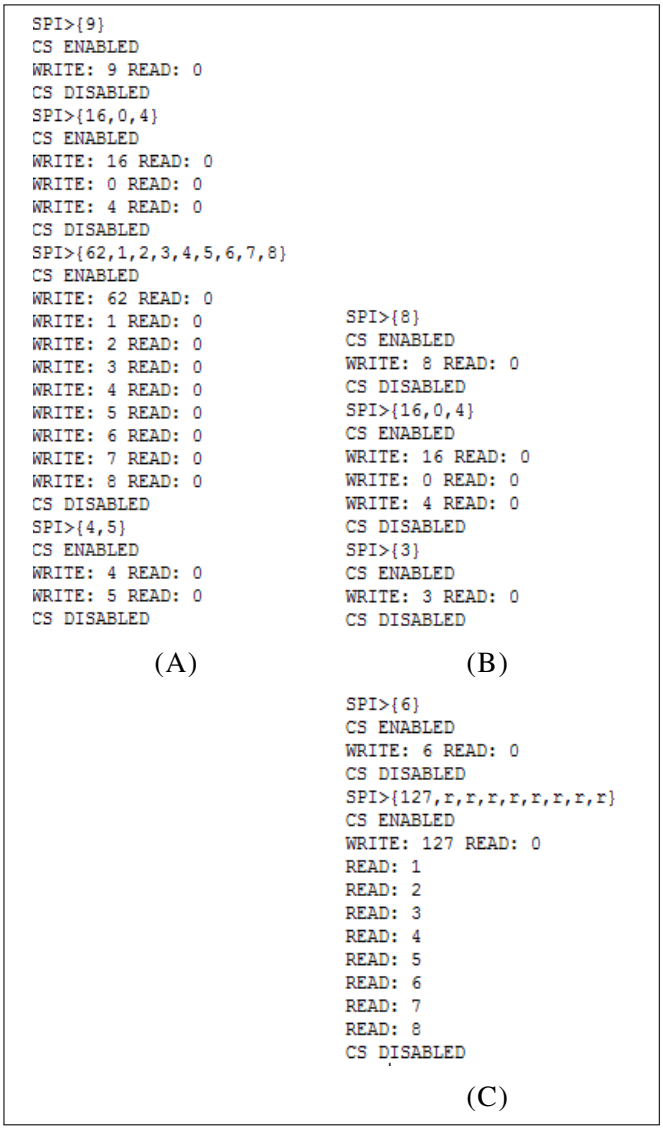

FIGURA 5.15. Resultados de la implementación en FPGA. A la izquierda se muestran los estímulos para el módem transmisor y a la derecha se muestran los estímulos del módem receptor. (A) y (B) ocurren en forma simultánea. (C) se ejecuta una vez finalizada la transmisión.

Una vez finalizada la transmisión, se verificó que el mensaje recibido fuera igual al mensaje enviado. De este modo se comprueba la correspondencia entre la simulación y la implementación del módem.

Testbench ping-pong

Una vez replicada la simulación de la Sección 5.1, se procedió a automatizar este proceso. El experimento consistió en transmitir una secuencia de datos aleatoria desde un módem a otro, y luego enviar de vuelta el mismo paquete utilizando otro esquema de modulación. El objetivo de esta prueba es comprobar que el módem pueda cambiar entre los modos de transmisión y recepción utilizando distintas modulaciones sin errores.

El paquete de prueba se escogió al azar y consiste de 16 números en el rango entre 0 y 255. Cada número se representa en binario utilizando 8 bits, por lo tanto la secuencia de datos corresponde a 128 bits.

Se utilizó la misma configuración de la figura 5.12, pero en esta ocasión, los módems A y B asumen los roles de transmisión y recepción e intercambian roles según sea necesario.

Para esta prueba se utilizó el *software* Buccaneer's Den (Roy, 2011), este *software* esta diseñado para operar el Bus Pirate V3 y es capaz de ejecutar macros. Las macros utilizados en este experimento se encuentran en el Anexo D y la secuencia de datos aleatorios utilizada es:

```
54 106 21 56 79 102 72 95 87 45  3 78 120 98 111 23
```

En la Tabla 5.2 se muestran el orden y los resultados de este experimento. Además se muestra la configuración en ambas FPGA y el mensaje recibido en la FPGA receptora.

5.3. Porcentaje de utilización de la FPGA

Para determinar el porcentaje de recursos utilizados en la FPGA se utilizó el software PlanAhead (Xilinx Inc., 2012a). Los resultados se pueden ver en la Tabla 5.3 y cada una de sus componentes se describe brevemente a continuación.

***Flip Flops*:** Se utilizan principalmente para almacenar información. En particular sirven para implementar máquinas de estado, como la utilizada en del *dispatcher*, almacenar información y armar el *pipeline* en la FPGA.

Sentido	Modulación	Configuración	Recepción
A → B	QPSK	CS ENABLED WRITE: 16 READ: 0 WRITE: 0 READ: 0 WRITE: 2 READ: 0 CS DISABLED	CS ENABLED WRITE: 127 READ: 0 READ 16 BYTES: 54 106 21 56 79 102 72 95 87 45 3 78 120 98 111 23 CS DISABLED
B → A	16-QAM	CS ENABLED WRITE: 16 READ: 0 WRITE: 0 READ: 0 WRITE: 4 READ: 0 CS DISABLED	CS ENABLED WRITE: 127 READ: 0 READ 16 BYTES: 54 106 21 56 79 102 72 95 87 45 3 78 120 98 111 23 CS DISABLED
A → B	64-QAM	CS ENABLED WRITE: 16 READ: 0 WRITE: 0 READ: 0 WRITE: 6 READ: 0 CS DISABLED	CS ENABLED WRITE: 127 READ: 0 READ 16 BYTES: 54 106 21 56 79 102 72 95 87 45 3 78 120 98 111 23 CS DISABLED
B → A	64-QAM	CS ENABLED WRITE: 16 READ: 0 WRITE: 0 READ: 0 WRITE: 6 READ: 0 CS DISABLED	CS ENABLED WRITE: 127 READ: 0 READ 16 BYTES: 54 106 21 56 79 102 72 95 87 45 3 78 120 98 111 23 CS DISABLED
A → B	16-QAM	CS ENABLED WRITE: 16 READ: 0 WRITE: 0 READ: 0 WRITE: 4 READ: 0 CS DISABLED	CS ENABLED WRITE: 127 READ: 0 READ 16 BYTES: 54 106 21 56 79 102 72 95 87 45 3 78 120 98 111 23 CS DISABLED
B → A	QPSK	CS ENABLED WRITE: 16 READ: 0 WRITE: 0 READ: 0 WRITE: 2 READ: 0 CS DISABLED	CS ENABLED WRITE: 127 READ: 0 READ 16 BYTES: 54 106 21 56 79 102 72 95 87 45 3 78 120 98 111 23 CS DISABLED

TABLA 5.2. Resultados del experimento ping-pong. La columna "Configuración" es común a ambas FPGA y depende del esquema de modulación. En todas las etapas se puede verificar que el mensaje recibido es siempre el mismo, independiente del esquema de modulación utilizado.

***Look-up Tables* (LUTs):** Permiten implementar funciones lógicas y funciones aritméticas básicas.

***Slices*:** Un *slice* contiene dos LUTs y dos *Flip Flops*. Un *slice* ocupado no siempre ocupa todos sus componentes internos. La cantidad de *Slices* ocupados es una buena métrica para determinar el porcentaje de utilización de una FPGA.

***Input/Output Blocks* (IOBs):** Los IOBs proveen la interfaz entre las conexiones externas de la FPGA (puertos, interruptores, botones, entre otros) con los componentes internos de la FPGA.

***Block* RAM (RAMB):** Provee el almacenamiento en la FPGA.

BUFGMUX (Clock Buffer/Multiplexer): Estos elementos se utilizan para evitar problemas de *fan-out* en la o las señales de *clock*, especialmente cuando se tienen muchos módulos implementados.

Utilización lógica	Usados	Disponibles	Porcentaje
Flip Flops	1528	9312	16,41 %
LUTs	1753	9312	18,83 %
Slices	1567	4656	33,66 %
IOBs	61	232	26,29 %
RAMBs	3	20	15,00 %
BUFGMUXs	2	24	8,33 %

TABLA 5.3. Resumen de la utilización del dispositivo Digilent Nexys 2 Spartan-3E 500.

El módem implementado ocupa una pequeña porción de la FPGA utilizada (ver Tabla 5.3), lo que muestra que se trata de un diseño compacto, además de ser escalable sin problemas para su operación con múltiples antenas.

Capítulo 6. CONCLUSIONES Y TRABAJO FUTURO

En esta tesis se presentó el diseño y la implementación de un módem de prueba para redes inalámbricas de sensores. Las principales contribuciones de este trabajo son:

1. La estructura modular del módem que permite la fácil inclusión de módulos de codificación de canal, encriptación y el soporte para varios esquemas de modulación MIMO.
2. La estructura de registros de configuración, disponible para ser modificada para albergar la configuración del módem y otros dispositivos.
3. El diseño de un módulo de control para el módem capaz de controlar elementos anexos, como múltiples antenas.

El uso e integración de diversas herramientas computacionales para la descripción de *hardware* utilizadas para la implementación, diseño y verificación del módem permitieron desarrollar un conjunto de pruebas estandarizadas para cualquier modificación o aplicación futura de este módem.

Las simulaciones computacionales permitieron verificar el diseño correcto de la unidad de control del módem y las pruebas sobre FPGA de desarrollo resultaron satisfactorias, tanto en la interfaz compatible con un microcontrolador a través de un puerto SPI maestro (como el MSP430 de TI), como las transmisiones en banda base digital realizadas hacia otra FPGA.

6.1. Trabajo futuro

Al diseñar el módem se consideró la compatibilidad con el microcontrolador MSP430 de TI. Para verificar esta compatibilidad será necesario escribir un *driver* adecuado y diseñar un conjunto de pruebas para corroborar su funcionamiento de manera similar al test detallado en la sección 5.2.

La estructura modular del módem permite incluir y modificar componentes fácilmente con tal de facilitar cualquier trabajo futuro. Este trabajo inclurá conectar múltiples radios

y sus respectivas señales de control. También es posible que se incluyan módulos para soportar múltiples esquemas de múltiples antenas, codificación de canal, encriptación, entre otras alternativas.

Una vez completadas todas estas tareas, se puede considerar eliminar componentes innecesarios para reducir el consumo energético de módem. Por ejemplo, si se elimina la porción de RAM de seguridad, se podrá simplificar la instrucción *RAM Access* y por lo tanto se simplificará el *dispatcher* y el *driver* para el microcontrolador MSP430.

Referencias

Atmel. (2009). *AT86RF230 Datasheet.* Recuperado el 21 de junio de 2012, de http://www.atmel.com/.

Atmel. (2011). *ATmega128L Datasheet.* Recuperado el 08 de junio de 2012, de http://www.atmel.com/.

Azami, F., Ghorssi, A., Hemesi, H., Mohammadi, A., y Abdipour, A. (2008). Design and implementation of a flexible 4×4 MIMO testbed. En *International symposium on telecommunications, 2008. ist 2008.* (p. 268 -272).

Bialkowski, K., y Uthansakul, P. (2011). Design of MIMO testbed with an FPGA board for fast signal processing. *1st IEEE International Conference on Wireless Broadband and Ultra Wideband Communications (AusWireless'06)*.

Chu, P. (2011). *FPGA Prototyping By Verilog Examples: Xilinx Spartan-3 Version*. Hoboken, NJ, Estados Unidos: John Wiley & Sons.

Cummings, C. E. (2002). The Fundamentals of Efficient Synthesizable Finite State Machine Design using NC-Verilog and BuildGates. *2002 International Cadence Usergroup Conference*.

Dangerous Prototypes. (2012). *Bus Pirate Documentation.* Recuperado el 26 de marzo de 2012, de http://dangerousprototypes.com/docs/Bus_Pirate.

Digilent Inc. (2010). *Adept 2 [Software].* Disponible en Digilent Inc. - Digital Design Engineer's Source: http://www.digilentinc.com/. (versión 2.9.4)

Goldsmith, A. (2005). *Wireless communications*. Estados Unidos: Cambridge University Press.

Hać, A. (2003). *Wireless sensor network designs*. Chichester, Inglaterra: John Wiley & Sons.

IEEE. (2003). *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications — Amendment 4: Further Higher Data Rate Extension in the 2.4 GHz Band* (n.º 802.11g). Nueva York, Estados Unidos: The Institute of Electrical and Electronics Engineers, Inc.

IEEE. (2005). *IEEE Standard for System Verilog- Unified Hardware Design, Specification, and Verification Language* (n.º 1364). Nueva York, Estados Unidos: The Institute of Electrical and Electronics Engineers, Inc.

IEEE. (2006). *Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs)* (n.º 802.15.4). Nueva York, Estados Unidos: The Institute of Electrical and Electronics Engineers, Inc.

IEEE. (2009). *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 5: Enhancements for Higher Throughput* (n.º 802.11n). Nueva York, Estados Unidos: The Institute of Electrical and Electronics Engineers, Inc.

Kim, D., y Torlak, M. (2008). Rapid prototyping of a cost effective and flexible 4×4 MIMO testbed. En *5th IEEE Sensor Array and Multichannel Signal Processing Workshop, 2008.* (p. 5 -8).

Meier, R. (2011). *CoolTerm [Software].* Disponible en Roger Meier's Freeware: http://freeware.the-meiers.org/. (versión 1.4.1)

Motorola Inc. (2003). *SPI Block Guide V03.06.*

Navabi, Z. (2005). *Verilog Digital System Design: RT Level Synthesis, Testbench, and Verification*. Estados Unidos: McGraw-Hill.

NXP Semiconductors. (2012). *UM10204 I^2C-bus specification and user manual.*

Palnitkar, S. (2003). *Verilog Hdl: A Guide to Digital Design and Synthesis*. SunSoft Press.

Rosas, F., y Oberli, C. (2012). Energy-efficient MIMO SVD Communications. En *PIMRC2012-Track 1: PHY and Fundamentals (PIMRC2012-PHY).* Sydney, Australia.

Roy, V. R. (2011). *Buccaneer's Den [Software].* Disponible en buccaneers-den - Gui for the Bus Pirate: http://code.google.com/p/buccaneers-den/. (versión 0.1.2)

Semtech. (2005). *XE1205 Datasheet.* Recuperado el 21 de junio de 2012, de http://www.semtech.com/.

Subtel MTT, Gobierno de Chile. (2005). *Estudio relativo a la actualización del Plan General de Uso del Espectro Radioeléctrico.* Santiago, Chile. Autor.

Texas Instruments Inc. (2010a). *CC2420 Datasheet.* Recuperado el 13 de abril de 2012, de http://www.ti.com/.

Texas Instruments Inc. (2010b). *CC2520 Datasheet.* Recuperado el 20 de junio de 2012, de http://www.ti.com/.

Texas Instruments Inc. (2012). *MSP430F2618 Datasheet.* Recuperado el 13 de abril de 2012, de http://www.ti.com/.

TIK WSN Research Group ETH Zurich. (2012). *The sensor network museum.*

Williams, J. (2008). *Digital VLSI Design With Verilog: A Textbook from Silicon Valley Technical Institute*. San José, CA, Estados Unidos: Springer.

Xilinx Inc. (2011). *Spartan-3 Generation FPGA User Guide.* Recuperado el 12 de mayo de 2012, de http://www.xilinx.com/.

Xilinx Inc. (2012a). *ISE Design Suite: System Edition [Software].* Disponible en All Programmable Technologies from Xilinx Inc.: http://www.xilinx.com/. (versión 14.1)

Xilinx Inc. (2012b). *LogiCORE IP FIFO Generator v8.4, Product Specification.* Recuperado el 05 de abril de 2012, de http://www.xilinx.com/.

Xilinx Inc. (2012c). *LogiCORE IP FIFO Generator v8.4, User Guide.* Recuperado el 05 de abril de 2012, de http://www.xilinx.com/.

Zolertia. (2011). *Z1 Platform.* Recuperado el 07 de diciembre de 2011, de http://zolertia.com/.

ANEXOS

ANEXO A. COMPOSICIÓN BIT A BIT DE LAS INSTRUCCIONES PARA EL MÓDEM

Las Tablas A.1, A.2, A.3 y A.4 indican como se deben componer las instrucciones enviadas al módem. Además muestran la información que debe contener cada instrucción y el orden en que deben ser transmitidas. Según el tipo de instrucción se necesitan de uno, dos o tres *bytes*.

Instrucción	*Strobe Command*							
bits	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
Byte 1	Reg 0	W 0	Strobe 0 0		Address $A_3 \ldots A_0$			

TABLA A.1. Composición bit a bit de la instrucción *Strobe Command*.

Instrucción	*FIFO Access*							
bits	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
Byte 1	Reg 0	R/W $1/0$	Address $A_5 \ldots A_0$					
Byte 2	Data $D_7 \ldots D_0$							

TABLA A.2. Composición *byte* a *byte* y bit a bit de la instrucción *FIFO Access*.

Instrucción	*Register Access*							
bits	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
Byte 1	Reg 0	R/W $1/0$	Address $A_5 \ldots A_0$					
Byte 2	Data I $D_{15} \ldots D_8$							
Byte 3	Data II $D_7 \ldots D_0$							

TABLA A.3. Composición *byte* a *byte* y bit a bit de la instrucción *Register Access*.

Instrucción	*RAM Access*							
bits	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
Byte 1	RAM 1	Address $A_6 \dots A_0$						
Byte 2	Bank $B_1 B_0$		R/W $1/0$	Dummy $X_4 \dots X_0$				
Byte 3	Data $D_7 \dots D_0$							

TABLA A.4. Composición *byte* a *byte* y bit a bit de la instrucción *RAM Access*.

ANEXO B. DIAGRAMAS DE CONSTELACIÓN

Se utilizaron los diagramas de constelación para modulaciones M-QAM descritos en el estándar IEEE-802.11n. (IEEE, 2009).

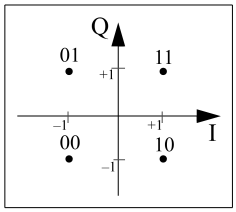

FIGURA B.1. Diagrama de constelación para la modulación QPSK (IEEE, 2009)

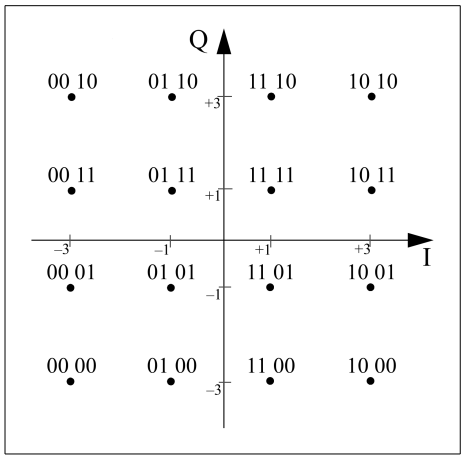

FIGURA B.2. Diagrama de constelación para la modulación 16-QAM (IEEE, 2009)

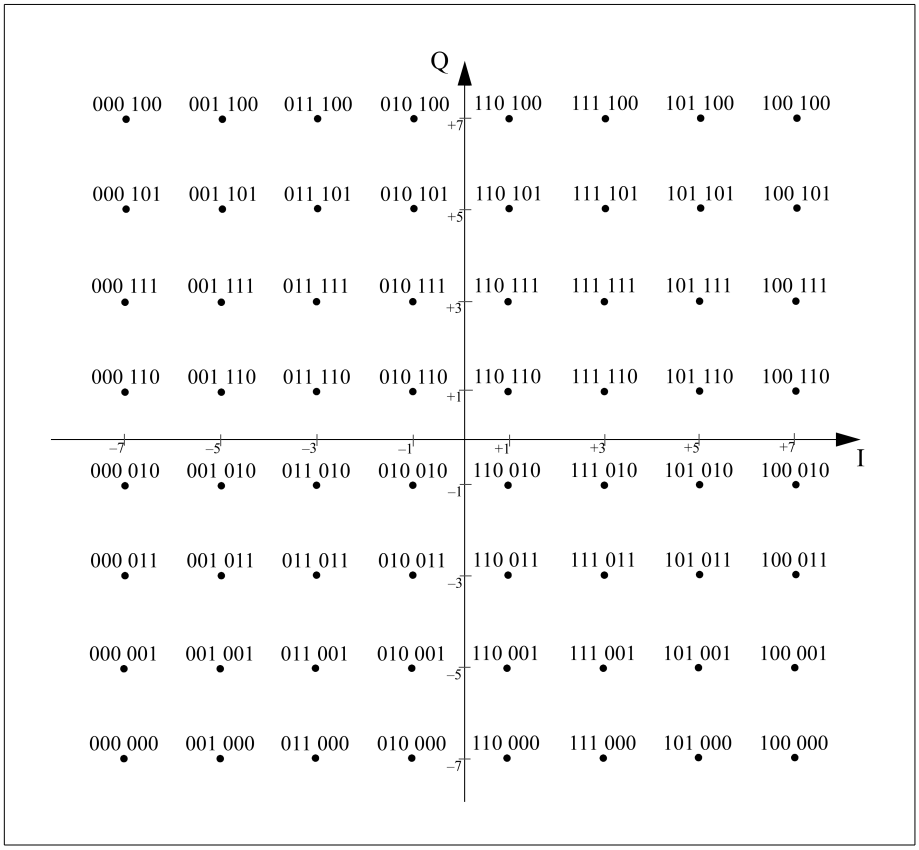

FIGURA B.3. Diagrama de constelación para la modulación 64-QAM. (IEEE, 2009)

ANEXO C. ESTRUCTURA DE ARCHIVOS DEL MÓDEM

La Figura C.1 muestra la estructura de archivos del código fuente del módem. A continuación se detalla el contenido de cada uno de estos archivos.

top.v: Archivo principal que instancia todos los demás archivos.

top.ucf: Archivo que asocia *inputs* y *outputs* a elementos de una FGPA en particular.

SPI_slave.v: Contiene al módulo SPI Esclavo.

dispatcher.v: Contiene al módulo Dispatcher.

mem.v: Contiene los módulos Memorias, Strobes de comando y Registros de acceso a las FIFO. También instancia y controla los archivos listados a continuación.

- **register.v:** Contiene el módulo Registros de configuración.
- **strobes.v:** Contiene el módulo de control del módem y los *strobes* de comando.
- **ram.xco:** *IP Core* que implementa la RAM Security.
- **FIFO_tx.xco:** *IP Core* que implementa la FIFO de transmisión.
- **FIFO_rx.xco:** *IP Core* que implementa la FIFO de recepción.
- **PtSdemod.v:** Módulo paralelo-serie que conecta al demodulador con la FIFO de recepción.
- **P2S.v:** Módulo paralelo-serie que conecta al *dispatcher* con la FIFO de transmisión.
- **QAMmod.v:** Módulo parametrizado que implementa los tres moduladores M-QAM.
- **QAMdemod.v:** Módulo parametrizado que implementa los tres demoduladores M-QAM.

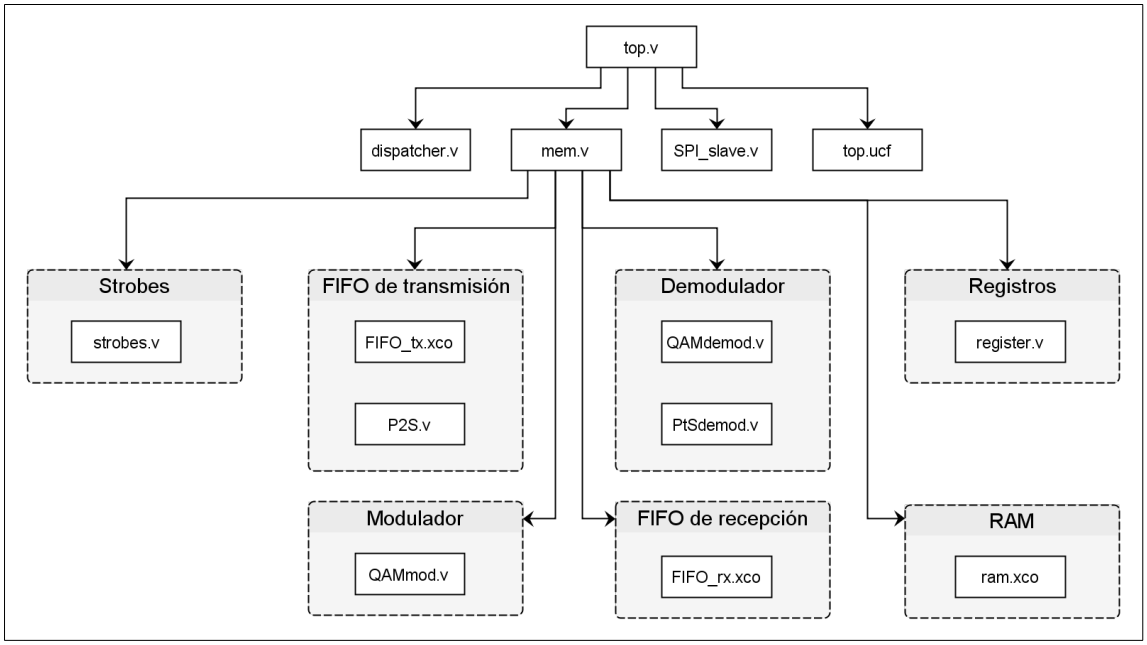

FIGURA C.1. Estructura de archivos que componen al módem.

ANEXO D. MACROS PARA EXPERIMENTO PING-PONG

Las Tablas D.1, D.2 y D.3 muestran las macros utilizadas en este experimento.

QPSK	
Tx	`{9} {16,0,2} {62,54,106,21,56,79,102,72, 95,87,45,3,78,120,98,111,23}`
	`{4,5}`
Rx	`{8} {16,0,2} {3}`
	`{6} {127,r:16}`

TABLA D.1. Macros utilizadas en el experimento ping-pong para QPSK.

16-QAM	
Tx	`{9} {16,0,4} {62,54,106,21,56,79,102,72, 95,87,45,3,78,120,98,111,23}`
	`{4,5}`
Rx	`{8} {16,0,4} {3}`
	`{6} {127,r:16}`

TABLA D.2. Macros utilizadas en el experimento ping-pong para 16-QAM.

64-QAM	
Tx	`{9} {16,0,6} {62,54,106,21,56,79,102,72, 95,87,45,3,78,120,98,111,23}`
	`{4,5}`
Rx	`{8} {16,0,6} {3}`
	`{6} {127,r:16}`

TABLA D.3. Macros utilizadas en el experimento ping-pong para 64-QAM.