UNIVERSIDAD DE VALLADOLID

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA DE TECNOLOGÍAS ESPECÍFICAS DE TELECOMUNICACIÓN

MENCIÓN EN SISTEMAS ELECTRÓNICOS

Diseño de un sistema SDR (Radio Definida por Software) mediante FPGA para la recepción de dispositivos RFID

Autor:

Daniel Álvarez Hernández

Tutor:

Jesús Arias Álvarez

Agradecimientos

Agradecer a mi madre todo el esfuerzo que ha hecho a lo largo de su vida para poder brindarme la oportunidad de llegar a ser lo que siempre he pretendido: un hijo del que poder sentirse orgullosa. Sin sus ánimos y educación nunca podría haber acabado la carrera.

También agradecer a mi hermana todo el apoyo y ayuda que me ha otorgado desde que aparecí en su vida ya que sin ella no sería la persona que soy hoy en día. De mayor siempre quise ser como ella, espero poder conseguirlo algún día.

Otra persona a la que me gustaría dar las gracias es a mi primo Carlos ya que probablemente es el gran culpable de que desde niño me interesase por la tecnología, concretamente en la electrónica. Él fue quien me invitó a estudiar teleco cuando aún no había oído hablar de su existencia. Gracias por enseñarme tantas cosas fascinantes.

Agradecer a Ana todo lo que ha hecho por mi en estos años ya que con su apoyo consigue que afronte la vida con mucho más optimismo e ilusión, siendo un pilar fundamental en mi vida.

A todos mis amigos y compañeros de estudio que me han permitido desconectar en los momentos más duros permitiéndome coger fuerzas para poder continuar con más ahínco: Gracias de corazón. Mención especial a Sara ya que es como una hermana para mí. Agradecer todos los momentos vividos juntos en San Blas.

También me gustaría darle las gracias a Jesús Arias, una persona cuyos conocimientos parecen no tener fin, por todo lo que me ha enseñado en la carrera y en concreto por su dedicación y ayuda para realizar este proyecto.

Por último, no podía olvidarme de la persona que mas me ha empujado a llegar hasta el final sin estar presente. Ojalá pudieses estar aquí para comprobar que finalmente mi deseo de niño se hizo realidad. Este esfuerzo es en honor a mi Padre.

Resumen

Hoy en día la Identificación por Radio Frecuencia (RFID) está muy extendida en áreas muy diversas de nuestra vida cotidiana como los sistemas para la identificación en el control de acceso a áreas restringidas, catalogación e identificación de objetos, seguimiento en la cadena productiva de un determinado artículo, identificación animal, pagos electrónicos o la industria del automóvil.

Tal y como queda patente en estos ejemplos de uso, el objetivo primordial de la tecnología RFID es transmitir una identidad única de forma similar a un código de barras.

Sin embargo, estos dispositivos presentan la ventaja de poder proporcionar información añadida a la identificación como enlaces a páginas web, información sobre el control de versiones o incluso lecturas de sensores de manera inalámbrica.

Este trabajo describe el diseño y desarrollo de un prototipo funcional para la lectura de dichos dispositivos RFID cuya frecuencia de trabajo se centra en la banda HF (3MHz-30MHz),concretamente en 13.56MHz.

Además, otro de los objetivos de este trabajo consiste en la implementación del lector utilizando los paradigmas de la Radio Definida por Software (SDR), en la cual, los elementos hardware típicos de los receptores de radio como puede ser el mezclador o el filtrado están implementados mediante lógica digital, proporcionando la posibilidad de gobernarlos por software de más alto nivel.

Dada la naturaleza de una SDR, se ha optado por sintetizarla en un dispositivo lógico programable como es una FPGA (Matriz de puertas lógicas programable), realmente versátil a la hora de implementar la lógica necesaria para el procesado digital de la señal.

Abstract

Nowadays, the Radio Frequency Identification (RFID) is very widespread in many diverse areas of our everyday life such as identification systems in access control to restricted areas, cataloging and identification of objects, tracking in the production chain of a certain item, animal identification or electronic payments.

As it is clear from these usage examples, the primary goal of RFID technology is to convey a unique identity similar to a barcode. However, these devices have the advantage of being able to provide information added to the identification such as web links, facts about version control or even wireless sensor readings .

This work describes the design and development of a functional prototype for reading these RFID devices whose working frequency is centered in the HF band (3MHz-30MHz), specifically at 13.56MHz.

In addition, another goals of this work consist of the reader's implementation using the paradigms of Software Defined Radio (SDR), where the typical hardware elements of radio receivers such as mixer or filtering are implemented by digital logic, providing the possibility to govern them by higher level software.

Given the nature of an SDR, a programmable logic device such as a FPGA (Field-programmable gate array) has been chosen to synthesize it. The FPGA is very versatile to perform the necessary logic for digital signal processing.

Índice General

Índice de figuras

Índice de tablas

Nomenclatura

ADC	Analog to Digital Converter
AFE	Analog Front End
ASK	Amplitud Shift Keying
BOM	Bill Of Material
CIC	Cascaded-Integrator Comb Filter
DAC	Digital to Analog Converter
DDC	Digital Down Converter
DDS	Direct Digital Synthesis
DSP	Digital Signal Processing
DUC	Digital Up Converter
EMC	ElectroMagnetic Compatibility
EMI	ElectroMagnetic Interference
FDA	Fully Differential Amplifier
FDT	Frame Delay Time
FIR	Finite Impulse Response filter
FPGA	Field Programable Gate Array
FSK	Frequency Shift Keying
HF	High Frequency
IPC	Institute of Printed Circuits
ISM	Industrial Scientific Medical band
ISO	International Organization for Standardization
LF	Low Frequency
LSB	Low Significative Bit
MSB	Most Significative Bit
NFC	Near Field Communication
OOK	On/Off Keying
OSI	Open Systems Interconnection
PCB	Printed Circuit Board

PCD	Proximity Coupling Devices
PICC	Proximity Card
PLL	Phase Locked Loop
PSK	Phase Shift Keying
RFID	Radio Frequency IDentification
RGT	Request Guard Time
SDR	Software Defined Radio
SMD	Surface-Mount Device
SNR	Signal-to-Noise Ratio
UART	Universal Asynchronous Receiver-Transmitter
UHF	Ultra High Frequency
UID	Unique IDentification number

1. Introducción

1.1 Objetivos

El objetivo principal de este trabajo consiste en, mediante todos los conocimientos posibles adquiridos durante el grado, realizar un proyecto que se adapte lo mejor posible a la simbiosis entre las telecomunicaciones y la electrónica.

Así pues, para la consecución del mismo han sido necesarias herramientas de distinta índole, como pueden ser los aspectos referidos a la teoría de la señal y teoría de la comunicación, pasando por la física de las ondas electromagnéticas propias de todo sistema de radiofrecuencia, herramientas del diseño electrónico digital y analógico, fundamentos telemáticos referidos al protocolo de comunicación referenciado por los estándares, herramientas de descripción hardware como puede ser Verilog , herramientas de programación de nivel más elevado como es C o Python, herramientas para la simulación y visualización de la señal como MatLab y así un sinfín de conocimientos de distintas ramas que en su conjunto tejen la tela de araña a la que llamamos telecomunicaciones.

El proyecto que aquí se plantea consiste en la implementación de una arquitectura propia de una Radio Definida por Software (SDR) en un dispositivo de lógica digital programable como son las FPGA (Field-programmable gate array) con el ánimo de establecer un canal de comunicación con los dispositivos RFID centrados en la frecuencia de 13.5MHz. En los sucesivos capítulos se explicará con más [1]detenimiento los fundamentos de dicha tecnología RFID así como de las radios SDR.

Una vez planteado el punto de partida en lo que a conocimientos fundamentales de estas tecnologías se refiere, se abordará la solución técnica concreta que se ha llevado a cabo para la realización tangible del sistema, desde el diseño esquemático de la parte analógica pasando por el rutado de la PCB hasta el planteamiento digital que conforma el transmisor y receptor implementado en la FPGA.

1.2 Fases del proyecto

A continuación, se presenta en qué fases se ha dividido el proyecto siguiendo la premisa “Divide y vencerás”. En la tabla 1.1 puede apreciarse un resumen de estas.

FASES DEL PROYECTO		
Fase 1: Especificación del proyecto	1.1 Estudio de los fundamentos de RFID	
	1.2 Estudio de los fundamentos de SDR	
	1.3 Planteamiento del sistema completo	
Fase 2: Diseño electrónica analógica	2.1 Búsqueda y selección de componentes	
	2.2 Diseño Esquemático	2.2.1 Diseño Transmisor
		2.2.2 Diseño Receptor
		2.2.3 Análisis consumo potencia
	2.3 Simulación	
	2.4 Diseño PCB	2.4.1 Creación de huellas
		2.4.2 Estudio Compatibilidad Electromagnética
		2.4.3 Reglas de diseño
		2.4.4 Rutado de conexiones
		2.4.5 Rutado de la antena
	2.5 Generación Gerber y Drill files	
	2.6 Confección B.O.M.	
Fase 3: Manufactura PCB	3.1 Envío a fabrica	
	3.2 Pedido B.O.M	
	3.3 Montaje PCB	
Fase 4: Verificación PCB	4.1 Verificación Transmisor	
	4.2 Verificación Receptor	
Fase 5: Diseño electrónica digital	5.1 Fundamentos FPGA	
	5.2 Diseño Transmisor	5.2.1 Diseño UART RX
		5.2.2 Diseño Modulador
	5.3 Simulación transmisor	
	5.4 Síntesis transmisor	
	5.5 Diseño receptor	5.5.1 Diseño UART TX
		5.5.2 Diseño cadena de recepción
	5.6 Simulación receptor	
	5.7 Síntesis receptor	
Fase 6: Ajuste del sistema		
Fase 7: Verificación del sistema completo		
Fase 8: Documentación final del proyecto		

Tabla 1.1:Fases del proyecto

2.Introducción a la tecnología RFID

En este capítulo se pretende dar una caracterización inicial de la tecnología mediante una primera aproximación a los elementos que conforman un sistema RFID y los principios básicos de operación referidos a los aspectos de la física electromagnética, velocidad de transmisión de los datos, codificación y modulación de los mismos, problemas de colisión de información, estandarización, etc. Además se presenta el funcionamiento del estándar ISO 1444-3 en el que se basa el sistema diseñado en este trabajo.

Un sistema RFID (Radio Frequency Identification) consiste en una tecnología que permite el intercambio de datos (Principalmente de identificación) entre un lector y una etiqueta , todo ello sin la necesidad de que exista un enlace mecánico o visual entre dichos dispositivos.

Las posibilidades que oferta un sistema que brinda una identificación automática de forma inalámbrica han sido aprovechadas por todo tipo de sectores, como aquellos en los que se requiere de una identificación personal para acceder a ciertas áreas restringidas (Peajes, acceso a edificios, entrada al metro, centros deportivos, pasaporte electrónico, etc), sector animal donde es necesario un seguimiento e identificación de las cabezas de ganado o mascotas domésticas, sector logístico donde es esencial llevar un registro minucioso del recorrido y localización de la mercancía, etc.

Siempre que se trata el concepto de identidad de manera intrínseca se debe hablar de seguridad, puesto que todos estos sectores no depositarían su confianza en un sistema incapaz de preservar la identidad de, por ejemplo, millones de cuentas bancarias.

Por ello, no se ha podido aprovechar todo el potencial de esta tecnología hasta que los mecanismos criptográficos , tanto físicos como en términos algorítmicos, no han alcanzado cierto grado de excelencia.

La comunicación entre el lector y la etiqueta está basada en la generación de ondas de radiofrecuencia. El lector inicia la conversación emitiendo una señal sinusoidal portadora que, según el tipo de sistema RFID, alimenta a la etiqueta puesto que esta suele ser pasiva. La información se añade a esa onda portadora modificando alguno de los parámetros característicos como son su amplitud, su frecuencia o su fase, es decir; aplicándole a la portadora una modulación. Las modulaciones típicas de estos sistemas son la ASK (Amplitude Shift Keying), FSK(Frecuency Shift Keying) y PSK (Phase Shift Keying).

Tras recibir la información pertinente modulada en la portadora que emite el lector, la etiqueta responde indicando su identificador único. En aplicaciones RFID de campo cercano, cuando la distancia entre el lector y la etiqueta es comparable a la longitud de onda de la portadora se produce un acoplamiento inductivo entre las antenas de cada dispositivo. Por otro lado, en aplicaciones de campo lejano el acoplamiento se produce por radiación.

En los subsiguientes apartados se tratará con más detalle los tipos de sistemas RFID según este tipo de características.

Una vez el lector ha obtenido el identificador único de la etiqueta, se puede iniciar otra fase a un nivel más elevado de la comunicación con la intención de extraer otra información relevante de la tarjeta como puede ser un historial médico de un paciente, una dirección a un portal web o la cartilla de esterilización de un animal.

Es importante mencionar que cabe la posibilidad de que, si hay dos etiquetas presentes en el radio del lector, pueden producirse colisiones entre la información de dichas etiquetas. Todos los sistemas RFID constan de mecanismos anticolisión para evitar estas situaciones.

2.1 Componentes de un sistema RFID

Como ya se ha mencionado anteriormente, los elementos que conforman un sistema RFID son esencialmente dos: un lector y una etiqueta.

2.1.1 Etiqueta RFID

Las etiquetas RFID reciben nombres muy diversos. A las etiquetas también se las nombra Tags o Transponders. Según el estándar concreto estas se describen con otras denominaciones, como por ejemplo PICC para el estándar ISO 14443 el cual es en el que se basa el sistema de este proyecto.

La información de la etiqueta puede ser únicamente leída pero también puede ser grabada. La escritura y/o lectura de la información depende de la memoria que tenga integrada la etiqueta. La mayoría de transponders incluyen una memoria EEPROM (Electrically Erasable Programmable Read-Only Memory) donde el fabricante incrusta su identificador único.

En la figura 1.1 se puede apreciar un ejemplo de la arquitectura interna de una etiqueta [1], en este caso de un sistema RFID de largo alcance en la banda de frecuencias UHF.

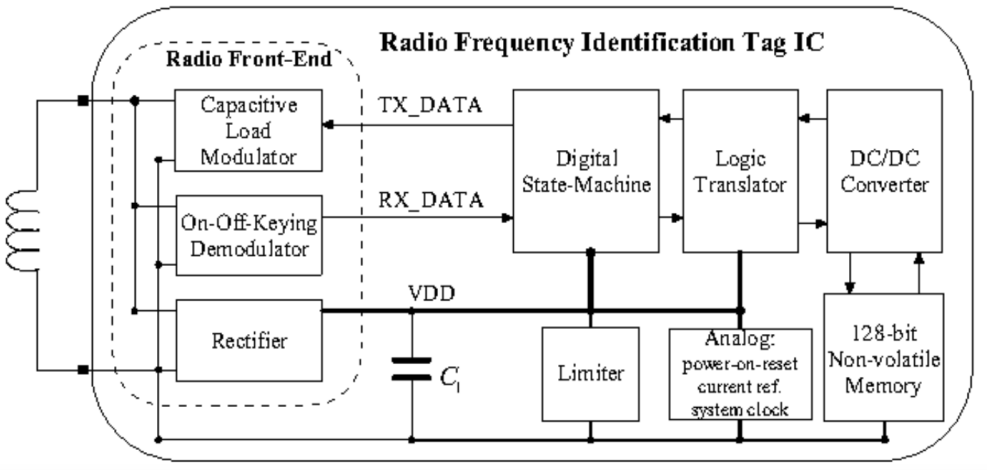

Figura 1.1: Arquitectura típica de una etiqueta RFID

Se puede apreciar la presencia de la memoria no volátil que contiene el identificador y demás información relevante según la aplicación dada a la tarjeta. El transponder debe de contar con algún tipo de memoria ROM que almacene la memoria de programa para efectuar la secuencia de comunicación según las órdenes recibidas por el lector.

También se puede apreciar el bloque encargado de modular la señal que se emite de vuelta al lector.

Uno de los elementos más importantes tanto del tag como del lector es la antena. A través de la antena , desde el punto de vista del transponder, tiene lugar la comunicación pero también puede ser el punto de entrada de la alimentación, según el tipo de sistema.

Una de las formas en las que se clasifican las etiquetas radica en su forma de alimentarse. Las etiquetas pasivas aprovechan la energía generada por la portadora del lector, lo que supone la gran ventaja de no tener que recurrir a baterías que necesiten recargarse y que acortan la vida del dispositivo.

Sin embargo , este tipo de alimentación acarrea ciertas desventajas ya que sin ir mas lejos afectan al tipo de codificación y a la modulación que debe presentar tanto la portadora del lector como la respuesta del tag. Otro de los aspectos a tener en cuenta para este sistema de alimentación es la distancia entre los entes implicados. Resulta fácil deducir que la electrónica de los transponders debe reducir a la mínima expresión el consumo de potencia necesario para su funcionamiento.

En la figura 1.2 aparece representado el diagrama de bloques de un sistema de alimentación concreto utilizado por un transponder [2]. En este ejemplo en concreto la eficiencia de conversión del rectificador no supera el 40%.

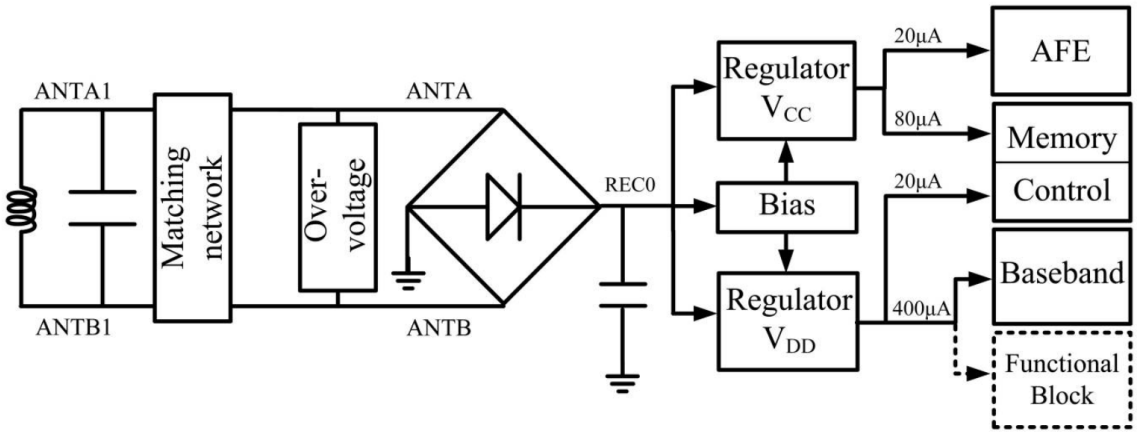

Figura 1.2: Bloques habituales del circuito de alimentación de un tag RFID

Por otro lado, las etiquetas con alimentación activa utilizan baterías que propician una comunicación a distancias mucho más grandes y velocidades mayores. Sin embargo, estas tarjetas tienen un tiempo de vida mucho más limitado y un precio bastante superior a las pasivas.

En lo que se refiere al formato físico, típicamente se tiende a pensar en un transponder RFID como una tarjeta, sticker o etiqueta. Sin embargo, muchos de los transponders están implementados en forma de ampolla, moneda o incluso botones. En la figura 1.3 aparecen recogidas las dimensiones físicas [3] que deben cumplir los transponders del estándar ISO 14443.

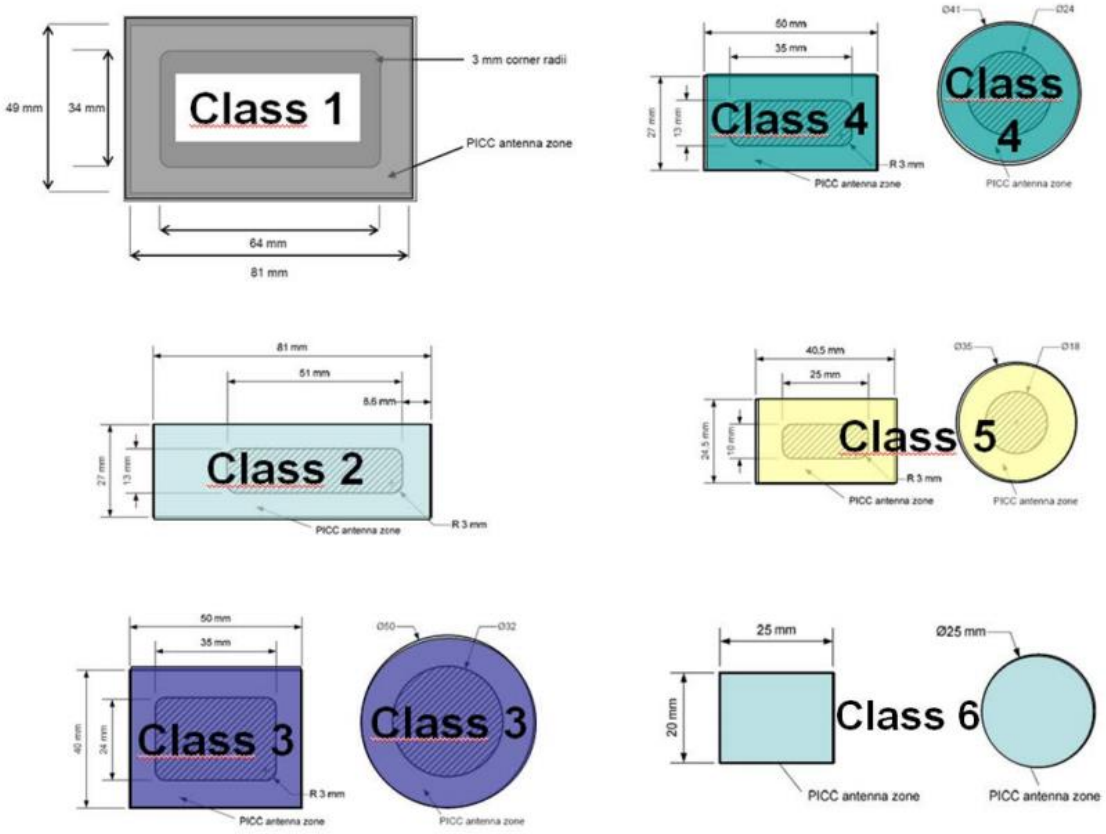

Figura 1.3: Dimensiones físicas de etiquetas RFID para el estándar ISO 14443

2.1.2 Lector RFID

El lector tiene encomendada la tarea de generar la señal portadora modulada, detectar la respuesta del tag y acondicionarla para su decodificación y corregir los errores que se hayan podido ir produciendo en la comunicación.

Los procesos de comprobación de errores más empleados en RFID son el LRC (Longitudinal Redundancy Check) y el CRC (Cyclic Redundancy Check). En sistemas que, como veremos a continuación, trabajan a frecuencias más altas se puede lograr una tasa de lectura de hasta 200 transponders por segundo.

El lector también tiene encomendada la tarea de alimentar a las etiquetas pasivas mientras transmite la información. Hay ciertos tipos de lectores que, con ánimo de facilitar esta tarea simultánea, utilizan dos antenas; una de ellas para el aspecto energético y otra para el canal de comunicaciones. Sin embargo, estos sistemas son más complejos y caros de implementar.

Hay varias formas de actuación de los lectores. Estos pueden sondear el medio de forma periódica en busca de la presencia de alguna etiqueta o bien pueden sondear el medio puntualmente cuando se tiene la certeza de que en ese instante pueda estar presente el transponder.

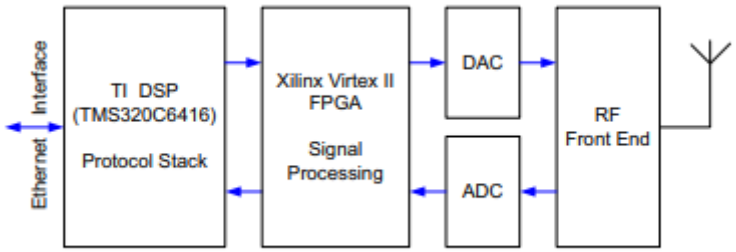

Figura 1.4: Propuesta de arquitectura para lector RFID

Tal y como se presenta en la figura 1.4 los bloques que conforman un lector [4]comienzan por el AFE (Analog Front End). El AFE lo integran la antena y todos aquellos elementos necesarios para acondicionar la señal tanto para la transmisión (Amplificación y filtrado) como para la recepción (LNA, mezclador, filtrado antialiasing, filtrado intermedio, etc).

Esta arquitectura no es rígida puesto que elementos del AFE pueden implementarse digitalmente en el procesador lógico, como puede ser el mezclado. Cada fabricante diseña sus lectores según la conveniencia de sus intereses. En el ejemplo arriba esquematizado, los datos recibidos por el AFE son digitalizados a través de un ADC (Analog to Digital Converter) para su procesamiento (En el ejemplo la FPGA se encargaría de dicha tesitura) y transmisión hacia un dispositivo que los trate con software de nivel aplicación. Este dispositivo puede estar embebido en el sistema que conforma el lector (Como, por ejemplo, un DSP) o bien tratarse de un ordenador de propósito general, conformando un sistema distribuido.

Para la emisión el lector recibe las ordenes desde el ordenador vía ethernet para desensamblarlos y codificarlos según el estándar empleado. Un DAC (Digital to Analog Converter) se encarga de transformar la señal digital en analógica, para entregársela al AFE y que le aplique el acondicionamiento adecuado necesario para la correcta transmisión por la antena.

2.2 Diferenciación de sistemas RFID

Los distintos tipos de sistemas RFID se pueden clasificar atendiendo a criterios muy diversos entre sí, como puede ser la alimentación pasiva o activa ya mencionada de los transponders, el volumen de datos de transmisión o su principio físico de funcionamiento.

En este apartado se pretende condensar estos criterios mencionados de modo que la clasificación quede compendiada según dos tipos de parámetros: La distancia de operación y la frecuencia de trabajo.

2.2.1 Distancia de operación

Si bien es cierto que los factores que condicionan principalmente el rango de alcance del sistema son la orientación del tag, la distancia entre transponders en el radio de operación y la velocidad del tag en el momento en el que se produce la lectura hay que tener en cuenta que el factor más determinante es el principio físico de funcionamiento.

Distancia de operación menor de 1 cm

Son los sistemas con una menor implementación en el panorama comercial de RFID. Su mayor aporte radica en que son sistemas con fuertes medidas de seguridad en lo que a protección de datos se refiere. Su rango de frecuencias suele llegar hasta los 30MHz. Las frecuencias más utilizadas oscilan entre 1 Mhz y 10Mhz. Esto es así debido a que la energía obtenida por los transponders pasivos es proporcional a la frecuencia de operación, cuanto mayor es dicha frecuencia mayor voltaje es capaz de rectificar la etiqueta.

El principio físico de este tipo de sistemas es el denominado "close coupling" que utiliza tanto campos eléctricos como campos magnéticos. Para que la comunicación tenga lugar, el transponder debe de estar rodeado por el lector, de modo que la bobina que conforma la antena del lector envuelva ala bobina del tag. El acoplamiento se produce de la misma manera en la que ocurre con un transformador, representando el lector las espiras primarias y el transponder las secundarias. Es un principio de funcionamiento bastante parecido al que utilizan los dispositivos del apartado que prosigue, el llamado acoplamiento inductivo.

Distancia de operación menor de 1m

Los sistemas capaces de trabajar en rangos de operación que van desde 0.1 cm hasta algo más de 1 m funcionan mediante acoplamiento inductivo. El acoplamiento inductivo consiste en un acoplamiento magnético tal y como ocurre con un transformador. La corriente que fluye por la bobina del lector induce un campo magnético a su alrededor que a su vez induce una corriente a través de la bobina del transponder, tal y como dicta la ley de Faraday.

Las bobinas que conforman las antenas de estos sistemas suelen ser de tamaño más grande puesto que la longitud de onda que emplean es sensiblemente mayor que la

distancia entre el tag y el lector. El campo electromagnético generado se puede interpretar como un campo magnético oscilante respecto a la distancia entre ambos dispositivos del sistema. Estamos hablando de longitudes de onda que pueden ir desde los 2400 m hasta los 22.4 m ya que el rango de frecuencias empleado para esta categoría está enmarcado entre los 125 KHz y los 13.56 MHz.

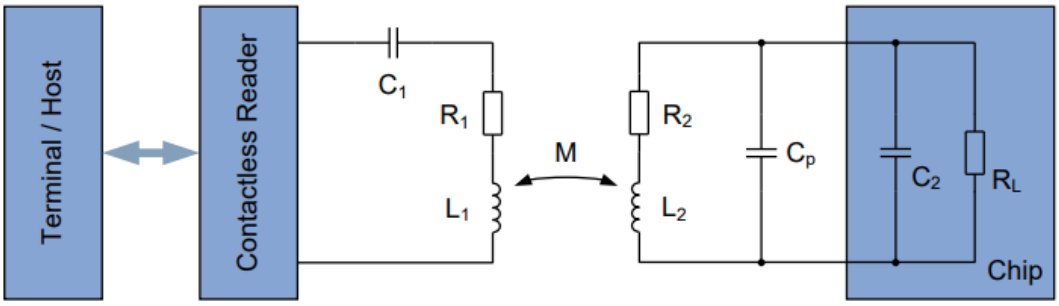

Figura 1.5 Circuito equivalente de las antenas del Tag y Lector.

En la figura 1.5 se representa el circuito resonante que conforma la antena [5]. Tanto el lector como el transponder emiten sus señales a la misma frecuencia a la que resuena el circuito, que no deja de ser un filtro pasa banda con un factor de calidad Q que no debe de ser demasiado elevado. Esta coincidencia proporciona un voltaje máximo debido a la sintonia del filtro pasabanda.

La frecuencia de trabajo, la relación entre el número de espiras del lector y transceptor, el área de la bobina, el ángulo entre bobinas y la distancia entre si son proporcionales a la eficiencia de la energía transmitida entre los dispositivos.

Puesto que el proyecto que se relata en el presente documento está basado en el principio físico del acoplamiento inductivo se reserva para capítulos posteriores una explicación mucho más detalla y cuantitativa del funcionamiento de dicho mecanismo que conforma la cápa física del sistema.

Distancia de operación mayor de 1m

Los sistemas cuyo rango de operación se encuentra por encima de 1 metro de distancia utilizan el principio físico del acoplamiento radiativo, también denominado “backscatter”, lo que les permite alcances de unos 15 metros para tags activos y de hasta 3 metros para transponders pasivos.

En el “backscatter” a medida que la onda electromagnética se aleja de la antena el campo magnético se va atenuando (Recordemos que el acoplamiento inductivo se aprovechaba de este campo magnético de campo cercano) hasta ser insignificante en regiones de campo lejano. Cuando las ondas electromagnéticas se encuentran con la antena del receptor de dimensiones de aproximadamente la mitad de la longitud de onda de dichas ondas electromagnéticas, estas se reflejan; además, con mayor intensidad puesto que los objetos que están en resonancia con el frente de ondas que los golpea

(Como es el caso de la antena del transponder a la frecuencia de operación) tienen una sección transversal de reflexión especialmente elevada.

Es en esta reflexión donde va incluida la información del tag. El campo electromagnético proveniente del emisor llega al transponder en una proporción pequeña del campo inicial, debido a las pérdidas por espacio libre.

Las pérdidas por espacio libre son la relación entre la potencia emitida por el transmisor y la potencia que realmente acaba llegando al receptor, todo ello a una frecuencia determinada. La energía recibida por el transponder disminuye con el cuadrado de la distancia entre los dos dispositivos. Estas pérdidas deben de tenerse en cuenta a la hora de dimensionar la potencia del emisor en el sistema.

Los sistemas que utilizan este principio físico trabajan en el rango de frecuencias de UHF, en concreto a 868 MHz en Europa y en el rango de las microondas, en concreto a 2.4 GHz y 5.8 GHz. Estas frecuencias permiten, por el tamaño de su longitud de onda, utilizar antenas realmente eficientes muy pequeñas que ocupen espacios reducidos.

En la tabla 1.2 se recoge una comparativa entre las atenuaciones sufridas según la distancia a ciertas frecuencias de funcionamiento.

Distancia	868 MHz	915MHz	2.4 GHz
0.3 m	18.6 dB	19 dB	27.6 dB
1 m	29 dB	29.5 dB	38dB
3 m	38.6 dB	39 dB	47.6 dB
10 m	49 dB	49.5 dB	58 dB

Tabla 1.2: Perdidas según una ganancia de 1.64 en la antena del tag (dipolo) y una ganancia de 1 en la antena del lector (Isotrópica)

Podemos resumir que la diferencial principal entre los sistemas que utilizan el acoplamiento inductivo y los que emplean el backscatter radica en que los primeros utilizan la energía generada por una antena en su alrededor (Campo cercano) y los segundos emplean ondas electromagnéticas radiadas hacia campos lejanos. La figura 1.6 resume las distancias de operación de estos principios físicos de funcionamiento [6].

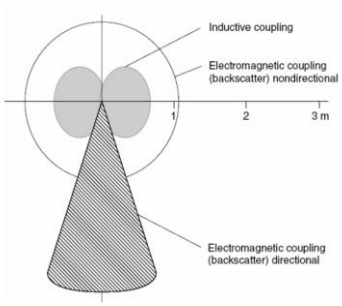

Figura1.6:Rangos de operación según principio de funcionamiento.

2.2.2 Rangos de frecuencia

Como todo sistema basado en las ondas de radio, los diferentes tipos de sistemas RFID deben enmarcarse dentro de la regulación que asigna y determina los usos de las distintas bandas de frecuencia. El funcionamiento de los demás sistemas de radio no debe verse perjudicado en ningún momento por las radiaciones electromagnéticas que emplea la tecnología RFID.

La banda de frecuencias asignada a este tipo de dispositivos se corresponde con la banda ISM (Industrial-Scientific-Medical) cuyo uso está reservado para aplicaciones médicas, industriales o de uso científico. Estas frecuencias no necesitan de licencia gubernamental para poder operar. Sin bien es cierto que son las más propensas a sufrir perturbaciones también son las mas utilizadas.

La banda ISM no se centra únicamente en un rango como, por ejemplo, el de HF (High Frequency) sino que está repartida por todo el espectro, teniendo rangos reservados en todas las bandas. En la figura 1.7 aparece representado un esquema con la presencia de las frecuencias ISM en las que operan los distintos tipos de sistemas RFID dentro de cada banda general de frecuencias además del principio físico que emplean [7].

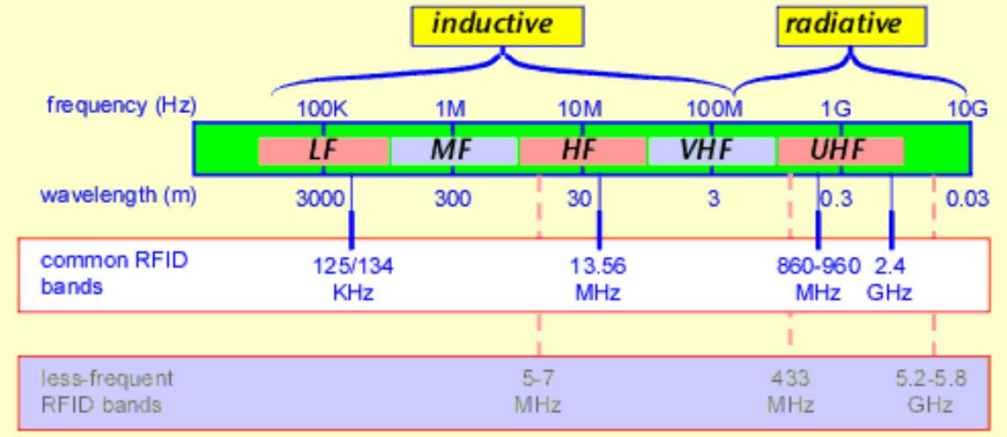

Figura 1.7: Frecuencias RFID de cada banda ISM

Low Frequency (LF) RFID

Los sistemas RFID de baja frecuencia operan en el rango de entre los 30 KHz y los 300KHz. Los más utilizados están centrados en 125KHz y 134 KHz. Los transponders son pasivos y utilizan el acoplamiento inductivo para su funcionamiento.

Las transferencias de datos son realmente bajas , entre 200bps y 1Kbps. Estos sistemas son muy resistentes a interferencias externas, con un rango de operación de hasta 10 cm en el mejor de los casos.

Las aplicaciones más comunes son el control de acceso y la identificación animal para la gestión de ganado o autentificación de mascotas domésticas.

Estos sistemas de baja frecuencia están recogidos por los estándares ISO 14223 y ISO/IEC 18000-2.

Tag	Frec. Portadora	Frec. Subportadora	H 10m
NEDAP	120 KHz	Fc ± 2 KHz	-28 dBuA/m
ISO 11785 FDX	134.2 KHz	Fc ± 4 KHz	-32 dBuA/m
ISO 11785 HDX	134.2 KHz	Fc ± 4 KHz	-28 dBuA/m
ISO 18000-2	125/134.2 KHz	Fc ± 4,64,0,-10 KHz	-32/-28 dBuA/m

Tabla 1.3: Frecuencias e intensidades para los estándares de la banda LF

High Frequency (HF) RFID

Los sistemas RFID de alta frecuencia operan en el rango de entre 3 y 30 MHz, centrándose en los 13.56MHz. Las etiquetas son pasivas y su funcionamiento se basa en el acoplamiento inductivo.

Dentro de los sistemas de HF se puede distinguir principalmente entre las "proximity cards" , que alcanzan hasta unos 10 cm de rango, y las "vicinity cards" que alcanzan hasta 1.5 m.

Los transponders manejan un volumen de datos que puede ir desde los 512 bits hasta 8 Kbits. Las velocidades típicas de transmisión alcanzan los 25 Kbps – 100 Kbps. Hay sistemas como el FeliCa que alcanzan tasas de hasta 13.56 Mbps.

Las aplicaciones más utilizadas son el control de accesos, seguimiento logístico de mercancías o los pagos electrónicos como los pagos NFC (Near Field Communication).

Los estándares que rigen los dispositivos RFID de alta frecuencia son el ISO 15693 para el "tracking" de objetos, ISO 15693 para vicinity cards, , ISO/IEC 18092 para el NFC. Las proximity cards se rigen por el estándar ISO/IEC 14443.

Tag	Frec. Portadora	Frec. Subportadora	H 10m
ISO 14443 Proximity	13.56 MHz	Fc ± 846 KHz	-41 dBuA/m
ISO 15693 Vicinity	13.56 MHz	Fc ± 423 KHz	-36 dBuA/m
ISO 18000-3	13.56 MHz	Fc ± 423 KHz	-36 dBuA/m
I-CODE Philips	13.56 MHz	Fc ± 423 KHz	-36 dBuA/m
Tag-It TIRIS	13.56 MHz	Fc ± 423 KHz	-36 dBuA/m
Microchip	13.56 MHz	Fc ± 70 KHz	-24 dBuA/m
Gemplus	13.56 MHz	Fc ± 212 KHz	-27 dBuA/m
Legic	13.56 MHz	Fc ± 212 KHz	-33 dBuA/m

Tabla 1.4: Frecuencias e intensidades para los estándares de la banda HF

El sistema desarrollado en este trabajo pertenece a la banda de HF y está regido bajo el estándar ISO 14443-A.

Ultra high frequency (UHF) RFID

Los sistemas RFID de ultra-alta frecuencia operan desde los 300MHz hasta los 3GHz, centrándose en los 868 MHz en Europa y en los 960 MHz para EEUU. Las etiquetas son tanto pasivas como activas y su funcionamiento se basa en el "backscatter".

Los rangos de operación pueden llegar hasta los 15 m en tags activos y 4 m para tags pasivos. Aunque son muy sensibles a las interferencias, los transponders son más fáciles y baratos de fabricar que los LF y HF.

Las aplicaciones de estos sistemas se centran en situaciones en las que se requiere una distancia de lectura mayor que los anteriores tipos de sistemas, como por ejemplo el control antirrobo de mercancías o la lectura de sensores embebidos en transponders RFID puesto que en UHF el volumen y la tasa de transmisión de datos es muy superior.

La frecuencia UHF esta principalmente regulada por un estándar global denominado EPC global Gen2 o ISO 1800-63.

2.3 El estándar 14443-A: Proximity Cards

A pesar de que puede soportar otros estándares RFID basados en la banda de los 13.56MHz, el sistema diseñado en este trabajo se centra en las denominadas Proximity Cards cuyo funcionamiento está regulado por la norma ISO 14443 [8]. Es por ello que antes de finalizar el apartado dedicado a la comprensión y contextualización de la tecnología RFID se hace especial hincapié en los fundamentos de los sistemas basados en esta norma para una óptima comprensión (Además de necesaria) del sistema planteado en el capítulo 4.

Este apartado se divide en dos partes; la primera centrada en lo que puede considerarse la capa física (En concordancia con la torre OSI definida por la ISO) desarrollada en el estándar ISO 14443-2 y una segunda parte que puede considerarse como una capa de transporte/enlace presentada en el estándar ISO 14443-3. El estándar tiene una tercera parte centrada en lo que puede considerarse la capa de aplicación. Sin embargo, no se hará hincapié en ella puesto que se aleja de los intereses y objetivos del sistema diseñado en este trabajo.

2.3.1 Interfaz de señal y potencia de radiofrecuencia

Esta parte del estándar habla acerca de las características de los campos y las señales que proporcionan energía a los transponders además de proveer la comunicación bidireccional entre el lector y las etiquetas.

Para el caso concreto de esta norma, a los lectores se les denomina PCD (Proximity Coupling Devices) y los transponders se les denomina PICC (Proximity Card).

Transferencia de Potencia

El PCD debe de generar un campo RF que se acople con el PICC para transmitirle la potencia necesaria para alimentar sus circuitos internos, ya mencionados en capítulos anteriores. El principio físico de funcionamiento es el también ya presentado acoplamiento inductivo.

La frecuencia de la portadora que alimenta al PICC es de 13.56MHZ con un margen de ±7 KHz.

El valor mínimo del campo generado (Sin modular) debe de tener un valor de H_{min}=1,5 A/m rms y el valor máximo del campo no debe de sobrepasar los H_{max}=7,5 A/m rms.

Hay dos métodos para calcular analíticamente dicho campo: El momento magnético y la Ley de Biot-Savart [9].

Una aproximación del campo magnético para el campo cercano viene dada por la ecuación 1.1 donde N hace referencia al número de espiras de la bobina de la antena, I a la corriente de la antena del lector, r al radio de esta y d a la distancia desde el centro de la antena en dirección perpendicular al plano que conforma la bobina.

$$H_{Campo\ Cercano} = \frac{N \times I}{2} \times \frac{r^2}{(\sqrt{r^2 + d^2})^3} \qquad (1.1)$$

Si se realiza un análisis gráfico de la ecuación 1.1 se obtiene la curva representada en la figura 1.8 [5] donde puede apreciarse la intensidad del campo magnético a medida que aumenta la distancia. El número de espiras es N=1, la intensidad de la bobina I=1,1125ª y el radio tiene un valor de r=0,0075 m.

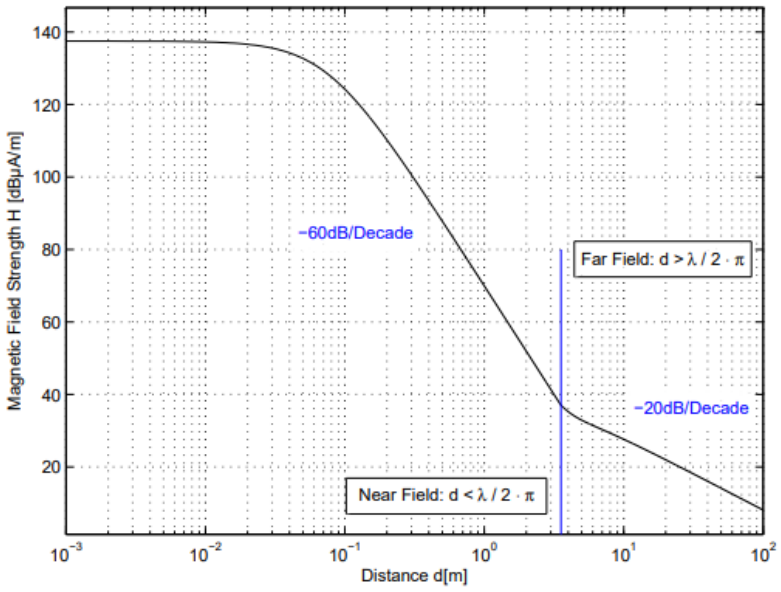

Figura 1.8: Fuerza del campo magnético V.S. distancia.

En ningún momento el lector debe generar un campo que no se encuentre dentro del rango marcado para H_{min} y H_{max}..

Interfaz de señal PCD hacia PICC

La comunicación entre el PCD y el PICC es half-duplex, el lector "habla" primero iniciando la comunicación. El bit rate durante la inicialización de la comunicación debe de ser de 106 Kbit/s, lo que equivale a 1/128 la frecuencia de la portadora.

La modulación que el PCD imprime a la portadora es ASK 100% con una codificación Miller modificada. Para ilustrar dicha codificación se definen las siguientes secuencias, resumidas en la figura 1.9:

- Secuencia X: La amplitud de la portadora permanece en baja durante un cuarto del tiempo de bit a partir de la mitad del tiempo de bit.
- Secuencia Y: La amplitud de la portadora no cambia durante un tiempo de bit completo (Permanece en alto).
- Secuencia Z: La amplitud de la portadora permanece en baja durante un cuarto del tiempo de bit al principio del tiempo de bit.

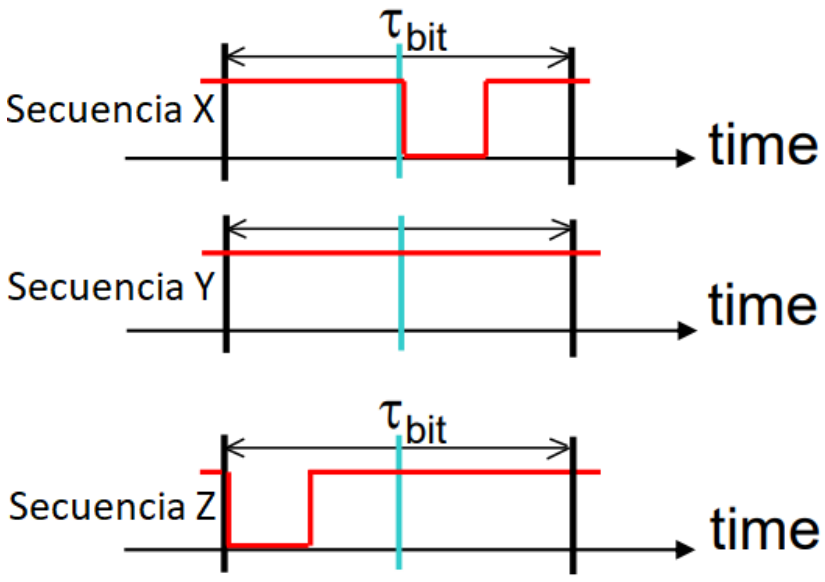

Figura 1.9: Secuencias para la modulación Miller

La lógica binaria de la codificación Miller se implementa de la siguiente manera, utilizando las secuencias descritas:

- Lógica “1”: Secuencia X.
- Lógica “0”: Secuencia Y con las siguientes excepciones:
    - Si hay dos o más “0” seguidos se utiliza la secuencia Z para el segundo “0” (Y los subsiguientes).
    - Si el primer bit después del bit de start es “0” este se codifica con la secuencia Z.
- Bit de start: Secuencia Z.
- Bit de end: Lógica “0”seguida de una secuencia Y.

Para ilustrar con un ejemplo la codificación que emplea el PCD para comunicarse con el PICC procederemos a emitir el carácter 0x26 en hexadecimal (Dicho carácter es el que emplea el PCD para iniciar la comunicación, véase el apartado 2.3.2.).

Hay que mencionar que primero se emite el bit menos significativo y que se emiten únicamente los 7 LSB. Por tanto, la codificación quedaría tal y como se expone en la figura 1.10.

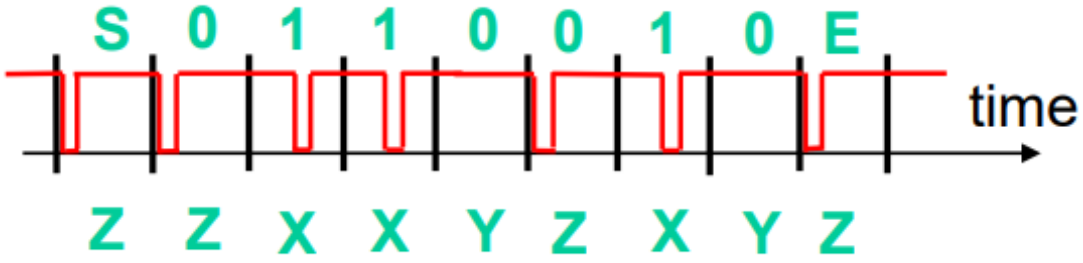

Figura 1.10: Codificación Miller modificado para el carácter 0x26

La envolvente de la portadora seguiría idealmente la forma expuesta en la figura 1.10. Sin embargo, la portadora no sigue unos cambios tan abruptos en su amplitud puesto que proviene del circuito resonante que conforma la antena. Esto implica que existe un factor de calidad Q que limita el ancho de banda y que provoca que la energía almacenada en el circuito resonante no se disipe de forma inmediata, si no siguiendo una caída exponencial.

Estos detalles referidos al factor Q se discuten en el apartado dedicado al diseño de la antena de nuestro sistema concreto.

La amplitud de la portadora debe de decrecer monótonamente en un tiempo determinado hasta llegar a situarse por debajo del 5% de la amplitud máxima. Algo similar ocurre con los tiempos de subida. En la figura 1.11 [8] aparece representada la forma de la envolvente junto con los tiempos que han de cumplirse.

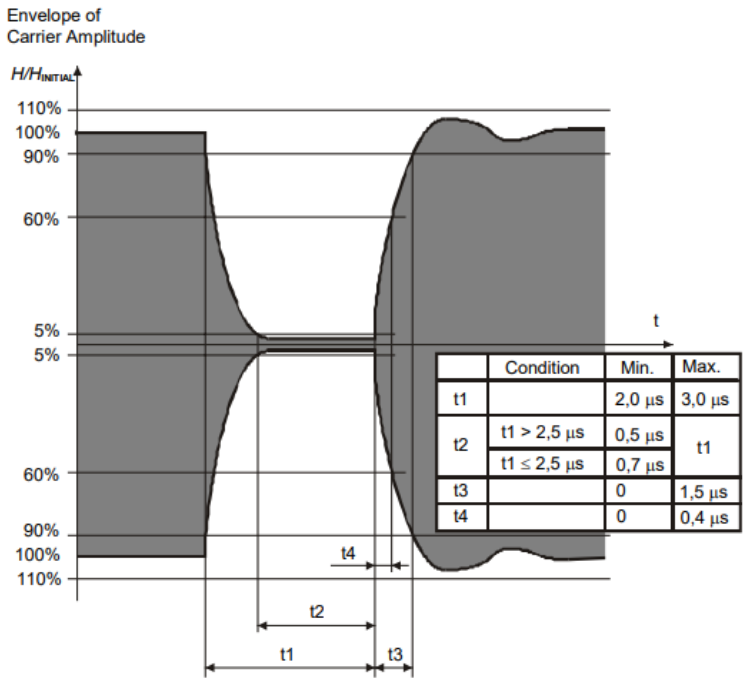

Figura 1.11: Envolvente de la señal cuando la codificación exige un nivel bajo de amplitud

En la figura 1.12 puede apreciarse un ejemplo de la apariencia real que tendría la portadora modulada con una serie de bits codificados en Miller modificado [10].

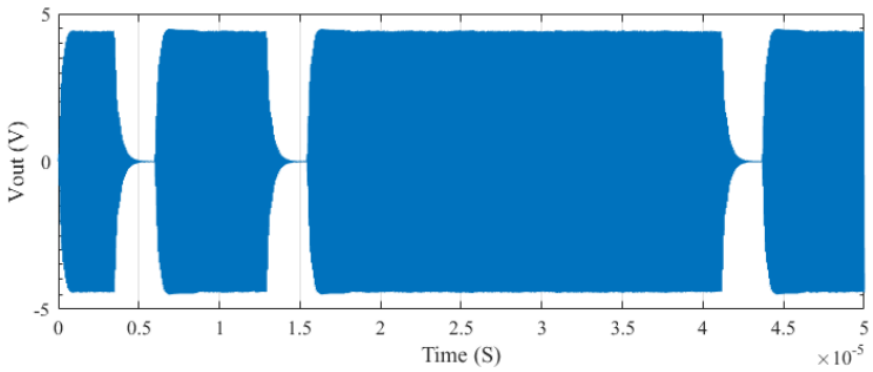

Figura 1.12: Apariencial real de la portadora con modulación ASK 100% y codificación Miller modificado.

Interfaz de señal PICC hacia PCD

El bit rate de la transmisión desde el PICC hacia el PCD durante el proceso de inicialización y anticolisión es de 106 Kbit/s ($F_{portadora}/128$). Sin embargo, la tasa de bits puede ser mayor para aplicaciones de alto nivel que requieran de mayor volumen de datos.

La comunicación desde el PICC hacia el PCD está basada en la llamada modulación de carga (Load Modulation). El acoplamiento entre ambas antenas se describe por la inductancia mutua M, tal y como se describía en la figura 1.5. El campo magnético generado por la antena del PCD induce un voltaje en la antena del PICC. Dicho voltaje

produce una corriente que genera un campo magnético en la antena del PICC el cual retroalimenta al campo del lector.

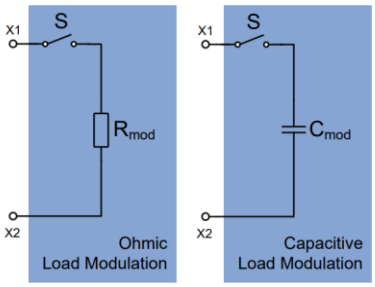

Figura 1.13: Circuito equivalente de la modulación de carga capacitiva y óhmica.

La modulación de carga se consigue alterando la impedancia del transponder, distinguiéndose entre una modulación de carga capacitiva o una modulación de carga óhmica. El circuito equivalente de dicha modulación aparece representado en la figura 1.13.

Debido a la inductancia mutua M, una variación en la corriente de la bobina del PICC produce una variación en la corriente de la bobina del PCD que debe ser detectada por el lector. Debido a las perdidas por el acoplamiento entre ambas antenas, la señal detectada es mucho más pequeña que la portadora (Alrededor de 80 dB más pequeña). Por este motivo, en lugar de incluir la información de regreso en la misma portadora la modulación de carga se aplica en una subportadora. La separación entre la subportadora y la portadora facilita en gran medida la recuperación de la información por parte del PCD.

La frecuencia de la subportadora es 1/16 de la frecuencia de la portadora, es decir, la frecuencia de la subportadora es de 847,5 KHz. En el proceso de la modulación se generan dos bandas laterales dispuestas simétricamente alrededor del espectro de la portadora que contienen los datos banda base que el PICC pretende transmitir. En la figura 1.14 se puede observar el espectro que se genera en la comunicación entre el lector y el tag [5].

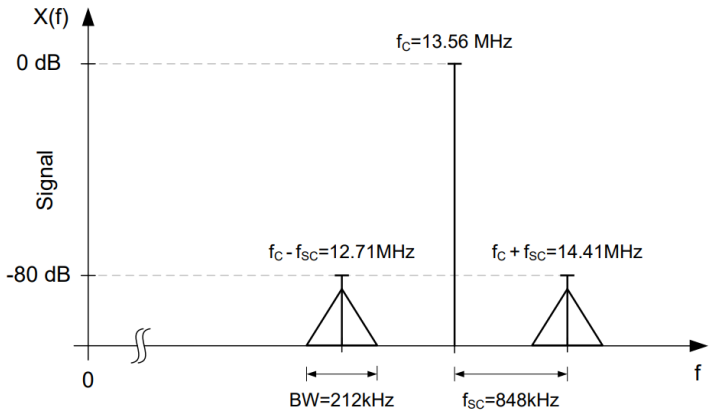

Figura 1.14: Componentes espectrales en la comunicación entre el PICC y el PCD

La modulación OOK (On/Off Keying) es la aplicada a la portadora con una codificación Manchester. La codificación Manchester es del tipo definido originalmente por E.G. Thomas en 1949 (También seguida por muchos otros autores como A.S. Tanenbaum) en la que el bit 1 se representa con una transición alto-bajo y el bit 0 se representa con una transición bajo-alto.

Para describir la codificación de los bits, se definen las siguientes secuencias de forma análoga a la comunicación desde el PCD hacia el PICC:

- Secuencia D: la portadora debe ser modulada con la subportadora durante la primera mitad del tiempo de bit.
- Secuencia E: la portadora debe ser modulada con la subportadora durante la segunda mitad del tiempo de bit.
- Secuencia F: La portadora no es modulada con la subportadora durante un tiempo de bit completo.

La lógica binaria emplea estas secuencias para codificarse de la siguiente manera, incluyendo el bit de start y de stop:

- Lógica “1”: Secuencia D.
- Lógica “0”: Secuencia E.
- Bit de start: Secuencia D.
- Bit de stop: Secuencia F.

La subportadora emplea 8 ciclos para codificar 1 tiempo de bit (Recordando que la tasa de bits es de 106 KHz, su duración es de 9.43 us). En la figura 1.15 aparece un ejemplo de la portadora con las secuencias que tendrían lugar para codificar el número binario 10010. Al igual que ocurría con la comunicación desde el PCD, el bit menos significativo es el primero en transmitirse.

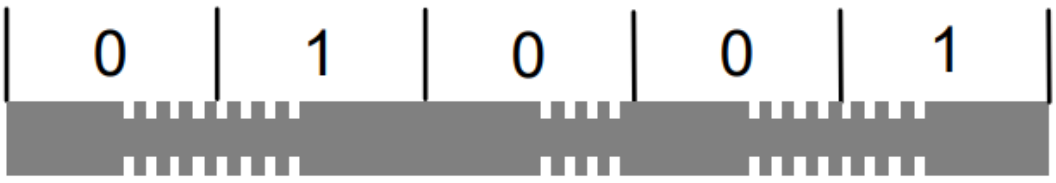

Figura 1.15: Modulación de la portadora correspondiente con la respuesta de un PICC.

En la tabla 1.5 se resumen las modulaciones y codificaciones del sistema para el estándar ISO 14443-A.

<table>
<tr><td rowspan="10">ISO 14443-A</td><td rowspan="5">PCD → PICC</td><td>Modulación</td><td>Frecuencia Portadora</td></tr>
<tr><td>ASK 100%</td><td rowspan="4">13,56 MHz</td></tr>
<tr><td>Codificación</td></tr>
<tr><td>Miller Modificado</td></tr>
<tr><td>Bit Rate</td></tr>
<tr><td rowspan="5">PICC → PCD</td><td>106 Kbit/s ($F_{Portadora}$/128)</td></tr>
<tr><td>Modulación</td><td>Frecuencia Subportadora</td></tr>
<tr><td>OOK</td><td rowspan="4">±847 KHz (Bandas laterales en 12,7125 MHz y 14,4075MHz)</td></tr>
<tr><td>Codificación</td></tr>
<tr><td>Manchester (E.G. Thomas)</td></tr>
<tr><td>Bit Rate</td></tr>
<tr><td>106 Kbit/s ($F_{Portadora}$/128)</td></tr>
</table>

Tabla 1.5: parámetros de la comunicación para ISO 14443-A

2.3.2 Inicialización de la comunicación

En este apartado se describe el formato de las tramas y los tiempos empleados durante la fase inicial de la comunicación entre el PICC y el PCD.

Inicialización desde el punto de vista del PCD

El PCD es el encargado de iniciar la comunicación. Se trata de una comunicación secuencial (Half-duplex) en la que el lector envía un comando y el transponder responde (Maestro-esclavo).

El tiempo que transcurre desde el último flanco del comando enviado por el PCD hasta el comienzo de la modulación que imprime el PICC en la respuesta de la subportadora en la portadora se define como FDT (Frame Delay Time).

El FDT puede variar según el valor del último bit transmitido y del número de bits transmitidos por el PCD. En la figura 1.16 se esquematiza la duración de dicho tiempo.

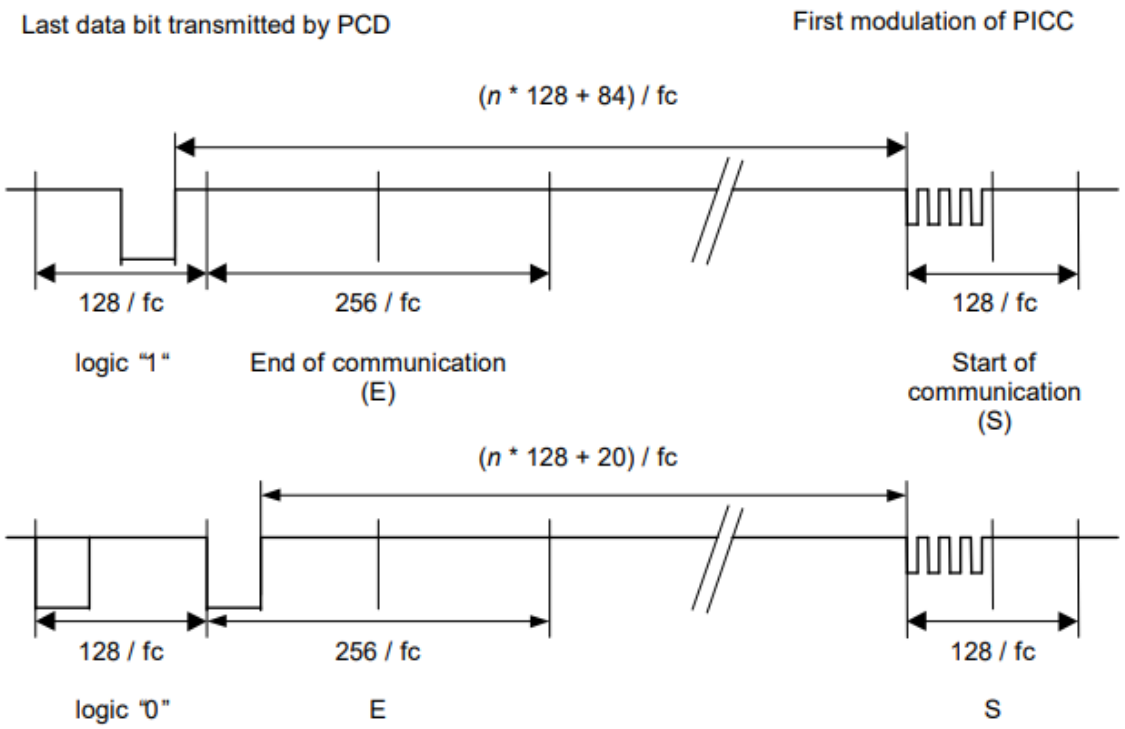

Figura 1.16: FDT de PICC a PCD

Los comandos que el PCD envía en la fase de anticolisión e inicialización son todos de 9 bit (Start + 7bit LSB + End), que es el a lo que hace alusión el parámetro n. Por tanto, se puede determinar que el FDT cuando el último bit enviado por el PCD es "0" tiene un valor de 86.43 us y cuando el último bit es "1" el FDT tiene un valor de 91.15 us.

El comando que el PCD envía para iniciar la comunicación es el ya mencionado carácter 0x26. A este comando se le denomina como REQA. Se envían los 7 bits menos significativos de REQA, empezando por el bit de menor peso. El formato de trama es el mostrado en la figura 1.17 y se codifica tal y como se expuso en el apartado 2.

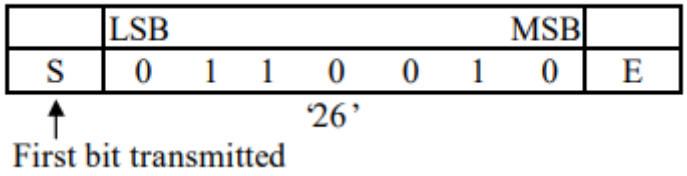

Figura 1.17: Formato de trama para REQA

Inicialización desde el punto de vista del PICC

También existe un FST referido al tiempo que discurre desde que el PICC emite una respuesta hasta el flanco inicial del siguiente comando del PCD. El PCD no puede emitir otro comando hacia el transponder hasta que no hayan transcurrido 1172 periodos de la

portadora, es decir; 83,43 us. Además, el PCD no debe de tener un límite superior en el tiempo de espera por una respuesta del PICC.

También es importante mencionar que como mínimo deben de transcurrir 0,516 ms (7000 ciclos de portadora) entre dos comandos REQA seguidos. A este tiempo se le denomina RGT (Request Guard Time).

Tras recibir el comando REQA, el PICC responde al lector con el denominado comando ATQA. Dicho comando tiene un formato de trama extendido cuya estructura es la que se ilustra en la figura 1.18. El PICC también emite el bit de menor peso primero. Cada byte de datos es seguido por un bit de paridad impar.

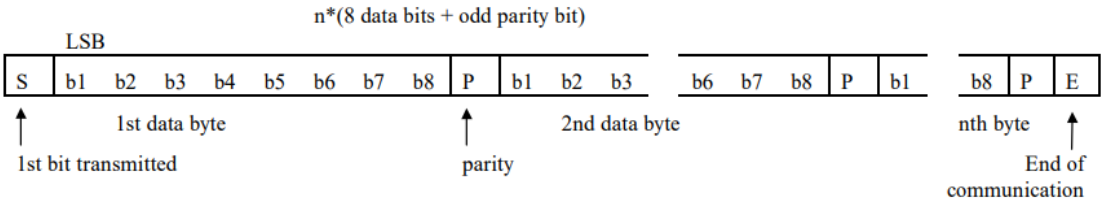

Figura 1.18: Trama estándar para la respuesta de un PICC

El contenido de la trama que conforma el comando ATQA es el expuesto en la figura 1.19. Los campos nombrados como RFU se refieren a “Reservado para Uso Futuro”. Los bits 8 y 7 indican al PCD el tamaño del UID. Cuando este campo contiene un valor de 0 (En valor decimal) el PICC indica que el tamaño de su UID es simple , es decir; 4 bytes. Si el valor es de 1 el tamaño es de 7 bytes y si el valor decimal es de 2 su tamaño será de 10 bytes. El valor 3 queda reservado para usos futuros. El campo “Bit frame anticollision” es el que emplea el sistema para detectar si se ha producido una colisión en la información por la presencia de varios PICCs en el radio de operación.

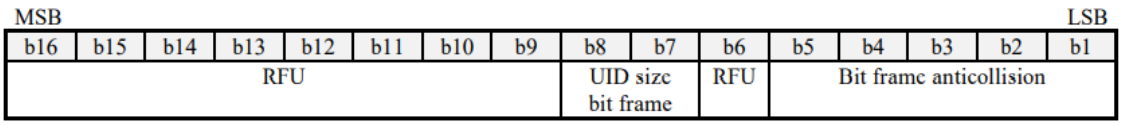

Figura 1.19: Contenido de la trama para ATQA

La respuesta ATQA consta de 2 bytes. Cada byte que forma parte de la respuesta del PICC esta seguido de un bit de paridad impar.

3. Introducción a la tecnología SDR

Una Radio Definida por Software (SDR por sus siglas en inglés) es un paradigma de diseño de sistemas de radio donde el principal objetivo radica en lograr la mayor independencia posible de los componentes hardware.

De este modo, se pretende reducir a la mínima expresión los componentes hardware típicos de las radios convencionales, empleando únicamente la electrónica necesaria para el acondicionamiento inicial de la señal y su digitalización.

La fuerza del paradigma radica en implementar el mezclado, generación de oscilador local, filtrado intermedio, demodulación, etc. mediante lógica combinacional y secuencial, lo que deriva indistintamente en manejar los parámetros clásicos para la recepción y transmisión de radiofrecuencia mediante software.

Joseph Mitola es considerado uno de los padres de esta idea a principios de la década de 1990, cuando impulsó la creación de la Radio Cognitiva [11]. Todo proceso cognitivo es capaz de percibir las condiciones del entorno para efectuar una planificación de modo que se pueda ejecutar decisiones de autoconfiguración para adaptarse dinámicamente a la demanda de recursos.

Si se aplica este concepto a los sistemas de enlaces de radiofrecuencia se obtienen sistemas en los que el emisor y receptor son capaces de decidir por sí mismos los valores de los parámetros propios a tener en cuenta en la comunicación como puede ser la frecuencia de trabajo logrando una mejora sustancial en la eficiencia en, por ejemplo, lo que se refiere al ancho de banda; repercutiendo en última instancia en un uso mucho más eficiente del espectro.

Toda esta capacidad autónoma de decisión no sería posible si los sistemas de radiofrecuencia estuviesen totalmente implementados vía hardware, es decir; la capacidad cognitiva de los SDR es únicamente posible gracias a la implementación por software de los parámetros típicos de la comunicación.

3.1 Arquitectura

Con ánimo de comprender el sentido de una arquitectura típica de un sistema SDR [12] en primer lugar se va a presentar una arquitectura tradicional de los receptores de radiofrecuencia hardware para, mediante las diferencias y similitudes entre ambos, lograr un mayor entendimiento del mismo.

3.1.1 Receptor tradicional

El receptor tradicional expuesto está basado en una arquitectura heterodina. El término heterodino proviene de la raíz griega "hetero" que significa diferente y de la raíz "dino" que hace alusión al termino potencia. Este origen etimológico permite comprender mejor el funcionamiento de la arquitectura puesto que lo que se hace en estos sistemas heterodinos es desplazar la frecuencia inicial de entrada sintonizada a otras frecuencias diferentes a la original, de orden más bajo.

Antes de realizar el desplazamiento de las frecuencias de entrada, estas señales deben acondicionarse puesto que llegan al receptor con una potencia baja y ruido espurio. El acondicionamiento suele consistir en un amplificador de bajo ruido (LNA, Low Noise Amplifier) y un filtro RF. Sin entrar en grandes detalles es importante que la primera etapa de la recepción tenga una figura de ruido [13] lo más baja posible puesto que esta condiciona el ruido del resto del sistema.

La siguiente etapa consiste en un mezclador. El mezclador tiene la tarea de desplazar las frecuencias de entrada a otra más baja, que será siempre la misma. Esto es posible gracias a la multiplicación de la señal de entrada (Analíticamente será un coseno de una determinada amplitud y frecuencia) con una señal oscilante local (Analíticamente también tendrá forma de coseno). Haciendo uso de las propiedades de las razones trigonométricas en la ecuación 2.1 se expone el resultado de multiplicar las señales mencionadas.

$$A_{in}\cos(f_{in} * t) * A_{OL}\cos(f_{OL} * t) = \frac{A_{in}A_{OL}}{2}(\cos((f_{in}+f_{OL}) * t) + \cos((f_{in}-f_{OL}) * t)) \quad (2.1)$$

Podemos observar que lo que se obtiene son dos nuevas señales, una de ellas con una frecuencia que es la suma de las frecuencias de la señal de entrada más la oscilación local y otra señal cuya frecuencia es la resta de las frecuencias mencionadas. La señal de interés será una de las dos, normalmente la de frecuencia más baja para facilitar el procesado posterior.

Para únicamente obtener la señal de frecuencia más baja, habrá que filtrar la salida del mezclador con un filtro pasa banda denominado de frecuencia intermedia. Este filtro siempre es fijo y es el que nos permite trabajar a partir de él siempre a la misma frecuencia sea cual sea la frecuencia de la señal inicial. Para sintonizar las distintas frecuencias que lleguen al receptor lo que se varía es la frecuencia del oscilador local.

El último elemento básico de un receptor tradicional es el demodulador cuya idiosincrasia depende de si la señal recibida tiene una modulación en amplitud (Empleando para la demodulación un detector de envolvente, por ejemplo) o en fase/frecuencia (Implementando un discriminador de frecuencias).

En la figura 2.1 pueden observarse los elementos que se ha descrito de forma muy básica a modo de contextualización de un receptor heterodino típico.

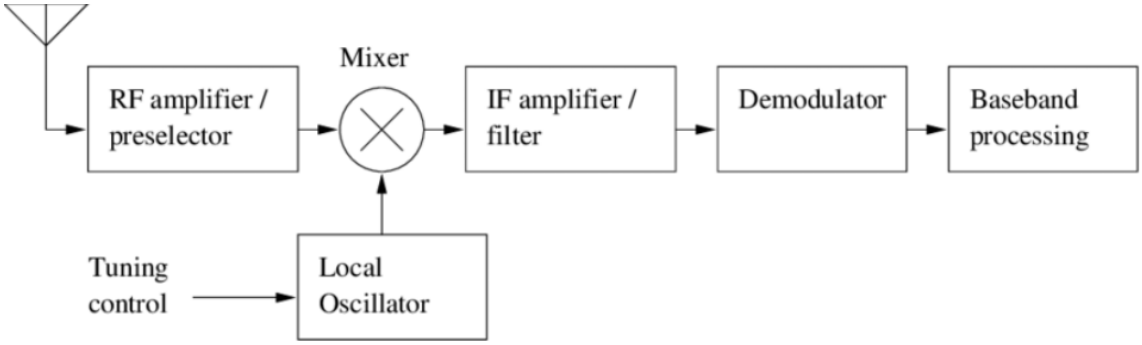

Figura 2.1:Arquitectura heterodina de receptor hardware tradicional

3.1.2 Receptor SDR

Las partes que conforman un receptor SDR son esencialmente las mismas en cuanto a su funcionalidad. Tal y como ocurre con los receptores tradicionales, la etapa inicial en la que la señal es acondicionada para aumentar su ganancia y limpiar el ruido de entrada siempre debe implementarse mediante hardware analógico dedicado, además de la necesaria adaptación de la señal a los requisitos de entrada del conversor analógico-digital en lo que se refiere al rango de entrada.

En la figura 2.2 aparecen representadas las distintas etapas del receptor. El ADC proporciona las muestras digitales correspondientes a su salida. A partir de este punto todo el procesamiento se realiza mediante puertas lógicas gobernadas por software. Las muestras del ADC se envían a un DDC (Digital Down Converter) donde se encuentran implementados el mezclador, el oscilador local y el filtro de frecuencia intermedia que en este caso es un filtro pasa bajos que deja concurrir la señal trasladada a banda base.

Las operaciones que realiza el DDC son similares a las realizadas por los componentes equivalentes del receptor analógico heterodino. El mezclador digital junto con el oscilador local desplaza las muestras digitales a la banda base, es decir, en torno a los 0Hz. El filtro digital pasa-bajos suele ser de tipo FIR (Finite Impulse Response) aunque esto no es una norma. Para implementar todas estas partes digitales el DDC utiliza un gran número de contadores, acumuladores, multiplicadores y registros de desplazamiento, entre otros.

Las señales de entrada se desplazan a su equivalente en banda base gracias a su descomposición en las componentes de fase y cuadratura mediante un detector Tayloe [14] que implementa el paradigma heterodino.

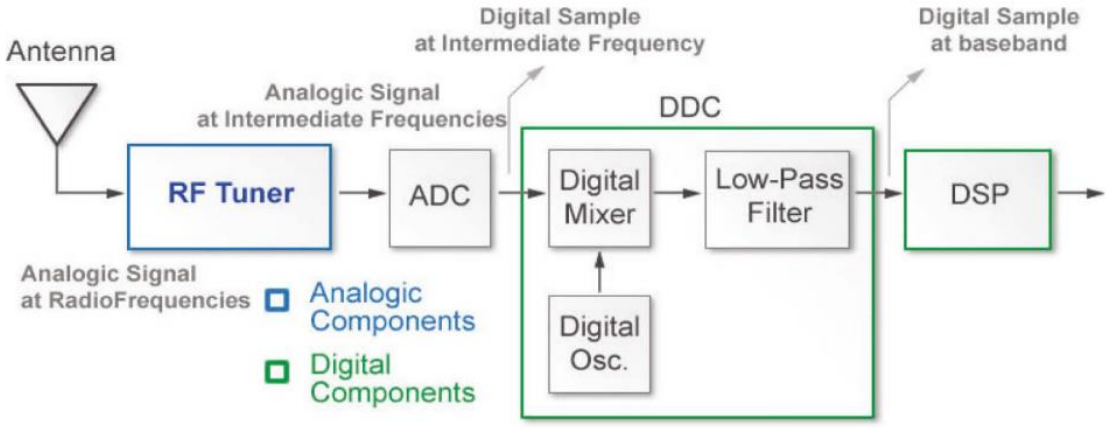

Figura 2.2: Etapas de un receptor SDR

También se lleva a cabo otro proceso conocido como decimación o diezmado. El objetivo del diezmado es reducir la frecuencia de muestreo. La nueva frecuencia de muestreo en la banda base es fruto de la división de la tasa de muestreo inicial entre un factor N, de modo que cada N muestras solo se coge una de ellas. Con el método de diezmado la tasa de muestreo final puede llegar a ser igual que el doble del ancho de banda de interés tal y como dicta el teorema de Nyquist [15].

La etapa final de la recepción consiste en obtener la información de la señal, es decir, en aplicar los procesos de demodulación y decodificación. Estas tareas suelen recaer en un DSP (Digital Signal Processing) que puede formar parte del propio dispositivo SDR o sin embargo estar implementado en un sistema de propósito general como un PC, en cuyo caso el sistema SDR sería distribuido.

3.1.3 Transmisor SDR

La secuencia que siguen los transmisores SDR con la señal es similar a la de los receptores, pero invirtiendo el sentido. El DSP provee la señal banda base al bloque DUC (Digital Up Converter) que la envía hacia un conversor digital-analógico .

Dentro del DUC hay un interpolador que realiza la operación inversa al diezmado comentado, proporcionando más muestras y por tanto aumentando la tasa de muestreo. El oscilador local en conjunción con un mezclador digital desplaza la señal banda base hacia una frecuencia superior.

El DAC entrega la señal ya analógica hacia la circuitería de radiofrecuencia que la acondiciona para ser transmitida con la potencia necesaria proporcionada por el amplificador hacia la antena. En la figura 2.3 se esquematiza las etapas del transmisor SDR.

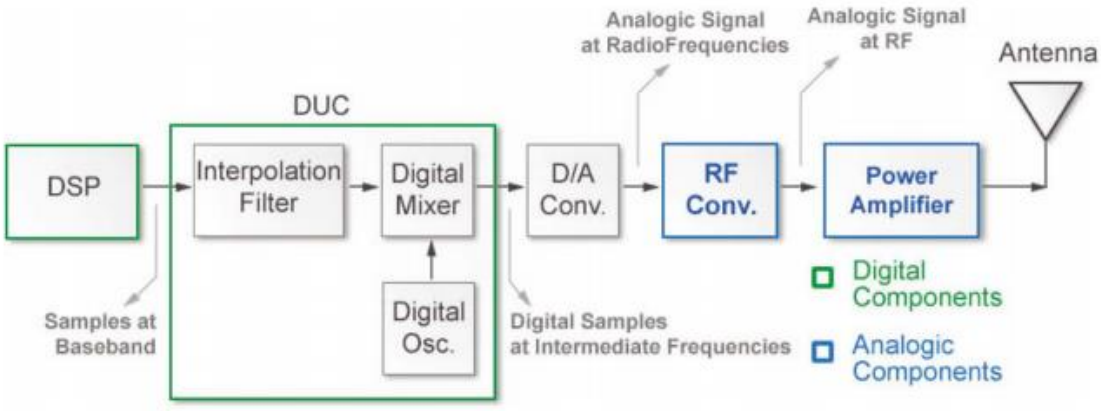

Figura 2.3: Etapas de un transmisor SDR

4. Vista general del sistema

Una vez se ha expuesto en los apartados anteriores los fundamentos que rigen a las tecnologías que se pretenden implementar en el proyecto en este apartado se proporciona una vista general de las características técnicas del sistema.

Tal y como se ha expuesto en el apartado 2.3 el sistema diseñado consiste en un lector RFID para los dispositivos regidos por la norma ISO 14443-A. Aunque se ha implementado un PCD para este estándar, el lector está preparado para soportar la lectura de cualquier transponder RFID que trabaje en la frecuencia de operación de los 13,56MHz, como las vicinity cards, los transponders NFC o el estándar FeliCa. Esto es posible gracias a la versatilidad de la electrónica analógica que conforma la etapa denominada como AFE (Analog Front End) en el sentido de que si el objetivo del trabajo fuese únicamente dar soporte a las proximity cards no se necesitaría, entre otras prestaciones, un DAC capaz de generar cualquier tipo de forma de onda, modulación y/o frecuencia de trabajo.

Por otro lado, la parte digital del sistema SDR debe de ser capaz de efectuar un proceso heterodino implementando un detector de cuadratura que desgrane la señal de entrada en su parte real y en su parte imaginaria. La señal que se desplazará a banda base será una de las bandas laterales en las que el PICC imprime su información, en concreto la banda lateral superior centrada en 14,4075 MHz mostrada en la figura 1.14. Para el filtrado se ha optado por un CIC (Cascaded Integrator Comb filter) que además de actuar como filtro pasabajos también realiza un diezmado que reduce la tasa de muestreo original de 4 veces la frecuencia de la banda lateral superior. Las etapas de desmodulación y decodificación entregan los datos al módulo UART de transmisión que las envía a un PC donde se implementarían las capas de aplicación.

La parte de transmisión SDR sigue un proceso similar pero en sentido inverso, generando la señal portadora para que, mediante el bloque modulador, aplicar la codificación correspondiente que exige el estándar y entregar los datos digitales al AFE para que este los transmita con su antena al PICC. En la figura 3.1 aparecen representados los bloques que conforman el sistema desarrollado en este trabajo.

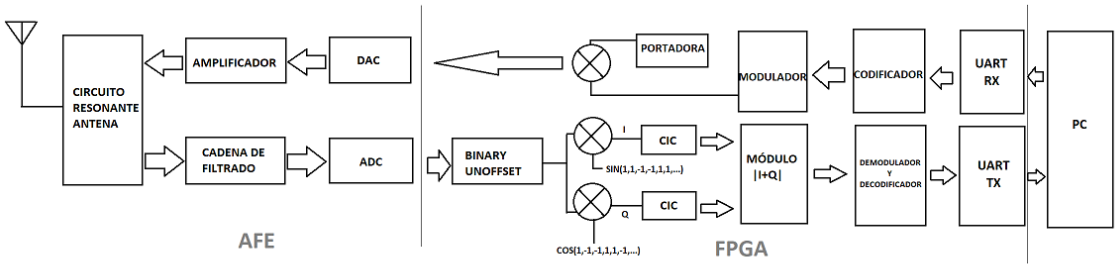

Figura 3.1: Diagrama de bloques del sistema desarrollado

Analog Front End

El circuito resonante de la antena no es mas que un filtro pasa banda conformado por un circuito RLC. En el siguiente apartado 5 se tratará con más detalle su implementación y el diseño de la antena, uno de los aspectos más críticos del sistema. En la figura 3.2 [16] se representa la impedancia típica de estos circuitos respecto de la frecuencia.

Aunque el PCD emite su portadora a 13.56 MHz, en la gráfica presentada la antena está sintonizada entorno a los 14,4075 MHz puesto que es donde se sitúa una de las bandas que contiene la información que nos interesa recibir del PICC.

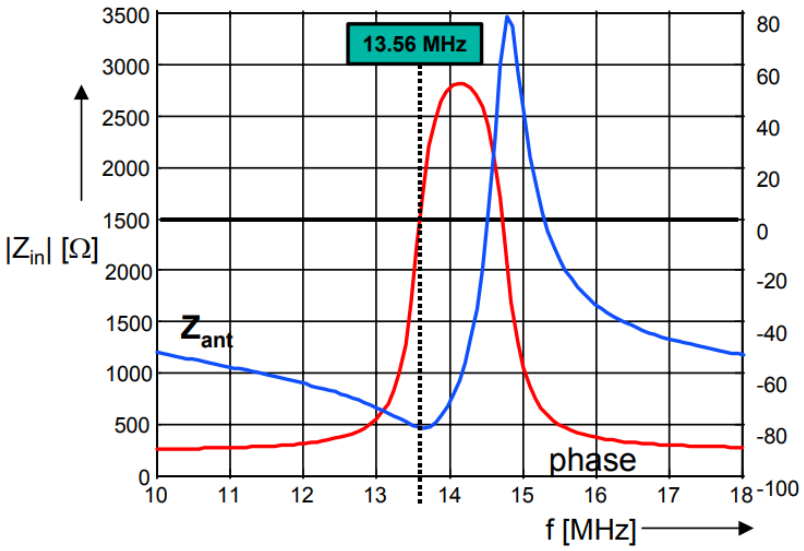

Figura 3.2: Impedancia de la antena frente a la frecuencia

La etapa amplificadora se encarga de dotar a la pequeña señal que entrega el DAC de la ganancia suficiente para cumplir con la intensidad del campo magnético que requiere el estándar. Dicho campo debe ser superior a 1.5 A/m rms pero nunca superior a 7.5 A/m rms.

Para la recepción, el ADC debe de tener suficiente resolución como para percibir la respuesta del PICC que, como se ha comentado en apartados anteriores, será mucho más débil que el nivel de la portadora. Además, tanto el DAC como el ADC deben de poder operar a una frecuencia mayor que la de la portadora, para la generación y muestreo de la misma. En concreto en este sistema se empleará una frecuencia de muestreo para la generación de la portadora 4 veces mayor. Para muestrear la señal de entrada, se empleará una frecuencia 4 veces mayor a la de la banda lateral superior, que es la que se procederá a sintonizar.

La señal recibida se acondiciona para entregársela al ADC mediante un filtrado paso bajos que también hace las veces de filtro antialiasing. Como la señal total recibida tiene demasiada amplitud, en el filtrado también se incluye una atenuación de la misma para cumplir con el rango de entrada del ADC.

Radio definida por software en la FPGA

Uno de los dispositivos más convenientes para implementar el procesamiento digital de la señal es una FPGA. En este proyecto se ha utilizado el modelo Lattice ICE40HX4K, en concreto el modelo con el encapsulado TQ144, que cuenta con herramientas de desarrollo libres enmarcadas en el proyecto ICESTORM. La FPGA utilizada está embebida en la placa de desarrollo ICECREAM [17] confeccionada por Jesús Arias, tutor de este trabajo. La placa ICECREAM permite utilizar casi todos los pines de la FPGA a nuestro antojo. La implementación de la lógica digital en la FPGA se ha realizado mediante el lenguaje de descripción hardware Verilog.

En lo referido a la recepción, el primer bloque "Binary Unoffset" se encarga de eliminar el nivel de continua de la señal digital entregada por el ADC. De este modo los valores digitales de la señal quedan representados en notación con signo, en concreto en notación complemento a dos.

Para una óptima obtención de la señan banda base original hace falta desgranarla en su parte real y en su parte imaginaria. Aunque en este caso concreto solo se necesita obtener la amplitud, es posible que en futuras implementaciones también se desee detectar la fase. Esta versatilidad solo es posible con un detector de cuadratura. Se puede representar la señal de entrada como un fasor tal y como muestra la figura 3.3.

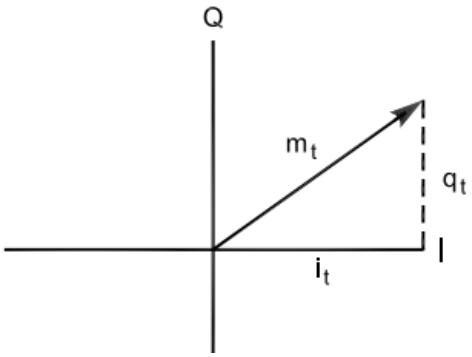

Figura 3.3: Señal de entrada en su forma I+jQ representada en el plano complejo

Para demodular las señales AM simplemente habría que calcular el módulo del fasor, cuyo valor se obtiene como dicta la ecuación 3.1.

$$m_t = \sqrt{I_t^2 + Q_t^2} \tag{3.1}$$

Por otro lado, si se desease demodular señales FM habria que aplicar la ecuación 3.2. para obtener la fase instantánea.

$$\emptyset_t = tan^{-1}\left(\frac{Q_t}{I_t}\right) \tag{3.2}$$

Para separar la señal de entrada en sus componentes I y Q se le hace pasar por dos canales paralelos de mezclado. El oscilador local genera dos sinusoides, desfasadas entre si 90º obteniéndose un seno y un coseno de la misma frecuencia y amplitud. Para la generación de la oscilación local suele emplearse la técnica DDS (Direct Digital Synthesis). Sin embargo, en este sistema se aprovecha el hecho de muestrear a 4 veces la frecuencia de la señal de interés ya que en este caso la generación de las muestras del seno y coseno locales se limita a la generación de secuencias de valores 1 y -1. Las tareas de mezclado se simplifican enormemente puesto que simplemente se cambiará el signo de las muestras de entrada que coincidan con los valores -1 de la oscilación local, prescindiendo de las multiplicaciones.

El filtrado se implementa mediante el mencionado CIC. Este tipo de filtros fueron inicialmente desarrollados por Eugene B. Hogenauer y aparte de incluir un filtro COMB también realiza tareas de diezmado, reduciendo la tasa de muestreo. Esto es efectivo siempre y cuando la nueva tasa de muestreo siga cumpliendo con el criterio de Nyquist. Una menor tasa de muestreo es de gran ayuda para reducir los requisitos computacionales de las etapas posteriores. A los filtros COMB también se les denominan peine por su apariencia, como se mostrará a continuación.

Un filtro peine (De primer orden) es la media dinámica de las últimas N muestras, siendo N el factor de diezmado. Otra forma de interpretarlo consiste en plantear un acumulador al que sumamos la muestra actual y restamos la muestra de N ciclos antes, tal y como muestra la ecuación 3.3:

$$y_i = y_{i-1} + x_i - x_{i-N} \tag{3.3}$$

Los filtros CIC, concordando con su naturaleza de filtro FIR, ofrecen una respuesta de fase lineal y una de sus mayores ventajas es que no requiere de multiplicaciones. Tal y como se aprecia en la figura 3.4 se pueden implementar con , según el orden del filtro, etapas integradoras seguidas de bloques que hacen las veces de derivador. En este caso el filtro es de orden 3.

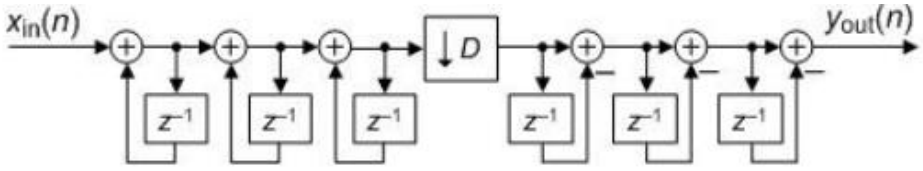

Figura 3.4: Filtro CIC de orden 3

La función de transferencia en términos generales la marca la ecuación 3.4., donde N se refiere al factor de decimación y L al número de etapas del filtro (Y por tanto a su orden).

$$\mathrm{H(z)} = \left(\frac{1 - z^{-N}}{1 - z^{-1}}\right)^{L} \tag{3.4}$$

El integrador es un acumulador al que se le suma la entrada y por tanto se acabará desbordando tarde o temprano. Su correcto funcionamiento requiere de este desborde y no debe de aplicarse una aritmética con saturación. La diferencia de las muestras que se integran será la correcta, aunque se desborde, siempre que este desbordamiento sea menor que el máximo valor con signo representable. Si el acumulador se saturase al valor máximo, la diferencia sería de cero [18].

En la figura 3.5. se muestra una primera aproximación al filtro CIC implementado en el sistema. La señal de color azul se corresponde con un filtro COMB de primer orden. Como la banda de paso de este tipo de filtros no es plana se les suele aplicar una segunda etapa de compensación con un filtro FIR (La señal naranja) cuya forma sea inversa a la caída del COMB. De esta manera se consigue una respuesta plana en la banda de paso y se atenúan mucho más las frecuencias centrales de cada lóbulo. El filtro final es el que aparece representado en color amarillo. En la figura 3.6 se representa únicamente la apariencia final del filtrado para una mayor comprensión. El cálculo de estos filtros se ha efectuado mediante MatLab.

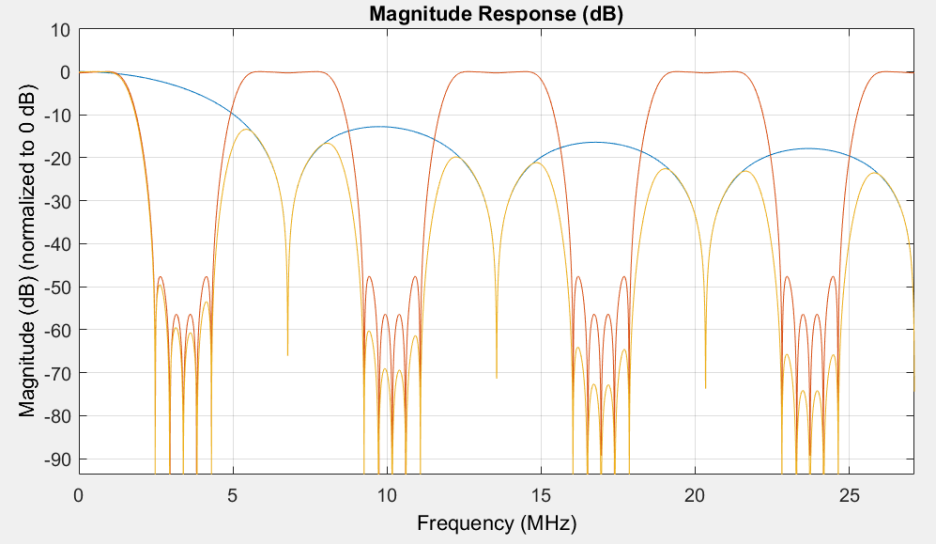

Figura 3.5: Filtro Comb, filtro FIR compensador y filtrado final

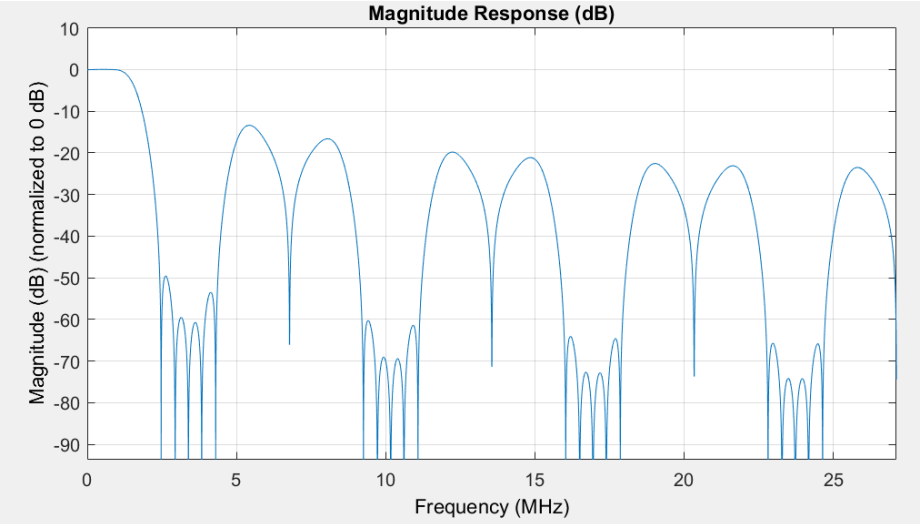

Figura 3.6: Detalle del filtrado final, combinación del COMB y FIR compensador

En el apartado 6 se tratarán los detalles concretos del orden, frecuencia de corte, ganancia, ancho de los registros, etc. del filtro, así como su implementación en la FPGA mediante Verilog.

La información que devuelve el PICC, en banda base tras el filtrado, sigue la codificación Manchester modulada mediante OOK. Hay que recordar que la tasa de bits es 106 Kbit/s (128 periodos de portadora). Por tanto, la modulación estará presente durante medio periodo del tiempo de bit. Una vez que se ha obtenido el módulo (La amplitud) de la señal a partir de su expresión en fase y en cuadratura, hay que fijar un nivel umbral para determinar los niveles lógicos de la señal, esto es, determinar cuando está en alta y cuando esta en baja. Sin embargo, no sería valido aplicar una técnica en la que el umbral se fija de forma estática puesto que la señal , además de provenir con un cierto nivel de ruido, se recibirá con diferentes amplitudes según la distancia de lectura.

Por tanto, para poder aplicar la decodificación Manchester, se diseña una técnica dinámica para determinar el umbral que nos dice cuándo ha tenido lugar una transición lógica. Una vez la información es decodificada se transmitirá vía UART hacia el PC que se encargaría de aplicar el protocolo de alto nivel. El PC envía los comandos que debe de transmitir el PCD mediante la UART. Estos son enviados al codificador Miller. El modulador aplicará la codificación a la portadora mediante una modulación ASK del 100% que finalmente se entregará al DAC para que los emita por la antena.

En los capítulos 5 y 6 se muestra en detalle cómo se ha implementado cada uno de los elementos que conforman el diagrama de bloques del sistema mostrado en la figura 3.1, tanto en términos de circuitos integrados, cálculos para el diseño de la antena , elementos de filtrado, etc. como en los términos de la electrónica digital empleada para implementar el tratamiento digital de la señal en la FPGA.

5.Analog front end

En el presente capítulo se expone con detalle los componentes utilizados así como los cálculos necesarios para implementar la etapa analógica necesaria para la transmisión y recepción de las señales que emplea el PCD para efectuar la comunicación con el transponder. Se presentará la captura esquemática del circuito y las pautas para realizar la placa de circuito impreso. El diagrama de bloques que conforma el AFE del sistema aparece expuesto en la figura 3.1. La PCB desarrollada se acoplará a modo de “shield” sobre la placa de desarrollo ICECREAM que embebe a la FPGA.

5.1 Búsqueda y selección de componentes

Para la elección de los componentes se ha buscado un compromiso entre las máximas prestaciones que estos pueden ofrecer dentro del precio más asequible posible.

5.1.1 Convertidor Digital Analógico (DAC)

Los dos requisitos más importantes que debe de cumplir el DAC son la tasa de muestreo y la interfaz paralelo. Puesto que la señal que se va a generar es principalmente una portadora de 13,56 MHz se requiere de una tasa de muestreo que como mínimo sea del doble para cumplir con el criterio de Nyquist. Aun así, emplear una tasa de muestreo para la generación de la portadora que sea del doble de su frecuencia no es buena idea puesto que estaríamos trabajando en el límite. La tasa de muestreo elegida es 4 veces la frecuencia de la portadora, es decir; necesitamos un DAC que pueda generar como mínimo 54,24 millones de muestras por segundo (54,24 MHz).

Por otro lado, se requiere de la mayor rapidez posible para transmitirle las muestras digitales desde la FPGA. Por ello, se desestimó la idea de emplear dispositivos con una interfaz serie del estilo SPI o I2C. La mejor opción es emplear un interfaz paralelo.

En última instancia, para realizar la búsqueda entre la ingente cantidad de dispositivos de este tipo del mercado, se impuso la condición de que tuviese 10 bits de resolución. El DAC elegido para el sistema es el AD9740 [19] de Analog Devices.

Para esta aplicación bastaría con un DAC de 2 o 3 niveles, pero esto no sería así si la frecuencia de muestreo no fuese exactamente 4 veces la frecuencia de la portadora. Además un DAC de estas características permite la implementación de cualquier protocolo RFID basado en la banda HF.

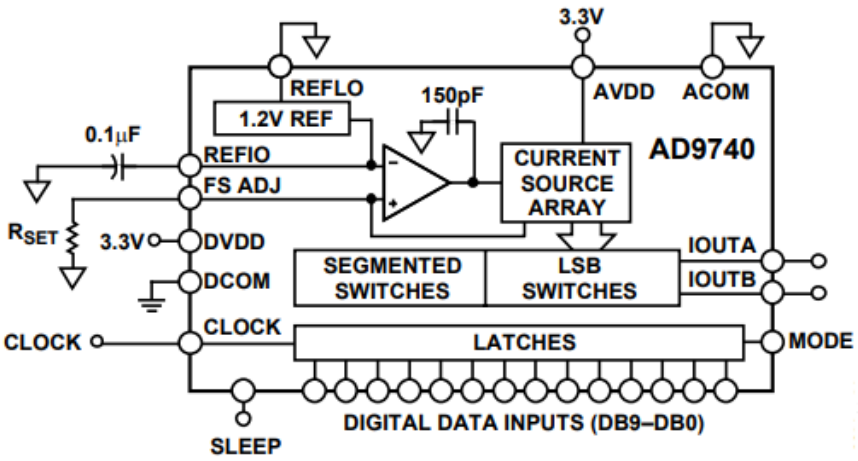

Figura 4.1:Bloques funcionales del DAC AD9740

En la figura 4.1 se muestra el diagrama de bloques del dispositivo. Las principales características de este DAC son las siguientes: Tasa de muestreo de hasta 210 Msps, interfaz paralela de 10 bit, alimentación single-ended de 3.3V, salida diferencial, control de corriente de salida, selección de modo sleep, selección de modo binario natural o complemento a dos para los bits de entrada.

Los mejores valores de distorsión para la señal de salida se obtienen cuando la señal como máximo tiene un voltaje pico a pico de 0,5 V. Teniendo en cuenta que se ha configurado el integrado para que cada una de las salidas diferenciales (Desplazadas entre si 180º) proporcione 20 mA (Fondo de escala) para obtener como máximo los 0.5 Vpp en la salida hay que conectar una carga de entorno 25 Ohms en cada una.

5.1.2 Amplificador diferencial

Para amplificar la salida del DAC hay que emplear algún dispositivo que sea capaz de imprimir cierta ganancia a la señal. Se ha optado por utilizar lo que se denomina FDA (Fully Differential Amplifier). Este tipo de amplificadores tienen tanto una entrada como una salida diferencial, aunque se pueden acondicionar para que tengan una configuración single-ended en cualquiera de las dos. La labor de esta etapa será la de amplificar la señal de entrada de 1Vpp diferencial hasta los 5Vpp diferenciales (2,5 Vpp en cada salida). Es decir, la ganancia deseada del amplificador es de 5. También otra de las tareas que se pretende implementar en esta etapa es la del filtrado de altas frecuencias, a pesar de que la etapa posterior de la antena es un filtro en si mismo.

El FDA escogido es el THS4521 de Texas Instruments [20]. Este dispositivo admite una alimentación de 3.3V (Se pretende unificar los voltajes de alimentación de todos los integrados para evitar utilizar distintos reguladores de tensión) y un ancho de banda de 22 MHz para una ganancia de 5 (Para una carga de 1 kOhm diferencial y una resistencia de realimentación también de 1 KOhm) tal y como se aprecia en la figura 4.2

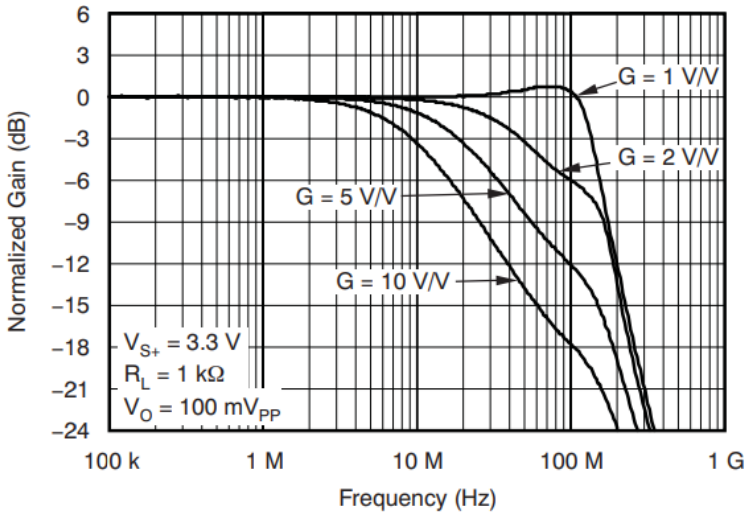

Figura 4.2:Diagrama de Bode del amplificador THS4521 alimentado a 3.3 V

La configuración del FDA se muestra en la figura 4.3. La impedancia de salida esta adaptada a la impedancia de la antena. Se recomienda que la impedancia de entrada la antena (700 Ohms diferenciales) sea ligeramente superior que la impedancia de salida de la etapa anterior, es por ello por lo que se ha establecido una impedancia de salida del FDA de 440 Ohms diferenciales. (Siguiendo los valores estándar para las resistencias de la serie E12 [21]). Para establecer una ganancia lo más cercano a 5 con los valores de la serie E12 la resistencia R_F adquiere un valor de 1.2 Kohm y la resistencia R_G de 220 Ohm.

Los condensadores C_{in} eliminan el nivel de continua que llega del DAC con un valor de 0.1 uF. Para limitar la amplitud de las posibles frecuencias alias que le lleguen a la antena se incluye un filtrado extra mediante C_F. Con un valor de 8,2 pF junto con la resistencia R_F , se forma un polo en 16 MHz que en consonancia con el propio ancho de banda del amplificador debería de atenuar notablemente las frecuencias entorno a los 54MHz de muestreo.

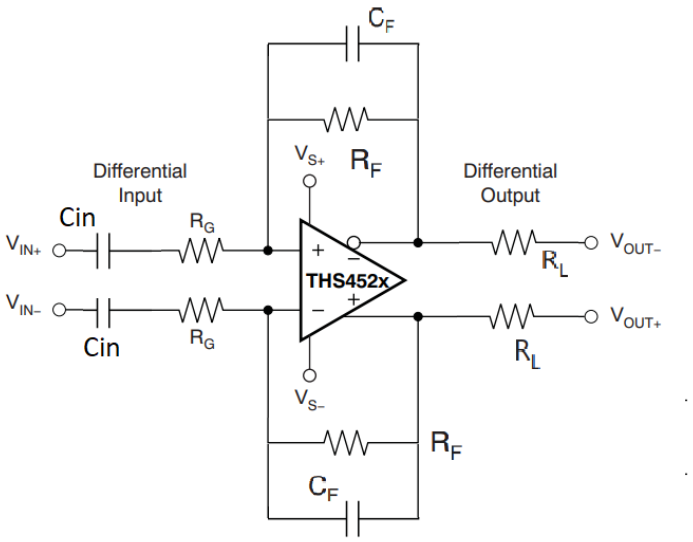

Figura 4.3: Configuración del FDA

5.1.3 Convertidor Analógico Digital (ADC)

Las prestaciones con las que debe cumplir del ADC son muy similares a las del DAC pero en sentido inverso. Debe de soportar una tasa de muestreo 4 veces superior a frecuencia donde esta centrada la banda superior de la subportadora que contiene la información del PICC (14,4075 MHz). Esto es, debe de soportar una tasa de muestreo de por lo menos 57,63 Msps. Además su interfaz debe de ser lo suficientemente rápida como para entregar las muestras a tiempo entre cada tiempo de adquisición, por ello también debe de contar con una interfaz paralelo.

El aspecto de la resolución es muy importante puesto que el dispositivo debe de ser capaz de detectar las pequeñas variaciones en la señal que provoca el PICC en su respuesta. Para un ADC de 10 bits y un voltaje de referencia normalizado a 1, la resolución viene dada por la ecuación 4.1. Hay que tener siempre en cuenta que es muy difícil aprovechar todo el rango de entrada del ADC al 100 % y que esto supone ciertas perdidas en la resolución.

$$\text{Resolución} = \frac{1}{2^{10} - 1} = 0{,}98\ mV \tag{4.1}$$

El convertidor analógico digital seleccionado se trata del ADC10065 de Texas Instruments [22]. Este integrado admite la alimentación pretendida de 3.3 V, tasa de muestreo de hasta 65 Msps, modo de standby, entrada diferencial, selección entre datos en binario natural o en complemento a dos, interfaz paralela y una característica muy interesante: la elección del rango de entrada entre 1 Vpp, 1.5 Vpp y 2 Vpp. Esta característica es muy útil y versátil puesto que los niveles de voltaje de la señal a recuperar que entrega el PICC siempre estarán sujetos a una cierta incertidumbre. El control del rango de entrada se efectuará tanto desde la propia FPGA como desde unos jumpers externos, manualmente.

En la figura 4.4 se representan los bloques funcionales del ADC.

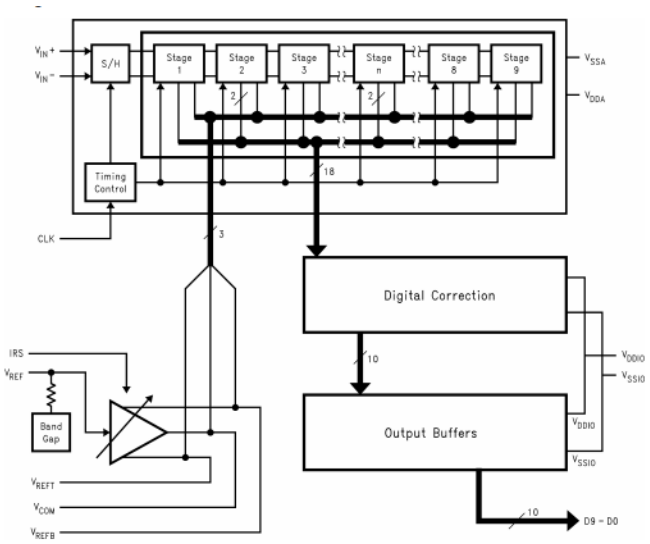

Figura 4.4: Bloques funcionales del ADC10065

5.1.4 Línea de filtrado en la recepción

Aunque el ADC seleccionado brinda la posibilidad de utilizar una entrada diferencial, la línea de recepción proveniente de la antena será single-ended. Además de filtrar la entrada, la línea de recepción incluye una atenuación puesto que la señal entregada a la antena por el amplificador también va directamente al ADC. La recepción parte de una de las ramas simétricas del circuito balanceado que conforma la antena, donde habrá una tensión entorno a los 2,5 Vpp.

El rango máximo del ADC es de 2 Vpp por lo que se le aplica una atenuación del 50% mediante un divisor de voltaje capacitivo. La relación entre los valores de los condensadores del divisor es de 1:1. Si se necesitase una atenuación distinta al 50% bastaría con cambiar la relación entre los valores de los condensadores que conforman el divisor, aparte de poder elegir el rango de entrada según nos convenga. En la figura 4.5 se puede apreciar el circuito de la línea de filtrado.

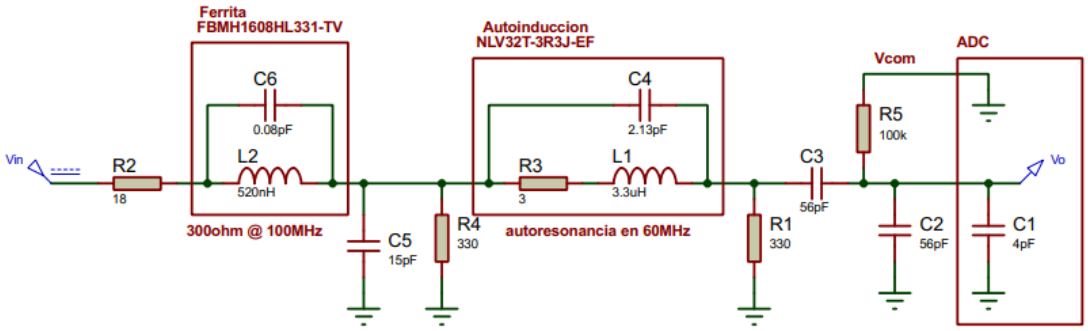

Figura 4.5:Cadena de filtrado y atenuación para la entrada del ADC

Los condensadores C2 y C3 son los que forman el divisor capacitivo. El nivel de continua que requiere el ADC de entorno a los 1.45 V los proporciona la propia salida VCOM del integrado mediante la resistencia R5. Aunque esta salida no debe cargarse el valor de la resistencia es lo suficientemente elevado como para establecer el nivel de continua en la señal de entrada sin que fluya corriente a través de ella. Este nivel DC es necesario para adaptar la señal al rango de entrada. La respuesta en frecuencia simulada de la cadena de filtrado puede observarse en la figura 4.6. Este filtrado limita la amplitud de las frecuencias alias que tienen lugar por el muestreo.

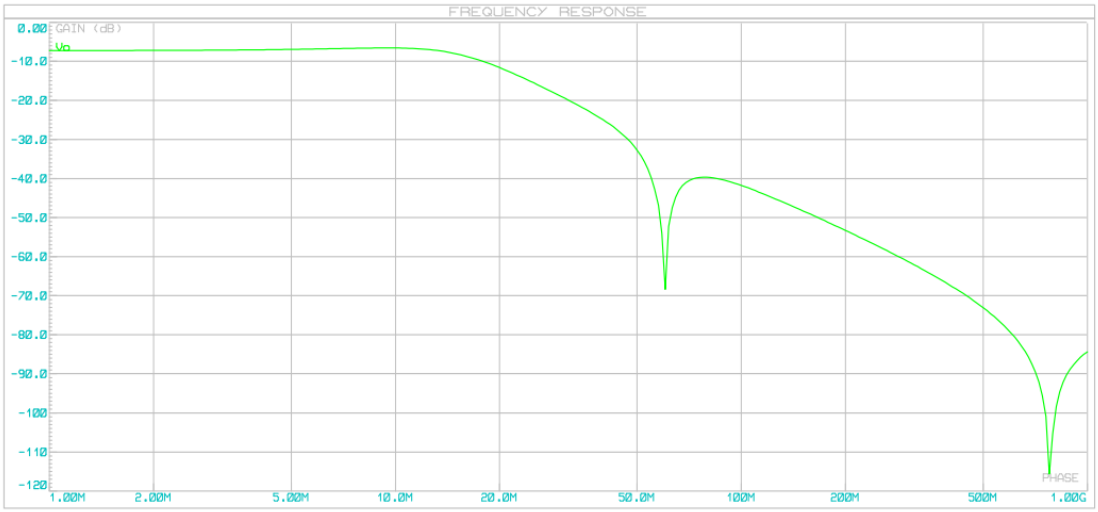

Figura 4.6: Respuesta en frecuencia de la cadena de filtrado

5.1.5 Generación del reloj

La frecuencia de la portadora debe ser de 13,56 MHz con un margen de ±7 KHz, según el estándar ISO 14443. Para cumplir con dicho requisito se ha optado por incluir un cristal de cuarzo a partir del cual obtener las distintas señales del reloj necesarias para la generación de la portadora y muestreo de la señal de entrada.

Al no existir en el mercado un cristal con una frecuencia exacta de 13,56 MHz se ha buscado un cristal cuya frecuencia sea un múltiplo exacto de la portadora. El cristal que mejor se adapta a nuestras pretensiones es de 27,12 MHz, justo el doble de la frecuencia de interés. Estos cristales son mucho más fáciles de encontrar por su utilización en aplicaciones de radiofrecuencia referidas a la CB (Banda ciudadana), muy extendida entre radioaficionados. La FPGA internamente generará las frecuencias de reloj necesarias (Recordemos que son 54,24 MHz para la generación de la portadora, 4 veces su frecuencia, y 57,63 MHz para el muestreo de la banda superior que contiene la subportadora) mediante los dos PLL que incluye. Para generar frecuencias de reloj más bajas basta con aplicar un divisor de frecuencias.

Sin embargo, la utilización de un cristal de cuarzo requiere de un circuito de acondicionamiento que es el que genera realmente la oscilación que conforma la señal del reloj. Para esta tarea se ha elegido el circuito integrado SN74LVC1404 [23] de Texas Instruments. El cristal tiene una capacidad de carga de 10 pF y una estabilidad de 50 ppm. El funcionamiento del integrado puede apreciarse en la figura 4.6.

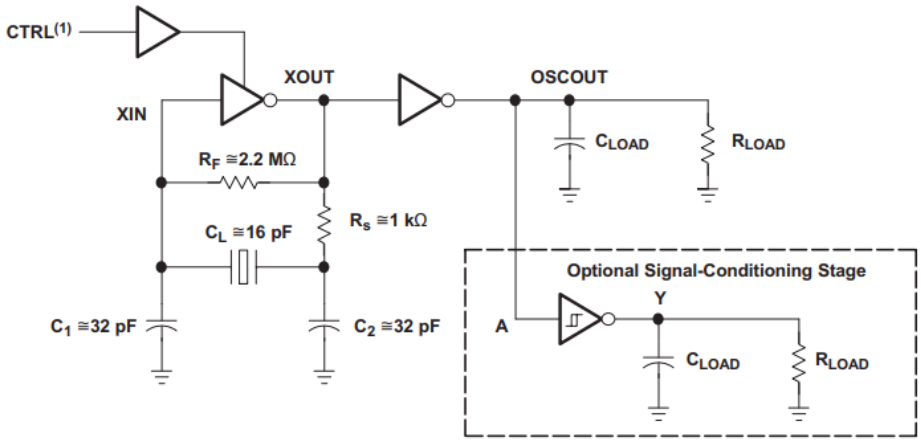

Figura 4.6: Bloques funcionales del controlador SN74LVC1404

El funcionamiento del oscilador se basa en el denominado Pierce Oscillator. La resistencia R_F actúa como resistencia de realimentación y su tarea es polarizar el inversor en su región lineal, forzando a los voltajes de entrada y salida a ser iguales. El inversor trabajara en la región de transición que es donde tiene ganancia.

El cristal, junto con C1 y C2 que tienen el mismo valor igual al doble de la capacidad de carga del cristal, forma un filtro pasabanda que proporciona un desfase de 180º y una ganancia desde la salida a la entrada a la frecuencia de resonancia. Esta característica junto con la ganancia negativa del inversor provoca una realimentación positiva que da lugar a una oscilación debido a la inestabilidad inducida en el punto de polarización del inversor.

La resistencia en serie R_s limita la corriente y reduce la oscilación de los armónicos además de aislar el inversor del cristal. Puesto que el cristal tiene una capacidad de carga de 10 pF, el valor de C1 y C2 en nuestro diseño es de 20 pF. Sin embargo, se ha optado por poner un valor de 15 pF puesto que los otros 5 pF restantes los aportan las capacidades parásitas. La oscilación generada se hace pasar por un comparador con histéresis para lograr mayor precisión, aunque su uso es opcional.

5.1.6 Alimentación

La alimentación de los circuitos integrados que conforman el AFE esta unificada a 3.3 V. La fuente de dicha alimentación proviene de los reguladores de tensión de la placa de la FPGA.

Dichos reguladores tienen una corriente máxima de 500 mA. Sin embargo, están limitados por la disipación de potencia. La temperatura interna no debe de pasar los 125ºC y la ambiente es de entorno a los 27ºC (Con una resistencia térmica de 163ºC/W) se obtiene una potencia máxima de (125-27)/163=0,6W. Estos reguladores se alimentan a su vez de un puerto USB que entrega 5V. Si la caída de tensión es de 5V-3.3V = 1.7V la corriente máxima que pueden suministrar es de 353 mA.

5.1.7 Diseño de la antena

Un diseño completo de la antena incluye el diseño de la bobina y circuito de resonancia, la adaptación de impedancias entre la antena y la etapa anterior y un filtrado paso bajos EMC (Electromagnetic Compatibility).

Antes de desgranar los pasos seguidos para el diseño, en la figura 4.7 se muestra la apariencia del circuito.

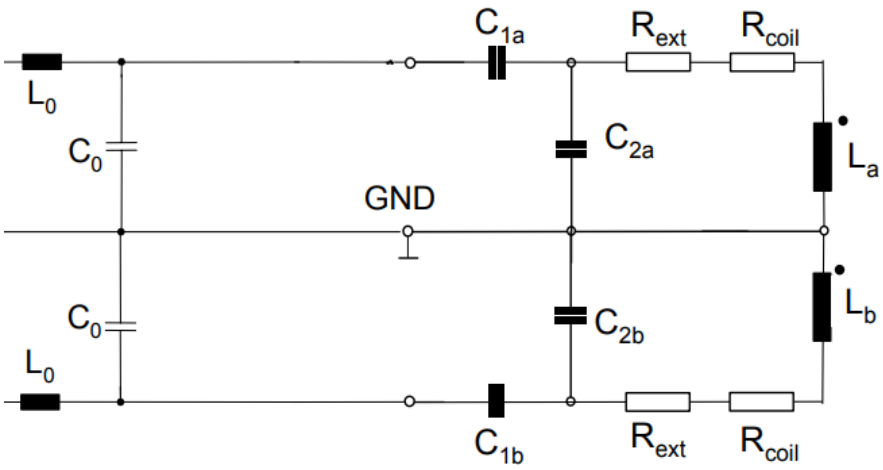

Figura 4.7:Circuito resonante que conforma la antena

Aprovechando la salida diferencial del amplificador FDA, la configuración de la antena es balanceada. Es muy importante mantener una simetría lo mas precisa posible entre las dos ramas del circuito a la hora de realizar el rutado de componentes y en concreto a la hora de realizar la conexión a tierra. La inductancia L_0 y el condensador C_0 conforman el filtrado antiarmónicos EMC. El ajuste de la recuencia de resonancia se efectua mediante C_1 y C_2. La principal función de C_1 es eliminar las componentes continuas que puedan llegar a la antena. En última instancia, la bobina se ha representado en dos partes, L_a y L_b, aunque en la realidad estará conformada por una única pieza.

La razón que subyace a esta representación es mantener la simetría del circuito balanceado, por lo que se debe conectar a tierra en el punto que coincida con la mitad de sus dimensiones. También se han representado la resistencia equivalente del embobinado, denominada Rcoil (Se corresponde con la mitad de la resistencia total equivalente) , y una resistencia exterior Rext que se incluye para poder controlar el factor de calidad Q del circuito resonante. Tal y como se expuso en el apartado 2.3.1, el factor de calidad Q es crítico a la hora de que la envolvente de la portadora cumpla con los tiempos requeridos por el PICC marcados por el estándar ISO 14443-A.

Por supuesto, la frecuencia de resonancia del circuito será de 13,56 MHz.

Para confeccionar el diseño completo del circuito se han llevado a cabo los siguientes pasos [16]:

1. Diseño de la bobina en términos de dimensiones y valor de su inductancia. Consideraciones para el factor de calidad Q.
2. Cálculo de los valores de los condensadores para la frecuencia de resonancia y adaptación de impedancias.
3. Conexión del circuito con el filtrado pasa bajos EMC.

Diseño de la bobina y factor Q

Los valores típicos en términos de inductancia de la bobina oscilan entre los 300 nH y los 3uH. Para nuestro sistema se ha pretendido dimensionar la bobina de tal forma que tenga un valor de 2.7 uH. Por tanto, habrá que adaptar el diseño mecánico del bobinado a dicho valor. Para mantener la simetría un ejemplo de diseño se muestra en la figura 4.8.

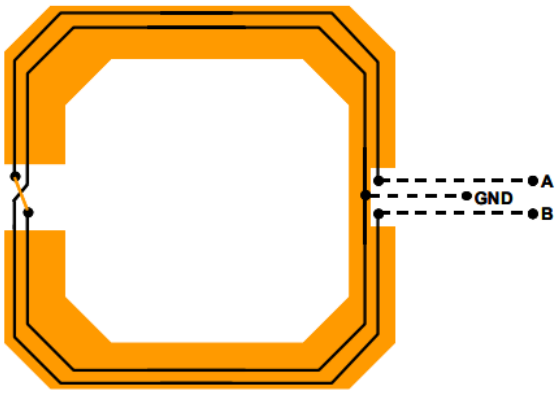

Figura 4.8: Ejemplo orientativo de la disposición de la bobina

En el ejemplo puede apreciarse que la bobina cuenta con dos espiras, una forma cuadrada y una determinada distancia y ancho de pistas. Este es un ejemplo orientativo, la información a la que hay que prestar atención de la figura 4.7 es la forma en la que mantiene la simetría con una conexión a tierra en el punto eléctrico medio, además de cómo se disponen los bornes de la bobina para que la circuitería se sitúe por fuera contribuyendo a una menor interferencia.

Las dimensiones de la bobina diseñada para el sistema de este trabajo han sido pensadas para lograr un mayor factor de acoplamiento con las tarjetas de clase 1 mostradas en la figura 1.3 cuyas dimensiones (81x49 mm) son las más utilizadas en los PICC y muy similares a las de una tarjeta de crédito que implementa la tecnología RFID (86,5 x 53,98 mm).Por tanto, nuestra bobina tendrá una forma rectangular, con las esquinas redondeadas, de unas dimensiones de 87 x 48 mm. Estas dimensiones elegidas también han sido condicionadas por el espacio disponible en la PCB final.

Entonces el objetivo radica en lograr una bobina con una inductancia de 2,7 uH y cuyo largo sean 87 mm y ancho 48 mm. Para lograr cumplir con estos requisitos hay que jugar con el ancho de las pistas (El espesor de las mismas estará estandarizado con un valor fijo de 35 um) y la distancia entre ellas. Otra variable que también hay que tener muy en cuenta es el número de espiras de la bobina.

Para calcular los valores de estas variables se hace uso de las siguientes ecuaciones. La ecuación 4.2 expresa el diámetro equivalente del conductor, que junto con la longitud media (Ecuación 4.3) y el ancho medio (Ecuación 4.4) nos permiten calcular la inductancia mutua que se produce entre los conductores paralelos entre sí (Ecuaciones 4.5 y 4.6). Para el cómputo total también habrá que considerar la autoinductancia de las pistas (Ecuaciones 4.7 y 4.8).

$$d = 2\sqrt{\frac{t * w}{\pi}} \tag{4.2}$$

$$a = \sqrt{\frac{a_0^2 + \left(a_0 - 2N(g+w)\right)^2}{2}} \tag{4.3}$$

$$b = \sqrt{\frac{b_0^2 + \left(b_0 - 2N(g+w)\right)^2}{2}} \tag{4.4}$$

$$M_1 = \frac{\mu_0}{2\pi}\left[a * ln\left(\frac{2ab}{d\left(a+\sqrt{a^2+b^2}\right)}\right) - 2b + \sqrt{a^2+b^2}\right] \tag{4.5}$$

$$M_2 = \frac{\mu_0}{2\pi}\left[b * ln\left(\frac{2ab}{d\left(b+\sqrt{a^2+b^2}\right)}\right) - 2a + \sqrt{a^2+b^2}\right] \tag{4.6}$$

$$L_1 = \frac{a * \mu_0}{16\pi} \tag{4.7}$$

$$L_2 = \frac{b * \mu_0}{16\pi} \tag{4.8}$$

La variable t de la ecuación 4.2 hace referencia al espesor del conductor que como se ha mencionado está fijado a 35 um. Las variables w y g se corresponden con el ancho de las pistas y la distancia entre pistas, respectivamente. La otra variable con la que debemos jugar es la N que referencia el número de espiras de la bobina. Se entiende que los valores que toman a_0 y b_0 se corresponden con las dimensiones de 87 x 48mm, respectivamente.

La inductancia total de la bobina utiliza las ecuaciones descritas anteriormente tal y como describe la ecuación 4.9, donde E representa el exponente de espira con un valor de 1.66.

$$L_{antena} = (2M_1 + 2M_2 + 2L_1 + 2L_2)N^E \quad (4.9)$$

Teniendo en cuenta estas fórmulas, se ha establecido un valor para el ancho de pista de 0,5 mm y un valor para la distancia entre pistas de 0,5 mm. El número de espiras para cumplir con los requisitos es de 4. Estos cálculos no dejan de mostrar una situación idealizada de diseño puesto que en la realidad factores como los márgenes de error en la fabricación de la PCB o el factor del redondeo de las esquinas de la espira provocan una desviación en el valor de la inductancia final de la bobina. Por tanto, a pesar de maximizar los esfuerzos a la hora de realizar los cálculos estos valores son orientativos y habrá que resintonizar el circuito de la antena (Cambiando el valor de los condensadores) debido a estas variaciones inevitables propias del mundo real, adaptándose a la inductancia final que tenga la bobina.

Para poder tener una referencia a la hora de establecer un valor para R_{ext} y por tanto controlar el factor de calidad Q del filtro también hay que calcular el valor teórico de la resistencia que presenta la bobina. Para ello se hace uso de la ecuación 4.10. La variable L expresa la longitud total de las pistas.

$$R_{coil} = \rho_{cobre}\frac{L}{t * w}[1 + \alpha_{cobre}(temp - 25)] = 2.2\Omega \quad (4.10)$$

$$\rho_{cobre} = 1{,}7 \text{ x } 10^{-6}\,\Omega/cm$$
$$\alpha_{cobre} = 3{,}9 \text{ x } 10^{-3}\,\Omega/\Omega/ºC$$

En la tabla 1.6 se muestra un resumen de los parámetros de la inductancia de la antena.

Inductancia	2,7 uH
Resistencia	2,2 ohm
Dimensiones	87x48 mm
Ancho de pista	0,5 mm
Distancia entre pistas	0,5 mm
Número de espiras	4

Tabla 2.1: Parámetros de la bobina de la antena

También hay que tener en cuenta el llamado efecto piel que sufren los conductores. Este fenómeno causa que la resistencia efectiva del conductor en corriente alterna aumente respecto a la resistencia que presenta a la corriente continua ya que la densidad de corriente se concentra en la superficie en lugar de ser homogénea en todo el conductor (De ahí su nombre).

Hay que comprobar a partir de qué profundidad superficial tiene lugar este efecto en el cobre a 13,56 MHz. La profundidad superficial se calcula con la ecuación 4.11.

$$\delta = \frac{1}{\sqrt{\pi f \mu_{cobre} \sigma_{cobre}}} = \frac{0{,}0179}{\sqrt{f}} = 0{,}187\ mm \tag{4.11}$$

El efecto piel para la frecuencia de interés no tendrá lugar hasta los 0,187 mm de espesor. El espesor de nuestras pistas es de 35 um por lo que no supone un problema para nuestro sistema.

En lo que respecta al factor de calidad Q, hay un límite superior. El data rate de la información es de 106 KHz. Por tanto, el ancho de banda deberá de ser de como mínimo el doble, es decir; de 212 KHz. La ecuación 4.12 marca el valor teórico máximo para el factor Q.

$$Q_{max} = \frac{f_o}{B_{min}} = \frac{13{,}56\ MHz}{0{,}212\ MHz} = 63 \tag{4.12}$$

Podemos efectuar una aproximación del valor R_{ext} en función de Q y de la resistencia equivalente de la bobina teniendo en cuenta las ecuaciones 4.13 y 4.14.

$$R_{coil} = \frac{2\pi f L}{Q} \tag{4.13}$$

$$R_{total} = 2R_{ext} + R_{coil} \tag{4.14}$$

Entonces, la resistencia externa R_{ext} puede expresarse tal y como dicta la expresión 4.15.

$$R_{ext} = \frac{1}{2}(R_{total} - R_{coil}) = \frac{2\pi f L}{2\ Q} - \frac{R_{coil}}{2} \tag{4.15}$$

Se puede utilizar en la ecuación 4.15 el valor máximo de Q=63 junto con los parámetros establecidos de la bobina (L hace referencia a su inductancia) para conocer un valor orientativo para R_{ext}. Sin embargo, esto solo sería una aproximación por el efecto (De menor influencia) de los demás elementos del circuito. Aun así esta estimación es necesaria para el cálculo de los condensadores que, en conjunción con la bobina, conforman el circuito resonante. El factor Q se deberá de ajustar manualmente una vez haya concluido el montaje de la PCB y se proceda a comprobar mediante un osciloscopio si la envolvente de la portadora cumple con los tiempos establecidos en el apartado 2.3.1.

Si la envolvente está demasiado tiempo en baja, siendo inferior al mínimo marcado para t_2 (Ver figura 1.11) el factor Q será demasiado alto y habría que incrementar R_{ext}. Si por otro lado t_2 sobrepasa su máximo, el factor Q es demasiado bajo y habría que decrementar R_{ext}.

Cálculo de los condensadores para la frecuencia de resonancia

Con los valores de la inductancia, la resistencia total y la impedancia deseada podemos calcular C_1 y C_2 mediante las ecuaciones 4.16 y 4.17. El valor Z_a hace alusión al valor de la impedancia deseada de una de las ramas del circuito resonante (El valor diferencial sería 2 veces Z_a). Se recomienda que, si la impedancia deseada es de 250 Ohm, Z_a se incremente un poco para aumentar la potencia de salida hacia la antena. Los cálculos se realizarán con Z_a= 350 Ohm.

$$C_2 = \frac{1}{2\pi f \sqrt{\left(\frac{2\pi f L}{1-\frac{R_{tot}}{Z_a}}\right)^2 - \frac{R_{tot}^2 + (2\pi f)^2 L^2}{1-\frac{R_{tot}}{Z_a}} + \frac{(2\pi f)^2 L}{1-\frac{R_{tot}}{Z_a}}}} \tag{4.16}$$

$$C_1 = \frac{R_{tot}^2 + \left(2\pi f L - \frac{1}{2\pi f C_2}\right)^2}{\frac{2\pi f L}{C_2}\left(\frac{1}{2\pi f C_2} - 2\pi f L\right) - \frac{R_{tot}^2}{C_2}} \tag{4.17}$$

Los valores de estos condensadores variarán a la hora de realizar las pruebas pertinentes para ver cuánto se ha desviado la frecuencia de resonancia del circuito de la frecuencia de resonancia deseada de 13,56 MHz. Sin embargo, los valores calculados son un buen punto de referencia para reajustar la antena hasta sintonizarla a la frecuencia de trabajo.

Filtrado pasa-bajos EMC

Para lograr mejores resultados la frecuencia de corte del filtro EMC debe situarse entorno a los 14,5 MHz (Frecuencia donde se encuentra centrada la banda lateral superior). El filtro EMC mejora la relación SNR (Signal-to-Noise Ratio) para la señal de recepción. También mejora la calidad de la señal de transmisión al eliminar el overshoot de los pulsos de emitidos desde el amplificador. La inductancia del filtro consta de un núcleo de ferrita. En la figura 4.9 se muestra la configuración final del circuito y en la tabla 2.2 se recogen los valores calculados para los componentes.

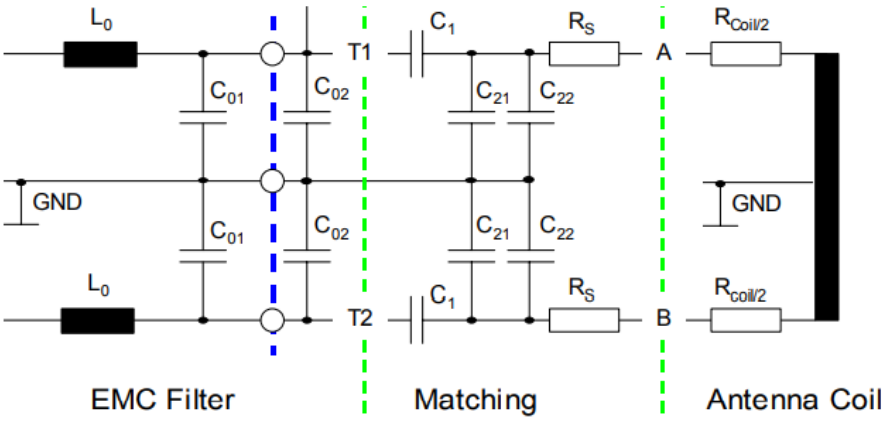

Figura 4.9: Configuración final del circuito resonante de la antena

L_0	1uH
C_{01}	68pF
C_{02}	56pF
C_1	12pF
C_{21}	82 pF
C_{22}	6.8pF
$Rs=R_{ext}$	3.3 ohm

Tabla 2.2: Valores de los componentes del circuito resonante de la antena

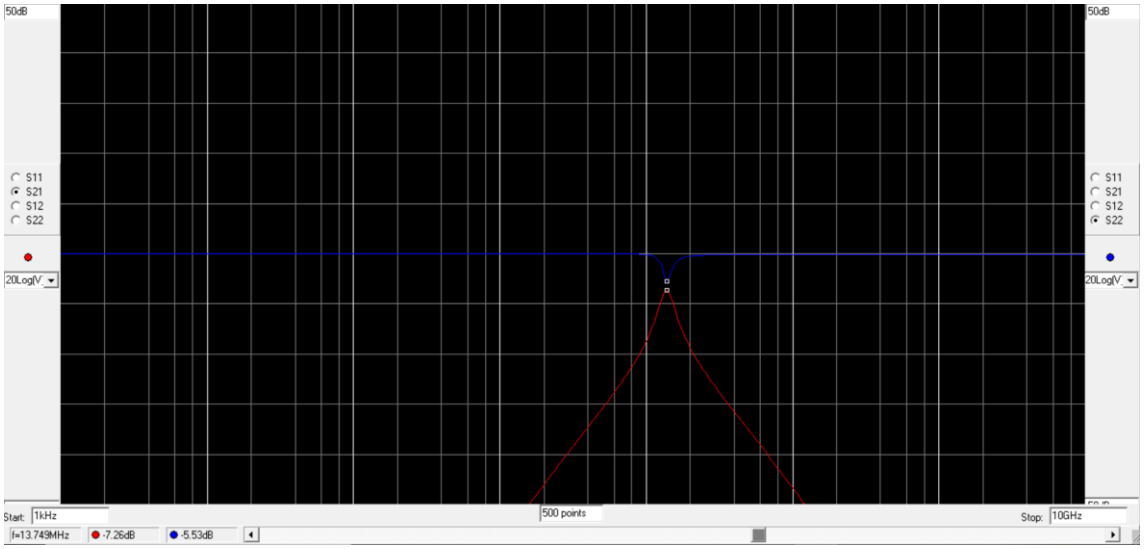

Figura 4.10: Simulación del circuito resonante de la antena

En la figura 4.10 se exponen los resultados de simular la resonancia del circuito. Podemos observar que está bastante centrado. Los valores de la ganancia, de 7,26 dB no son del todo fiables. Esta simulación es un buen punto de partida para la posterior resintonización del circuito real.

5.2 Captura esquemática

Una vez se han expuesto los componentes que conforman el AFE así como los cálculos necesarios para su configuración se procede a realizar la captura esquemática del circuito completo.

La herramienta utilizada tanto para la captura esquemática como para la confección del layout de la PCB (Printed Circuit Board) es Altium Designer, con la licencia que brinda a los estudiantes. Se ha elegido esta herramienta con el objetivo de familiarizarse con ella en vistas al futuro puesto que es de las más utilizadas y valoradas en el entorno profesional de la industria electrónica.

En la figura 4.11 se muestra la captura esquemática para la parte transmisora y en la figura 4.12 para la parte de recepción.

Los circuitos integrados que conforman el DAC y ADC requieren de una fuente de alimentación separada para su circuitería analógica interna y para la digital, un hecho común en aquellos chips que manejan señales mixtas. Esta disgregación en las alimentaciones por parte del fabricante invita al diseñador a filtrar por separado las líneas de las fuentes debido a que los elementos de la circuitería digital conmutan mucho más rápido que los elementos analógicos, demandando picos de corriente de frecuencia más elevada a la fuente. Estos picos pueden resultar fatales a la parte analógica del integrado si ambas alimentaciones estuviesen unificadas. Puede apreciarse en la figura 4.11 que el filtrado de la fuente para las líneas analógicas es más estricto que para las líneas digitales.

Las conexiones marcadas con rojo se refieren a la alimentación analógica y las marcadas en amarillo a la alimentación digital. Algo similar ocurre para las tierras donde se hace una distinción entre tierra analógica y tierra digital. Aunque tierra no hay más que una, estas están separadas y se unen en un punto específico del plano de masas. Este aspecto se discutirá en el siguiente apartado 5.3 donde se desarrolla el diseño del layout. El color azul remarca las tierras analógicas y el verde las digitales. Esto es de gran ayuda a la hora de confeccionar el rutado y la puesta a tierra de los integrados. También se puede observar que todos los integrados incluyen sus condensadores de desacoplo.

Se han incluido 3 jumpers de selección, P2, P3 y P4, para el control manual de las funciones ya mencionadas para el formato de los datos del ADC y del DAC. Con P2 y P3 se controla si dicho formato es binario natural o por el contrario complemento a dos. Hay que recordar que esta selección también es accesible desde la FPGA mediante el pinout que se muestra a continuación El jumper P4 permite al usuario elegir el rango de entrada del ADC entre los valores 1V, 1,5V o 2V.

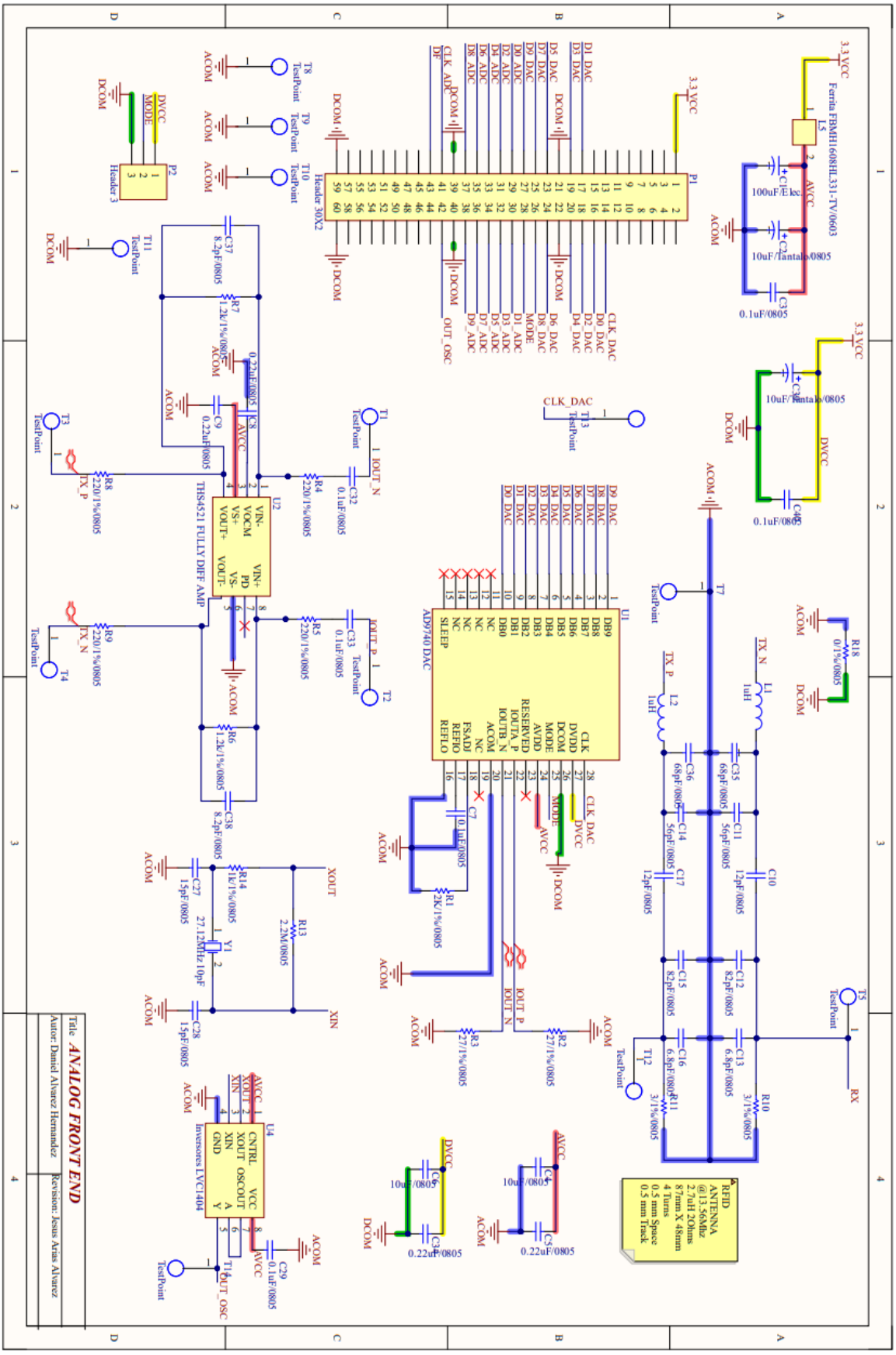

Figura 4.11: Captura esquemática de la parte transmisora

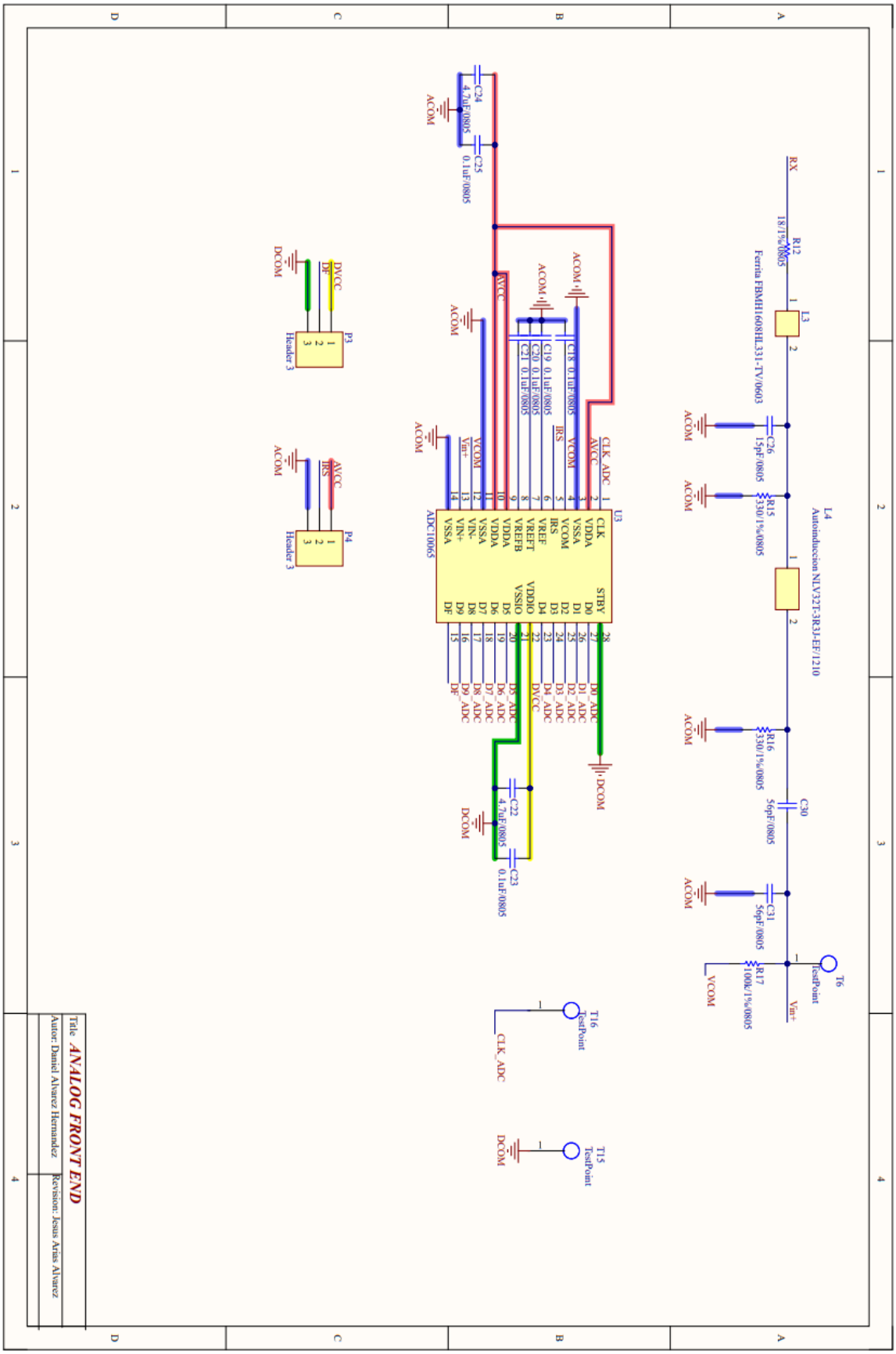

Figura 4.12: Captura esquemática de la parte receptora

Los cálculos de los valores de las resistencias y condensadores se han efectuado teniendo en cuenta los valores estandarizados por la serie E12. Mención especial a la tolerancia de los condensadores que conforman el circuito de la antena. Con ánimo de lograr la mayor precisión posible se han escogido tolerancias del 2%. Esto supone un coste añadido al proyecto que vale la pena asumir puesto que la antena es la parte más crítica del AFE.

Finalmente se han añadido unos testpoints que permiten acceder a las pistas de más interés (Como pueden ser los puntos de alimentación, varios puntos de tierra, la señal del reloj, las líneas de señal antes y después de entrar al amplificador, etc.) para poder realizar mediciones con el multímetro y el osciloscopio.

5.2.1 Consumo de potencia

En lo que respecta al consumo de potencia, en la tabla 2.3 se recoge el consumo de cada componente y el consumo total de corriente y potencia máximas.

	Corriente Máxima	Potencia Maáxima
ADC	30 mA	99Mw
FDA	55mA	182mW
DAC	50mA	165mW
OSCILLATOR	32mA	106mW
TOTAL	167mA	552mW

Tabla 2.3: Consumo de potencia

Los reguladores de la ICECREAM pueden otorgar hasta 352 mA por lo que estamos dentro del rango con un consumo máximo posible de 167 mA.

5.2.2 Pinout del AFE

Para conectar los pines digitales del DAC y ADC junto con las señales de control de sus modos, así como la señal del reloj al pinout de la FPGA se ha utilizado un cabezal macho de 30 x 2 posiciones. A la hora de establecer qué señal va en qué pin del cabezal P1 se ha tenido en cuenta la relación de la tabla 2.3 que muestra que pin del empaquetado de la FPGA ICE40HX1K/4K va conectado a que pin del DAC, ADC y CLK. Para conocer las asignaciones de los pines hembra del cabezal izquierdo de la placa ICECREAM al pinout del empaquetado de la FPGA se ha recurrido al layout mostrado en la figura 4.13.

Pin Function	Pin Type	Bank	FPGA(TQ144) 144-Pin TQFP Pin Number	ANALOG FRONT END	
IOL_6B_GBIN7	GBIN	3	20	CLK_DAC	
IOL_7B	DPIO	3	22	D0_DAC	
IOL_8A	DPIO	3	23	D1_DAC	
IOL_8B	DPIO	3	24	D2_DAC	
IOL_9A	DPIO	3	25	D3_DAC	
IOL_9B	DPIO	3	26	D4_DAC	
IOL_10A	DPIO	3	28	D5_DAC	DAC
IOL_10B	DPIO	3	29	D6_DAC	
IOL_11A	DPIO	3	31	D7_DAC	
IOL_11B	DPIO	3	32	D8_DAC	
IOL_12A	DPIO	3	33	D9_DAC	
IOL_12B	DPIO	3	34	MODE	
IOB_36_GBIN4	GBIN	2	52	OSC_OUT	XTAL
IOB_24	PIO	2	37	D0_ADC	
IOB_25	PIO	2	38	D1_ADC	
IOB_26	PIO	2	39	D2_ADC	
IOB_27	PIO	2	41	D3_ADC	
IOB_28	PIO	2	42	D4_ADC	
IOB_29	PIO	2	43	D5_ADC	ADC
IOB_30	PIO	2	44	D6_ADC	
IOB_31	PIO	2	45	D7_ADC	
IOB_32	PIO	2	47	D8_ADC	
IOB_33	PIO	2	48	D9_ADC	
IOB_35_GBIN5	GBIN	2	49	CLK_ADC	
IOB_37	PIO	2	56	ISR_ADC	

Tabla 2.3: Pinout de la FPGA asociado al pinout del AFE

5.3 Diseño PCB

Una vez completada la captura esquemática el siguiente paso a seguir consiste en elaborar el layout que conforma la PCB. La PCB diseñada cuenta con unas dimensiones máximas de 10 x 10 mm. Estas dimensiones están limitadas en cierto modo por el fabricante puesto que otras medidas mayores suponen un sobrecoste inasumible para este proyecto. Afortunadamente nuestro sistema cuenta con espacio suficiente con dichas medidas.

Como ya se ha mencionado, la placa diseñada está pensada para acoplarse al modelo ICECREAM de la FPGA diseñada por Jesús Arias, en concreto sobre sus cabezales de la parte izquierda. Se puede apreciar el layout de la placa de desarrollo en la figura 4.13.

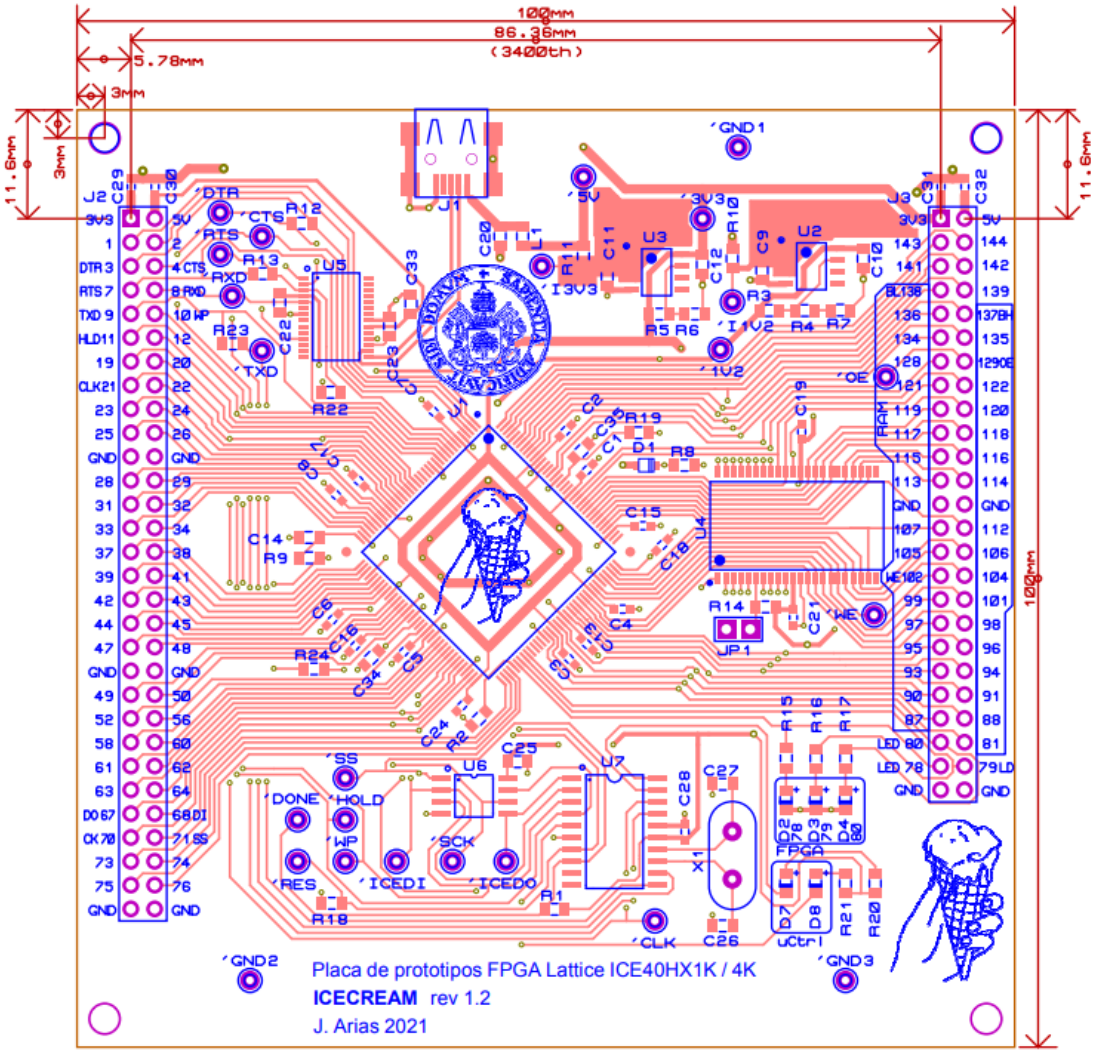

Figura 4.13: Layout de la cara superior de la placa ICECREAM

5.3.1 Huellas de componentes

Prácticamente todos los componentes pasivos son del tipo SMD (Surface-Mount Device) y cuentan con una huella 0805 (En la escala métrica). Las únicas excepciones son las inductancias L3 y L5, que cuentan con una huella 0603, y las inductancias L1,L2 y L4 con una huella 1210. Los componentes del tipo trhoughole (Agujero pasante) son el condensador del filtrado C1, los cabezales de los jumpers y el header de interfaz con la placa de la FPGA (Ambos tipos con un pitch de 2.54 mm) y el cristal de cuarzo para la generación de reloj.

Las huellas de los circuitos integrados fueron creadas manualmente siguiendo las dimensiones mecánicas indicadas en sus respectivos datasheet. En la parte superior izquierda de la figura 4.14 se puede observar la huella del DAC AD9740 que cuenta con un empaquetado SOIC28. El FDA THS4521 cuenta con un empaquetado SOIC8 y su huella puede apreciarse en la parte superior derecha. El empaquetado del ADC10065 también cuenta con 28 pines pero con un pitch más reducido. Se trata de un empaquetado TSOP28, en la parte inferior derecha de la figura. La huella del controlador del cristal es para un empaquetado SOP8, apreciable en la parte inferior izquierda.

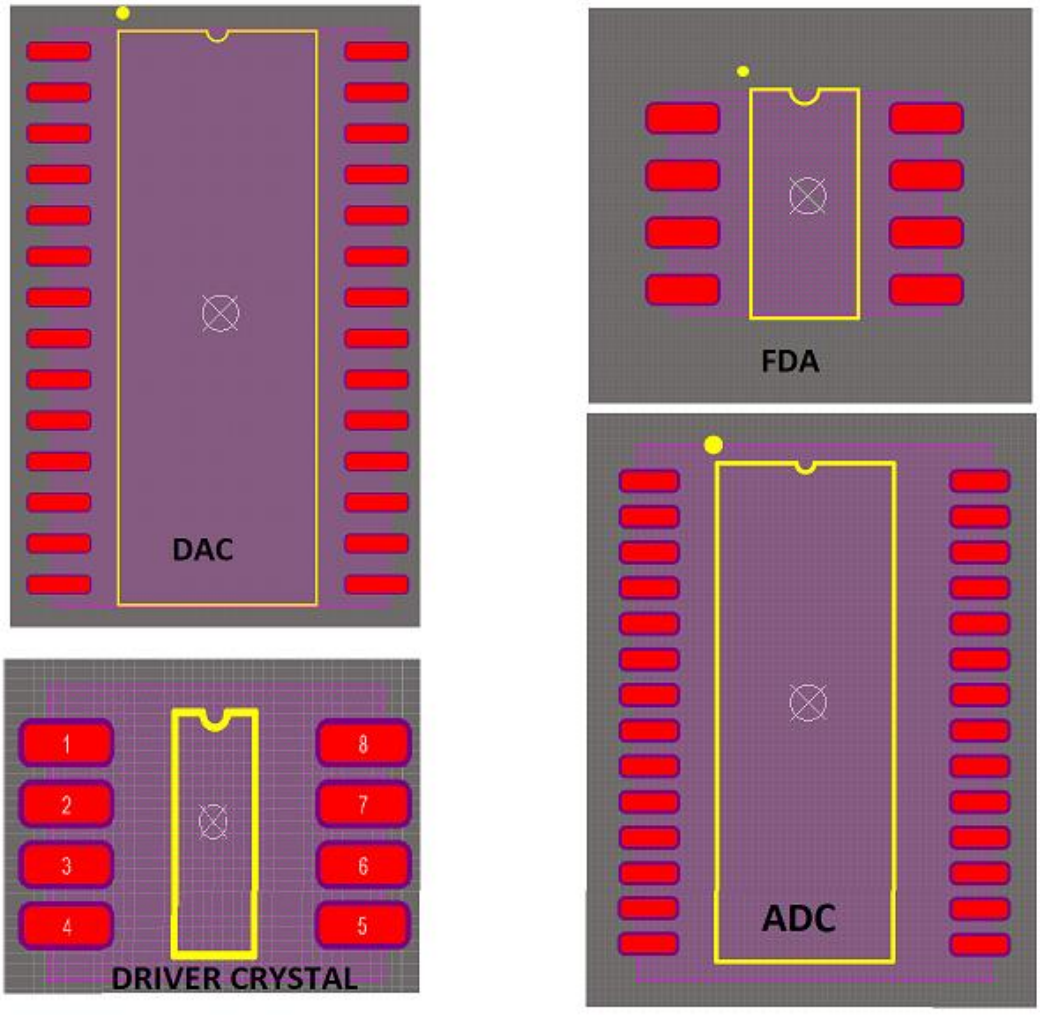

Figura 4.14: Huellas de los circuitos integrados del AFE

5.3.2 Posicionado de los componentes

Para establecer la posición de los circuitos integrados se ha tomado como criterio la separación de la circuitería digital de la analógica. Sin embargo, el DAC y ADC tienen ambos tipos de circuitería. Afortunadamente la disposición de sus patillas ha sido favorable en este aspecto puesto que la mitad pueden considerarse analógicas y la otra mitad digitales, al implementar la interfaz paralela de la palabra de 10 bits. Se han orientado las patillas digitales hacia el cabezal de conexión con la FPGA, dispuesto a la derecha. Por otro lado, la ruta que sigue la señal analógica es muy similar a la dispuesta en los diagramas de bloques del AFE de la figura 3.1.

En la parte más occidental de la misma se encuentra la bobina de la antena, que ocupa prácticamente la mitad del espacio disponible. En la figura 4.15 puede observarse la apariencia final de la antena cuyas dimensiones son las establecidas en la tabla 2.1. El rutado de la antena se ha llevado a cabo de forma muy estricta en lo que ha dimensiones se refiere, para poder acercarse lo máximo posible al valor deseado de 2,7 uH. Puede apreciarse la conexión a tierra (Por la cara inferior marcada en azul) en el punto eléctrico donde la simetría es perfecta. Los elementos del circuito resonante también siguen una simetría estricta impuesta por la naturaleza balanceada de la antena.

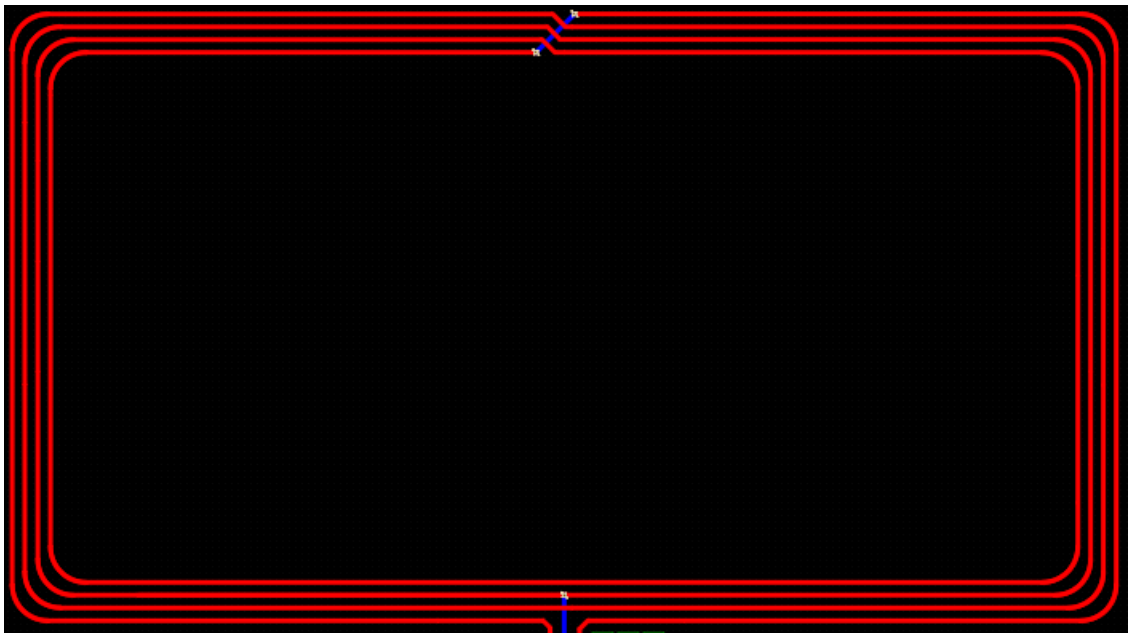

Figura 4.15: Rutado de la bobina que conforma la antena

En la figura 4.16 se muestra la capa correspondiente a la serigrafía. De esta forma puede apreciarse mucho mejor la colocación descrita de los componentes cuya designación es la marcada en la captura esquemática de las figuras 4.11 y 4.12. La serigrafía también incluye el logo de la escuela y el nombre del autor y tutor del proyecto.

La colocación final de los elementos de la PCB obedece a ciertas reglas que se han seguido para cumplir con los requisitos que impone la compatibilidad electromagnética, descritas a continuación en el siguiente apartado.

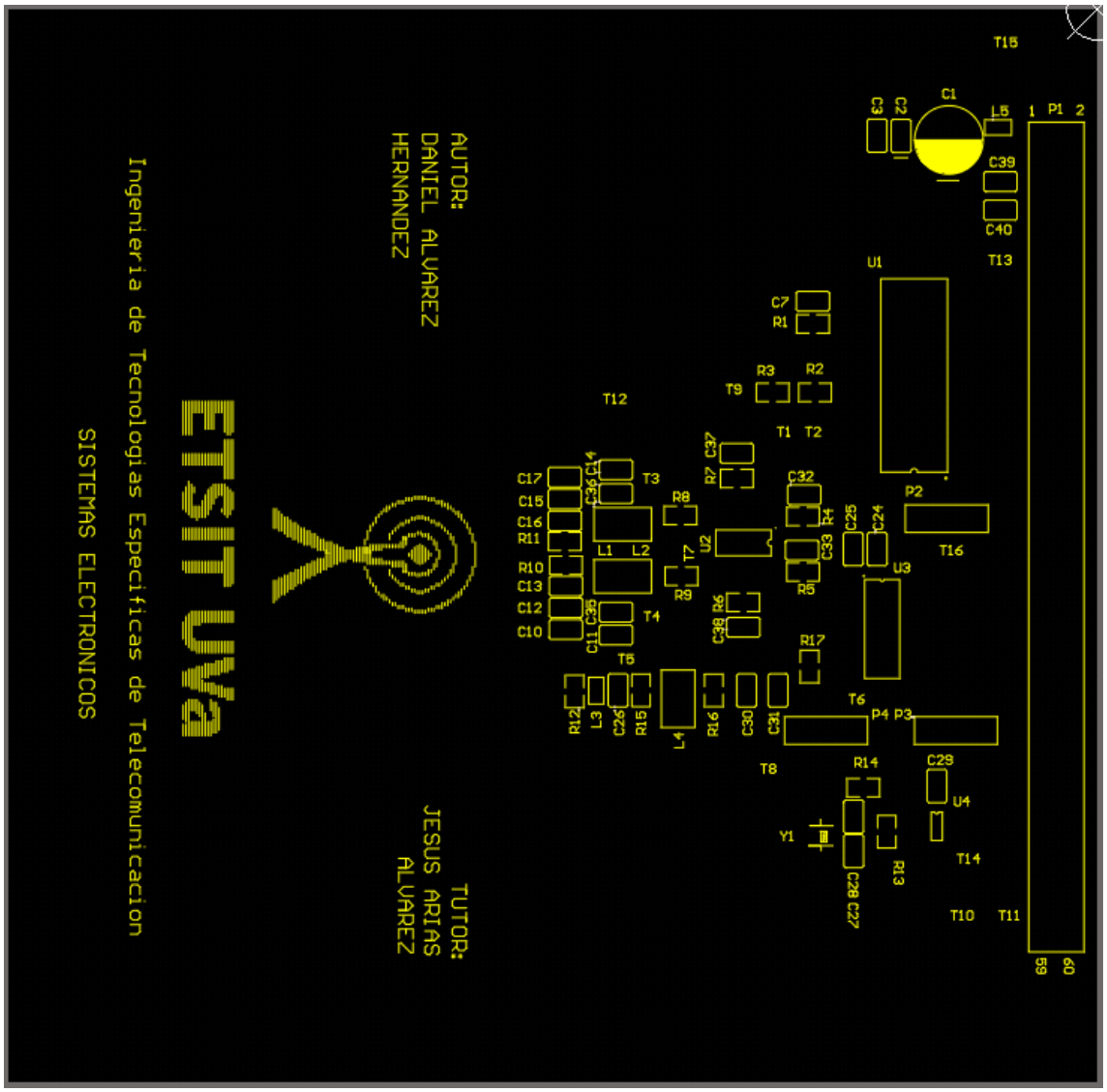

Figura 4.16: Serigrafía de la capa superior de la PCB

5.3.3 Compatibilidad Electromagnética

A la hora de efectuar el posicionamiento y rutado de la PCB es muy importante seguir ciertas pautas referidas a la compatibilidad electromagnética en concordancia con las normas IPC (Institute of Printed Circuits).

Las pistas de interconexión deben de ser lo más cortas posible con la intención de reducir su resistividad. Además deben de estar lo suficientemente alejadas entre si para evitar acoplamientos capacitivos e inductivos. Hay que evitar que las pistas por donde viajan señales de frecuencias altas realicen giros bruscos de 90º .

El área que forman las pistas debe de ser lo más pequeño posible para evitar los bucles de corriente y reducir las emisiones EMI (ElectroMagnetic Interference).Si las pistas actúan como antena radiante pueden emitir señales hacia el exterior e interferir en otros equipos. Teniendo en cuenta que una antena es igual de buena emisora que receptora también las interferencias externas pueden afectar al funcionamiento normal del circuito.

Los condensadores de desacoplo en los puntos de alimentación y de voltaje de referencia de cada integrado deben de estar lo más cerca posible a estos. Aunque no es buena idea incluir elementos en la capa inferior de la PCB pensando en términos de producción, en el diseño de este sistema se han incluido algunos condensadores de desacoplo por esta cara para poder cumplir con la premisa de situarlos lo más cerca posible del dispositivo a proteger. Esto no supone ningún problema puesto que este diseño no se va a producir en masa.

Tal y como ya se ha mencionado con la alimentación separada para los circuitos analógicos y para los digitales, también hay que hacer lo propio con el plano de masa en circuitos donde están presentes señales mixtas. Existe un extenso debate acerca de cómo habría que implementar esta separación de masas y de cómo habría que unirlas puesto que tierra no hay mas que una y hay que unificar el potencial a cero. La estrategia que se ha llevado a cabo en este diseño consiste en la llamada unión de estrella [24], representada en la figura 4.17.

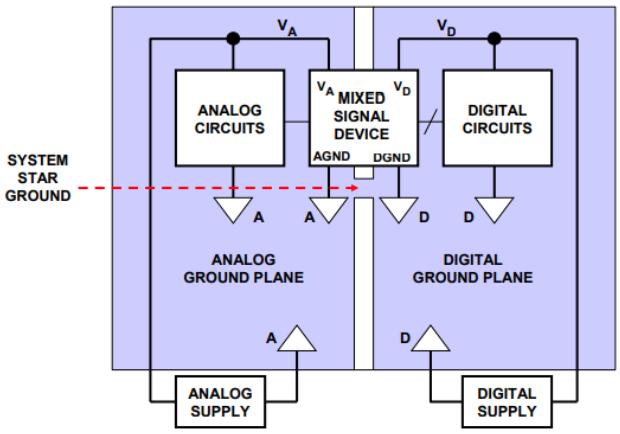

Figura 4.17: Unión en estrella de la tierra digital y la analógica.

La estrategia de la unión en estrella es eficaz cuando únicamente hay presente un dispositivo de señal mixta, en cuyo caso la unión de los planos de masa tiene lugar justo por debajo del integrado. Aunque nuestro sistema conste con dos integrados de señal mixta como son el DAC y el ADC (En cuyo caso la medida pierde eficacia), la unión entre los planos se ha efectuado en el punto medio entre ambos cuidando de no rutar pistas que corten la brecha que los separa. En la figura 4.18 puede apreciarse la unión de los planos de masa implementados en la capa inferior de la PCB.

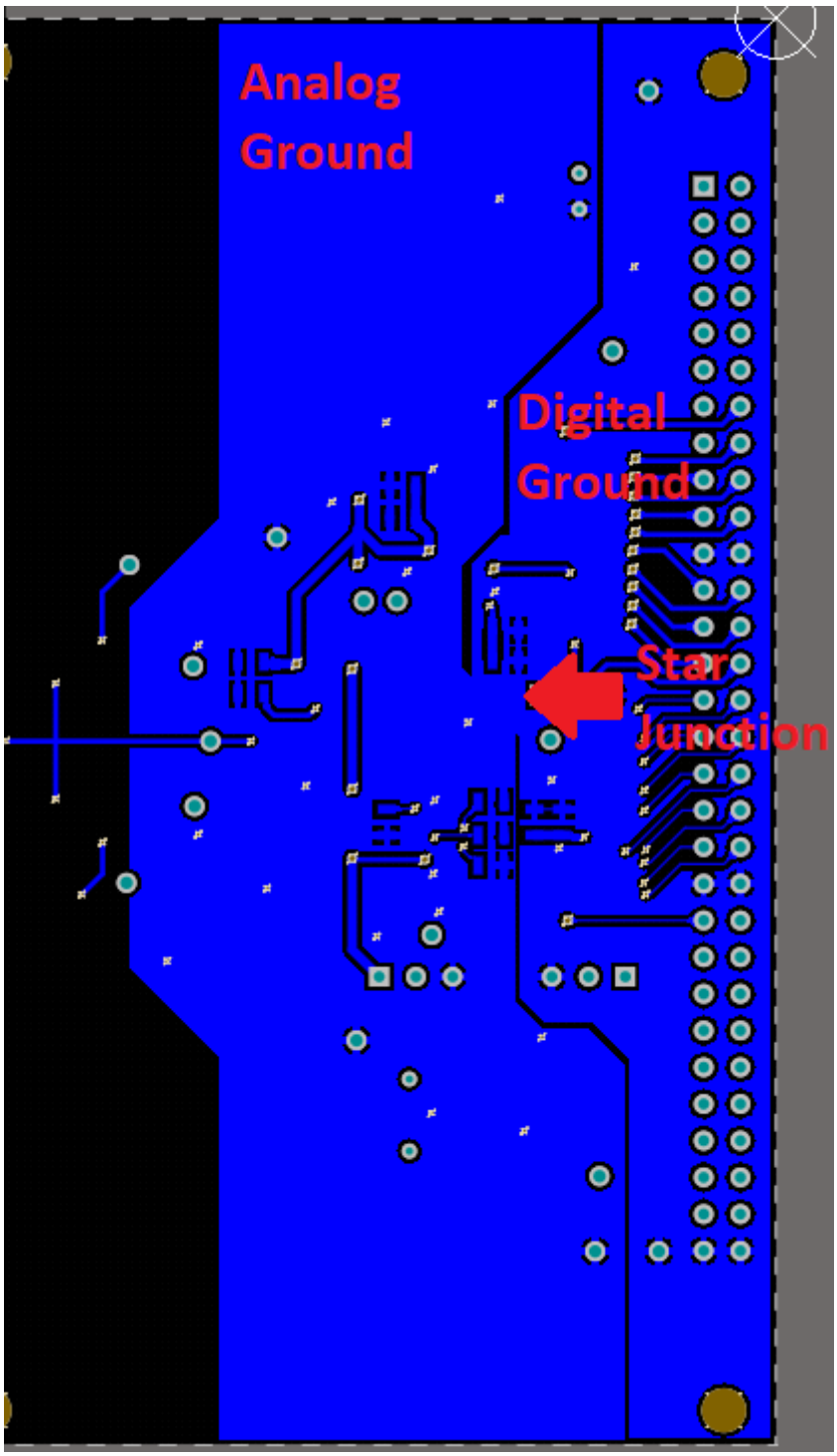

Figura 4.18: Unión en estrella de los planos de masa del AFE

Otro aspecto importante respecto a las masas es el referido al plano situado por debajo de la bobina de la antena. En la gráfica superior de la figura 4.19 [25] puede comprobarse como afecta el plano de masas al campo generado por la bobina. La intensidad del campo magnético desaparece a partir del radio en el que se encuentra el plano. Este efecto puede no importar mucho al diseñador puesto que es posible que en el sistema no se efectúen lecturas desde la parte inferior de la antena e incluso sea realmente beneficioso por si hubiese otra circuitería por debajo del lector.

Inicialmente el lector diseñado en este trabajo se iba a situar justo por encima de la placa ICESCREAM por lo que el efecto del campo de la antena debía ser "estrangulado" utilizando este método. Sin embargo, para evitar riesgos innecesarios y facilitar el rutado de los componentes finalmente se optó por situar el AFE a un costado de la placa de desarrollo de la FPGA. Por tanto, la utilización de un plano de masas por debajo de la bobina no es necesaria y únicamente lo que se logra es limitar la intensidad del campo y por tanto acortar la mínima distancia de lectura, tal y como queda patente al compararla situación con la mostrada en la gráfica inferior de la figura 4.19.

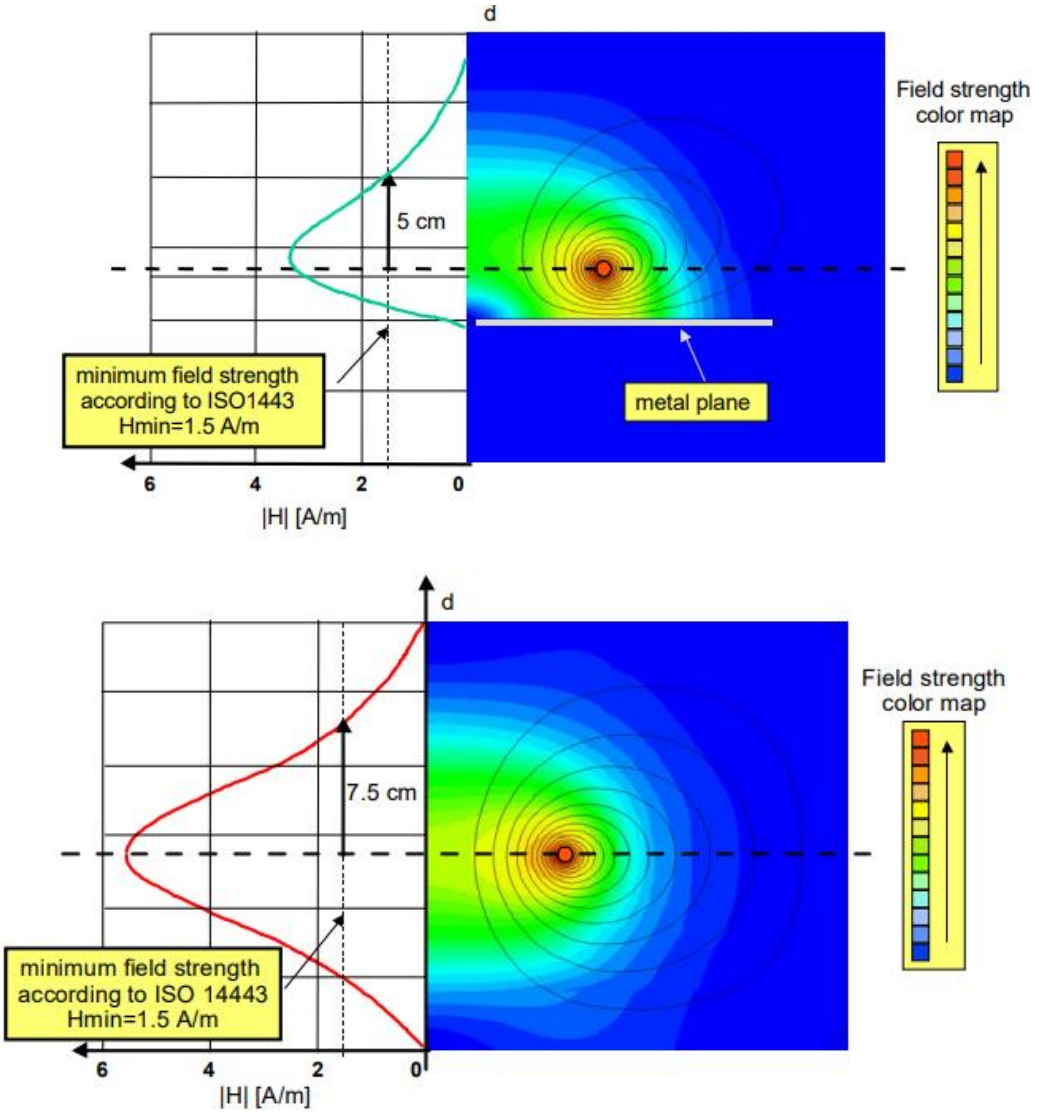

Figura 4.19: Efecto del plano de masa sobre la bobina de la antena

5.3.4 Fabricación PCB

El layout completo de la PCB puede apreciarse en la figura 4.20. A partir del diseño completo se generan los gerber files y los drill files que necesita el fabricante para la fabricación.

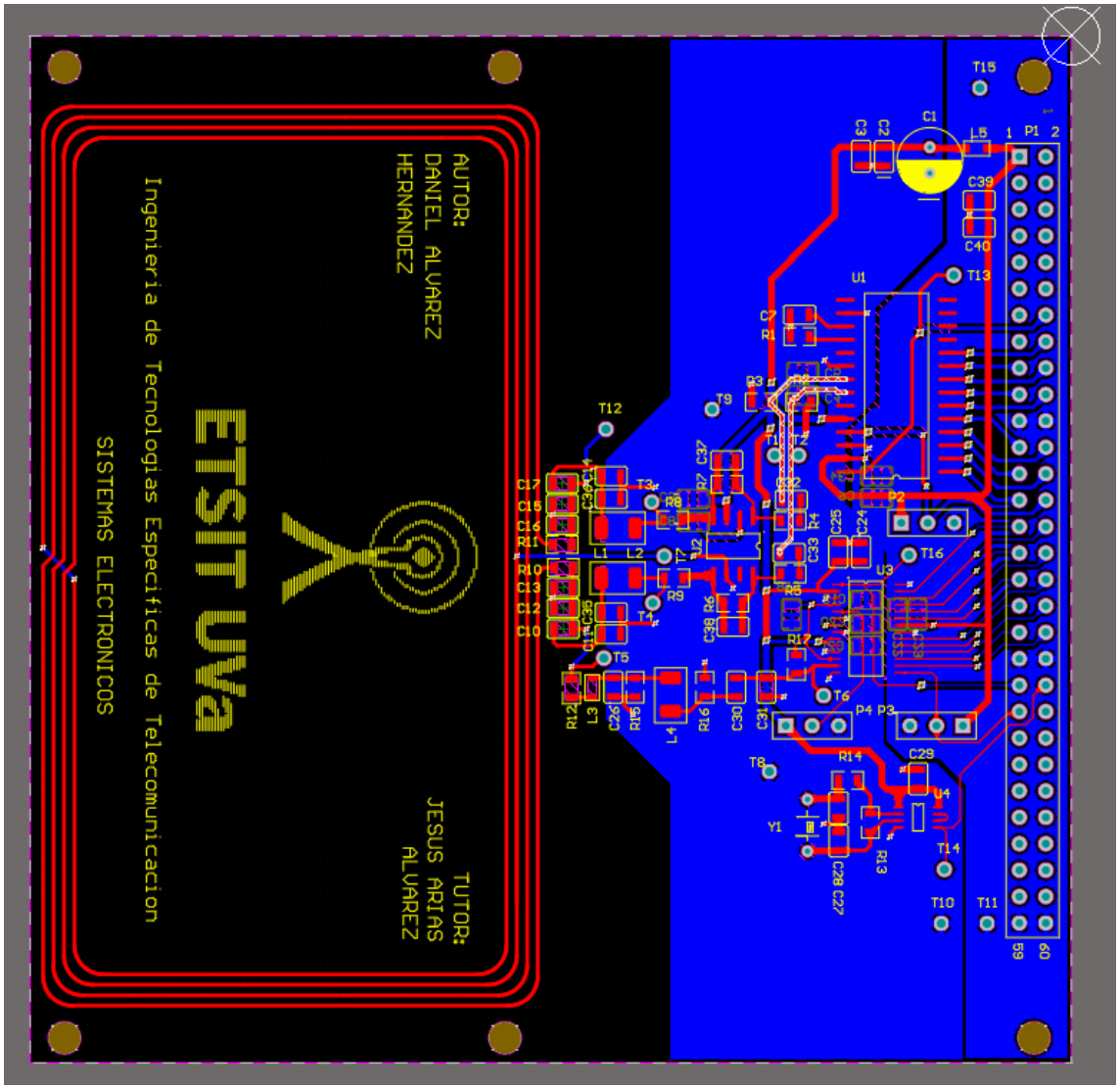

Figura 4.20: Layout completo del AFE

Figura 4.21: Representación 3D de la PCB

Bill Of Materials

Una forma eficaz de organizar la lista de componentes para su control y poder realizar su pedido al distribuidor electrónico es mediante la denominada BOM (Bill Of Materials). En la BOM de la tabla 2.4 se recoge el nombre del componente, su designación en el esquematico, su huella y su numéro logístico referido a un determinado proveedor.

Comment	Description	Designator	Footprint	Quantity	Supplier	Supplier Part Numer
100uF/Elect.	Polarized Capacitor (Radial)	C1	CAP TH 2.5 PITCH ELECT	1	DigiKey	493-1548-ND
10uF/0805/Tantalo	Polarized Capacitor (Radial)	C2, C39	0805C	2	DigiKey	478-8115-1-ND
0.1uF/0805	Capacitor	C3, C7, C18, C19, C20, C21, C23, C25, C29, C32, C33, C40	0805C	12	DigiKey	399-9158-1-ND
10uF/0805	Capacitor	C4, C6	0805C	2	DigiKey	399-4925-1-ND
0.22uF/0805	Capacitor	C5, C8, C9, C34	0805C	4	DigiKey	399-9206-1-ND
*12pF/0805/2%	Capacitor	C10, C17	0805C	2	DigiKey	1276-2580-1-ND
*56pF/0805/2%/NPO	Capacitor	C11, C14, C30, C31	0805C	4	DigiKey	399-17451-1-ND
*82pF/0805/2%	Capacitor	C12, C15	0805C	2	DigiKey	399-9272-1-ND
*6.8pF/0805/2%	Capacitor	C13, C16	0805C	2	DigiKey	399-10076-1-ND
4.7uF/0805	Capacitor	C22, C24	0805C	2	DigiKey	399-3134-1-ND
15pF/0805	Capacitor	C26	0805C	1	DigiKey	399-1111-1-ND
*15pF/0805/2%	Capacitor	C27, C28	0805C	2	DigiKey	399-14557-1-ND
*68pF/0805/2%	Capacitor	C35, C36	0805C	2	DigiKey	399-14622-1-ND
8.2pF/0805	Capacitor	C37, C38	0805C	2	DigiKey	399-14635-1-ND
1uH	Inductor	L1, L2	1210I	2	DigiKey	445-16564-1-ND
Ferrita FBMH1608HL331-TV/0603		L3, L5	0603FB	2	DigiKey	587-3815-1-ND
Autoinduccion NLV32T-3R3J-EF/1210		L4	1210I	1	DigiKey	445-16592-1-ND
Header 30X2	Header, 30-Pin, Dual row	P1	HDR2X30	2	DigiKey	S1012EC-30-ND
Header 3	Header, 3-Pin	P2, P3, P4	HDR1X3	3	DigiKey	732-5316-ND
2K/1%/0805	Resistor	R1	0805R	1	DigiKey	A126358CT-ND
27/1%/0805	Resistor	R2, R3	0805R	2	DigiKey	A126357CT-ND
220/1%/0805	Resistor	R4, R5	0805R	2	DigiKey	311-220CRCT-ND
1.2k/1%/0805	Resistor	R6, R7	0805R	2	DigiKey	A129750CT-ND
220/1%/0805	Resistor	R8, R9	0805R	2	DigiKey	311-220CRCT-ND
4/1%/0805	Resistor	R10, R11	0805R	2	DigiKey	541-4.02CCCT-ND
18/1%/0805	Resistor	R12	0805R	1	DigiKey	A129728CT-ND
2.2M/0805	Resistor	R13	0805R	1	DigiKey	RMCF0805FT2M20CT-ND
1k/1%/0805	Resistor	R14	0805R	1	DigiKey	RNCP0805FTD1K00CT-ND
330/1%/0805	Resistor	R15, R16	0805R	2	DigiKey	A129743CT-ND
100k/1%/0805	Resistor	R17	0805R	1	DigiKey	RMCF0805FT100KCT-ND
0/1%/0805	Resistor	R18	0805R	1	DigiKey	RMCF0805ZT0R00CT-ND
AD9740 DAC		U1	SOIC28 AD9740	1	DigiKey	AD9740ARZ-ND
THS4521 FULLY DIFF AMP		U2	SOIC8 THS4521	1	DigiKey	296-39226-1-ND
ADC10065		U3	TSOP28	1	DigiKey	296-35231-1-ND
Inversores LVC1404		U4	SOP8 74LVC1404 - duplicate	1	DigiKey	296-17257-1-ND
27.12MHz 10pF	Crystal Oscillator	Y1	XTAL 27.12MHz	1	DigiKey	1923-1441-ND

Tabla 2.4: Bill Of Materials

Manufactura

Una vez se ha efectuado el pedido de los componentes y la PCB se procede a soldarlos manualmente en la placa. Primero se procede a soldar los circuitos integrados. Mas tarde los componentes SMD pasivos y por último los elementos de agujero pasante como el cristal y los cabezales.

Hay que mencionar que la soldadura de los componentes se ha efectuado por etapas (Primero la parte del DAC, luego la etapa amplificadora, etc.) para verificar individualmente su correcto funcionamiento y en caso de detectar fallos poder tenerlos más localizados. Este asunto se desarrolla en detalle en el apartado 5.5. En las figuras 4.22 y 4.23 aparece fotografiada la PCB final junto con la ICESCREAM.

Figura 4.22: Fotografía de la PCB del AFE

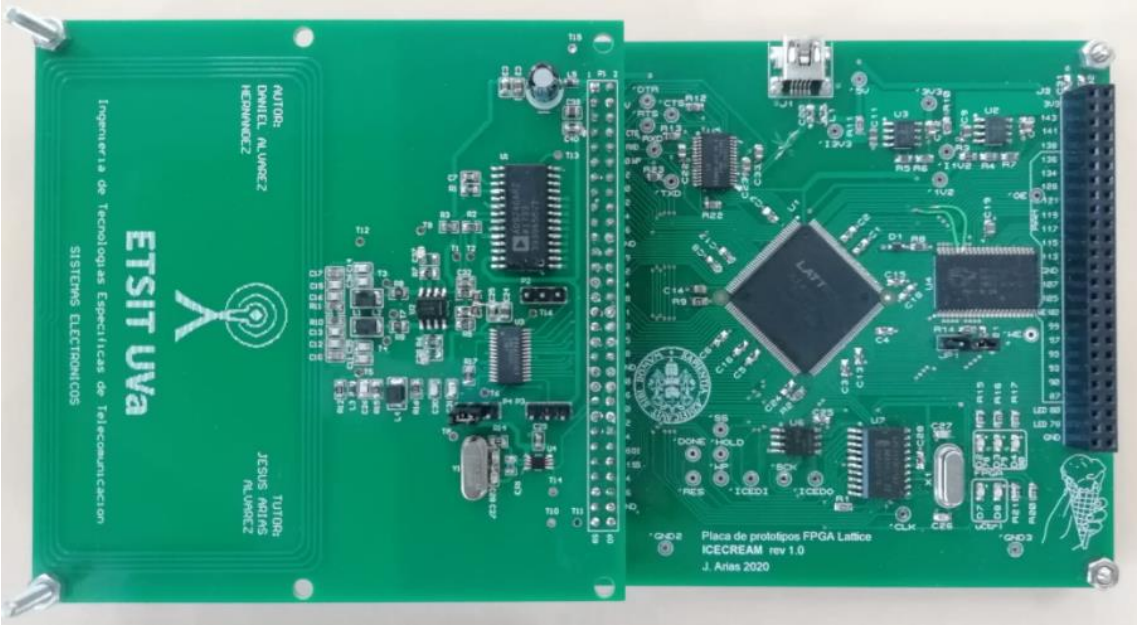

Figura 4.23: Fotografía del AFE acoplado a la PCB de la ICESCREAM

5.4 Verificación y modificaciones hardware

A medida que se han ido soldando los componentes de los diferentes bloques funcionales se han ido testeando para comprobar su correcto funcionamiento.

Verificación del ADC y DAC

Para comprobar el buen funcionamiento de estos dispositivos se generó una señal mediante un integrado 555 muestreada por el ADC. Dichas muestras se enviaron directamente hacia el DAC para contrastar su señal con la del integrado. Las pruebas fueron satisfactorias determinando pues el correcto funcionamiento de ambos dispositivos.

Verificación de la generación de la señal de reloj

A la hora de realizar el pedido de los componentes se cometieron dos errores en lo que se refiere al circuito que genera la señal de reloj de 27,12 MHz. El circuito integrado que implementa los inversores del oscilador no coincidía con la huella de la PCB puesto que pertenecía a un empaquetado diferente. A pesar de ello se soldó en la huella disponible no sin el requerimiento de cierta habilidad a la hora de disponer las patillas en la posición correcta.

El otro error cometido en este aspecto es el referido al cristal de cuarzo. El cristal que formaba parte de la BOM original oscilaba en su tercer armónico en lugar del fundamental. Por tanto, la frecuencia del reloj era de entorno a los 9 MHz. Para solucionar este error no hubo más remedio que pedir de nuevo los cristales que oscilasen en su tono fundamental de 27,12 MHz. En la figura 4.24 se puede apreciar la señal de reloj con un periodo de 36,87 ns.

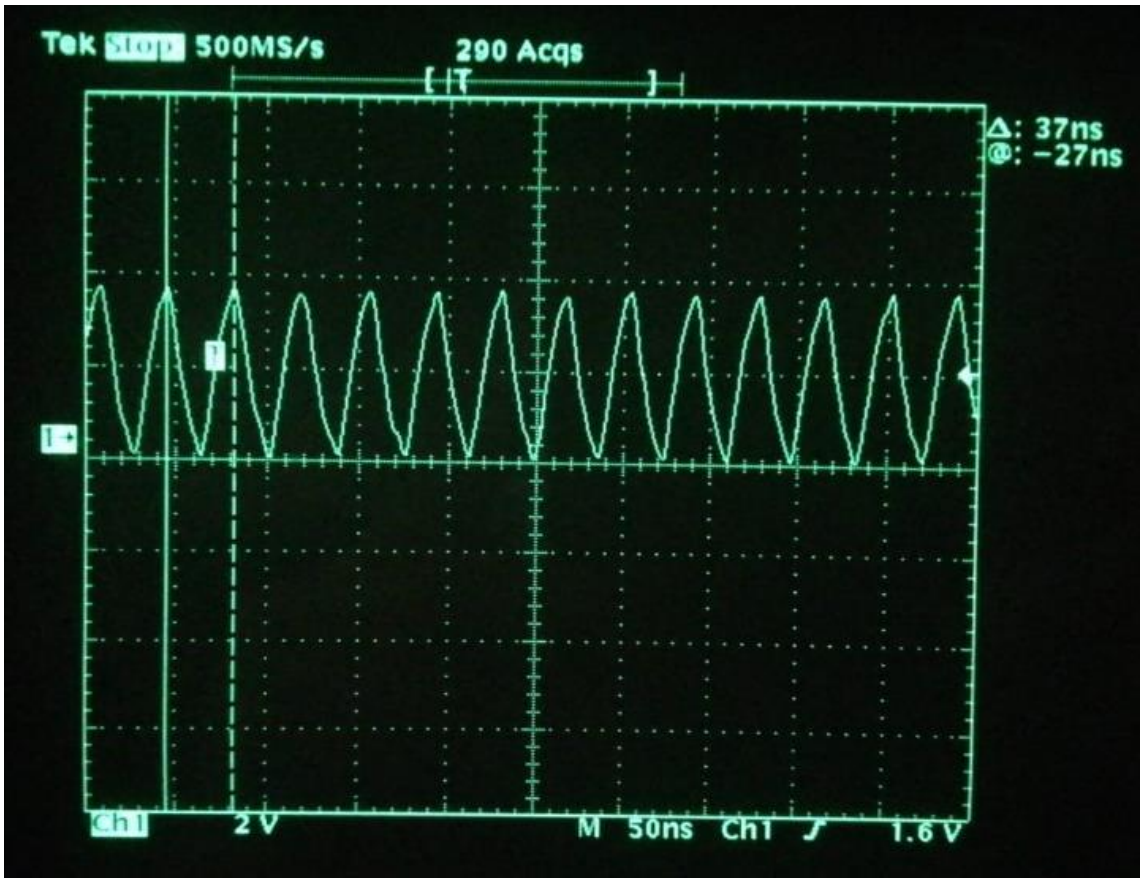

Figura 4.24: Frecuencia de reloj proveniente del AFE

Sintonización del circuito resonante de la antena

Una de las partes más críticas del diseño del AFE es la correcta sintonización del circuito resonante centrado en 13,56 MHz. A pesar de realizar un diseño de las pistas de la bobina lo más preciso posible factores como el redondeo de las esquinas, resolución del fabricante a la hora de confeccionar el rutado, capacidades parasitas, fallos en la simetría, etc. propician que la resonancia real no esté centrada exactamente en la frecuencia de interés.

Se procedió a medir la reactancia de la bobina a 50 KHz obteniendo un valor de Z=0,854 + 0,900j Ohm. Esto equivale a tener una bobina de 2,865 uH (El diseño inicial pretendía un valor de 2,7uH) con una resistencia equivalente de 0,854 Ohm. Estas medidas son un buen punto de referencia para estimar la desviación de la frecuencia central del circuito tanque, aunque no son del todo precisas puesto que las mediciones solo pudieron realizarse a 50 KHz en lugar de los 13,56 MHz del sistema.

Para comprobar la desviación de la frecuencia de resonancia se efectuó un barrido de frecuencias generado desde la FPGA hacia el DAC. Para la medición con el osciloscopio se empleó un diodo en serie con la sonda puesto que la capacidad de la sonda altera la frecuencia de resonancia. En la figura 4.25 aparece la medición del barrido de frecuencias.

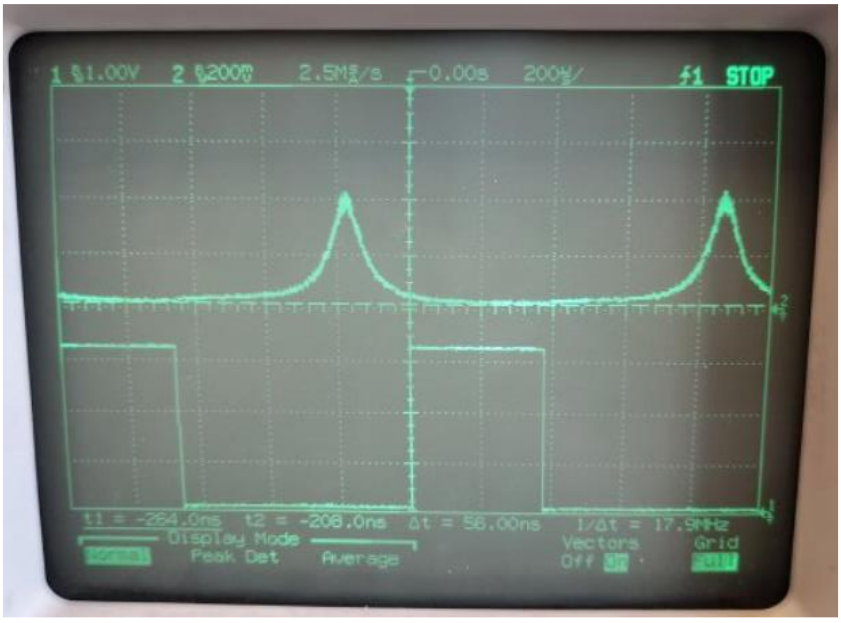

Figura 4.25: Barrido de frecuencias para determinar la resonancia de la antena.

La onda rectangular de la figura 4.25 tiene el flanco de bajada cuando comienza un barrido (Entre 9,88 MHz a 15,6 MHz) y el de subida justo a 13,56 MHz. La resonancia de la antena se sitúa en 12,6 MHz en lugar de los 13,56 MHz pretendidos. Esto supone una desviación de la frecuencia deseada del 7%.

Para ajustar la frecuencia de resonancia se modificó el valor de los condensadores que conforman el circuito tanque, en concreto el valor de los condensadores C12 y C15 del esquemático de la figura 4.11. Dichos valores originales de 82pF se cambiaron por 68 pF, teniendo en cuenta las mediciones antes mencionadas de la reactancia de la bobina. Tras este nuevo ajuste se volvió a efectuar el barrido de frecuencias, cuyo resultado se observa en la figura 4.26.

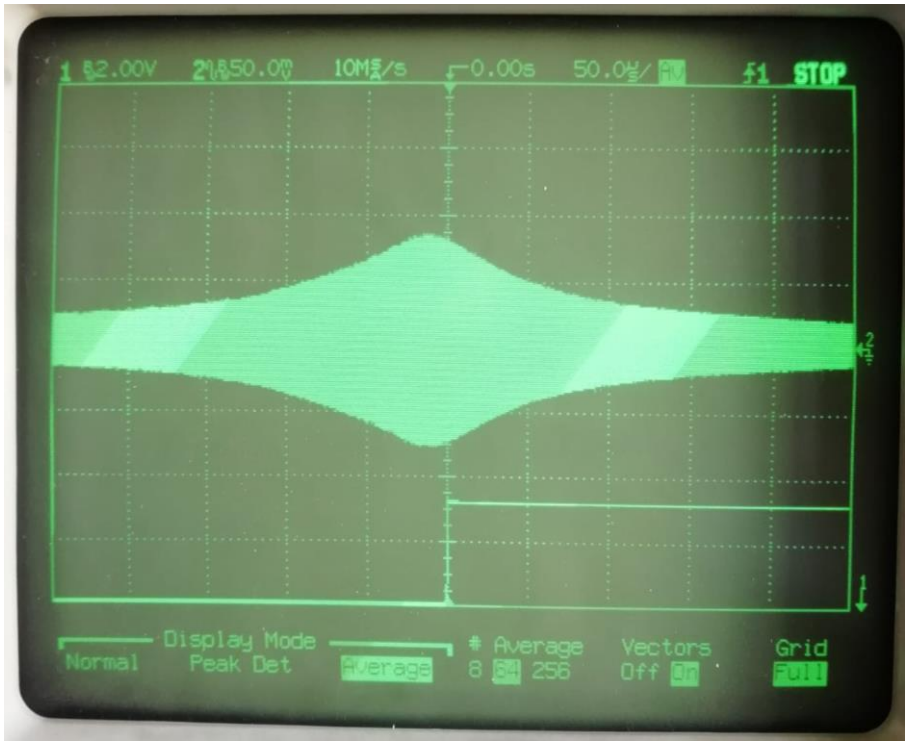

Figura 4.26: Segundo barrido de frecuencias para la obtención de la resonancia

Con los nuevos valores en C12 y C15 la frecuencia de resonancia es de 13,49MHz. Este valor se desvía un 0.5% de la frecuencia central de 13,56MHz. Esta desviación es bastante más asumible que la anterior y se puede afirmar que el circuito resonante ha quedado sintonizado.

Verificación del factor de calidad Q del circuito resonante

Tal y como se ha mencionado en apartados anteriores, el factor Q del circuito resonante determina los tiempos de subida y de bajada de la envolvente de la portadora. Estos tiempos deben de cumplir los márgenes impuestos por el estándar 14443 A tal y como se expuso en la figura 1.11. Las resistencias externas R10 y R11 controlan dicho factor de calidad, en este caso con un valor de 3,3 Ohm.

En la figura 4.27 aparece la medida de t_1 que debe de tener un valor entre los 2 us y los 3us. En este caso se determina que t_1 cumple con el rango puesto que tiene un valor de 2,36 us.

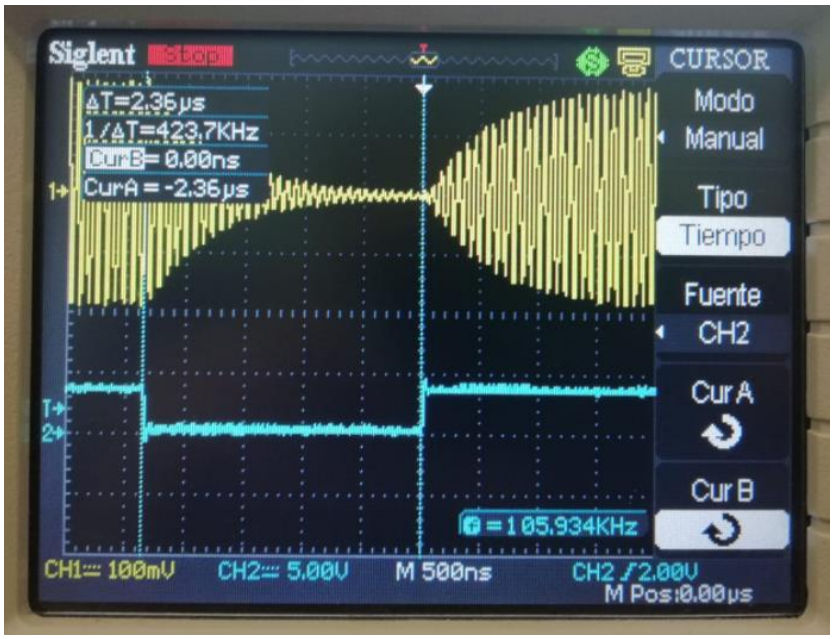

Figura 4.27: Medida de t_1

Si t_1 es menor que 2,5 us el valor de t_2 debe de estar dentro del rango de entre los 0,7us como mínimo y t_1 como máximo. En la figura 4.28 puede comprobarse que el valor de t_2 es de 0,82 us y que por tanto cumple con la condición.

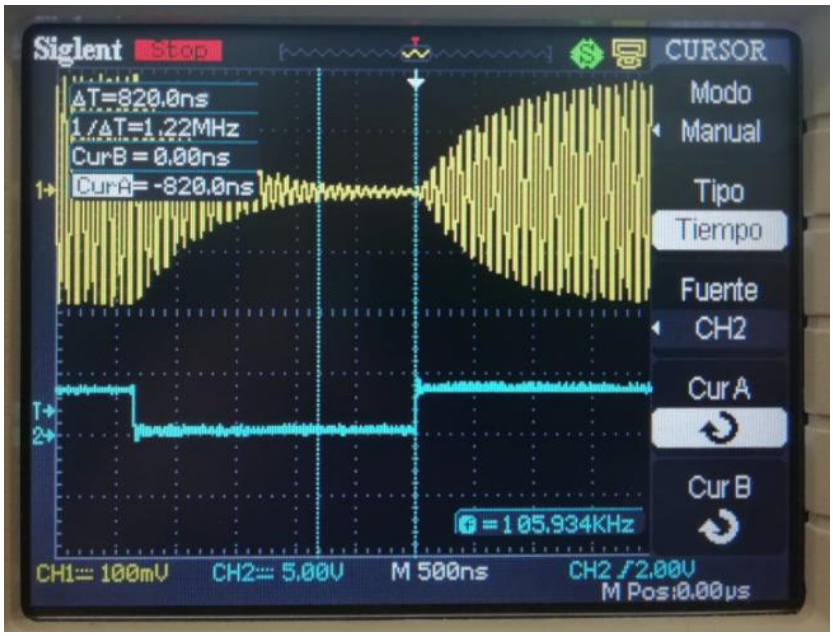

Figura 4.28: Medida de t_2

La condición que debe de cumplir t_3 consiste en que su valor debe de ser mayor que 0 us y menor que 1,5 us. Tal y como queda patente en la figura 4.29 el valor de t_3 cumple con lo esperado con un valor de 1,32 us.

Figura 4.29: Medida de t_3

Corrección en el pinout del AFE

Inicialmente el diseño del AFE se pensó para una versión de la ICECREAM que contaba con una FPGA Laticce ICE40 HX1K. En este modelo el pin 50 está accesible como uno de los pines globales que internamente recorren toda la FPGA por lo que su utilización es muy útil para el rutado de las señales del reloj puesto que estas deben de ir a un gran número de módulos síncronos. Por ese motivo se conectó el reloj de 27,12MHz generado en el AFE al pin 50.

Sin embargo, la versión final utilizada de la ICESCREAM cuenta con el modelo Laticce ICE40 HX4K. La característica por la que finalmente se optó por este modelo fue principalmente la incorporación de dos PLL necesarios para generar las 2 frecuencias principales del sistema, una para la parte receptora (57,63 MHz) y otra para la emisora (54,24 MHz). También otro aliciente a tener en cuenta a la hora de decantarse por este modelo fue el número de celdas lógicas con el que cuenta, muy superior al modelo de 1k que finalmente se habría quedado corto.

El problema que se plantea radica en que el pin 50 en el modelo 4k no está accesible para el usuario. Por tanto, la señal de reloj rutada a dicho pin se cambió a otro pin accesible de la FPGA que contase con la naturaleza de rutado global tal y como tenía el pin 50. El único pin accesible con dichas características en la versión 4k es el pin 52. Desgraciadamente esto acarrea otro problema puesto que dicho pin no está libre, siendo utilizado por la FPGA para controlar el formato de los bits del ADC (El pin DF controla el formato Straight Binary o Two's Complement). Afortunadamente dicho formato de bits también puede controlarse mediante el jumper P3, por lo que dicha funcionalidad no se acaba perdiendo.

La solución final ha consistido en soldar un puente desde el pin 50 al 52 del header del AFE y en separar la pista que estaba conectada al pin 52 proveniente del ADC rajándola con un cutter.

Modificación de la ganancia del FDA

El amplificador diferencial THS4521 no aporta la ganancia estimada de 5. En una primera instancia se retiraron los condensadores C17 y C18 los cuales formaban un filtro pasa bajos y por tanto podrían estar limitando demasiado el ancho de banda. Aunque su retirada supuso un aumento en la ganancia de la señal, este aumento resultó ínfimo. Otra solución podría consistir en cambiar la relación entre las resistencias de entrada y las resistencias de realimentación, Sin embargo, esta medida distorsiona la señal por lo que finalmente se ha optado por reducir el valor de las resistencias de salida R8 y R9.

Con los nuevos valores de 10 Ohm para cada una se obtiene una señal en la antena de en torno a 2 Vpp para cada cana, tal y como aparece en la figura 4.30. Por tanto, la señal diferencial consta de una amplitud de 4 Vpp en la antena. Con la anterior amplitud de apenas 500 mV la tarjeta no era capaz de emitir una respuesta lo suficientemente grande como para ser detectada por el lector.

Al reducir la resistencia de salida la antena demanda más corriente. Puede resultar nefasto que la demanda de corriente sea superior a la corriente proporcionada por los reguladores o incluso superior a la corriente que puede proporcionar a su salida el FDA. Sin embargo, esto no ha supuesto un problema a la hora de implementar la solución final.

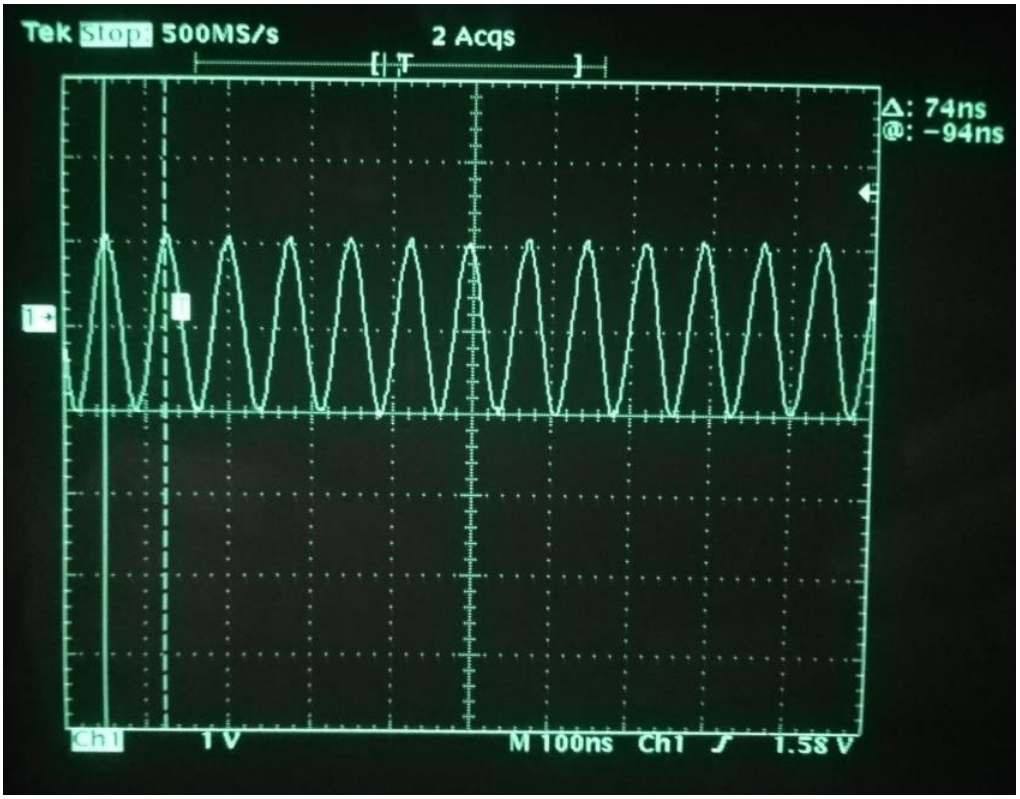

Figura 4.30: Amplitud de la portadora en uno de los canales de la antena.

6. Diseño SDR en la FPGA

La lógica digital que conforma el sistema SDR se ha implementado una FPGA de Laticce, en concreto en el modelo ICE40 HX4K. Este modelo cuenta con dos PLL necesarios para generar las dos frecuencias de reloj en las que se basan el sistema de recepción y el sistema de transmisión. El sistema de recepción trabaja a 57,63 MHz que es justo 4 veces la frecuencia de la banda superior que se sintoniza. El sistema de transmisión trabaja a 54,24 MHz que es 4 veces la frecuencia de la portadora.

Según el fabricante Laticce, este modelo HX4K cuenta con 3520 celdas lógicas utilizando sus herramientas oficiales. Sin embargo, este modelo realmente cuenta con las mismas celdas lógicas que su versión superior HX8K, es decir; con 7680 celdas lógicas siempre y cuando se empleen las herramientas de software libre del proyecto IceStorm [26].

Con la FPGA se controlarán los bits enviados al DAC del AFE así como su modo de funcionamiento. También recibe los datos del ADC y la frecuencia de reloj base de 27,12MHz empleada por los PLL.

En la figura 5.1 se representa el diagrama de bloques general del sistema completo. El sistema cuenta con un bloque inicializador que genera una señal de reset de 4 ciclos de portadora de duración para los bloques síncronos de cada subsistema. La exposición detallada de los bloques de transmisión y de recepción se desarrolla en los siguientes subapartados.

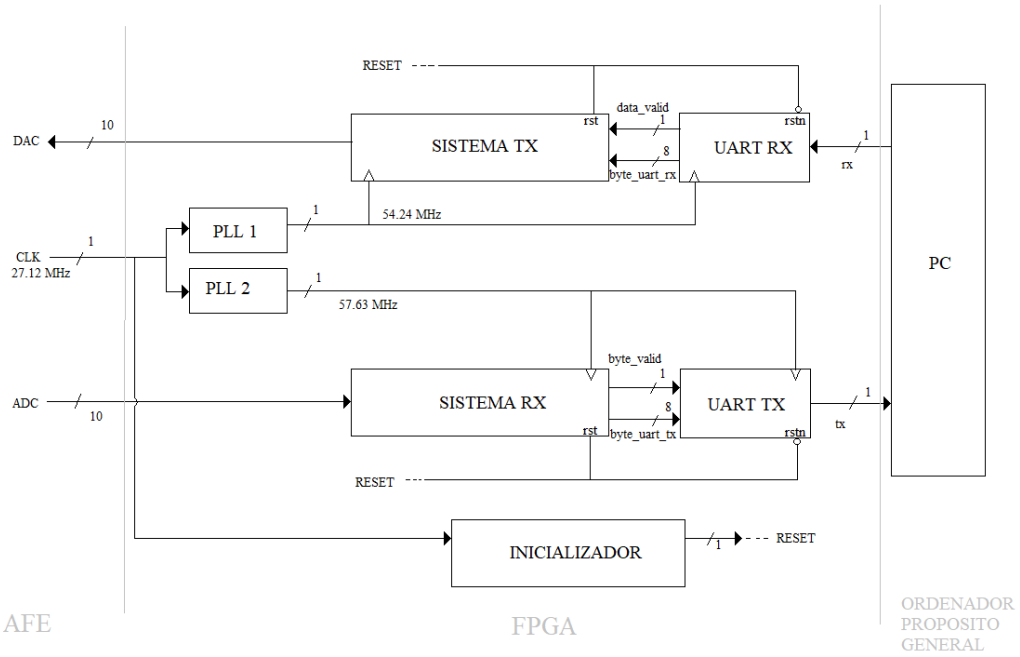

Figura 5.1: Diagrama de bloques general del sistema SDR en la FPGA

6.1 Transmisor

La parte transmisora tiene la tarea de emitir los comandos que le llegan al sistema desde la UART de recepción. Una vez ha llegado un comando, este pasa al bloque codificador que se encarga de generar una secuencia de bits que siga las directrices de la codificación Miller modificado detallada en la sección 2.3.1, concretamente en la figura 1.10.

Esta secuencia de bits codificados pasa al bloque modulador que se encarga de agregarlos a la portadora mediante la modulación ASK 100%. La palabra digital que conforma la portadora modulada consta de 10 bits que es el ancho de palabra que maneja la interfaz paralela del DAC. El formato de la palabra de bits transmitida al DAC es "straight binary", es decir; valores con signo positivo (Entre 0 y 1023) y un offset con valor 512.

En la figura 5.2 se representa el diagrama de bloques que conforman el transmisor.

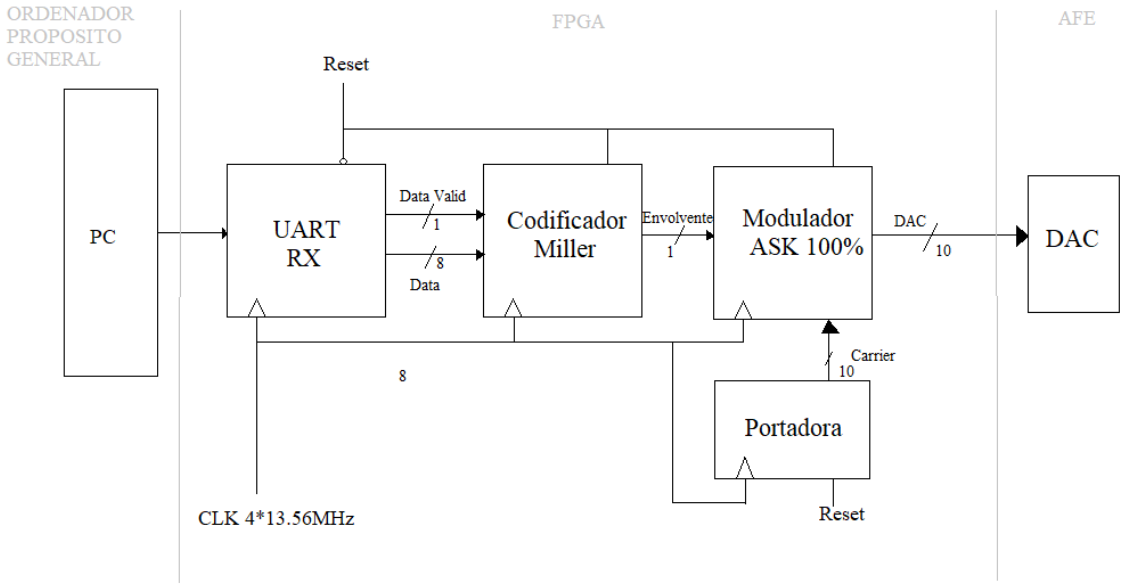

Figura 5.2: Diagrama de bloques del sistema transmisor

6.1.1 Generación del reloj

El reloj que emplean todos los bloques del transmisor es de 4*13,56MHz (54,24MHz) proveniente del PLL interno de la FPGA cuya frecuencia de referencia son los 27,12MHz del oscilador del AFE.

Por tanto, todas las acciones síncronas tendrás como referencia los flancos positivos de dicha frecuencia. Se ha elegido una frecuencia de reloj que sea 4 veces la de la

portadora para la generación de la misma. El mínimo de muestras por ciclo para generar la portadora y entregársela al DAC es de 2 según el criterio de Nyquist. Los contadores internos que utiliza el codificador se incrementan hasta 512 cuentas (128 ciclos de portadora * 4 muestras por ciclo) para determinar la duración de un tiempo de bit de 106 KHz.

La UART RX también genera los baudios de la comunicación a partir del reloj de 54,25MHZ. Para ello emplea un divisor de 235 logrando un baudrate de 230400 Hz.

6.1.2 UART RX

Los bloques que conforman la UART de recepción se pueden apreciar en la figura 5.3 [27]. Las estradas de la UART son el reloj de 54,24 MHz, una señal de reset negada (rstn) y la línea de recepción serie asíncrona (rx). Como salidas cuenta con un flag (rcv) de 1 ancho de pulso de duración que indica la disponibilidad del carácter que ha llegado y el propio carácter de 8 bits (Data).

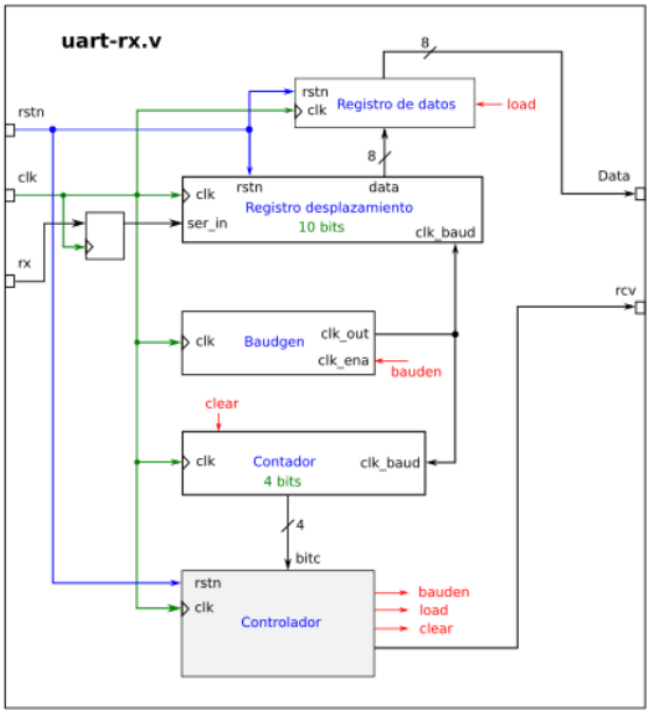

Figura 5.3: Bloques de la UART RX

La señal rcv permanece en bajo cuando la línea esta libre y no hay un carácter transmitido. Cuando por la línea de recepción llega un carácter el receptor lee los bits muestreándolos a la mitad del tiempo de bit gracias a la configuración del generador de baudios. Al leer los bits en la mitad de su periodo se minimiza el error puesto que el dato será mucho más estable en ese instante de tiempo. El receptor va almacenando los datos en un registro desplazador. Cuando se recibe el bit de stop por la línea de recepción el

dato se captura en un registro de salida y al siguiente ciclo de reloj la señal rcv índica que el dato ya puede extraerse de la UART.

La ruta de datos de la UART está gobernada por un controlador modelado como una máquina de estados, representada en la figura 5.4 [27].

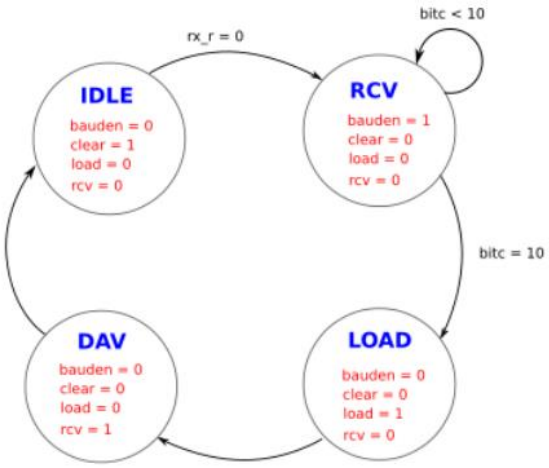

Figura 5.4: Maquina de estados que gobierna la UART RX

El estado IDLE representa el estado de reposo en el que se espera recibir el bit de start por la línea de recepción asíncrona. Cuando se recibe el bit de start la maquina pasa al siguiente estado RCV. Se permanecerá en este estado hasta que se hayan recibido todos los datos. El contador bitc cuenta los bits que se van recibiendo, activado mediante la orden baudgen. Estos bits se van almacenando en el registro de desplazamiento y cuando se ha completado la recepción (1 bit start + 8 bits datos + 1 bit stop) el contador bitc tendrá el valor10 y se pasará al siguiente estado de LOAD. En este estado se activa la orden load para que se almacene en un registro de salida el dato recibido dando paso al siguiente estado DAV que pone en alta el flag rcv indicando a los bloques posteriores a la UART que ya pueden capturar el dato recibido.

6.1.3 Codificador Miller y Modulador

En la figura 5.5 aparecen representados los bloques que conforman el codificador Miller.

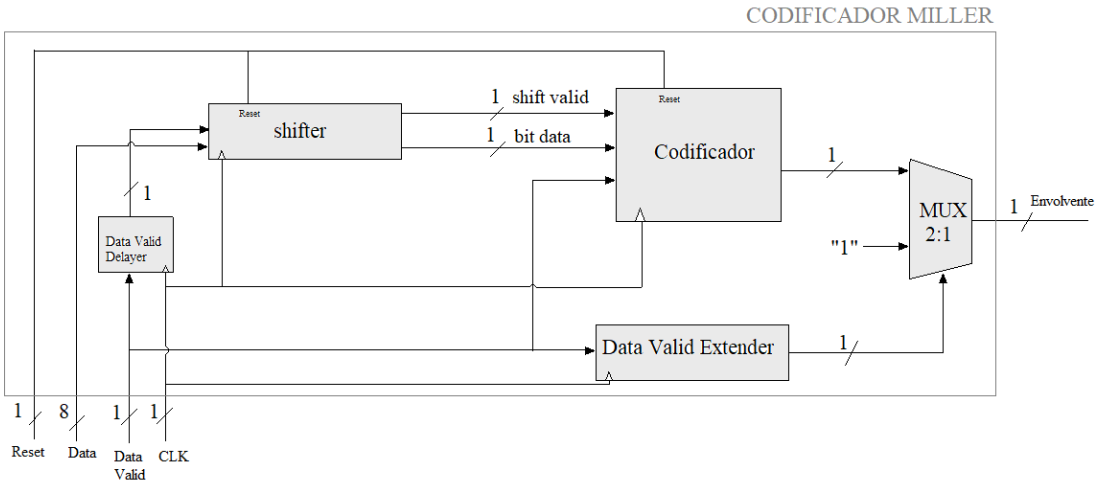

Figura 5.5: Diagrama de bloques del codificador Miller

Cuando no se ha recibido ningún dato por la UART la señal Data Valid se mantiene en baja. Cuando Data Valid indica la disponibilidad de un nuevo dato estableciéndose en alta durante un ciclo de reloj el bloque Data Valid Extender proporciona en su salida una señal que se mantendrá en alta durante el tiempo que dura la codificación de un byte ,es decir; el tiempo que dura la codificación del bit de start, de los 8 bits del comando y del bit de stop; en total unos 94,34 us. Esta señal gobierna la entrada select de un multiplexor de modo que cuando no hay datos que codificar a la salida del multiplexor aparece la entrada 0. La entrada 0 del mux está conectada siempre a un valor lógico "1". Por lo tanto, cuando no hay datos que codificar la salida del módulo completo y por tanto la envolvente de la portadora estará en alta quedando esta con una amplitud máxima.

Este bloque Data Valid Extender se ha incluido puesto que cuando no llegan datos a transmitir desde la UART al codificador le llegan continuamente "0"s a su entrada y por tanto los codificará con la secuencia pertinente detallada en la sección 2.3.1. Si no incluimos el multiplexor controlado por Data Valid Extender estos ceros codificados se modularían en la portadora cuando en realidad no habría llegado ningún dato que se corresponda con dichas secuencias de "0"s.

Durante el tiempo que está en alta Data Valid Extender el multiplexor tendrá a su salida la entra 1, es decir, tendrá a su salida los bits codificados que conforman la envolvente de la portadora modulada con la codificación Miller del comando que ha llegado por la línea de recepción asíncrona.

Cuando llega un dato, este se registra en un desplazador que va entregando los bits al codificador con una tasa del tiempo de bit de 106 KHz. Cada vez que hay un bit que entregar al decodificador el flag shift valid se pone en alta durante un ciclo del reloj del sistema. El desplazador conoce la existencia de un carácter disponible para transmitir gracias al flag de entrada Data Valid. Sin embargo, como hay que codificar primeramente el bit de start se incluye un retardo en esta señal que le llega al desplazador para que, cuando al codificador le llegue el flag Data Valid, inmediatamente codifique un bit de start y un tiempo de bit más tarde codifique la secuencia que le llega desde el desplazador. Si no se incluyese este retardo en Data Valid el codificador entregaría directamente en su

salida el byte codificado sin el bit de start. El funcionamiento del codificador se expone con mayor claridad en el apartado 6.2 donde la visualización de las señales implicadas aporta una visión esclarecedora de la codificación.

El bloque codificador en sí esta implementado como una máquina de estados en la que según sea el valor del bit de entrada a la salida se obtiene la secuencia X, Y o Z ya mencionada anteriormente.

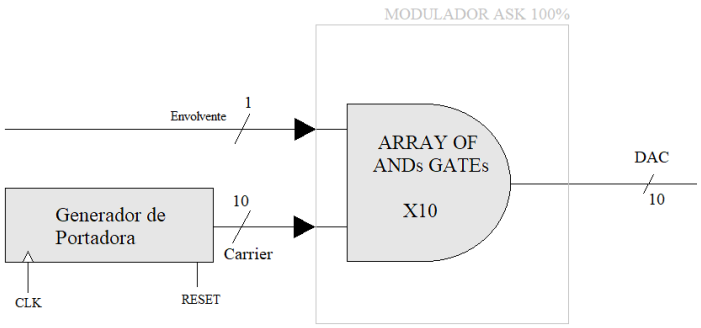

Figura 5.6: Bloques del modulador ASK 100%

Los bloques funcionales del modulador aparecen representados en la figura 5.6. El modulador simplemente consiste en una matriz de puertas lógicas AND, cada una de ellas con dos entradas. En una de esas entradas de cada puerta está presente la envolvente proveniente del codificador. En la otra entrada de cada puerta estará conectado cada bit de la palabra de 10 bits que conforma la portadora. Es decir, en la puerta AND[0] estarán conectadas en sus entradas el bit de la envolvente y el bit Carrier[0]. En la puerta AND[1] estarán conectados en sus entradas el bit de la envolvente y el bit Carrier[1] y así sucesivamente.

De esta forma, cuando la envolvente está en alta, la salida hacia el DAC es la portadora con su amplitud al 100%. Cuando la envolvente está en baja, la amplitud de salida hacia el DAC será cero.

6.1.4 Generación de la portadora

Para generar la portadora de 13.56MHz se utiliza un contador. Como el reloj del sistema de transmisión es de 54,24MHz se generan cuatro muestras por ciclo de portadora. Por tanto, se utilizará un contador de módulo 4 de modo que a medida que se incrementan las cuentas el valor de la portadora será de 1023, 512, 0, 512 y así sucesivamente. La portadora conformada consta de 3 niveles diferentes tal y como se representa en la figura 5.7. Esto facilita las cosas al DAC puesto que no necesita de un slew rate más pronunciado además de generar una señal cuadrada con menos armónicos obteniéndose un tono fundamental de 13.56 MHz menos distorsionado.

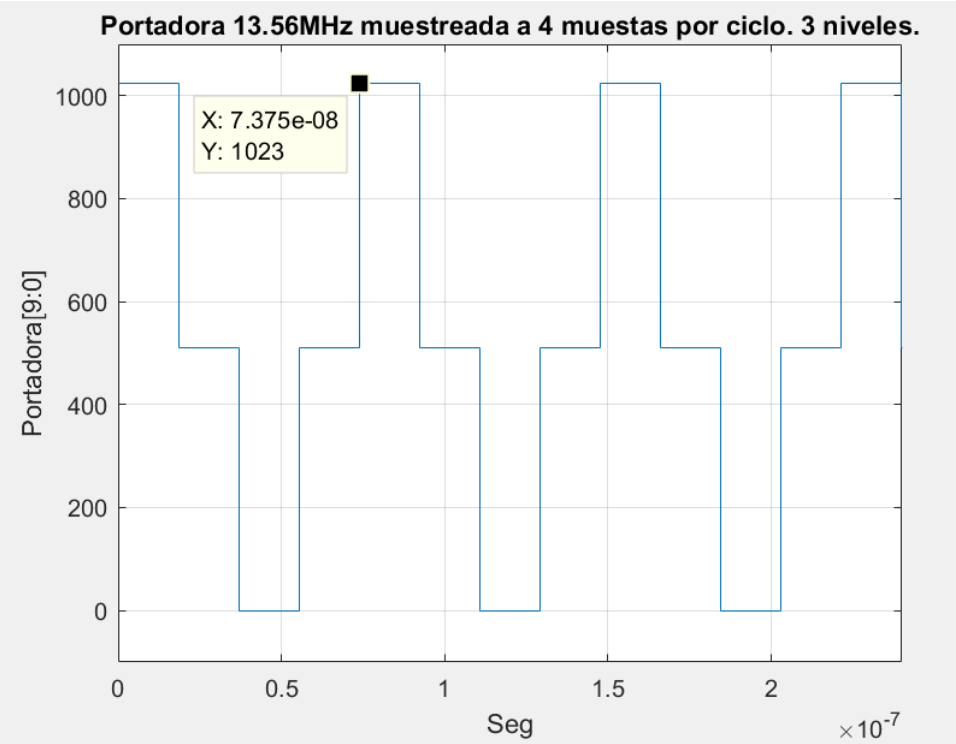

Figura 5.7: Muestras de la portadora generada

6.2 Simulación del Transmisor

Para constatar el correcto funcionamiento de los módulos del transmisor diseñados en Verilog se ha efectuado una serie de simulaciones que permiten visualizar la forma de las señales a modo de cronograma mediante el programa Gtkwave, recogidas en la figura 5.8.

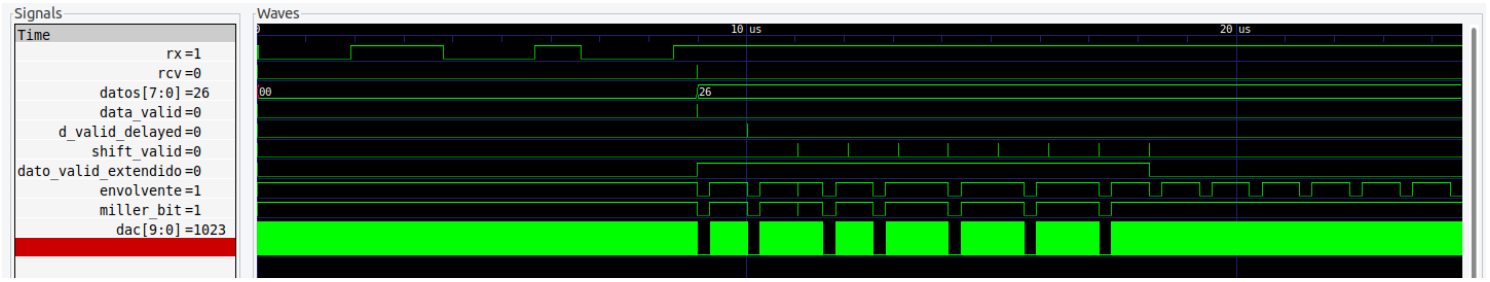

Figura 5.8: Simulación del transmisor

La primera señal representada se corresponde con la línea de recepción asíncrona de la UART. Los bits representados son el bit de start seguido del carácter 0x26 en hexadecimal, correspondiente con el comando REQA además del bit de stop. Hay que recordar que primero se transmiten los bits menos significativos. La tasa de transmisión es de 115200 baudios. Se puede apreciar cómo la señal rcv se avisa durante un ciclo de reloj de 54,24 MHzde la disponibilidad del dato recibido.

Hasta que el dato 0x26 no se ha acabado de recibir la señal dato_valid_extendido permanece en un nivel bajo y por tanto, la salida del codificador (Señal miller_bit en la figura 5.8) permanece en alta propiciando que la amplitud de la portadora sea del 100%. Debido a la escala de tiempos no se aprecian los valores de la portadora en la simulación. Sin embargo, estos se corresponden literalmente con los expuestos en la figura 5.7.

Cuando data_valid alerta de la llegada del carácter, el codificador Miller codifica el bit de start (Antes de la llegada de los bits del comando provenientes del desplazador) con la secuencia Z en la cual, la envolvente permanece en baja durante el primer cuarto del tiempo de bit de 106KHz.

A medida que shift_valid va dando paso a los bits del desplazador la máquina de estados del decodificador decide qué secuencias aplicar según el valor de los bits que van entrando. Durante todo este tiempo de codificación del carácter la señal data_valid_extendido permanece en alta propiciando que miller_bit adquiera el valor de la señal envolvente. Puede apreciarse que cuando se termina la transmisión del carácter completo la señal envolvente sigue codificada con la secuencia correspondiente a los bits de valor “0” que le llegan desde el desplazador puesto que este ya se ha vaciado, pero aun así sigue entregando valores. Se puede apreciar en esta situación la utilidad de la señal data_valid_extended que evita que estos “0”s residuales modulen la portadora.

En la simulación no se tiene en cuenta el efecto amortiguador de la antena cuando la portadora adquiere una amplitud de cero, por lo tanto, los cambios en la envolvente final se representan de forma brusca.

6.3 Receptor

El receptor tiene la tarea de sintonizar la banda lateral superior de 14,4075 MHz con la información del PICC trasladándola a banda base. Una vez se ha efectuado dicha traslación, se demodula la señal OOK para obtener la codificación Manchester. El decodificador le entrega los datos a un bloque que desensambla la trama recibida, comprobando la paridad. Si esta comprobación es correcta, cada byte recibido se envía a una UART de transmisión y esta se encarga de enviar los datos al PC. En la figura 5.9 se presenta el diagrama de bloques del sistema receptor.

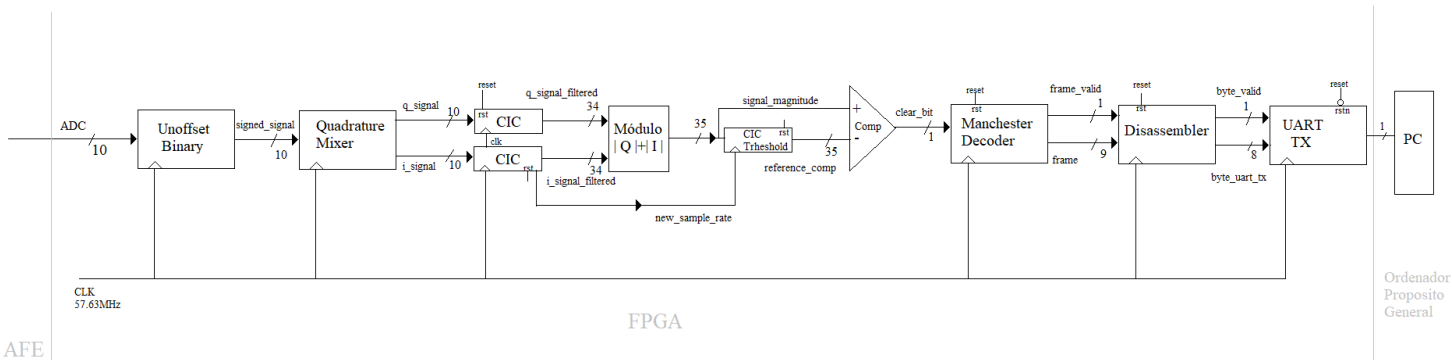

Figura 5.9: Diagrama de bloques del sistema receptor

6.3.1 Generación del reloj

El reloj que emplean todos los bloques del transmisor es de 4*14,4075MHz (57,63MHz) proveniente del PLL interno de la FPGA cuya frecuencia de referencia son los 27,12MHz del oscilador del AFE.

Por tanto, todas las acciones síncronas tendrás como referencia los flancos positivos de dicha frecuencia. Se ha elegido una frecuencia de reloj que sea 4 veces la de la subportadora para facilitar la sintonización de la misma, tal y como se expone en el apartado 6.3.3. Los contadores internos que utiliza el decodificador se incrementan hasta 544 cuentas (136 ciclos de la banda lateral superior * 4 muestras por ciclo) para determinar la duración de un tiempo de bit de 106 KHz.

La UART TX también genera los baudios de la comunicación a partir del reloj de 57,63MHZ. Para ello emplea un divisor de 250 logrando un baudrate de 230400 Hz.

6.3.2 Binary Unoffset

El primer bloque de la cadena de recepción tiene como fin eliminar el nivel de continua de la señal de entrada. Los datos muestreados por el ADC no tienen signo (Son siempre positivos) y poseen un valor entre 0 y 1023, es decir, incluyen un offset binario.

Al eliminar el nivel de continua o offset, los datos quedarían representados con valores tanto positivos como negativos en formato complemento a dos, oscilando entre -512 y 511 (Siempre y cuando se aproveche todo el rango de entrada del ADC). Para convertir el formato binary offset en complemento a dos basta con invertir el bit más significativo de las muestras de entrada. En la tabla 3.1 se muestra un ejemplo de dicha operación que esclarece también por qué los valores expresados en complemento a dos no quedan del todo simétricos.

Decimal	Offset Binario	Complemento a dos
7	1111	0111
6	1110	0110
5	1101	0101
4	1100	0100
3	1011	0011
2	1010	0010
1	1001	0001
0	1000	0000
−1	0111	1111
−2	0110	1110
−3	0101	1101
−4	0100	1100
−5	0011	1011
−6	0010	1010
−7	0001	1001
−8	0000	1000

Tabla 3.1: Formato offset binary frente complemento a dos

6.3.3 Mezclador de cuadratura

El mezclador desgrana la señal de entrada en su componente en fase y en cuadratura además de aplicar un proceso de heterodinación a la señal, desplazándola así a banda base (Véase el apartado 3.1). Para obtener la componente Q e I de la señal hay que multiplicarla por dos sinusoides desfasadas entre sí 90º, es decir; por un coseno y un seno. Si además la frecuencia de dichas sinusoides es la misma que la frecuencia de la señal de entrada entonces obtendremos a la salida una señal cuya frecuencia es del doble de la original y una señal cuya frecuencia es cero, es decir; un nivel de continua que representa la amplitud de la componente Q y la amplitud de la componente I.

Puesto que la información que deseamos recuperar está en la amplitud de la señal trasladada a banda base se debe filtrar la señal cuya frecuencia es del doble de la de entrada además del resto de armónicos que sean podido generar en el proceso de mezclado.

Son dos los problemas principales que se plantean en el mezclador: la generación de las sinusoides locales y la multiplicación de las muestras de las sinusoides con las muestras de entrada.

Para generar las sinusoides se puede optar por mecanismos como la síntesis digital directa DDS. Sin embargo, con ánimo de simplificar los requisitos computacionales del sistema se ha optado por un método más sencillo que también soluciona el problema de la multiplicación entre muestras.

El método consiste en muestrear la señal de entrada con una frecuencia que sea 4 veces la frecuencia de la señal de interés. De esta forma la señal de entrada tendrá 4 muestras por ciclo y, por tanto, se tendrá que generar una oscilación local que también cuente con 4 muestras por ciclo y de la misma frecuencia. Para simplificar aun más las cosas, los valores de las muestras de las sinusoides locales tienen únicamente dos niveles de amplitud normalizada, esto es, 1 y -1. De esta forma las muestras de entrada se multiplican por 1 (O lo que es lo mismo, se dejan tal y como están) o por -1 (O lo que es lo mismo, únicamente se cambia su signo). De esta manera se puede apreciar cómo se prescinde de la operación multiplicación.

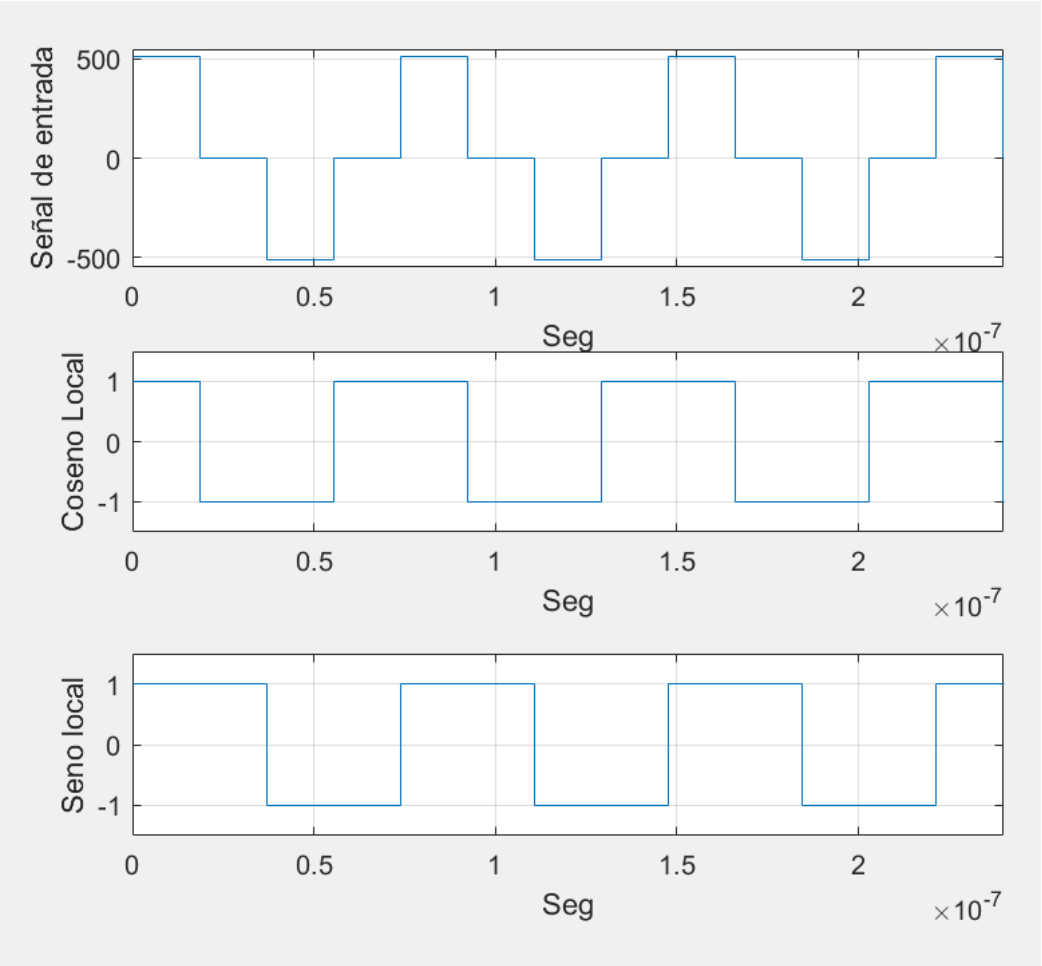

Figura 5.10: Muestras ideales de la señal d eentrada junto con las sinusoides locales desfasadas 90ª

Los valores que toma periódicamente el coseno son 1, -1, -1, 1, etc. y los valores que toma el seno son 1, 1, -1, -1, etc. El módulo que implementa el mezclador cuenta con un contador de módulo 4. Cada valor que va adquiriendo el contador determina los valores expuestos del seno y el coseno. Cuando el contador vale 0 el coseno vale 1 y el seno 1 y por tanto la muestra de entrada se deja tal y como está tanto para la salida Q como I. Cuando el contador vale 1 el coseno adquiere el valor -1 y el seno 1 por lo que la muestra de entrada mantiene su valor en su salida Q e invierte su valor en su salida I. Cuando el contador vale 2 el coseno vale -1 y el seno vale -1 por lo que tanto la salida Q como I invierten el valor de la muestra de entrada. Cuando el contador vale 3 el coseno vale 1 y el seno -1 por lo que la salida Q mantiene el valor de la muestra de entrada y la salida I lo invierte. El contador reinicia la cuenta y el proceso se repite periódicamente.

En la figura 5.10 se puede apreciar la forma de la señal de entrada muestreada junto con las sinusoides locales.

6.3.4 Filtros CIC paso-bajo

Para filtrar las componentes I y Q se emplean los filtros pasa bajos decimadores CIC expuestos en el apartado 4. Se han confeccionado unos filtros de orden 4 y decimación 64. Para determinar los bits necesarios para las variables del filtro se recurre a la ecuación 5.1 donde N son los bits de la variable de entrada (10 bits), L el orden del filtro y M la decimación aplicada.

$$B = N + L * \log_2 M \tag{5.1}$$

El filtro diseñado necesita unas variables de 34 bits. La ganancia del filtro es de 64^4, es decir; de 16777216. Consta de 4 etapas integradoras y 4 derivadoras, obteniéndose una muestra de cada 64. La frecuencia de muestreo se ve reducida a 900,469 KHz. La apariencia del filtro se observa en la figura 5.11.

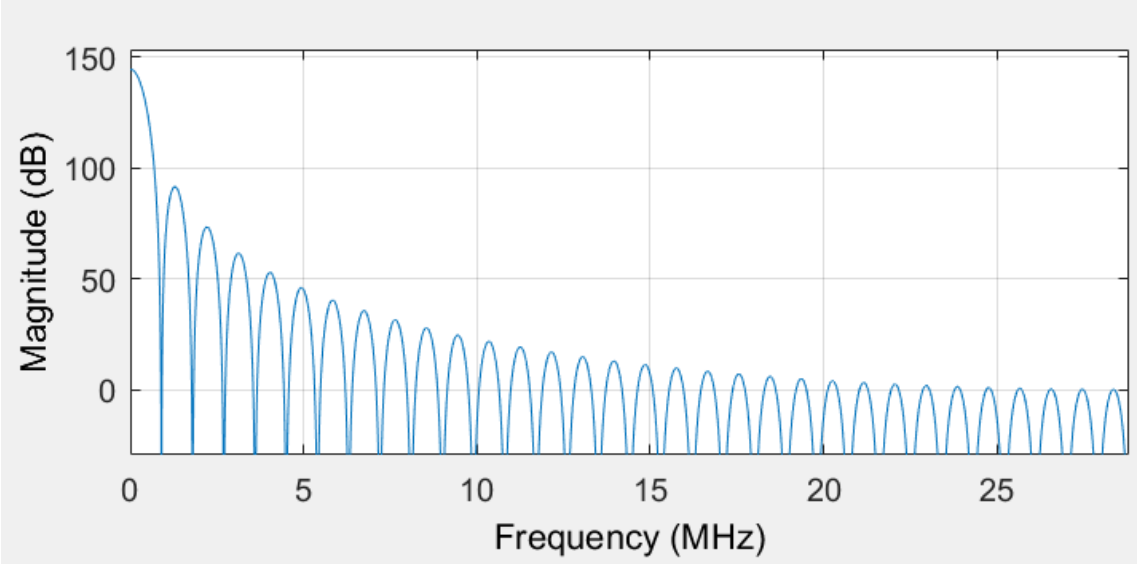

Figura 5.11: Filtro CIC de orden 4 y decimación 64

El primer notch del filtro aparece a 900,469 KHz y los subsiguientes se sitúan en múltiplos de dicha frecuencia, eliminando la frecuencia de la portadora, de la banda lateral superior y de muestreo, entre otras. Un diseño más preciso requiere de una decimación de 68 dado que para la frecuencia de muestreo inicial el primer notch se situaría en 847,5KHz que es justo la frecuencia de la subportadora y sus múltiplos coinciden con la portadora, la tasa de transmisión de los datos y demás armónicos que se generan tras el mezclado. Sin embargo, se optó finalmente por emplear una decimación que fuese una potencia exacta de 2 puesto que los resultados en la simulación de la cadena receptora y en el prototipo real resultaron mejores con la decimación de 64. La elección de un filtro de orden 4 y decimación 64 no es fruto del azar puesto que se han realizado una gran cantidad de pruebas para optimizar el mejor orden del filtro con una decimación no demasiado elevada para que las variables del mismo no resultasen demasiado elevadas.

6.3.5 Demodulador

Para demodular la señal OOK hay que extraer la amplitud de las señales I y Q, es decir; su magnitud, por lo que hay que obtener el módulo del fasor tal y como se exponía en la ecuación 3.1. Sin embargo, el cálculo del cuadrado de cada componente junto con la raíz cuadrada de su suma implica un esfuerzo computacional a la FPGA que es mejor evitar en la medida de lo posible.

Para obtener la magnitud de I y Q simplemente se ha optado por obtener el valor absoluto de cada componente y sumar ambos: $|Q| + |I|$.

Sin embargo, esta magnitud de la señal de entrada está representada mediante una palabra de 35 bits (34 bits de cada componente I y Q que se extiende a 35 bits puesto que la suma puede desbordar los 34 bits) con presencia de ruido. Esto provoca que el ancho de los pulsos demodulados sea demasiado irregular además de que es necesaria una envolvente de un único bit que simplemente refleje un estado de alta cuando hay modulación presente y de baja cuando no hay ninguna modulación en la amplitud de la subportadora.

Para transformar esa magnitud en un único bit totalmente “limpio” se utiliza un comparador de forma que, si la magnitud de la señal supera un cierto valor umbral empleado como referencia, el bit de salida se pone en alta. Si la magnitud está por debajo del umbral el bit se establece en baja. Esto supone un problema: se debe establecer el valor umbral que sirve como referencia al comparador de forma dinámica en función del valor máximo y mínimo que posee la magnitud demodulada de la señal de entrada en cada instante puesto que si se establece de forma estática habría ocasiones que, aunque la demodulación sea correcta, la magnitud de entrada no sería lo suficientemente elevada como para sobrepasar el umbral estático o por el contrario habría ocasiones en que tendría un valor demasiado elevado y la salida siempre estaría en alta. Un umbral estático también afectaría al correcto ancho de los pulsos codificados en Manchester.

Para poder establecer el umbral dinámico hay que efectuar una media dinámica de los valores de la envolvente entregada por el módulo que calcula la magnitud. Esta tarea la realiza un filtro pasa bajos puesto que el nivel de continua de los pulsos que conforman la magnitud de la señal de entrada será dicho valor medio. Para implementar este filtro también se emplea un CIC decimador. Sin embargo, en esta ocasión, el filtro debe de tener un ancho de banda más restrictivo y lo que es más importante aún: su ganancia debe de estar normalizada para no adulterar el valor de referencia entregado al comparador puesto que si no este no se correspondería con el valor medio de la magnitud de la señal.

El CIC utilizado cuenta con un orden de 4 y una decimación de 256 lo que supone una ganancia de 256^4. Por tanto, para normalizar la ganancia habría que dividir el valor devuelto por el filtro por 256 ^4, o lo que es lo mismo, desplazar los bits del valor de salida a la derecha un número de $\log_2(256^4)=32$ posiciones (Desplazar a la derecha un número binario equivale a dividirlo por potencias de dos). En la figura 5.12 aparece representado el CIC utilizado para el cálculo del umbral dinámico que sirve de referencia al comparador. Hay que tener en cuenta que ahora la frecuencia de muestreo ya no es de 57,63 MHz sino de 900,469 KHz situándose el primer notch en 3,5 KHz.

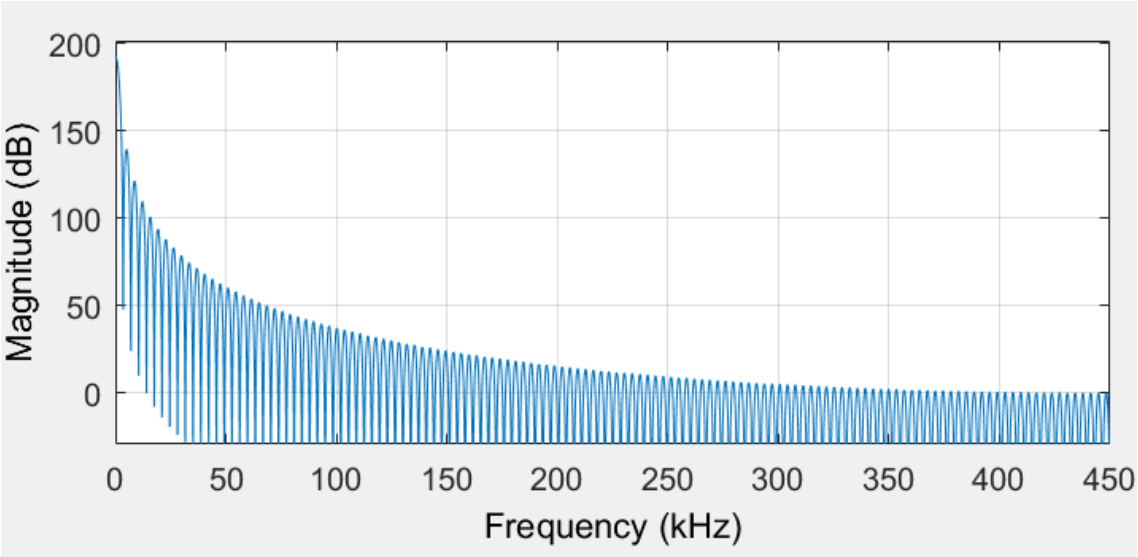

Figura 5.12: Filtro CIC de orden 4 y decimación 256

6.3.6 Decodificador Manchester

El decodificador convierte los pulsos de bits codificados en Manchester en una palabra de 8 bytes + 1 bit de paridad. Hay que recordar que la trama recibida por el PICC está conformada por un bit de start, que sirve para sincronizar la tasa de bit de 106 KHz, paquetes de 8 bits de información + 1 bit de paridad impar por cada byte transmitido y un bit que indica el final de la comunicación manteniéndose en baja durante todo el tiempo de bit.

El decodificador comprueba con una frecuencia de 57,63 MHz el valor del bit en crudo de entrada y lo comprueba con el valor del bit en el instante anterior. Cuando no hay ninguna respuesta presente el bit de entrada permanece en baja. De esta forma, cuando llega el bit de start el decodificador denota el inicio de la respuesta puesto que el valor anterior del bit será 0 y el valor actual 1. En este momento se produce la sincronización iniciando el receptor un contador para determinar el tiempo de bit de 106 KHz. Un tiempo de bit de 9,43 us se corresponde con 544 ciclos del reloj del sistema de recepción, es decir; 544 cuentas del contador de sincronización.

El decodificador, en cada tiempo de bit, determina el valor del bit transmitido comprobando el valor del bit de entrada en la mitad de cada mitad del tiempo de bit. Es decir, el decodificador comprueba el valor de la entrada en 1/4 de Tbit y en 3/4 de Tbit tal y como se representa en la figura 5.13. La cuenta de sincronización para 1/4 de Tbit se corresponde con 135 y para 3/4 de Tbit se corresponde con 407 cuentas. Si el valor en la primera mitad del tiempo de bit es 1 y en la segunda mitad es 0, entonces el valor recibido es un 1 lógico. Si el valor del bit de entrada en la primera mitad es 0 y en la segunda mitad es 1, entonces el bit transmitido por el PICC es un 0 lógico.

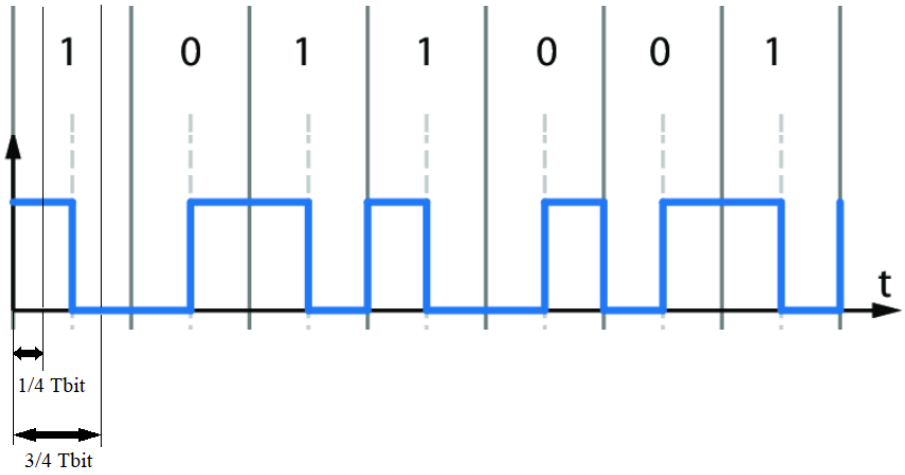

Figura 5.13: Tiempos de bit en la codificación Manchester

El decodificador va depositando los bits decodificados en un registro de 9 bits, empezando por su parte LSB puesto que primero se trasmiten los LSB. Cuando se han recibido los 9 bits (1 byte + 1 bit paridad) se emite un flag de un ciclo de reloj de duración para indicar al bloque posterior de la disponibilidad de un nuevo dato decodificado.

Es necesario comprobar el valor del bit de entrada en las dos mitades del tiempo de bit (Realmente en otras condiciones valdría con comprobarlo únicamente en una de las dos mitades) puesto que la única forma de saber que la respuesta ha llegado a su fin es comprobar que en ambas mitades del Tbit el bit de entrada permanece en baja tal y como dicta la secuencia de end of frame. Si en todo el Tbit la entrada permanece en baja se produce una violación del código Manchester. Esta violación es precisamente la que utiliza el PICC para indicar que su respuesta ha llegado al final. A lo largo de toda la decodificación también se comprueba si se producen violaciones del código Manchester como por ejemplo que durante todo el T bit la entrada permanezca en alta, es decir, ambas mitades en 1 en cuyo caso se indicaría de la incidencia con un flag de error.

6.3.7 Desensamblador

A los datos proporcionados por el decodificador se le deben aplicar una comprobación de errores basada en un bit de paridad impar. De la palabra de 9 bits recibida, el bit MSB es el que indica la paridad. Para comprobarla se suman los bits de valor 1. Si el resultado es impar, entonces el bit de paridad recibido debería de ser 0. Si la suma de bits de valor 1 es par, el bit de paridad recibido debería de ser 1.

Si la comprobación resulta correcta, se indica al bloque posterior con un flag en alta durante un ciclo de reloj de la disponibilidad del dato de 1 byte.

6.3.7 UART TX

Los bloques que conforman la UART de transmisión se pueden apreciar en la figura 5.14 [28]. Como entradas cuenta con el reloj de 57,63 MHz, una señal de reset negado (Cuando rstn está en 0 se efectúa un reset síncrono de la unidad de transmisión), la señal de start por la cual se indica a la UART de que un nuevo dato está disponible y el propio dato de 8 bits. Como salidas cuenta con un flag ready que indica la disponibilidad de la UART para seguir emitiendo datos y la salida tx que es la línea de transmisión asíncrona.

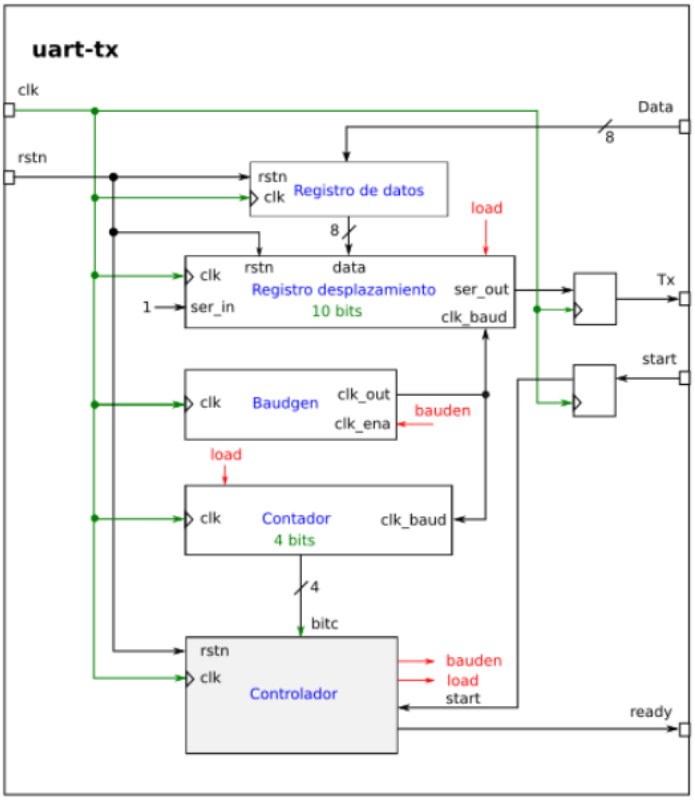

Figura 5.14 Bloques de la UART TX

A la hora de transmitir primero se pone el byte de datos en la entrada y se activa el flag de start. La UART entonces comienza a transmitir el dato por la línea tx mientras la señal ready se establece en 0 indicando que la uart está ocupada. Cuando el carácter se ha enviado el flag ready se vuelve a poner en alta. Hay dos microórdenes que genera el controlador. Con bauden activa el temporizador de bits y con load carga el registro de desplazamiento además de establecer a cero el contador de bits.

6.4 Simulación del Receptor

Para poder comprobar el correcto funcionamiento de los distintos bloques de recepción antes de sintetizarlos en la FPGA se han efectuado una serie de simulaciones. Para poder llevarlas a cabo también se ha simulado la respuesta del PICC mediante el software MatLab. En la figura 5.15 aparece representada la respuesta del PICC.

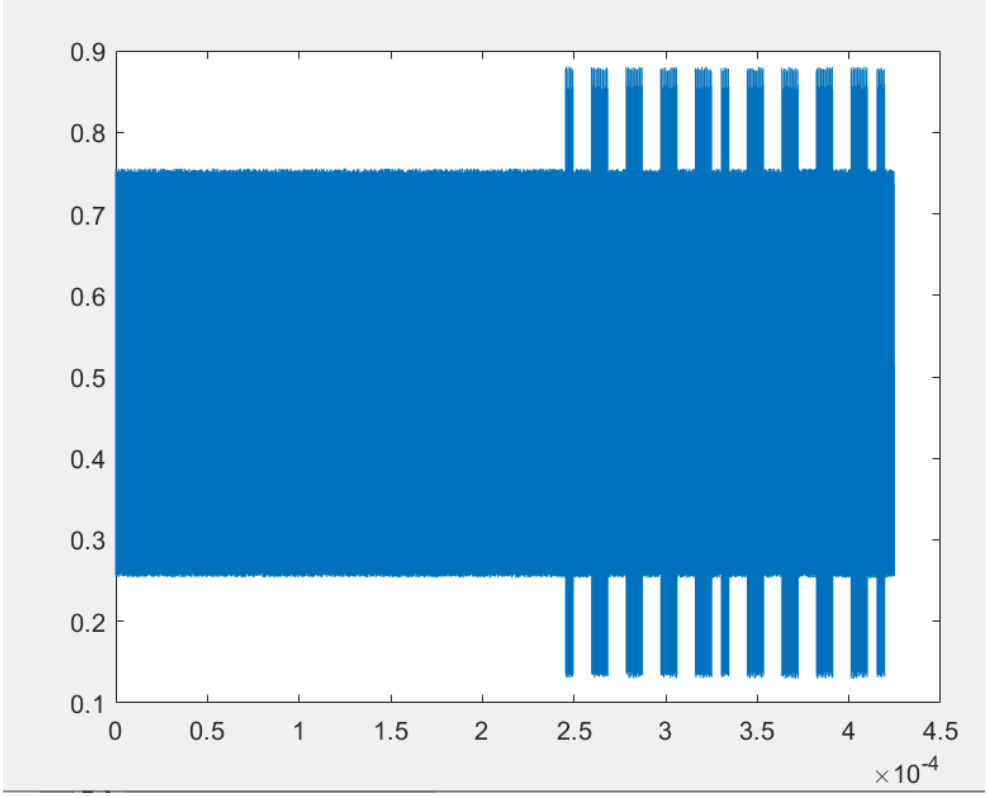

Figura 5.15: Respuesta del PICC generada mediante MatLab para la simulación del receptor

En la figura 5.15 se ha representado la respuesta de la tarjeta con una amplitud exagerada para que pueda apreciarse mejor. Además, la amplitud aparece normalizada y en el eje x se representa el dominio del tiempo en segundos. A la señal también se le ha añadido un nivel de ruido de -45 dB respecto del nivel de señal de la portadora. Las muestras generadas para la simulación son de 10 bits y el nivel de la señal de las bandas laterales es del 1% respecto al nivel de señal de la portadora. La trama que envía el PICC en la simulación es de 2 byte + 1 bit de paridad por byte y se trata del carácter 0xAAAA, por lo que cada byte quedaría como 0x1AA teniendo en cuenta el bit de paridad impar.

A las muestras de la señal generada con MatLab se le aplica toda la cadena de recepción. Los resultados de la simulación pueden observarse en la figura 5.16.

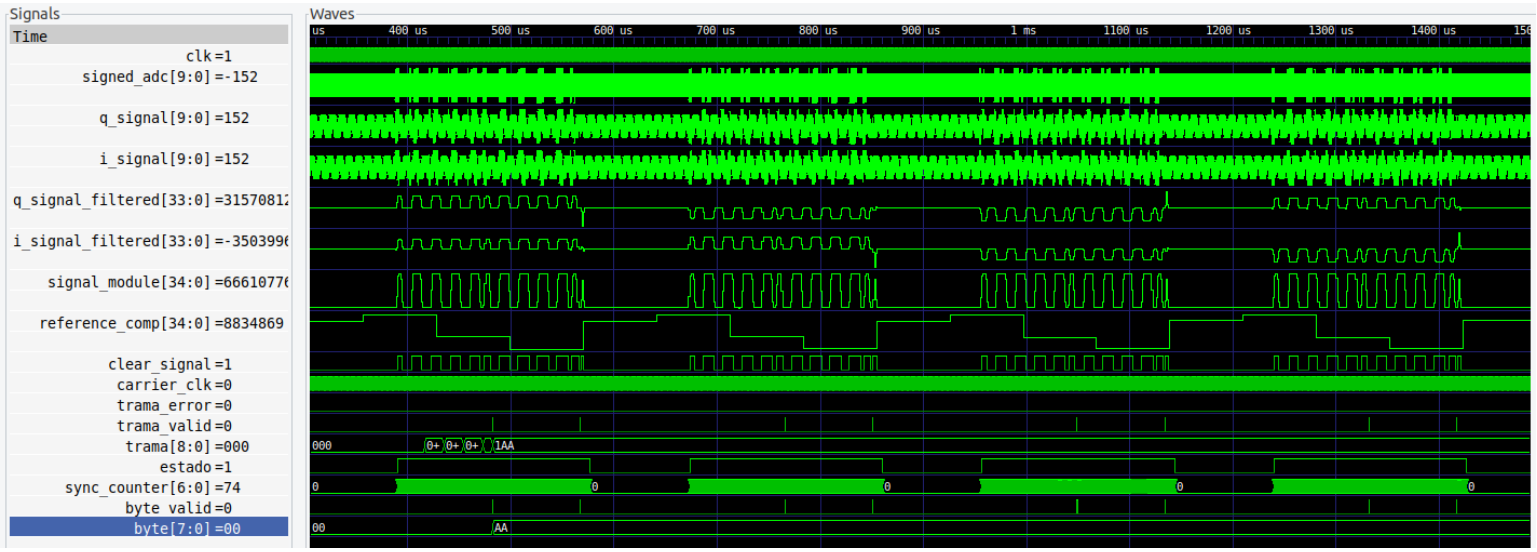

Figura 5.16: Simulación de la cadena receptora

La señal signed_adc[9:0] se corresponde con las muestras proporcionadas por el ADC transformadas a formato de complemento a dos. Esta señal pasa por el mezclador de cuadratura obteniéndose la q_signal[9:0] e i_signal[9:0]. En la figura su apariencia aparece distorsionada y no puede apreciarse claramente las frecuencias obtenidas tras el mezclado. Sin embargo, tras aplicarle el filtrado CIC puede observarse en q_signal_filtered[33:0] e i_signal_filtered[33:0] la envolvente de la componente Q y de la componente I respectivamente. Las envolventes obtenidas pueden tener valores negativos tal y como queda patente en la simulación. Esta es otra de las razones por las que se debe de aplicar un módulo que calcule la magnitud de la señal de entrada, quedando la envolvente representada únicamente con valores positivos. La labor de este módulo puede apreciarse en la señal signal_module[34:0].

Para calcular el umbralque utiliza el comparador como referencia a la señal signal_module[34:0] se le aplica el oro filtrado pasa bajos CIC de modo que la referencia del comparador es el valor medio de las muestras del módulo. Este valor umbral se corresponde con la señal reference_comp[34:0]. Por desgracia, no se ha podido representar el valor de esta señal a escala con el resto de las señales por lo que su apariencia en la simulación es desproporcionada.

Una vez que la envolvente irregular proveniente de signal_module[34:0] pasa por el comparador, se obtiene una serie de pulsos de 1 bit representados en la señal clear_signal. Puede apreciarse la presencia de un glitch espurio al final de la respuesta del PICC. Este glitch es fruto de la simulación puesto que la ausencia de la subportadora se produce de forma brusca cuando en la realidad este cambio se produce de forma amortiguada por los efectos del factor de calidad Q del circuito resonante que conforma la antena. En la implementación real este glitch no llega a tener lugar.

Tras obtener la demodulación se procede a aplicar la decodificación. La salida del decodificador Manchester puede apreciarse en la señal trama[8:0]. Esta señal cuenta con el byte de datos transmitidos por el PICC de 0xAA y con el bit de paridad 1, por lo tanto, el dato entregado al bloque que comprueba la paridad es de 0x1AA. También puede apreciarse el flag trama_valid que indica de la disponibilidad del dato decodificado.

Finalmente, el módulo desensamblador comprueba la paridad y le entrega el byte 0xAA a la UART de transmisión, alertándola de la validez del byte mediante el flag byte_valid.

7. Herramientas utilizadas

La simulación de la cadena de filtrado de la parte de recepción del AFE expuesta en la figura 4.6 se ha llevado a cabo mediante el softwate Proteus.

Toda la captura esquemática, así como la confección del layout de la PCB , generación de gerber files y la BOM se ha llevado a cabo mediante el software de diseño electrónico Altium, con su licencia para estudiantes.

La simulación del comportamiento del circuito resonante que conforma la antena se ha efectuado con el programa RFSIM muy útil a la hora de confeccionar diseños de filtros y redes de adaptación de impedancias para circuitos de radio frecuencia. La simulación de la figura 4.10 está efectuada con dicho programa.

Para el diseño y confección de la respuesta en frecuencia de los filtros decimadores se ha utilizado MatLab. También se ha utilizado dicho entorno para la generación y representación de las muestras que conforman la respuesta del PICC así como la señal generada por el PCD. MatLab ha resultado de gran utilidad a la hora de poder simular el comportamiento de la SDR antes de llevar a cabo su síntesis en la FPGA.

La compilación de los archivos Verilog para su simulación se ha efectuado con Icarus Verilog y la visualización de las señales generadas se ha realizado mediante GTKWave.

Las herramientas empleadas para la síntesis y rutado del diseño en Verilog forman parte del proyecto de software libre IceStorm. Dicho proyecto permite cargar el bitstream generado en las FPGAs de la familia Laticce ICE40 HX sin tener que utilizar las herramientas oficiales del fabricante. La síntesis es generada mediante el programa yosys mientras que el place & route se lleva a cabo mediante el programa nextpnr. Estas herramientas solo trabajan mediante líneas de comandos. Para cargar el archivo .bin en la FPGA se emplea el programa IceLoad confeccionado por Jesús Arias. Todas estas herramientas se ejecutan en conjunto mediante un archivo makefile. Todo el desarrollo del SDR se ha efectuado en Linux.

La placa de desarrollo que contiene el FPGA Laticce ICE40 HX4K también ha sido confeccionada por Jesús Arias. La motivación principal de crear dicha placa de desarrollo radica en la disponibilidad completa de todos los pines de la FPGA puesto que hay varios modelos comerciales que ocupan la mayoría de sus pines con periféricos que en muchas ocasiones lo único que hacen es estorbar. Dicha placa de desarrollo se denomina ICECREAM.

La soldadura de los componentes y el testeo de los mismos se ha llevado a cabo con las herramientas del laboratorio de electrónica de la universidad: soldadores, microscopio, multímetro, etc. La visualización de las señales hubiese sido imposible sin la disponibilidad de un osciloscopio. Los osciloscopios empleados han sido un Tektronix TDS 320 de 100 MHz y 500 Ms/seg y un Siglent SDS115CM de 100 MHz y 1Gs/s.

8. Resultados

Para realizar las pruebas de lectura con el dispositivo se ha empleado una tarjeta MIFARE Classic MF1S503yX que cuenta con 4 bytes para el UID. En su datasheet se puede obtener el valor de la respuesta ATQA, que en este caso es de 0x0004, tal y como se muestra en la figura 6.1.

Table 11. ATQA response of the MF1S50yyX/V1

		Bit Number															
Sales Type	Hex Value	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
MF1S500yX	00 44h	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0
MF1S503yX	00 04h	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
MF1S700yX	00 42_h	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0
MF1S703yX	00 02_h	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Figura 6.1: ATQA para la tarjeta MIFARE Classic utilizada en las pruebas

Para iniciar la comunicación, enviamos el comando REQA=0x26 al PICC. En la figura 6.2 puede apreciarse la portadora modulada con dicho comando. La señal superior es el dato recibido por la línea de recepción asíncrona. Hay que recordar que en la línea RX de la UART primero se transmiten los bits LSB.

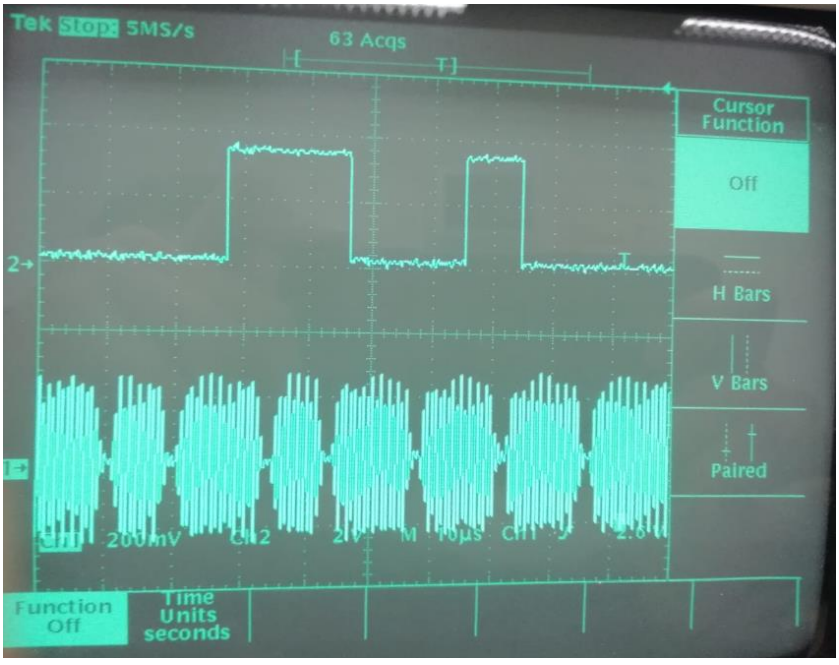

Figura 6.2: Carácter REQA=0x26 en la línea de recepción asíncrona y modulado en la portadora

En la figura 6.3 puede apreciarse la forma de la envolvente codificada en Miller modificado. La señal superior es el flag que indica la duración del carácter que modula la portadora, sin contar el bit de end of frame. Puede apreciarse que la duración es de 84,87us. La apariencia de la señal es la esperada tal y como se expuso en la figura 1.10.

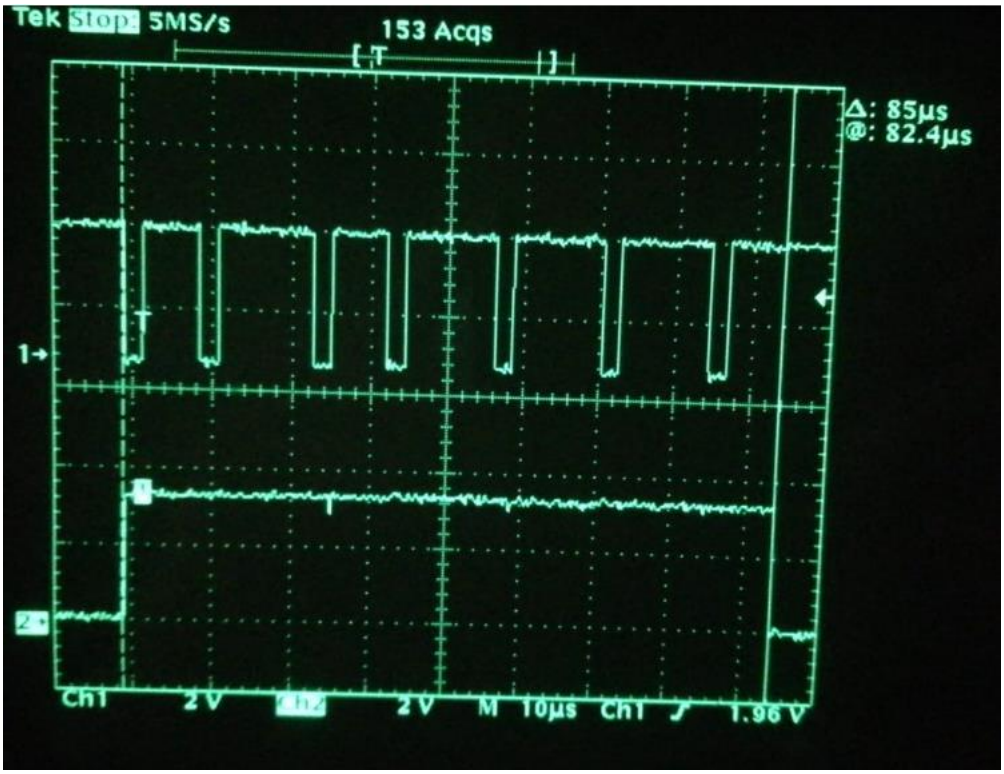

Figura 6.3: Envolvente de la modulación Miller junto con la duración del carácter transmitido

La respuesta del PICC se muestra en la figura 6.4. Puede comprobarse que el carácter recibido se corresponde con el comando 0x0004 (Hay que tener en cuenta que primero se recibe el bit LSB) que es justo el ATQA de nuestra tarjeta. Para hacer la comprobación se puede decodificar manualmente la envolvente Manchester demodulada. La trama recibida en binario se corresponde con la secuencia 00100000 0 00000000 1, sin contar el bit de start. El primer byte tiene un bit de paridad impar igual a cero y el segundo byte tiene un bit de paridad igual a 1. La duración de la respuesta del PICC está indicada con el flag que conforma la señal inferior. Se puede confirmar una duración de la respuesta esperada puesto que los 186 us de la misma se corresponden con 20 tiempos de bit (Realmente serían 188,6 us pero el flag que indica la duración de la respuesta se pone en baja inmediatamente al detectar la violación del código Manchester que indica el end of frame y eso se produce en 3/4 del Tbit de dicho end of frame).

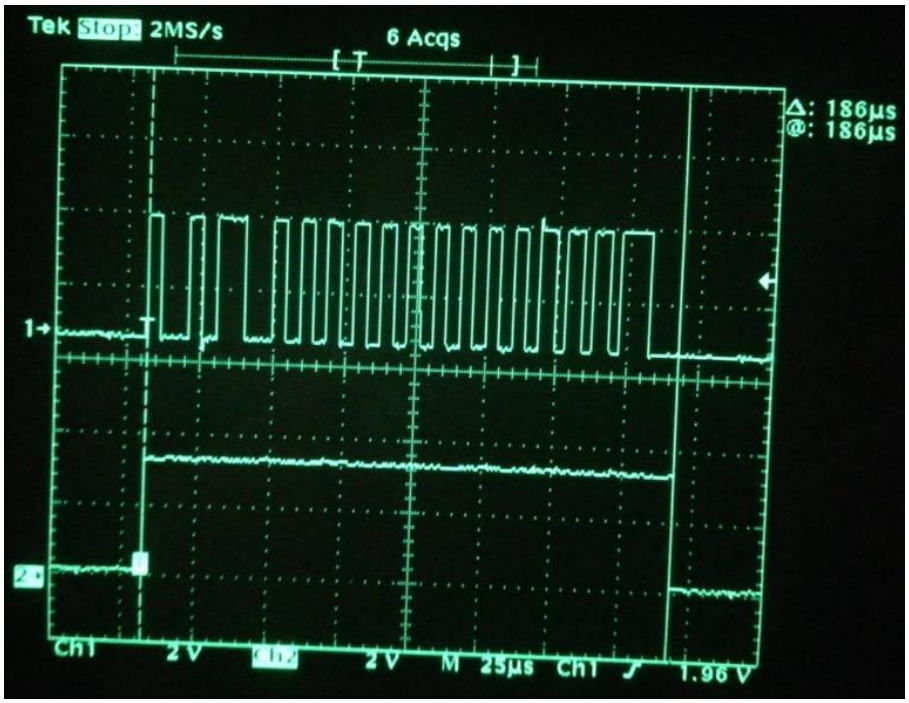

Figura 6.4: Codificación Manchester de la respuesta del PICC

La figura 6.5 representa la envolvente del comando REQA enviado al PICC seguido de la envolvente de la respuesta ATQA. La señal de la parte inferior es el flag que indica el estado de la decodificación Manchester. Puesto que el comando REQA no esta codificado en Manchester, puede apreciarse como el flag se establece en baja y vuelve a un estado alto al intentar decodificarse. Sin embargo, con la respuesta del PICC el flag se mantiene en alta hasta el final puesto que se decodifica satisfactoriamente.

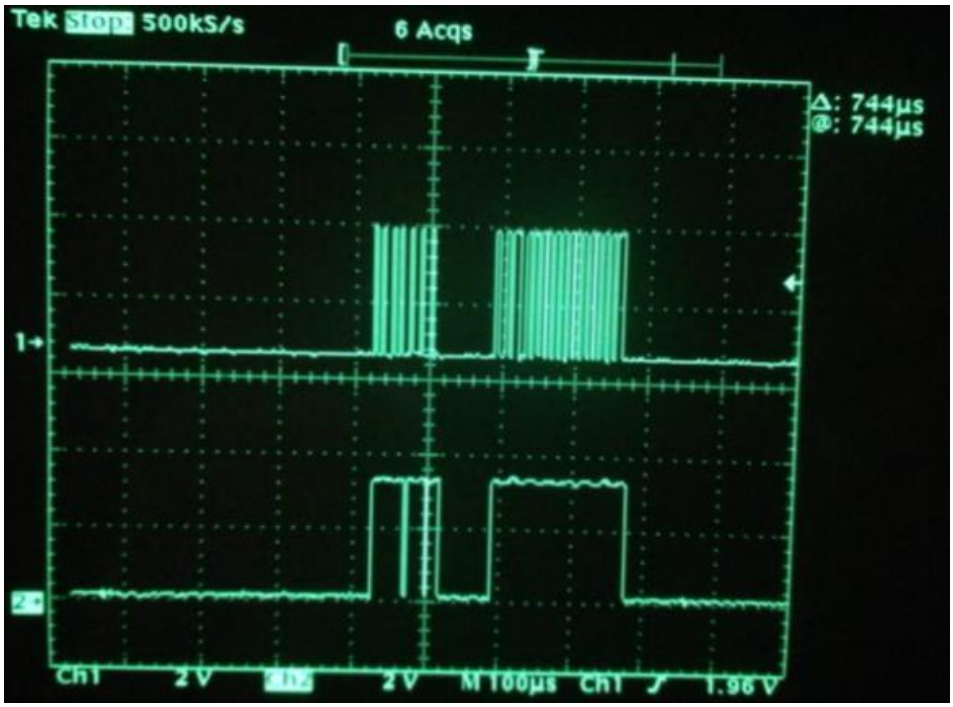

Figura 6.5: Envolvente Miller del REQA seguida de la envolvente Manchester del ATQA

En la figura 6.6 se adjunta una captura de los caracteres emitidos por la UART de transmisión del PCD correspondientes a la respuesta ATQA del PICC. La captura se corresponde con una de las pruebas en las que se envía al transponder de forma periódica el comando REQA, en concreto cada 0,604 ms respetando el tiempo mínimo entre comandos REQA de 7000 ciclos de portadora (0,516 ms).

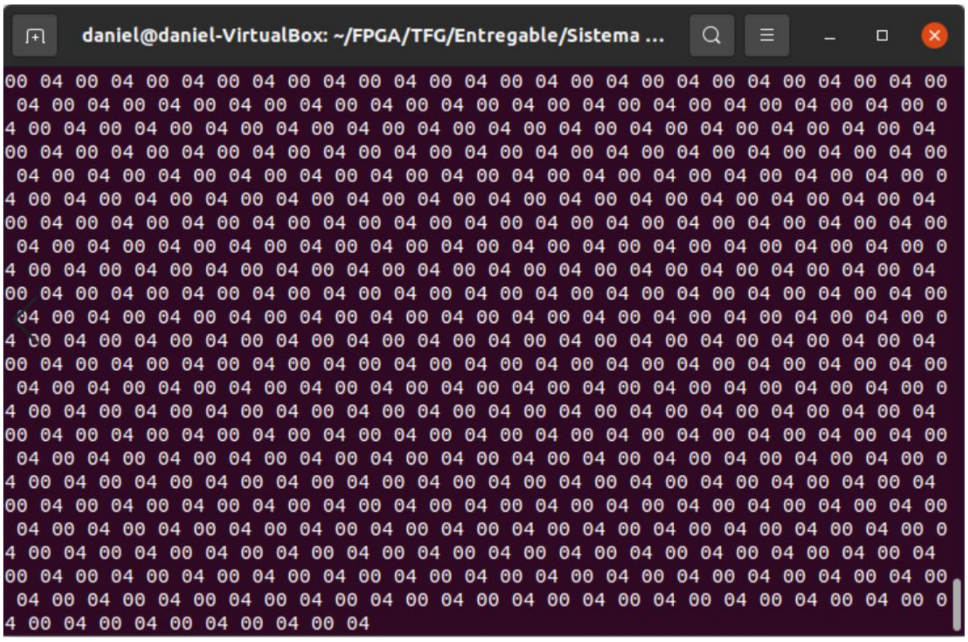

Figura 6.6: Respuesta del PIC emitida por la UART hacia el PC

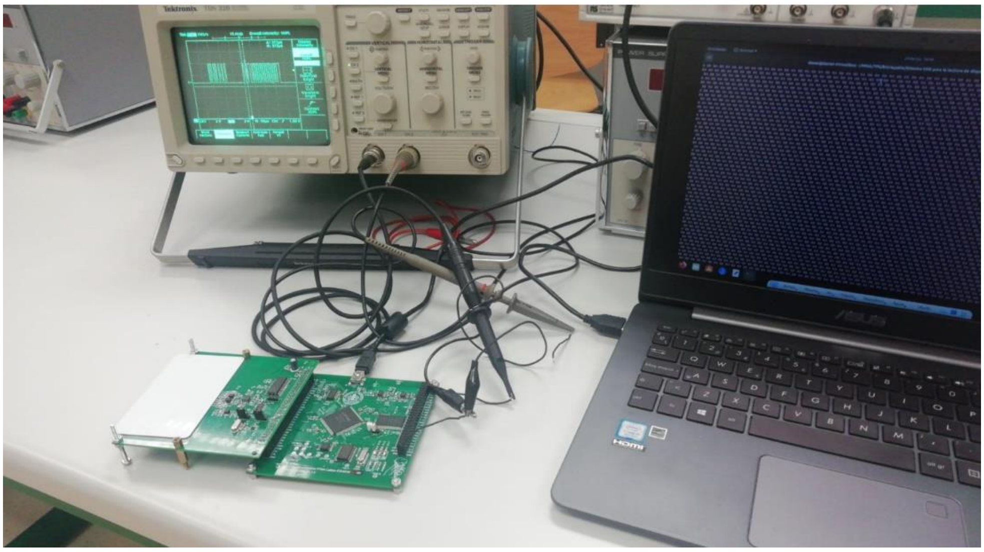

Figura 6.7: Entorno de trabajo en el laboratorio de electrónica

9. Conclusiones y líneas futuras

Este proyecto es el resultado de varios meses de trabajo en los cuales se han puesto en práctica una gran cantidad de conocimientos adquiridos a lo largo del grado, conocimientos de distinta índole referidos a la rama de la electrónica analógica y digital, programación o tratamiento digital de la señal , entre otros. Se puede concluir que el prototipo diseñado cumple con las especificaciones requeridas por los sistemas RFID de la banda HF, concretamente del estándar 14443-A además de constatar la gran versatilidad de los dispositivos de lógica programable como son las FPGAs. La arquitectura implementada del sistema SDR cumple con su cometido, pudiéndose adaptar con ligeros cambios a otras bandas de frecuencia y modulaciones de señal. En cuanto a las líneas futuras, algunas de las mejoras se pueden llevar a cabo son las siguientes :

- Implementación de las capas de alto nivel del estándar ISO 14443-A mediante lenguajes de programación alto nivel en un ordenador de propósito general.
- Implementación de otros protocolos basados en la banda HF como el estándar ISO 14443-B o el estándar NFC.
- En lugar de implementar el rol de lector PCD se podría implementar la funcionalidad de transponder PICC, emulando el comportamiento de la tarjeta.
- Implementación del dispositivo como “husmeador”, sin entrar en juego en la comunicación.
- Implementación de estándares basados en la banda LF (Cambiando el circuito resonante de la antena).
- Implementación del dispositivo como un testeador de tarjetas y lectores RFID, marcado por el estándar ISO 10373-6.

Todas estas propuestas son posibles con el mismo hardware del AFE puesto que la elección de sus componentes permite una gran versatilidad a la hora de generar casi cualquier forma de onda, así como para digitalizar las señales recibidas.

Bibliografía

[1] «A 2.45-GHz RFID tag with on-chip antenna,» [En línea]. Available: https://www.semanticscholar.org/paper/A-2.45-GHz-RFID-tag-with-on-chip-antenna-Yeoh-Choi/cb0c633d00dd650fe48ef35d80c578b82853d954.

[2] «Design and Implementation of a RF Powering Circuit for RFID Tags or Other Batteryless Embedded Devices,» [En línea]. Available: https://www.researchgate.net/publication/264865754_Design_and_Implementation_of_a_RF_Powering_Circuit_for_RFID_Tags_or_Other_Batteryless_Embedded_Devices.

[3] «AN10841 MIFARE Plus Card Coil Design Application note,» [En línea]. Available: https://www.nxp.com/docs/en/application-note/AN10841.pdf.

[4] «A Digital Receiver Architecture for RFID Readers,» [En línea]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4577685&tag=1.

[5] «Digital Demodulator Architecture of a Contactless Reader System for HF RFID Applications Supporting Data Rates up to 13.56 Mbit/sec,» [En línea]. Available: https://diglib.tugraz.at/download.php?id=576a7d790f04a&location=browse.

[6] «ESTUDIO, DISEÑO Y SIMULACIÓN DE UN SISTEMA DE RFID BASADO EN EPC,» [En línea]. Available: https://upcommons.upc.edu/bitstream/handle/2099.1/3552/40883-2.pdf.

[7] «The RF in RFID: physical layer operation of passive UHF tags and readers,» [En línea]. Available: http://www.enigmatic-consulting.com/Communications_articles/RFID/RFID_frequencies.html.

[8] «ISO 14443,» [En línea]. Available: https://www.iso.org/obp/ui/#iso:std:iso-iec:14443:-1:ed-4:v1:en.

[9] «Campo Magnético,» [En línea]. Available: http://www.esi2.us.es/DFA/F1(GIOI)/Apuntes/2011-12/Fisica%20II/5_Campo_magnetico_gioi_1112.pdf.

[10] «A multi-standard analog front-end circuit for 13.56MHz RFID reader,» [En línea]. Available: https://ieeexplore.ieee.org/document/8122314.

[11] «Radio Cognitiva. La radio se vuelve inteligente,» [En línea]. Available: https://www2.coitt.es/res/revistas/04d%20Radio%20cognitiva.pdf.

[12] «Software Defined Radio: Basic Principles and Applications,» [En línea]. Available: https://www.researchgate.net/publication/303253115_Software_Defined_Radio_Basic_Principles_and_Applications.

[13] «Noise temperature, Noise Figure (NF) and noise factor (f),» [En línea]. Available: https://www.satsig.net/noise.htm.

[14] «A Software-Defined Radio for the Masses,» [En línea]. Available: https://www.arrl.org/files/file/Technology/tis/info/pdf/020708qex013.pdf.

[15] «Nyquist Sampling Theorem,» [En línea]. Available: http://www.eecs.umich.edu/courses/eecs206/archive/f02/public/lec/lect20.pdf.

[16] « Micore Reader IC Family; Directly Matched Antenna Design,» [En línea]. Available: http://www.chinaidcard.com/uploadfiles/files/20170512162955_7594.pdf.

[17] «Placa FPGA ICECREAM,» [En línea]. Available: https://www.ele.uva.es/~jesus/ICECREAM/index.html.

[18] «Convertidores ΣΔ para microcontroladores,» [En línea]. Available: https://www.ele.uva.es/~jesus/microdelta.pdf.

[19] «AD9740 Datasheet,» [En línea]. Available: https://www.analog.com/media/en/technical-documentation/data-sheets/AD9740.pdf.

[20] «THS4521 Datasheet,» [En línea]. Available: https://www.ti.com/lit/ds/symlink/ths4521.pdf?ts=1622218311002&ref_url=https%253A%252F%252Fwww.google.es%252F.

[21] «E12 standard values,» [En línea]. Available: https://www.electronics-notes.com/articles/electronic_components/resistors/standard-resistor-values-e-series-e3-e6-e12-e24-e48-e96.php.

[22] «ADC10065 Datasheet,» [En línea]. Available: https://www.ti.com/lit/ds/symlink/adc10065.pdf?ts=1622218469041&ref_url=https%253A%252F%252Fwww.google.es%252F.

[23] «SN74LVC1404 Datasheet,» [En línea]. Available: https://www.ti.com/lit/ds/symlink/sn74lvc1404.pdf.

[24] «Grounding Data Converters and Solving the Mystery of "AGND" and "DGND",» [En línea]. Available: https://www.analog.com/media/en/training-seminars/tutorials/MT-031.pdf.

[25] «Proximity Antennas Application Note Mifare,» [En línea]. Available: https://www.manualslib.com/manual/797710/Mifare-14443a.html.

[26] P. IceStorm. [En línea]. Available: http://www.clifford.at/icestorm/.

[27] C. 2. U. d. r. s. asíncrona. [En línea]. Available: https://github.com/Obijuan/open-fpga-verilog-tutorial/wiki/Cap%C3%ADtulo-25%3A-Unidad-de-recepci%C3%B3n-serie-as%C3%ADncrona.

[28] C. 2. U. d. t. s. asíncrona. [En línea]. Available: https://github.com/Obijuan/open-fpga-verilog-tutorial/wiki/Cap%C3%ADtulo-24%3A-Unidad-de-transmisi%C3%B3n-serie-as%C3%ADncrona.

Apéndices

Apéndice: PICC_SIGNAL.m

```
%%%%%%%
%Generamos la subporadora de 847.5 KHz
%Generamos un periodo y lo replicamos
%La tasa de muestreo sera 4*14.4075MHz, en 1 periodo de subcarrier
entran
%68 muestras de la fs
onePeriodSubcarrier =[ones(1, 34), zeros(1, 34)];
numCycles = 240;%Para generar 30 periodos de Tbit
% 68 representan 0.00000117994 segundos, averiguamos el dt
dt = 0.00000117994/ numel(onePeriodSubcarrier);
subcarrier= repmat(onePeriodSubcarrier, [1, numCycles]);

t = dt * (0 : (numel(subcarrier) - 1));
figure (1)
plot(t, subcarrier);

manchester=[zeros(1,5984),   ones(1,272),zeros(1,272)
,zeros(1,272),ones(1,272),ones(1,272),zeros(1,272),zeros(1,272),ones(1
,272),ones(1,272),zeros(1,272),zeros(1,272),ones(1,272),ones(1,272),ze
ros(1,272),zeros(1,272),ones(1,272),ones(1,272),zeros(1,272),ones(1,27
2),zeros(1,272),zeros(1,272),ones(1,272),ones(1,272),zeros(1,272),zero
s(1,272),ones(1,272),ones(1,272),zeros(1,272),zeros(1,272),ones(1,272)
,ones(1,272),zeros(1,272),zeros(1,272),ones(1,272),ones(1,272),zeros(1
,272),ones(1,272),zeros(1,272)];
subcarrier_modulated=manchester .* subcarrier;
subcarrier_modulated=0.01*subcarrier_modulated;
figure(2)
plot(t,subcarrier_modulated);
%%Generamos la portadora de 13.56 MHz muestreada a 4*14.4075MHz

%carrier=0.5+0.5*cos(2*pi*13.56*1000000*t);
carrier=cos(2*pi*13.56*1000000*t);
figure(3)
plot(t,carrier);
xlim([0,0.000001])
ylim([-3,3]);

mixer_out=carrier.*subcarrier_modulated;
figure(4)
plot(t,mixer_out);

%%Calculo de fft de mixer out
Y = fft(mixer_out+0.05*rand(size(t)));
L=numel(mixer_out);
P2 = abs(Y/L);
P1 = P2(1:L/2+1);
P1(2:end-1) = 2*P1(2:end-1);
f = (1/(4*14.4074*1000000))*(0:(L/2))/L;
figure(5)
plot(f,P1)
%%%%%%%
%Calculo del espectro completo que entregaria el adc
signal_picc=mixer_out + carrier;
signal_picc=0.5+(0.5/2)*signal_picc;
figure(6)
plot(t,signal_picc)
```

```
%Calculo de la se□con ruido
signal_picc=signal_picc+0.001*rand(size(t));
figure(7)
plot(t,signal_picc)
%%Calculo de fft de mixer out
Y = fft(signal_picc);
L=numel(signal_picc);
P2 = abs(Y/L);
P1 = P2(1:L/2+1);
P1(2:end-1) = 2*P1(2:end-1);
f = (1/(4*14.4074*1000000))*(0:(L/2))/L;
figure(8)
plot(f,P1)

%%%
%Arreglo para que las muestras se representen en una variable de 10
bits
B=11;
signal_picc=signal_picc';
signal_picc=round(signal_picc*power(2,B-1)-1); %% Fixed point
%%%%%
%Guardo las muestras en txt
 fileID = fopen('PICC_SIGNAL.txt','w');
 fprintf(fileID,'%d\n',signal_picc);
 fclose(fileID);
```

Apéndice: system_tx_tb.v

```
/*
    Top system tb
*/
`include "baudgen.vh"
`timescale 1ns/10ps
module system_rx_tb();

localparam BAUD = `B115200;

localparam BITRATE = (BAUD << 1);
//----------------------------------------
//-- Tarea para enviar caracteres serie
//----------------------------------------
  task send_car;
    input [7:0] car;
  begin
    rx <= 0;                  //-- Bit start
    #BITRATE rx <= car[0];    //-- Bit 0
    #BITRATE rx <= car[1];    //-- Bit 1
    #BITRATE rx <= car[2];    //-- Bit 2
    #BITRATE rx <= car[3];    //-- Bit 3
    #BITRATE rx <= car[4];    //-- Bit 4
    #BITRATE rx <= car[5];    //-- Bit 5
    #BITRATE rx <= car[6];    //-- Bit 6
    #BITRATE rx <= car[7];    //-- Bit 7
    #BITRATE rx <= 1;         //-- Bit stop
    #BITRATE rx <= 1;         //-- Esperar a que se envie bit de stop
  end
  endtask


reg clk=0;
```

```
reg rx=1;
wire [9:0] output_to_dac;

system_rx #(BAUD)
    system_rx_tb(.rx(rx),
                 .clock(clk),
                 .dac(output_to_dac)
                );

always
  #1 clk <= ~clk;

initial begin

  //-- Fichero donde almacenar los resultados
  $dumpfile("system_rx_tb.vcd");
  $dumpvars(0, system_rx_tb);

  //-- Enviar datos de prueba
  #30    send_car(8'h26);

  #30000   send_car(8'h26);

  #60000   send_car(8'h26);


  #200000 $display("FIN de la simulacion");
  $finish;
end
endmodule
```

Apéndice: system_rx_tb.v

```
`timescale 1ns/10ps
`default_nettype none

module system_tb();

integer fileIdr;
integer scan_file;
reg  [9:0] d_matlab_handler;
reg [9:0] muestras [24479:0];

reg [9:0] in;
reg[14:0] i;
reg[14:0] j;

reg clock=0;
//reg portadora_clk=0;
reg rst=0;
wire comparator_output;
wire [7:0] byte_out;
wire byt_val;
wire trama_err;
wire parity_err;



system system_tb(.clk(clock),
```

```
//         .carrier_clk(portadora_clk),
            .reset(rst),
            .adc(in),
           .byte_to_uart(byte_out),
           .valid_byte(byt_val),
           .trama_error(trama_err),
           .parity_error(parity_err));


initial
    begin
    //-- Definir el fichero donde volcar los datos
        $dumpfile("system_tb.vcd");
        //-- Volcar todos los datos a ese fichero (al finalizar la
simulacion)
        $dumpvars(0, system_tb);

        fileIdr=$fopen("PICC_SIGNAL_RT_0_01_NOISE60dB.txt","r");

    end

initial
    begin
    i<=14'd0;
    in<=10'd0;

    #5 rst<= 1'b1;
    #33 rst<= 1'b0;
    end
initial
    begin

        for(j=0;j<24480;j=j+1'b1)
        begin
        scan_file = $fscanf(fileIdr, "%d\n", d_matlab_handler);
        muestras[j]<= d_matlab_handler;

        end
    end
always
    begin
    #8.6760 clock<=~clock;
    end
//always #36.8732 portadora_clk<=~portadora_clk;

always @ (posedge clock)
begin


        if(i>15'd24479)
        begin
        i <= 15'd0;
        end else
        begin

         in<=muestras[i];
         i <= i + 1'b1;
         end
```

```
end
initial
begin
#2748837 $finish;
//#10488.37 $finish;
end

endmodule
```

Apéndice: binary_offset.v

```
`default_nettype none
module binaryoffset (
    input celeka,
    input reset_binaryoffset,
    input [9:0] input_adc,
    output signed [9:0] signed_output

);
reg signed [9:0] s_output;
always @(posedge celeka)
begin
    if(reset_binaryoffset==1'b1)
    begin
    s_output<=10'd0;
    end else
    begin
    s_output<={~input_adc[9],input_adc[8:0]};//invirtiendo el MSB se
elimina el offset de la señal de entrada entregada por el ADC
                            //quedando los datos en formato de
complemento a dos
    end
end
assign signed_output=s_output;

endmodule
```

Apéndice: carrier.v

```
`default_nettype none
module carrier_module( input wire reset,
           input wire clk,          //-- Reloj del sistema 54,24MHz
           output reg [9:0]carrier
                    );
reg [1:0] counter;//contador de ciclos de reloj
always @(posedge clk or posedge reset)
begin
    if(reset)
    begin
    carrier<=0;
    counter<=0;
    end
    else
    begin
        if((counter==2'b00))
        begin
        carrier<=10'd1023;
        end
        if((counter==2'b01))
```

```
        begin
        carrier<=10'd512;
        end
        if((counter==2'b10))
        begin
        carrier<=10'd0;
        end
        if((counter==2'b11))
        begin
        carrier<=10'd512;
        end
        counter<=counter+1'b1;
    end


end
endmodule
```

Apéndice: cic.v

```
`default_nettype none

//----------------------------------------------------------------
// CIC de 4 orden , decimacion 1/64, ganancia 64^4, n_bits_in +
order*log2(decimation)= 34 bits para las variables del filtro
//----------------------------------------------------------------


module Integrated
(
    input clk, rst,
    input signed [9:0] Xin,//10 bits variable de entrada
    output signed [33:0] Xout
);

//Cada integrador requiere de un registro y un sumador
wire signed [33:0] I1, I2, I3, I4;
reg signed [33:0] d1, d2, d3,d4;

// 1 integrador
always @ (posedge clk or posedge rst)
    if (rst) d1 <= 0;
    else d1 <= I1;              // registro
assign I1 = d1 + {{24{Xin[9]}},Xin};  //suma

//2 integrador
always @ (posedge clk or posedge rst)
    if (rst) d2 <= 0;
    else d2 <= I2;            // registro
assign I2 = I1 + d2;          //suma

//3 integrador
always @ (posedge clk or posedge rst)
    if (rst) d3 <= 0;
    else d3 <= I3;            // registro
assign I3 = I2 + d3;          //suma

//4 integrador
always @ (posedge clk or posedge rst)
    if (rst) d4 <= 0;
```

```
    else d4 <= I4;             // registro
assign I4 = I3 + d4;           // suma

assign Xout = I4;               //salida del integrador

endmodule



module Decimate
(
    input clk,
    input rst,
    input signed [33:0] din,
    output tvalid,
    output signed [33:0] dout
);

reg tvalid_reg;
reg [5:0] cnt;      //COntador para la decimacion
reg signed [33:0] dout_reg;

always @ (posedge clk or posedge rst)
    if (rst) begin
        cnt <= 0;
        tvalid_reg <= 1'b0;
        dout_reg <= 0;
    end
    else begin
        if (cnt == 63) begin   //Cada 64 muestras
            tvalid_reg <= 1'b1;
            dout_reg <= din;
            cnt <= 0;
        end
        else begin
            tvalid_reg <= 1'b0;
            cnt <= cnt + 1'b1;
        end
    end

assign dout = dout_reg;
assign tvalid = tvalid_reg;

endmodule



module Comb
(
    input clk,
    input rst,
    input signed [33:0] din,
    input tvalid,
    output signed [33:0] dout);

reg signed [33:0] d1,d2,d3,d4,d5;
wire signed [33:0] C1, C2,C3, dout_reg;

always @ (posedge clk or posedge rst)
    if (rst) begin
        d1 <= 0;
```

```
            d2 <= 0;
            d3 <= 0;
            d4 <= 0;
            d5 <= 0;
        end
        else begin
            if (tvalid) begin
                d1 <= din;
                d2 <= d1;
                d3 <= C1;
                d4 <= C2;
                d5 <= C3;
            end
        end

assign C1 = d1 - d2;        //Restador
assign C2 = C1 - d3;
assign C3 = C2 - d4;
assign dout_reg = C3 - d5;
assign dout = dout_reg;

endmodule


module CIC (
    input clk,
    input rst,
    input signed [9:0] din,
    output tvalid,
    output signed [33:0] dout
);

wire n_sample_rate;
wire [33:0] x_out_integrador;
wire [33:0] x_out_dec;

Integrated int(
    .clk(clk),
    .rst(rst),
    .Xin(din),
    .Xout(x_out_integrador)
);

Decimate dec(
    .clk(clk),
    .rst(rst),
    .din(x_out_integrador),
    .tvalid(n_sample_rate),
    .dout(x_out_dec)
);

Comb combb
(
    .clk(clk),
    .rst(rst),
    .din(x_out_dec),
    .tvalid(n_sample_rate),
    .dout(dout)
);
assign tvalid = n_sample_rate;
```

```
endmodule
```

Apéndice: cic_threshold.v

```
`default_nettype none

//----------------------------------------------------------------
// CIC de 4 orden , decimacion 1/256, ganancia 256^4,     67 bits para
las variables del filtro. La ganancia a la salida del filtro esta
normalizada
//----------------------------------------------------------------

module Integrated_threshold
(
    input clk, rst,
    input signed [34:0] Xin,
    output signed [66:0] Xout
);


wire signed [66:0] I1, I2, I3, I4;
reg signed [66:0] d1, d2, d3,d4;


always @ (posedge clk or posedge rst)
    if (rst) d1 <= 0;
    else d1 <= I1;
assign I1 = d1 + {{32{Xin[34]}},Xin};


always @ (posedge clk or posedge rst)
    if (rst) d2 <= 0;
    else d2 <= I2;
assign I2 = I1 + d2;


always @ (posedge clk or posedge rst)
    if (rst) d3 <= 0;
    else d3 <= I3;
assign I3 = I2 + d3;


always @ (posedge clk or posedge rst)
    if (rst) d4 <= 0;
    else d4 <= I4;
assign I4 = I3 + d4;

assign Xout = I4;

endmodule



module Decimate_threshold
(
    input clk,
    input rst,
```

```
    input signed [66:0] din,
    output tvalid,
    output signed [66:0] dout
);

reg tvalid_reg;
reg [7:0] cnt;
reg signed [66:0] dout_reg;

always @ (posedge clk or posedge rst)
    if (rst) begin
        cnt <= 0;
        tvalid_reg <= 1'b0;
        dout_reg <= 0;
    end
    else begin
        if (cnt == 255) begin
            tvalid_reg <= 1'b1;
            dout_reg <= din;
            cnt <= 0;
        end
        else begin
            tvalid_reg <= 1'b0;
            cnt <= cnt + 1'b1;
        end
    end

assign dout = dout_reg;
assign tvalid = tvalid_reg;

endmodule



module Comb_threshold
(
    input clk,
    input rst,
    input signed [66:0] din,
    input tvalid,
    output signed [66:0] dout);

reg signed [66:0] d1,d2,d3,d4,d5;
wire signed [66:0] C1, C2,C3, dout_reg;

always @ (posedge clk or posedge rst)
    if (rst) begin
        d1 <= 0;
        d2 <= 0;
        d3 <= 0;
        d4 <= 0;
        d5 <= 0;
    end
    else begin
        if (tvalid) begin
            d1 <= din;
            d2 <= d1;
            d3 <= C1;
            d4 <= C2;
            d5 <= C3;
        end
```

```
    end

assign C1 = d1 - d2;
assign C2 = C1 - d3;
assign C3 = C2 - d4;
assign dout_reg = C3 - d5;
assign dout = dout_reg;

endmodule


module CIC_threshold (
    input reloj,
    input reseteo,
    input signed [34:0] din,
    output tvalid,
    output signed [34:0] dout
);

wire n_sample_rate;
wire [66:0] x_out_integrador;
wire [66:0] x_out_dec;
wire [66:0] out;

Integrated_threshold int_threshold(
    .clk(reloj),
    .rst(reseteo),
    .Xin(din),
    .Xout(x_out_integrador)
);

Decimate_threshold dec_threshold(
    .clk(reloj),
    .rst(reseteo),
    .din(x_out_integrador),
    .tvalid(n_sample_rate),
    .dout(x_out_dec)
);

Comb_threshold combb_threshold
(
    .clk(reloj),
    .rst(reseteo),
    .din(x_out_dec),
    .tvalid(n_sample_rate),
    .dout(out)
);
assign tvalid = n_sample_rate;
assign dout = (out>>32);//la señal de referencia tendra una ganancia
de 32 puesto que viene de un cic, hay que atenuarla 32 veces(Desplazar
5 veces a la derecha)

endmodule
```

Apéndice: baudgen.v

```
//----------------------------------------------------------------------
---------
//-- Generacion de baudios
```

```
//-- Señal cuadrada, de periodo igual a la frecuencia de los baudios
indicados
//-- El ancho del pulso positivo es de 1 ciclo de reloj
//--
//-- (c) BQ. August 2015. written by Juan Gonzalez (obijuan)
//------------------------------------------------------------------
---------
//-- GPL license
//------------------------------------------------------------------
---------
`include "baudgen.vh"

module baudgen
    #(
    parameter BAUD = `B230400
    )

(input wire clk,
               input wire clk_ena,
               output wire clk_out);


localparam N = $clog2(BAUD);


reg [N-1:0] divcounter = 0;

always @(posedge clk)

  if (clk_ena)

    divcounter <= (divcounter == BAUD - 1) ? 0 : divcounter + 1;
  else

    divcounter <= BAUD - 1;

assign clk_out = (divcounter == 0) ? clk_ena : 0;


endmodule
```

Apéndice: baudgen.vh

```
//-- Valores de los divisores para conseguir estos BAUDIOS:
`define B1500000 38
`define B230400 250
`define B115200 500
`define BR230400 235
```

Apéndice: baudgen_rx.v

```
//------------------------------------------------------------------
---------
//-- Generacion de baudios
//-- Señal cuadrada, de periodo igual a la frecuencia de los baudios
indicados
//-- El ancho del pulso positivo es de 1 ciclo de reloj
//--
//-- Una vez habilitado, el pulso se genera justo en la mitad del
periodo
//-- Esto es necesario para implementar el receptor
```

```
//--
//-- (c) BQ. August 2015. written by Juan Gonzalez (obijuan)
//-----------------------------------------------------------------
---------
//-- GPL license
//-----------------------------------------------------------------
---------
`include "baudgen.vh"

//-- ENTRADAS:
//--     -clk: Senal de reloj del sistema (12 MHZ en la iceStick)
//--     -clk_ena: Habilitacion.
//--             1. funcionamiento normal. Emitiendo pulsos
//--             0: Inicializado y parado. No se emiten pulsos
//
//-- SALIDAS:
//--     - clk_out. Señal de salida para lograr la velocidad en
baudios indicada
//--                 Anchura de 1 periodo de clk. SALIDA NO REGISTRADA
module baudgen_rx
    #(
                parameter BAUD = `BR230400
    )
    (
                input wire clk,
                input wire clk_ena,
                output wire clk_out);


//-- Numero de bits para almacenar el divisor de baudios
localparam N = $clog2(BAUD);

//-- Valor para generar pulso en la mitad del periodo
localparam M2 = (BAUD >> 1);

//-- Registro para implementar el contador modulo M
reg [N-1:0] divcounter = 0;

//-- Contador módulo M
always @(posedge clk)

  if (clk_ena)
    //-- Funcionamiento normal
    divcounter <= (divcounter == BAUD - 1) ? 0 : divcounter + 1;
  else
    //-- Contador "congelado" al valor maximo
    divcounter <= BAUD - 1;

//-- Sacar un pulso de anchura 1 ciclo de reloj si el generador
//-- esta habilitado (clk_ena == 1)
//-- en caso contrario se saca 0
//-- Se pone a uno en la mitad del periodo
assign clk_out = (divcounter == M2) ? clk_ena : 0;


endmodule
```

Apéndice: comparator.v

```
`default_nettype none
module comparator (input [34:0] magnitude_in,
            input [34:0] reference_in,
            output clear_signal);
assign clear_signal = (magnitude_in >reference_in) ? (1'b1):(1'b0);

endmodule
```

Apéndice: decoder_manchester.v

```
`default_nettype none

module manchester_decoder (
                input clock,
                input reset,
                input raw_bit,
                output reg [8:0] frame,
                output reg strobe,
                output est,
                output reg error_flag
                    );
reg estado=0; //estado=1 Decodificando
          //estado=0 No Data
reg muestra_anterior=0;

reg [9:0] sync_counter=0;//contador de 544 cilos de reloj, para
determinar el Tbit

reg [3:0] bit_counter=0;//contador de los bits decodificados, 9 por
frame


reg first_half=0;
reg first_half_bis=0;
reg second_half=0;

reg start_flag=0;


//Se necesita comprobar cada mitad de Tbit para ver si se ha llegado
el end of comunication, de ahi los dos contadores para cada mitad de
Tbit
always @(posedge clock)
begin
    if(reset)
    begin

    frame <=0;
    strobe<=0;
    error_flag<=0;
    sync_counter<=0;
    end
    else
    begin
    case(estado)
    1'b0:
        begin
            if((raw_bit==1)&&(muestra_anterior==0))
```

```
            begin
                estado<=1;//La modulación esta presente, se procede a
decodificar

                start_flag<=1;

            end
            muestra_anterior<=raw_bit;
            sync_counter<=0;
            error_flag<=0;
        end
    1'b1:
        begin

            if(sync_counter==10'd128)//1/4 de Tbit 135 cuentas
            begin


                first_half<=raw_bit;
            end


            if(sync_counter==10'd384)//3/4 de Tbit 407 cuentas
            begin


                if((first_half==1)&&(raw_bit==0))
                begin

                    if(start_flag==0)
                    begin
                        frame[bit_counter]<=1'b1;
                        bit_counter<=bit_counter+1;//hay que
incrementar el bit_counter en cada caso precisamente por el flag de
start, descuadraria si no la
                                            //posicion delregistro de
salida
                    end
                end


                if((first_half==0)&&(raw_bit==1))
                begin
                    frame[bit_counter]<=1'b0;
                    bit_counter<=bit_counter+1;
                end

                if((first_half==0)&&(raw_bit==0))
                begin
                    estado<=0;
                    bit_counter<=0;
                    sync_counter<=0;//esto seria redundante si lo
establecemos a cero en estado 0

                end
                if((first_half==1)&&(raw_bit==1))
                begin
                error_flag<=1;
                estado<=0;
                bit_counter<=0;
                sync_counter<=0;
```

```
                end


            end
            if(bit_counter==9)
            begin
             bit_counter<=0;
             strobe<=1;
            end
            else
            begin
            strobe<=0;
            end
            if(sync_counter==10'd511)
            begin
            start_flag<=0;
            sync_counter=0;
            end
            sync_counter<=sync_counter+1;


        end
    endcase
    end
end
assign est = estado;
endmodule
```

Apéndice:frame_disassembler.v

```
`default_nettype none


module disassembler (
            input clk,
            input reset,
            input frm_valid,
            input [8:0] frame,
            output reg parity_error_flag,
            output reg byte_valid,
            output reg [7:0] byte);
always@(posedge clk)
begin
    if(reset)
    begin
    byte=0;
    byte_valid=0;
    parity_error_flag=0;
    byte_valid=0;
    end
    else
    begin
        byte_valid<=0;
        parity_error_flag<=0;
        if(frm_valid)
        begin
            if((^~frame[7:0])==frame[8])
            begin
                byte<=frame[7:0];
```

```
                byte_valid<=1;

            end
            else
            begin
            parity_error_flag<=1;
            end
        end
    end
end
endmodule
```

Apéndice: inicializador.v

```
`default_nettype none

/*
    Modulo para resetear el resto de modulos sincronos
    4 ciclos de 27,12 MHz de reset:
    2 ciclos de portadora de reset
    8 ciclos de CLKDAC de reset
    8.5 ciclos de CLKADC de reset
*/
module init( input clk,//27,12MHz
         output reg init_pulse
        );

reg [1:0] count=0;
reg control=1;
//reg rst=0;

always @ (posedge clk)
begin
    if(control==1)
    begin
    init_pulse<=1'b1;
        if(count==2'b11)
        begin
        control<=0;
        end
    count<=count+1'b1;

    end
    else
    begin
    init_pulse<=1'b0;

    end
end

endmodule
```

Apéndice: magnitud.v

```
`default_nettype none

module magnitude (
           input signed [33:0] q,
           input signed [33:0] i,
```

```
           output signed [34:0] signal_magnitude//la suma de los
modulos desbordan las variables de entrada
          );

wire signed [33:0] temp_q;
wire signed [33:0] temp_i;

assign temp_q= (q<0) ? ((~q)):(q);

assign temp_i= (i<0) ? ((~i)):(i);
assign signal_magnitude=temp_q+temp_i;

endmodule
```

Apéndice: main.v

```
`default_nettype none
`include "baudgen.vh"

module main(
    input   CLK27,  //RELOJ 27,12 MHz
    output  CLKDAC, //RELOJ 4*13,56MHz= 54,24MHz
    output  [9:0]DDAC,  //Señal transmitida al PICC (Portadora +
modulación miller)
    output  MODEDAC,    //1=twos complement  0=straight binary
    output  CLKADC, // RELOJ 4*14,4075 MHz = 57,63MHz
    input   [9:0]DADC,  //Datos recibidos del PICC
    output bit_miller,  //Envolvente de la modulación Miller de la
poradora
    output dv_extended, //Flag que indica el estado de la modulacion
miller, 1=modulando, 0= no modulando
    output bit_manchester,  //Envolvente de la demodulación Manchester
de la subportadora
    output est,     //Flag que indica el estado de la decodificación
Manchester, 1= decodificando, 0=No decodificando
    output tx,      //Linea de transmision serie asincrona
    input rx,       //Linea de recepcion serie asincrona
    output reg [3:0] leds
);


localparam BAUDRX = `BR230400;  //Baudios para la UART RX

parameter BAUD = `B230400;  //Baudios para la UART TX

wire clk_dac, clk_adc;      //Relojes entregados por los pll
wire rst;           //señal de reset para los módulos secuenciales
wire byte_val;          //Flag que indica la disponibilidad de un byte
para su transmision por la UART TX
wire [7:0] byte_uart_tx;    //Byte transmitido porla UART
wire uart_tx_ready;     //Flag que indica que la UART de transmision
ya esta disponible para la transmision de otro dato

wire [7:0] byte_uart_rx;    //Byte que llega por la UART de recepcion
reg data_valid_uart=0;      //Flag para indicarle al sistema de
transmision la disponibilad de un byte en la UART de recepcion



/************************************************************
```

```
    Modulos de los PLL que generan
    54,24 MHz y 57,63 MHz respectivamente
*****************************************************************/
pll_1 pll1 (.clock_in(CLK27), .clock_out(clk_dac));
pll_1 pll2 (.clock_in(CLK27), .clock_out(clk_adc));

/*****************************************************************
    Modulo que genera un pulso de reset
    para inicializar los modulos secuenciales sincronos
    de duracion 4 ciclos de portadora
*****************************************************************/
init inicializador( .clk(CLK27),//27,12MHz
                .init_pulse(rst));

/*****************************************************************
    UART RX
*****************************************************************/
wire temp;
uart_rx #(BAUDRX)
  RX0 (.clk(clk_dac),
       .rstn(~rst),
       .rx(rx),
       .rcv(temp),
       .data(byte_uart_rx)
      );
reg [7:0] byte=8'h26;
wire d_valid;
always @(posedge clk_dac)   data_valid_uart <= temp;

/*****************************************************************
    Modulo que implementa todos los elementos de la cadena
    de transmision:
    -shifter
    -Modified Miller coder
    -Modulator ASK 100%
    -Carrier Generator
*****************************************************************/

system_tx systemtx(
           .clock(clk_dac),         //54,24 MHz
           .reset(rst),
           .dato_valido(data_valid_uart),
           .pcd_comand(byte_uart_rx),
           .data_valid_extendido(dv_extended),
           .millerbit(bit_miller),
           .dac(DDAC)
        );

/*****************************************************************
    Modulo que implementa todos los elementos de la cadena
    de recepcion:
    -Unoffset binary
    -Quadrature Mixer
    -CIC filters
    -Magnitud
    -CIC threshold
    -Comparator
    -Demodulator
    -Manchester Decoder
    -Frame dissasembler
*****************************************************************/
```

```
system_rx systemrx(    .clk(clk_adc),           //57,63 MHz
                .reset(rst),
                .adc(DADC),
                .byte_to_uart(byte_uart_tx),
                .valid_byte(byte_val),
                .estado(est),
                .manchester_bit(bit_manchester));

/*****************************************************************
    UART TX
*****************************************************************/
uart_tx #(.BAUD(BAUD))
  TX0 (
    .clk(clk_adc),
    .rstn(~rst),
    .data(byte_uart_tx),
    .start(byte_val),
    .ready(uart_tx_ready),
    .tx(tx)
  );

always @(posedge clk_dac)
    //-- Capturar el dato cuando se reciba
    if (data_valid_uart== 1'b1)
      leds <= byte_uart_rx[3:0];




assign CLKDAC=~clk_dac;
assign CLKADC=~clk_adc;

assign MODEDAC=0;//1=twos complement  0=straight binary


endmodule
```

Apéndice: modulador_shifter.v

```
`default_nettype none

module shifter(         input load,
                input reset,
                input [7:0] data_in,
                input clock,//54.24MHz
                output reg bit_valid,
                output data_out
                );
reg [7:0]temp;
reg [8:0]carrier_counter;
reg [3:0]bit_counter;
reg control_counter;
always @ (posedge clock )
begin
   if (reset)
   begin
          temp <= 8'd0;
      carrier_counter<=9'd0;
      bit_counter<=4'd0;
      control_counter<=1'b0;
```

```
      bit_valid<=1'b0;
    end
    else
    begin
        if (load)
         begin
           temp <= data_in;
           bit_counter<=3'd0;
           control_counter<=1'b1;
           carrier_counter<=9'd0;
         end
        else
         begin
          if(carrier_counter>510)
              begin
                  temp <= {1'b0, temp[7:1]};
                      if(control_counter)
                      begin
                  bit_valid <=1'b1;
                  end
                  if(bit_counter>6)
                  begin
                  control_counter<=1'b0;
                  end
              bit_counter<=bit_counter+1'b1;
              carrier_counter<=9'd0;
              end

          else
          begin
              bit_valid <= 1'b0;
              carrier_counter<= carrier_counter + 1'b1;
         end
             end

    end
end

assign data_out = temp[0];
endmodule


module delay(   input d_valid_in,
         input reset,
         input reloj,
         output d_valid_out
         );
reg[511:0] temp;
always@(posedge reloj or posedge reset)
begin
    if(reset)
    begin

        temp<=511'd0;
    end
    else
    begin
        if(d_valid_in)
        begin
        temp[511]<=1'b1;
        end
```

```
        else
        begin
        temp <= {1'b0, temp[511:1]};
        end
    end
end
assign d_valid_out = temp[0];
endmodule


module extended_d_valid( input reset,
              input clok,
              input dt_valid,
              output reg d_valid_extendido
                );
reg control;
reg [12:0] counter;
always @(posedge clok or posedge reset)
begin
    if(reset)
    begin
    d_valid_extendido<=1'b0;
    control<=1'b0;
    counter<=0;
    end
    else
    begin
        if(dt_valid)
        begin
        control<=1'b1;
        end
        if(control)
        begin
            if(counter>4604)
            begin
            control<=1'b0;
            counter<=0;
            end
            else
            begin
            d_valid_extendido<=1'b1;
            counter<=counter+1'b1;
            end
        end
        else
        begin
            d_valid_extendido<=1'b0;
        end
    end
end

endmodule


module mux( D0, D1, S, Y);
input wire D0, D1, S;
output reg Y;

always @(D0 or D1 or S)
begin
```

```
if(S)
Y<= D1;
else
Y<=D0;

end

endmodule
module modulador_miller(            input reset,
                        input s_valid,//bit de flag valido procedente
del shifter
                        input d_valid,//bit de flag palabra completa
valida que se carga en el shifter
                        input d_shift,//bit procedente del shifter
                        input clk_carrier,
//                      output estado,
                        output reg  bit_modulador//bit que modula la
envolvente de la portadora
                        );

reg [2:0] state;//000: Estado IDLE
                //001:Estado Z
                //010:Estado X
                //011:Estado Y
                //100:Estado END

reg [8:0] carrier_counter;//contador de modulo 128 para conocer
cuantos ciclos de portadora se han efectuado en 1 Tbit
reg [3:0] bit_counter;
reg bit_anterior;


always @(posedge clk_carrier)
begin


    if(reset)
        begin
        bit_modulador <= 1'b1;
        state <= 3'b000;//idle
        carrier_counter <= 9'd0;
        bit_counter <= 4'd0;
        bit_anterior <= 1'b0;
        end
    else
        begin




        if(d_valid)
        begin
        state<=3'b001;//start, secuencia Z
        bit_counter<=4'd0;
        //bit_modulador<=1;


        carrier_counter<=0;
        //bit_anterior <= 1'b0;
```

```
        end
        if(s_valid)
        begin
        if(bit_counter>7)
    begin
    state<=3'b000;//IDLE


    end
    else
    begin
    case(d_shift)
    1'b0:
        begin
            if(bit_counter==4'd0)
            begin
            state<=3'b001;//Si el primer bit tras START es cero, se
debe codificar con Z
            end
            else if(bit_anterior==1'b0)
            begin
            state<=3'b001;//Si el primer bit tras START es cero, se
debe codificar con Z
            end
            else
            begin
            state<=3'b011;//estado Y
            end
        end
    1'b1:
        begin
            state<= 3'b010;//secuencia X
        end
    endcase
    bit_counter <= bit_counter + 1'b1;
    bit_anterior<= d_shift;
    end
    carrier_counter<=9'd0;//cada 106KHz reseteamos el contador de
portadora




        end
        case(state)
        3'b000://IDLE
            begin
            bit_modulador<=1'b1;
            end
        3'b001://SECUENCIA Z
            begin
            if(carrier_counter<128)//primer cuarto de Tbit
                begin
                bit_modulador<=1'b0;
                end
            else
                begin
                bit_modulador<=1'b1;
                end
            end
```

```
        3'b010://SECUENCIA X
            begin
            if((carrier_counter>256)&&(carrier_counter<384))//primer
cuarto de Tbit
                begin
                bit_moduladorr<=1'b0;
                end
            else
                begin
                bit_moduladorr<=1'b1;
                end
            end
        3'b011://SECUENCIA Y
            begin
            bit_moduladorr<=1'b1;
            end
        endcase
        carrier_counter<=carrier_counter+1'b1;
        end

end


endmodule


module modulador_shifter (input data_valid,
              input rst,
              input [7:0] data,
              input clk,
              output dt_val_extended,
              output mill_bit,
              output reg [9:0] daC
             );

wire d_valid_delayed;
wire shift_valid;
wire data_shifter;
wire dato_valid_extendido;
wire envolvente;
wire miller_bit;
wire [9:0] portadora;
wire [9:0] dac;
delay retardador(   .reloj(clk),
            .reset(rst),
            .d_valid_in(data_valid),
            .d_valid_out(d_valid_delayed)
            );

extended_d_valid extended_valid(.reset(rst),
              .clok(clk),
              .dt_valid(data_valid),
              .d_valid_extendido(dato_valid_extendido)
               );

shifter desplazador(    .reset(rst),
            .load(d_valid_delayed),
            .data_in(data),
            .clock(clk),
            .bit_valid(shift_valid),
            .data_out(data_shifter)
```

```
            );

modulador_miller modulador(         .reset(rst),
                        .s_valid(shift_valid),//bit de flag valido
procedente del shifter
                        .d_valid(data_valid),//bit de flag palabra
completa valida que se carga en el shifter
                        .d_shift(data_shifter),//bit procedente del
shifter
                        .clk_carrier(clk),
                        .bit_modulador(envolvente)//bit que modula la
envolvente de la portadora
                        );
mux mux_2_1( 1'b1, envolvente, dato_valid_extendido, miller_bit);
carrier_module carrier( .reset(rst),
             .clk(clk),         //-- Reloj del sistema 54,24MHz
             .carrier(portadora)
             );

assign dt_val_extended=dato_valid_extendido;
assign mill_bit = miller_bit;



and and_0 (dac[0],miller_bit ,portadora[0]);
and and_1 (dac[1],miller_bit ,portadora[1]);
and and_2 (dac[2],miller_bit ,portadora[2]);
and and_3 (dac[3],miller_bit ,portadora[3]);
and and_4 (dac[4],miller_bit ,portadora[4]);
and and_5 (dac[5],miller_bit ,portadora[5]);
and and_6 (dac[6],miller_bit ,portadora[6]);
and and_7 (dac[7],miller_bit ,portadora[7]);
and and_8 (dac[8],miller_bit ,portadora[8]);
and and_9 (dac[9],miller_bit ,portadora[9]);

always @(posedge clk)
begin
if(rst)
begin
daC<=0;
end
else
begin
daC<=dac;
end

end
endmodule
```

Apéndice: /**

```
 * PLL configuration
 *
 * This Verilog module was generated automatically
 * using the icepll tool from the IceStorm project.
 * Use at your own risk.
 *
 * Given input frequency:        27.120 MHz
 * Requested output frequency:   54.240 MHz
 * Achieved output frequency:    54.240 MHz
```

```
 */

module pll_1(
    input  clock_in,
    output clock_out,
    output locked
    );

SB_PLL40_CORE #(
        .FEEDBACK_PATH("SIMPLE"),
        .DIVR(4'b0000),      // DIVR =  0
        .DIVF(7'b0011111),   // DIVF = 31
        .DIVQ(3'b100),       // DIVQ =  4
        .FILTER_RANGE(3'b011)    // FILTER_RANGE = 3
    ) uut (
        .LOCK(locked),
        .RESETB(1'b1),
        .BYPASS(1'b0),
        .REFERENCECLK(clock_in),
        .PLLOUTCORE(clock_out)
        );

endmodule
 /*
 * Given input frequency:        27.120 MHz
 * Requested output frequency:   57.630 MHz
 * Achieved output frequency:    57.630 MHz
 */
module pll_2(
    input  clock_in,
    output clock_out,
    output locked
    );

SB_PLL40_CORE #(
        .FEEDBACK_PATH("SIMPLE"),
        .DIVR(4'b0000),      // DIVR =  0
        .DIVF(7'b0100001),   // DIVF = 33
        .DIVQ(3'b100),       // DIVQ =  4
        .FILTER_RANGE(3'b011)    // FILTER_RANGE = 3
    ) uut (
        .LOCK(locked),
        .RESETB(1'b1),
        .BYPASS(1'b0),
        .REFERENCECLK(clock_in),
        .PLLOUTCORE(clock_out)
        );

endmodule
```

Apéndice: quadrature_mixer.v

```
`default_nettype none

module quadrature_mixer(
    input clk_mix,
    input reset_mix,
    input signed [9:0] signed_input,
    output signed[9:0] down_output_q,
    output signed[9:0] down_output_i

);
```

```
reg [1:0] count_mixer=2'd0;
always @(posedge clk_mix) count_mixer<=count_mixer+1;
//cos=1,-1,-1,1,... sen=1,1,-1,-1,... count=0,1,2,3,0,1,2,3,...
assign down_output_q = ((count_mixer==1)||(count_mixer==2)) ?
((~signed_input)+1) : signed_input;
assign down_output_i = ((count_mixer==2)||(count_mixer==3)) ?
((~signed_input)+1) : signed_input;
endmodule
```

Apéndice: system_rx.v

```
`default_nettype none

module system_rx(input clk,
           input reset,
           input [9:0] adc,
           output [7:0]byte_to_uart,
           output valid_byte,
           output estado,
           output manchester_bit);


wire signed [9:0] signed_adc;
wire signed [9:0] q_signal;
wire signed [9:0] i_signal;
wire signed [33:0] q_signal_filtered;
wire signed [33:0] i_signal_filtered;
wire nuevo_fs_q;
wire nuevo_fs_i;
wire [34:0] signal_module;
wire nuevo_fs_umbral;
wire [34:0] reference_comp;
reg signal_comparator;
wire carrier_clk;
wire trama_valid;
wire [8:0] trama;

wire trama_error;
wire parity_error;
wire trm_valid;


binaryoffset unoffset_adc(
    .celeka(clk),
    .reset_binaryoffset(reset),
    .input_adc(adc),
    .signed_output(signed_adc)

);

quadrature_mixer mixer(
    .clk_mix(clk),
    .reset_mix(reset),
    .signed_input(signed_adc),
    .down_output_q(q_signal),
    .down_output_i(i_signal)

);
```

```
CIC cic_q
(
    .clk(clk),
    .rst(reset),        //Reset
    .din(q_signal),         //Datos de entrada muestreados a
4*14.4075MHz
    .tvalid(nuevo_fs_q),        //Señal validadora de los datos de
salida, marca el nuevo sample rate, sera 4*14,4075/64
    .dout(q_signal_filtered)  //datos filtrados de salida
);

CIC cic_i
(
    .clk(clk),
    .rst(reset),
    .din(i_signal),
    .tvalid(nuevo_fs_i),
    .dout(i_signal_filtered)
);

magnitude magnitud(
           .q(q_signal_filtered),
           .i(i_signal_filtered),
           .signal_magnitude(signal_module) //Modulo de la señal
conmpleja de entrada = |Q|+|I|
          );

CIC_threshold cic_umbral
(
    .reloj(nuevo_fs_q),
    .reseteo(reset),
    .din(signal_module),
    .tvalid(nuevo_fs_umbral),
    .dout(reference_comp)   //Valor medio de la señal demodulada en
banda base (Magnitud) para el calculo del umbral dinamico
                    //que emplea el comparador para establecer el bit
manchester de salida
);

wire temp;
comparator comp (   .magnitude_in(signal_module),
            .reference_in(reference_comp),
            .clear_signal(temp));

always @(posedge clk)   signal_comparator <= temp; //se registra la
salida del comparador para eliminar los glitches del circuito
combinacional

manchester_decoder decoder(
                .clock(clk),
                .reset(reset),
                .raw_bit(signal_comparator),
                .frame(trama),
                .strobe(trama_valid),
                .est(estado),
                .error_flag(trama_error)
                    );


disassembler parity_check (
            .clk(clk),
```

```
            .reset(reset),
            .frm_valid(trama_valid),
            .frame(trama),
            .parity_error_flag(parity_error),
            .byte_valid(valid_byte),
            .byte(byte_to_uart));

assign manchester_bit = signal_comparator;
assign trm_valid=trama_valid;
endmodule
```

Apéndice: system_tx.v

```
`default_nettype none

/*module generador_d_valid( input clk,//4*13.56MHz
             input reset,
             output reg d_valid_alert);

reg [14:0] count =0;
always @(posedge clk or posedge reset)
begin
    if(reset)
    begin
    d_valid_alert<=1'b0;
    count<=0;
    end
    else
    begin
        if(count==32767)//RGT(Tiempo minimo) es de 7000 ciclos de
portadora, osea 0,516 ms, Cada ciclo de carrier tiene 4 muestras..
                //Para encajar con un contador potencia de 2
enviaremos el REQA cada 0,604 ms, es decir, cada, 2^15=32768(Minimo
28000)
        begin
        d_valid_alert<=1'b1;
        end
        else
        begin
        d_valid_alert<=1'b0;
        end
        count<=count +1'b1;
    end
end
endmodule*/

module system_tx(
         input clock,
         input reset,
         input dato_valido,
         input [7:0] pcd_comand,
         output data_valid_extendido,
         output millerbit,
         output [9:0] dac
        );

//wire dato_valido;

//reg [7:0] REQA =8'h26;
```

```
/*generador_d_valid gen_d_valid( .clk(clock),
                    .reset(reset),
                    .d_valid_alert(dato_valido));*/

modulador_shifter modulador(.data_valid(dato_valido),
                .rst(reset),
                .data(pcd_comand),
                .clk(clock),
                .dt_val_extended(data_valid_extendido),
                .mill_bit(millerbit),
                .daC(dac)
            );
endmodule
```

Apéndice: uart_rx.v

```
//-----------------------------------------------------------------------
--------
//-- Unidad de recepcion serie asincrona
//----------------------------------------
//-- (C) BQ. October 2015. Written by Juan Gonzalez (Obijuan)
//-- GPL license
//-----------------------------------------------------------------------
--------
//-- Comprobado su funcionamiento a todas las velocidades estandares:
//-- 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200
//-----------------------------------------------------------------------
--------
//-- Although this transmitter has been written from the scratch, it
has been
//-- inspired by the one developed in the swapforth proyect by James
Bowman
//--
//-- https://github.com/jamesbowman/swapforth
//--
//-----------------------------------------------------------------------
--------

`default_nettype none

`include "baudgen.vh"

module uart_rx
    #(
    parameter BAUD = `BR230400
    )
    (input wire clk,          //-- Reloj del sistema
                input wire rstn,         //-- Reset
                input wire rx,           //-- Linea de recepcion serie
                output reg rcv,          //-- Indicar Dato disponible
                output reg [7:0] data); //-- Dato recibo


//-- Reloj para la recepcion
wire clk_baud;

//-- Linea de recepcion registrada
//-- Para cumplir reglas de diseño sincrono
reg rx_r;
```

```
//-- Microordenes
reg bauden;  //-- Activar señal de reloj de datos
reg clear;   //-- Poner a cero contador de bits
reg load;    //-- Cargar dato recibido


//-----------------------------------------------------------------
//--     RUTA DE DATOS
//-----------------------------------------------------------------

//-- Registrar la señal de recepcion de datos
//-- Para cumplir con las normas de diseño sincrono
always @(posedge clk)
  rx_r <= rx;

//-- Divisor para la generacion del reloj de llegada de datos
baudgen_rx #(.BAUD(BAUD))
  baudgen0 (
    .clk(clk),
    .clk_ena(bauden),
    .clk_out(clk_baud)
  );

//-- Contador de bits
reg [3:0] bitc;

always @(posedge clk)
  if (clear)
    bitc <= 4'd0;
  else if (clear == 0 && clk_baud == 1)
    bitc <= bitc + 1;


//-- Registro de desplazamiento para almacenar los bits recibidos
reg [9:0] raw_data;

always @(posedge clk)
  if (clk_baud == 1) begin
    raw_data = {rx_r, raw_data[9:1]};
  end

//-- Registro de datos. Almacenar el dato recibido
always @(posedge clk)
  if (rstn == 0)
    data <= 0;
  else if (load)
    data <= raw_data[8:1];

//--------------------------------------
//-- CONTROLADOR
//--------------------------------------
localparam IDLE = 2'd0;  //-- Estado de reposo
localparam RECV = 2'd1;  //-- Recibiendo datos
localparam LOAD = 2'd2;  //-- Almacenamiento del dato recibido
localparam DAV = 2'd3;   //-- Señalizar dato disponible

reg [1:0] state;

//-- Transiciones entre estados
always @(posedge clk)
```

```
  if (rstn == 0)
    state <= IDLE;

  else
    case (state)

      //-- Resposo
      IDLE :
        //-- Al llegar el bit de start se pasa al estado siguiente
        if (rx_r == 0)
          state <= RECV;
        else
          state <= IDLE;

      //--- Recibiendo datos
      RECV:
        //-- Vamos por el ultimo bit: pasar al siguiente estado
        if (bitc == 4'd10)
          state <= LOAD;
        else
          state <= RECV;

      //-- Almacenamiento del dato
      LOAD:
        state <= DAV;

      //-- Señalizar dato disponible
      DAV:
        state <= IDLE;

    default:
      state <= IDLE;
    endcase


//-- Salidas de microordenes
always @* begin
  bauden <= (state == RECV) ? 1 : 0;
  clear <= (state == IDLE) ? 1 : 0;
  load <= (state == LOAD) ? 1 : 0;
  rcv <= (state == DAV) ? 1 : 0;
end


endmodule
```

Apéndice: uart_tx.v

```
//-------------------------------------------------------------------
--------
//-- Unidad de transmision serie asincrona
//-----------------------------------------
//-- (C) BQ. September 2015. Written by Juan Gonzalez (Obijuan)
//-- GPL license
//-------------------------------------------------------------------
--------

`default_nettype none
```

```
`include "baudgen.vh"


module uart_tx
    #(
    parameter BAUD = `B230400
    )
    (
         input wire clk,
         input wire rstn,
         input wire start,
         input wire [7:0] data,
         output reg tx,
         output wire ready
       );


reg start_r=0;
wire clk_baud;
reg [3:0] bitc=0;
reg [7:0] data_r=0;
wire load;
wire baud_en;

//---------------------------------------
//-- RUTA DE DATOS
//---------------------------------------


always @(posedge clk)
  start_r <= start;

always @(posedge clk)
  if (start == 1 && state == IDLE)
    data_r <= data;


reg [9:0] shifter;


always @(posedge clk)

  if (rstn == 0)
    shifter <= 10'b11_1111_1111;


  else if (load == 1)
    shifter <= {data_r,2'b01};


  else if (load == 0 && clk_baud == 1)
    shifter <= {1'b1, shifter[9:1]};

always @(posedge clk)
  if (load == 1)
    bitc <= 0;
  else if (load == 0 && clk_baud == 1)
    bitc <= bitc + 1;


always @(posedge clk)
```

```
  tx <= shifter[0];


baudgen #(.BAUD(BAUD))
  BAUD0 (
    .clk(clk),
    .clk_ena(baud_en),
    .clk_out(clk_baud)
  );

//------------------------------
//-- CONTROLADOR
//------------------------------

//-- Estados del automata finito del controlador
localparam IDLE  = 0;  //-- Estado de reposo
localparam START = 1;  //-- Comienzo de transmision
localparam TRANS = 2;  //-- Estado: transmitiendo dato


reg [1:0] state;


always @(posedge clk)


  if (rstn == 0)
    state <= IDLE;

  else

    case (state)
      IDLE:
        if (start_r == 1)
          state <= START;
        else
          state <= IDLE;

      START:
        state <= TRANS;
      TRANS:
        if (bitc == 11)
          state <= IDLE;
        else
          state <= TRANS;

      default:
        state <= IDLE;

    endcase


assign load = (state == START) ? 1 : 0;
assign baud_en = (state == IDLE) ? 0 : 1;

assign ready = (state == IDLE) ? 1 : 0;

endmodule
```

Apendice: main_reqa_periodico.v

```
`default_nettype none
`include "baudgen.vh"

module main(
    input   CLK27,  //RELOJ 27,12 MHz
    output  CLKDAC, //RELOJ 4*13,56MHz= 54,24MHz
    output  [9:0]DDAC,  //Señal transmitida al PICC (Portadora +
modulación miller)
    output  MODEDAC,    //1=twos complement  0=straight binary
    output  CLKADC, // RELOJ 4*14,4075 MHz = 57,63MHz
    input   [9:0]DADC,  //Datos recibidos del PICC
    output bit_miller,  //Envolvente de la modulación Miller de la
poradora
    output dv_extended, //Flag que indica el estado de la modulacion
miller, 1=modulando, 0= no modulando
    output bit_manchester,  //Envolvente de la demodulación Manchester
de la subportadora
    output est,     //Flag que indica el estado de la decodificación
Manchester, 1= decodificando, 0=No decodificando
    output tx       //Linea de transmision serie asincrona
);

parameter BAUD = `B230400;  //Baudios para la UART
wire clk_dac, clk_adc;      //Relojes entregados por los pll
wire rst;           //señal de reset para los módulos secuenciales
wire byte_val;          //Flag que indica la disponibilidad de un byte
para su transmision por la UART TX
wire [7:0] byte_uart_tx;    //Byte transmitido porla UART
wire uart_tx_ready;     //Flag que indica que la UART de transmision
ya esta disponible para la transmision de otro dato


/*************************************************************
    Modulos de los PLL que generan
    54,24 MHz y 57,63 MHz respectivamente
*************************************************************/
pll_1 pll1 (.clock_in(CLK27), .clock_out(clk_dac));
pll_1 pll2 (.clock_in(CLK27), .clock_out(clk_adc));

/*************************************************************
    Modulo que genera un pulso de reset
    para inicializar los modulos secuenciales sincronos
    de duracion 4 ciclos de portadora
*************************************************************/
init inicializador( .clk(CLK27),//27,12MHz
                .init_pulse(rst));

/*************************************************************
    Modulo que implementa todos los elementos de la cadena
    de transmision:
    -shifter
    -Modified Miller coder
    -Modulator ASK 100%
    -Carrier Generator
*************************************************************/
system_tx systemtx(
           .clock(clk_dac),         //54,24 MHz
           .reset(rst),
           .data_valid_extendido(dv_extended),
           .millerbit(bit_miller),
           .dac(DDAC)
```

```
        );
/*****************************************************************
    Modulo que implementa todos los elementos de la cadena
    de recepcion:
    -Unoffset binary
    -Quadrature Mixer
    -CIC filters
    -Magnitud
    -CIC threshold
    -Comparator
    -Demodulator
    -Manchester Decoder
    -Frame dissasembler
*****************************************************************/
system_rx systemrx(    .clk(clk_adc),           //57,63 MHz
                .reset(rst),
                .adc(DADC),
                .byte_to_uart(byte_uart_tx),
                .valid_byte(byte_val),
                .estado(est),
                .manchester_bit(bit_manchester));

/*****************************************************************
    UART TX
*****************************************************************/
uart_tx #(.BAUD(BAUD))
  TX0 (
    .clk(clk_adc),
    .rstn(~rst),
    .data(byte_uart_tx),
    .start(byte_val),
    .ready(uart_tx_ready),
    .tx(tx)
  );



assign CLKDAC=~clk_dac;
assign CLKADC=~clk_adc;

assign MODEDAC=0;//1=twos complement  0=straight binary


endmodule
```

Apéndice: pines.pcf

```
set_io CLK27 52
set_io CLKDAC 20
set_io DDAC[0] 22
set_io DDAC[1] 23
set_io DDAC[2] 24
set_io DDAC[3] 25
set_io DDAC[4] 26
set_io DDAC[5] 28
set_io DDAC[6] 29
set_io DDAC[7] 31
set_io DDAC[8] 32
set_io DDAC[9] 33
```

```
set_io MODEDAC 34

set_io CLKADC 49
set_io DADC[0] 37
set_io DADC[1] 38
set_io DADC[2] 39
set_io DADC[3] 41
set_io DADC[4] 42
set_io DADC[5] 43
set_io DADC[6] 44
set_io DADC[7] 45
set_io DADC[8] 47
set_io DADC[9] 48

set_io leds[3] 76
set_io leds[2] 80
set_io leds[1] 79
set_io leds[0] 78

set_io tx 8
set_io rx 9


set_io bit_miller 143
set_io dv_extended 142

set_io bit_manchester 144
set_io est 90
```

Apéndice: makefile

```
#-----------------------------------------
#-- Establecer nombre del componente
#-----------------------------------------

DEPSINT= binaryoffset.v quadrature_mixer.v cic.v magnitud.v
cic_threshold_4_256.v comparator.v system_rx.v system_tx.v
modulador_shifter.v carrier.v decoder_manchester.v inicializador.v
pll.v frame_disassembler.v baudgen.v baudgen_rx.v uart_rx.v uart_tx.v
#DEPSINT= pll.v system_rx.v


#-------------------------------------------------
#-  make sint
#-------------------------------------------------
#-  Objetivo para realizar la sintetis completa
#- y dejar el diseno listo para su grabacion en
#- la FPGA
#-------------------------------------------------

sint: main.bin
    #../iceRAM/iceram $<
    sudo ~/FPGA/lpc11loader/iceload -s 230400 -x -c $<

burn: main.bin
    sudo ~/FPGA/lpc11loader/iceload -p $<
```

```
term:
    sudo ../lpc11loader/iceload -t



#------------------------------
#-- Sintesis completa
#------------------------------
main.bin: pines.pcf main.v $(DEPSINT) Makefile

    #-- Sintesis
    yosys -p "synth_ice40 -relut" -o main.json main.v $(DEPSINT)
>sint.log
    #-- Place & route
    nextpnr-ice40 --hx4k --pcf pines.pcf --json main.json --asc
main.txt --package tq144 2>pnr.log
    #-- Generar binario final, listo para descargar en fgpa
    icepack main.txt main.bin

#-- Limpiar todo
clean:
    rm -f *.bin *.txt *.blif *.out *.vcd *.json *.log *.o65 *~

.PHONY: all clean
```