Trabajo Fin de Máster

Máster en Electrónica, Tratamiento de Señal y Comunicaciones

# Implementación y análisis de SEU de un transmisor de comunicaciones inalámbricas para aplicaciones intra-satélite

Autor: Javier Barrientos Rojas
Tutor: Hipólito Guzmán Miranda

**Departamento de Ingeniería Electrónica**
**Escuela Técnica Superior de Ingeniería**
**Universidad de Sevilla**

Sevilla, Noviembre 2015

Trabajo Fin de Máster
Máster en Electrónica, Tratamiento de Señal y Comunicaciones

# Implementación y análisis de SEU de un transmisor de comunicaciones inalámbricas para aplicaciones intra-satélite

Autor:
Javier Barrientos Rojas

Tutor:
Hipólito Guzmán Miranda
Profesor Ayudante Doctor

Departamento de Ingeniería Electrónica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, Noviembre 2015

Trabajo Fin de Máster: Implementación y análisis de SEU de un transmisor de comunicaciones inalámbricas para aplicaciones intra-satélite


Autor: Javier Barrientos Rojas

Tutor: Hipólito Guzmán Miranda


El tribunal nombrado para juzgar el Trabajo arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, Noviembre 2015

El Secretario del Tribunal

*A mi mujer*

# Agradecimientos

A todo el equipo de personas que forman parte del proyecto Edelweiss: Vicente, Patricio, Miguel Ángel y sobre todo a Hipólito con quien nunca dejo de aprender, agradecer la confianza depositada en mí y conseguir que trabajar en un proyecto de investigación se convierta en algo ameno y sencillo de realizar.

A mis padres por su apoyo incondicional.

A mi mujer por sus palabras de ánimo, su compañía en los buenos y malos momentos ha influido enormemente en la elaboración de este trabajo.

# Resumen

Uno de los inconvenientes más importantes en cuanto a embarcar sistemas de comunicaciones en satélites, es el excesivo peso que aportan a causa, en gran medida, del cableado necesario para su instalación. Una de las soluciones más atractivas que se estudian es la de emplear sistemas inalámbricos por motivos obvios. Este trabajo forma parte de un proyecto de investigación, en el que se trata de diseñar un sistema de comunicaciones inalámbrico de alta eficiencia. Para ello se realizan una serie de pruebas de inyección de fallos sobre el diseño, con el objeto de conocer la robustez estructural del sistema. Dicha característica representa la sensibilidad del diseño ante posibles ataques que pueda sufrir el sistema a causa de una radiación externa en un entorno hostil, como puede ser el caso de una aplicación inter-satelital.

# Índice

# Índice de figuras

# Índice de tablas

# Capítulo 1

# Introducción

*En este capítulo introductorio se explica la motivación del desarrollo del presente trabajo. Ubicado en el ámbito de las aplicaciones espaciales, se detallan los efectos negativos que producen los llamados Eventos Singulares sobre este tipo de aplicaciones y cómo pueden llegar a afectar a cualquier dispositivo electrónico. Terminando por analizar los principales protocolos para comunicaciones inalámbricas que se ajusten dentro de un sistema intra-satélite y definir el concepto de robustez estructural, parámetro que ayudará a entender la capacidad de endurecimiento del que dispone el diseño del transmisor sin emplear ninguna técnica de mitigación de fallos.*

## 1.1. Marco del trabajo

El presente trabajo se desarrolla dentro del proyecto de excelencia de la Junta de Andalucía: "Edelweiss: Diseño de un sistema de comunicaciones inalámbricas intra-satélite de alta eficiencia" (P11-TIC-7095). Proyecto en el que uno de sus principales objetivos es diseñar un sistema de comunicaciones inalámbrico para aplicaciones espaciales y cuyo transmisor es el trabajo que se presenta en este documento.

El equipo de trabajo lo integran varias personas del Departamento de Ingeniería Electrónica de la Universidad de Sevilla y el alumno plasma aquí su contribución con el diseño digital del hardware del transmisor y el sistema de verificación que se emplea para cotejar y validar los resultados obtenidos.

## 1.2. Problemática

El proyecto Edelweiss trata de solventar algunos de los principales problemas que afectan al encarecimiento de cualquier tipo de misión espacial que conlleve el lanzamiento de un satélite sea cual sea su función y la órbita en la que vaya a situarse; a saber: peso, consumo de potencia disponible, ensamblaje e integración de equipos, así como el esfuerzo que se realiza en términos de verificación.

Este tipo de misiones están sujetas a fuertes restricciones de consumo, peso y fiabilidad y tratando de que el conjunto sea económicamente factible para poder ser realizada. Los diferentes dispositivos e instrumentos que se instalan en un satélite, se comunican mediante cableado y este hecho aumenta tanto el peso como la complejidad de la carga útil total, contribuyendo negativamente en el coste de la misión.

La idea de la eliminación del cableado en los satélites no se puede considerar novedosa pero sí el desarrollo de técnicas y su implementación en misiones lanzadas como ensayos o demostradores tecnológicos. Este es el caso, por ejemplo, del satélite Optos lanzado a finales del año 2013 y desarrollado por el INTA, que entre sus principales experimentos a realizar durante la misión, contempla el uso de un sistema de comunicaciones a bordo basado en óptica difusa con el objeto de eliminar el cableado interno [Ref. 1.1].

Otra problemática asociada, de manera general a cualquier dispositivo que vaya a funcionar tanto en las diferentes capas que conforman la atmósfera terrestre como en el espacio exterior, son los **efectos de la radiación** sobre los circuitos electrónicos. Se trata de fenómenos físicos complejos que se producen tras la posible colisión entre partículas provenientes de la radiación y las partículas que componen la estructura molecular de un componente electrónico (principalmente sobre transistores).

Es la atmósfera la que actúa como una pantalla protectora que retiene la mayor parte de esta radiación y los rayos UV que inciden sobre el planeta. Las fuentes de radiación son diversas, entre las principales se pueden destacar las siguientes:

- Protones y electrones atrapados en los cinturones de Van Allen.

- El sol como fuente de dos importantes fenómenos:

    - Iones pesados atrapados en la magnetosfera a causa del viento solar.
    - Iones pesados y protones procedentes de llamaradas solares.

- Rayos cósmicos, cuya composición es aproximadamente de un 85% protones, 14% partículas alfa y un 1% de iones pesados.

Existen numerosos estudios sobre cada una de las diferentes fuentes de radiación ya nombradas. Se conocen como anillos de radiación (llamados cinturones de Van Allen), a dos grandes zonas circundando el planeta Tierra compuestos principalmente por electrones y protones atrapados [Ref. 1.2]. Se ha comprobado que para el anillo interior se alcanza una energía no superior a los 5 MeV y que en el anillo exterior se pueden alcanzar hasta los 7 MeV, la mayoría de iones que componen estos anillos son ligeros y de baja penetración [Ref. 1.3].

El sol es la mayor fuente de radiación que afecta al planeta Tierra, y destacan dos fenómenos principalmente. En primer lugar, las llamaradas solares son fuertes explosiones capaces de liberar una altísima cantidad de energía en forma de calor; estas explosiones provocan la aceleración de electrones, protones e iones pesados a velocidades cercanas a la de la luz [Ref. 1.4]. En segundo lugar, el viento solar que se forma a partir de la expulsión de electrones de la corona solar, formándose un plasma que contiene protones en su mayor parte y un pequeño porcentaje de iones pesados.

Los cambios producidos en la densidad o la velocidad del viento solar provocan perturbaciones sobre la magnetosfera terrestre y dan origen a tormentas magnéticas que pueden afectar a satélites en órbita [Ref. 1.5]. Para terminar, cabe resaltar la influencia de los rayos cósmicos como otra fuente importante de radiación, ya que contienen núcleos de iones pesados de muy alta energía [Ref. 1.6].

El problema de los efectos de la radiación no es algo exclusivo a sistemas que funcionan en órbitas alrededor del planeta Tierra o en el espacio exterior. La evolución tecnológica que se viene desarrollando en los últimos años da como lugar a la fabricación de dispositivos microelectrónicos cada vez más rápidos y más pequeños.

Como consecuencia de esta evolución, se concentran un mayor número de transistores en una porción de silicio del orden de $mm^2$ y esta reducción de tamaño disminuye el valor de la energía necesaria para producir fallos en la electrónica, lo que conlleva a que sean mucho más sensibles a la radiación. Aunque estos efectos se encuentran en menor medida a nivel del mar que fuera de la atmósfera, pero es un factor a tener en cuenta ya que puede provocar fallos en los circuitos y es algo que debe evitarse en sistemas aeronáuticos o de automoción, así como militares. Por ello, se están desarrollando técnicas de protección de los dispositivos electrónicos ante radiación para uso a nivel del mar, ya que la microelectrónica se encuentra en multitud de lugares, transportes y objetos cotidianos.

## 1.3. Efectos de la Radiación

Una vez planteada la motivación del presente trabajo, en este apartado se trata de desarrollar con algo más de detalle cuáles son los efectos que la radiación puede llegar a provocar en un circuito electrónico, resaltando aspectos como por ejemplo si dichos efectos son totalmente destructivos o no y si existen técnicas para mitigarlos.

El estudio de la radiación ha dado lugar a diferenciar entre dos grandes categorías según sea el tipo de daño que provoca sobre un determinado circuito electrónico. Por un lado, este tipo de efectos se tratan de modelar de alguna forma para comprender mejor su naturaleza y poder desarrollar técnicas para mitigarlos. La idea de modelar este tipo de fenómenos físicos tan complejos entre la interacción de partículas provenientes de la radiación y la estructura molecular de un dispositivo electrónico no es sencilla, por ello se intentan modelar ciertos parámetros circuitales (tensión, corriente o carga en ciertos nodos) ya que resulta más sencillo plantear de esta manera el cómo afectan los modelos de error sobre un diseño de un circuito electrónico.

Estos modelos se suelen hacer en el dominio digital, principalmente para simplificar el problema y además aprovechar las capacidades de los simuladores de lógica digital y emuladores hardware mediante el uso de las FPGA. Por otro lado, algunos de estos errores provocados por la radiación pueden llegar a destruir físicamente partes del circuito. Los principales motivos que diferencian a las dos categorías de errores debidos a la radiación se detallan a continuación:

- **Errores Físicos (*Hard Errors*) -** Pueden llegar a destruir físicamente ciertas partes de la estructura de un dispositivo electrónico y además es imposible poder modelarlo y solventarlo de manera digital.

- **Errores Lógicos (*Soft Errors*) –** Es posible hacer un modelo digital de su comportamiento y con ello poder analizarlos y hallar estrategias para mitigarlos.

### 1.3.1. Mecanismos Físicos de Daños

Existen tres grupos donde se diferencian claramente cuáles son los mecanismos o la naturaleza de los daños provocados sobre un circuito electrónico debido a la radiación.

- **Dosis Ionizante Total -** TiD (del inglés *Total ionizing Dose*)

Produce errores del tipo físico ya que tiene un efecto acumulativo sobre el circuito al que ataca; es decir, se trata de una ionización a largo plazo que provoca cambios en la tensión umbral de transistores MOS debido a cargas atrapadas entre el óxido y el semiconductor y que a su vez pueden escapar por efecto túnel degradando las prestaciones del dispositivo. También genera corrientes de fuga o '*leakage*' entre difusiones vecinas de tipo N, lo que puede llevar a un incremento del consumo del transistor MOS.

- **Daño por Desplazamiento** – DD (del inglés *Displacement Damage*)

Produce cambios en la estructura cristalina del material semiconductor tales como un aumento de recombinación de pares e-h, agotamiento de portadores minoritarios y un empeoramiento de las propiedades analógicas de las uniones semiconductoras afectadas; degenerando las propiedades del dispositivo. Los transistores bipolares y dispositivos ópticos son más sensibles a este mecanismo de daño ya que son dependientes de los portadores minoritarios en sus regiones de base y a causa de la recombinación sufren pérdidas de ganancia del transistor.

- **Efectos de Eventos Únicos** - SEE (del inglés *Single Event Effects*)

A diferencia de los mecanismos anteriores que se caracterizan por una acumulación gradual debida a la radiación, estos errores son transitorios y bien localizados, ocasionados por impactos de partículas ionizantes sobre ciertas partes de un circuito. En este caso, son los dispositivos digitales los más afectados ya que pueden provocar cambios de valor en registros y provocar respuestas no esperadas en el sistema implementado. Este tipo de errores es el que se va a emular en este trabajo utilizando una herramienta de inyección de fallos propia, para analizar la robustez del transmisor que se va a diseñar.

## 1.3.2. Efectos de Eventos Únicos (SEE)

Estos daños son los más comunes en dispositivos digitales y sus efectos pueden ser: aumento del consumo, pérdida de funcionalidad (crear situaciones de bloqueo), en general, una degradación de los parámetros del dispositivo y sobre todo la propagación de bits erróneos en memorias y lógica digital. Dentro de este grupo se incluyen varios tipos de efectos que a su vez se dividen según su naturaleza sea de errores tipo físico o lógico. No es objeto de este trabajo describir de manera detallada cada uno de los efectos que existen en esta categoría y por ello, en la tabla 1.1 se muestra un resumen donde se clasifican, según su naturaleza, dejando como único efecto a analizar los de tipo SEU, ya que son los que van a formar parte en el desarrollo de este trabajo.

| Efecto | Naturaleza |
|---|---|
| **SEL** ***Single Event Latchup*** | Físico |
| **SEGR** ***Single Event Gate Rupture*** | Físico |
| **SEB** ***Single Even Burnout*** | Físico |
| **SEU** ***Single Event Upset*** | Lógico |
| **MBU** ***Multi Bit Upset*** | Lógico |
| **SET** ***Single Event Transient*** | Lógico |
| **SEMU** ***Single Event Multi Upset*** | Lógico |

**Tabla 1.1 - Clasificación de SEE**

#### 1.3.2.1. Conmutación por Evento Único

Conocido como SEU (del inglés *Single Event Upset*), provoca un cambio de estado (o nivel lógico: de '0' a '1', o de '1' a '0') de un elemento biestable. Dicho cambio se produce por el impacto de una única partícula ionizante o bien por reacciones nucleares producidas por un único protón. No todas las regiones del biestable son sensibles como para que el impacto de la partícula ionizante produzca un cambio de estado, solamente los drenadores de los transistores que se encuentran cortados pueden acarrear que dicho efecto se lleve a cabo, se denominan nodos sensibles. El impacto de la radiación en estos nodos sensibles genera pares e-h que inducen un pulso de corriente instantánea de duración del orden de picosegundos. La carga generada por la partícula ionizante es recogida por la estructura del biestable y si dicha carga es superior a la denominada carga crítica $\mathbf{Q_c}$ (valor de carga media necesaria para cambiar el estado del biestable), entonces el estado lógico del biestable cambia.

Al ser de naturaleza lógica, lo que produce son errores transitorios que no llegan a destruir físicamente un dispositivo electrónico y además puede ser modelado de manera digital. El modelo básico que se emplea para estudiar este tipo de efecto es el denominado **Bit-Flip**, que representa un cambio o inversión del valor lógico de una celda de memoria; es decir, cambios de '0' a '1', o de '1' a '0', teniendo en cuenta que dicho cambio se produce en un instante único y determinado de tiempo.

## 1.4. Sistemas Inalámbricos para Comunicaciones Intra-satélites

Como se indica al comienzo del capítulo, el presente trabajo muestra el diseño de un transmisor que forma parte de un sistema de comunicaciones inalámbricas para aplicaciones intra-satélites. En este punto se pretende mostrar una visión de los principales sistemas de este tipo aplicados a satélites, algunos de ellos ya han sido embarcados en alguna misión y otros se encuentran en fase de investigación ya que uno de los principales problemas que se encuentran para poder ser instalados y que funcionen de forma autónoma, es el hecho de que a día de hoy falta por desarrollarse una tecnología de pilas o baterías para espacio capaces de dar la potencia necesaria a este tipo de sistemas inalámbricos para su correcto funcionamiento. En el capítulo 2 se realiza una comparativa entre los diferentes sistemas que se describen a continuación, con la idea de establecer los motivos del protocolo de comunicaciones elegido para desarrollar el proyecto Edelweiss.

### 1.4.1. Enlaces Inalámbricos Ópticos

Esta tecnología bautizada por el INTA (Instituto Nacional de Técnica Aeroespacial) como OWLS (*Optical Wireless Links for intra-Satellite communications*), busca una comunicación intrasatelital sin cables basada en una transmisión óptica de la información pero sin emplear la fibra óptica como elemento de conexión si no utilizando sólo emisores y detectores ópticos transmitiendo en el espacio abierto. El inicio de esta tecnología data del año 1999 cuando el INTA la propuso como solución

de la interconexión de micro y nano sistemas en el interior de las plataformas espaciales. El primer sistema con tecnología OWLS en vuelo tuvo lugar en 2004 con el lanzamiento por parte del INTA del satélite NANOSAT 01; más recientemente en septiembre de 2007, se lanzó una cápsula espacial llamada FOTON M-3, dentro de la cual se encuentra un experimento del INTA (propuesto por la Agencia Espacial Europea) denominado FOTON-OWLS con el fin de optimizar este tipo de comunicaciones sin cables y así aligerar el peso en la medida de lo posible [Ref. 1.7].

#### 1.4.1.1. Enlaces Inalámbricos por Infrarrojos (IR)

Una de las tecnologías, dentro de las ópticas, que se plantean es el uso de los enlaces IR como una vía complementaria a la RF (*Radio Frequency*). Entre sus ventajas destacan que no es necesaria una alineación entre transmisores y receptores, se pueden alcanzar altas tasas de datos, el espectro IR no está sujeto a regulaciones gubernamentales de ningún tipo, no se ve afectada por EMI (*Electro Magnetic Interference*) y además es una tecnología de bajo coste.

Es precisamente la inmunidad a EMI la que hace mejor esta tecnología frente a la RF en cuanto a la comunicación de datos dentro de un satélite [Ref. 1.8]. El desafío se centra en la investigación de diferentes puntos con idea de poder desarrollar este tipo de sistemas e implementarlos en alguna misión o experimento espacial:

- Cobertura completa del satélite: La distribución de la carga dentro de un satélite puede crear zonas sombreadas que dificulten la comunicación con algún receptor. Se precisa investigar sobre repetidores que permitan salvar estas zonas y así obtener una cobertura completa. Estos repetidores deben de ser muy robustos y permitir varios caminos redundantes para asegurar las comunicaciones entre cualquier transmisor y transmisor colocados dentro del satélite. A su vez, deben de ser elementos de bajo coste y de fácil instalación.

- Redes difusas de baja-media velocidad: Como se ha comentado, esta tecnología no necesita de ningún tipo de alineación entre transmisor y receptor gracias a que usa enlaces de naturaleza difusa. Por tanto, se precisa de una red con topología difusa y con ello el desarrollo tanto de un hardware como un software aptos para este tipo de comunicaciones. Además se necesita disponer de un protocolo de comunicaciones para manejar los diferentes nodos, ya sean equipos terminales o repetidores.

- Enlaces de alta velocidad: Para el caso en el que haya enlaces con líneas de visión directa, se usan diodos tipo láser disponible hoy día en el mercado, se pueden conseguir tasas de datos del orden de decenas de Gbps. Se necesitan desarrollar tanto el hardware como el software para implementar las aplicaciones de transferencia de datos.

- Integración en el entorno espacial: Un sistema IR debe cumplir con dos aspectos principalmente para ser instalado en un satélite y soportar las

situaciones extremas que se dan en un torno tan hostil como es el espacio exterior. Por un lado, evitar en la medida de lo posible la necesidad de tener partes mecánicas asociadas al sistema IR. Para ello se emplean circuitos electrónicos y optoelectrónicos integrados en el mismo equipo de comunicaciones, y al tratarse de un sistema inalámbrico se evitan los cables y sus problemas de conexionado. Y por otro lado, se desea que sea un sistema de bajo consumo de potencia por lo que es preferible que se desarrollen circuitos optoelectrónicos de muy baja potencia.

- Selección de componentes y su certificación para espacio: Todo el hardware necesario para un sistema de comunicaciones inalámbrico IR debe hacerse usando COTS (*Components Off The Shelf*). Esto implica que una vez se tengan desarrollados los emisores y receptores para este tipo de enlaces, es necesario diseñar y llevar a cabo su correspondiente certificación para espacio y así disponer de unos dispositivos aptos para embarcarlos en vuelo.

La ESA (*European Space Agency*) desarrolló un estudio junto con un demostrador en el año 2000, en el que se disponía de dos redes IR trabajando de forma simultánea. Una de las redes comunicaba una unidad central con varios sensores empleando un canal inalámbrico IR de baja velocidad (tasa de datos de 19200 bps). La otra red establecía una conexión punto a punto de alta velocidad para transmitir una señal de video analógico (modulada a una frecuencia cercana a los 500 MHz).

Este experimento demostró que es posible la comunicación en este tipo de sistemas inalámbricos y que están disponibles enlaces de gran ancho de banda para esta tecnología [Ref. 1.9].

### 1.4.2. Enlaces Inalámbricos por Radiofrecuencia

El empleo de tecnologías RF (*Radio Frequency*) en sistemas de comunicaciones para espacio puede dar solución a ciertas aplicaciones mediante el uso de motas inalámbricas, como por ejemplo, la recopilación de telemetría en general.

Los protocolos inalámbricos COTS que se han investigado, diferencian su uso entre dos ámbitos. Por una parte, las comunicaciones intrasatelitales, donde destacan ZigBee (IEEE 802.15.4) [Ref. 1.10] y Bluetooth (IEEE 802.15.1) [Ref. 1.11]; y por otra parte, las comunicaciones intersatelitales donde encajan los protocolos WiFi (IEEE 802.11) [Ref. 1.12] y WiMax (IEEE 802.16) [Ref. 1.13].

Para el proyecto Edelweiss, se necesita un protocolo de comunicaciones intra-satélite, por tanto se dispone de dos alternativas: ZigBee y Bluetooth.

Los principales criterios de selección para un protocolo inalámbrico para comunicaciones intra-satelitales son: robustez ante fallos provocados por radiación e interferencias externas, poder dar soporte a varios nodos, tasas de datos aceptables, requerimientos de potencia asumibles y en la medida de lo posible minimizar el coste y una sencilla instalación.

En el siguiente capítulo se desarrolla una mejor comparativa entre estas dos opciones y se dan los motivos para la elección de uno de ellos para el desarrollo del presente trabajo.

## 1.5. Robustez Estructural

El concepto de robustez estructural, SR en adelante (del inglés *Structural Robustness*), es la medida más común para determinar la eficiencia frente a los efectos de la radiación (tipos SEE principalmente) sobre la estructura de un sistema genérico. Otra manera de definirlo es la capacidad que tiene un sistema para auto regenerar su comportamiento ante un fallo tipo SEU. Todo sistema se puede caracterizar mediante un valor conocido como AVF (del inglés *Architectural Vulnerability Factor*), definido como la probabilidad de que un fallo en una zona concreta dentro de un circuito pueda dar como resultado un error. Este parámetro es muy interesante en estudios para espacio porque es lógico pensar que la capacidad que tiene un protocolo de comunicaciones inalámbrico de corregir errores de bit, puede ser de ayuda para mitigar los efectos tipo SEU. Por tanto, se precisa calcular el AVF de un circuito mediante un ensayo de inyección de fallos sobre una versión no mitigada del mismo para conocer la SR del sistema.

Más adelante se mostrará como el protocolo de comunicaciones seleccionado tiene una robustez inherente a su arquitectura, que le hace ser capaz de corregir muchos de los errores en la señal recibida causados tanto por las interferencias en el canal de comunicaciones como por fallos inducidos en el propio transmisor.

# Capítulo 2

# Solución Propuesta

*Vista las diferentes opciones que se emplean o investigan a día de hoy para disponer de comunicaciones inalámbricas intra-satelitales, en este capítulo se trata de justificar el protocolo seleccionado en la solución propuesta. Mostrando al final la arquitectura, a nivel de bloques, de dicha solución cuyo desarrollo se describe en el siguiente capítulo.*

## 2.1. Comparativa entre ZigBee y Bluetooth

Tal y como se ha comentado en el capítulo anterior, el presente trabajo forma parte de un sistema de comunicaciones inalámbricas para aplicaciones intra-satelitales y es en este ámbito donde ZigBee y Bluetooth se encuentran como los principales protocolos que se han investigado hasta ahora para su posible uso en misiones espaciales. Lo que se plantea en este punto es justificar el por qué se ha decidido por uno de los dos, antes de entrar en el diseño del transmisor, ya que las propias estructuras de los diferentes módulos que los componen dependen totalmente del protocolo seleccionado.

Ambos protocolos tienen ciertos aspectos en común, pertenecen a la categoría del IEEE 802.15 redes inalámbricas de área personal o WPAN (*Wireless Personal Area Network*) y trabajan en la banda de frecuencia de 2.4 GHz (aunque ZigBee puede trabajar en otras bandas también, esta es la más usual por tener licencia de libre uso a nivel mundial). Se detallan a continuación las principales características de cada uno de estos protocolos, con idea de tener posteriormente un cuadro comparativo donde mostrar las razones por las cuales se realiza la elección del que resulta más adecuado al proyecto de investigación que se va a realizar.

### 2.1.1. Bluetooth

Este protocolo de comunicaciones, definido en el estándar IEEE 802.15.1 aunque el organismo que gestiona las evoluciones del protocolo y la certificación de equipos es el denominado *Bluetooth Special Interest Group* (B-SIG), posibilita la transmisión de datos y de voz entre diferentes dispositivos mediante un enlace RF (*Radio Frequency*) en la banda de los 2.4 GHz. La técnica de modulación empleada es la de salto en frecuencia con espectro expandido o FHSS (del inglés, *Frequency Hopping Spread Spectrum*); entre sus principales características cabe resaltar su alta tasa de velocidad de transmisión (hasta 1 Mbit/s) y el rango de distancias de separación entre los dispositivos (de 1 a 100 metros).

Bluetooth está concebido para ser utilizado de manera exclusiva en redes de tipo Ad hoc, redes descentralizadas enfocadas a una comunicación equipo a equipo donde cada mota participa en el encaminamiento mediante el reenvío de la información hacia otras motas. Son un tipo de redes muy fáciles de implementar y la adhesión de una nueva mota es tan simple como situarla dentro del rango de alcance de la red. Por el contrario, aunque su ancho de banda permite transmitir una mayor cantidad de datos con respecto a ZigBee, como consecuencia emplea un mayor consumo de energía, elemento crítico en aplicaciones espaciales.

### 2.1.2. ZigBee

Esta tecnología fue diseñada por la Alianza ZigBee, está basada en el estándar IEEE 802.15.4 pero modificado para poder funcionar también en redes con tipología '*mesh'* (mallado). Se trata de un protocolo de comunicaciones inalámbrico pensado para establecer comunicaciones a baja velocidad entre dos o varios dispositivos (llegándose a poder formar redes con cientos de dispositivos comunicándose entre sí).

La Alianza ZigBee[1] está formada por cientos de compañías que tratan de encontrar una solución a la necesidad de disponer de un estándar para comunicaciones a baja velocidad pero con un bajo coste de implementación y que además los dispositivos involucrados en la red requieran muy bajo consumo. Algunas de las compañías involucradas en esta alianza son Philips, LG, Texas Instruments, ARM, Schneider, Cisco, Siemens y varias empresas españolas como Indra, que ha entrado recientemente[2].

Entre sus principales características cabe resaltar que su velocidad de transmisión se mueve entre 25-250 kbps, trabaja en la frecuencia de 2.4 GHz (aunque también puede trabajar en las frecuencias de 868 MHz y 915 MHz), es un protocolo asíncrono half dúplex, emplea una modulación de espectro expandido de secuencia directa (DSSS, del inglés *Direct Sequence Spread Spectrum*), soporta redes tipo *mesh*, punto a multipunto y árbol, está pensado para que disponga de dispositivos que trabajan en modo de bajo consumo alargando la duración de sus baterías, es un protocolo fiable ya que la red se organiza y puede repararse de forma automática (las tramas de datos se encaminan de forma dinámica) y por último destacar que es un protocolo seguro ya que permite implementar técnicas de autenticación y encriptación.

Desde el punto de vista de aplicación, como protocolo para un sistema de comunicaciones embarcado en un satélite, resulta una opción muy atractiva por su versatilidad, simplicidad de implementación y sobre todo su bajo consumo.

### 2.1.3. Cuadro comparativo

Se pasa a mostrar en la tabla 2.1 una comparación entre ambos protocolos para seguidamente detallar cuál es la opción elegida para emplear en el desarrollo de este trabajo.

| Protocolo | Bluetooth | ZigBee |
|---|---|---|
| Estándar | 802.15.1 | 802.15.4 |
| Tasa de transmisión máxima (kbps) | 1000 | 250 |
| Rango de transmisión (m) | 1-100 | 1-100 |
| Nº de nodos en la red | hasta 10 | ilimitado ($2^{6+}$) |
| Vida media de baterías (días) | 1-10 | 100-1000+ |
| Tiempo típico de acceso a la red | 3 segundos | 30 milisegundos |
| Mejor característica | Coste | Seguridad, Consumo, Coste |

**Tabla 2.1 – Comparativa entre protocolos**

[1] Para más información, acudir a su web: *http://www.zigbee.org/*
[2] Nota de prensa a fecha de 14 de Octubre de 2015:
*http://www.indracompany.com/noticia/indra-se-adhiere-zigbee-alliance-impulsar-desarrollo-estandares-comunicaciones-iot*

A la vista del cuadro comparativo, teniendo en cuenta que la velocidad de transmisión no es un requisito crítico en una red de sensores inalámbricos que se pueda embarcar en un satélite, se establece que el protocolo seleccionado es ZigBee. Básicamente las características de bajo consumo y su versatilidad para formar diferentes tipos de redes con múltiples dispositivos, convierten a este protocolo en una solución muy adecuada al problema de implementar un sistema de comunicaciones inalámbrico para aplicaciones intra-satelitales.

## 2.2. Capa física del protocolo ZigBee

ZigBee como cualquier otro protocolo de comunicación, posee una torre de protocolos con la cual se pueden desarrollar todo tipo de aplicaciones a nivel de usuario que funcionen bajo una transmisión radio compatible con el estándar ZigBee. En este trabajo sólo se va a implementar la capa física ya que sólo resulta de interés el hecho de poder transmitir una trama y analizar si el receptor es capaz de recuperar la información de usuario aun cuando la trama ha sufrido ataques mediante una inyección de fallos.

Las funciones de la capa física del estándar 802.15.4 son:

- Activación y desactivación del transmisor/receptor
- Detección de energía dentro de cada canal específico
- CCA (*Copying Collision Avoidance*) para CSMA (*Carrier Sense Multiple Access*)
- Selección de la frecuencia de canal
- Transmisión y recepción de datos

En el estándar también se proporcionan las especificaciones para la transmisión en las distintas bandas de frecuencia donde puede trabajar. Se muestra en la tabla 2.2 dichas especificaciones centradas en la frecuencia de 2.4 GHz ya que es la que resulta de mayor interés (es la más utilizada por ser común en la normativa europea y americana).

| **Frecuencia capa física (MHz)** | **Banda de frecuencia (MHz)** | **Parámetros de espectro expandido** | | **Parámetros del flujo de datos** | | |
|---|---|---|---|---|---|---|
| | | *Tasa de chips (kchips/s)* | *Modulación* | *Tasa de bit (kbps)* | *Tasa de símbolo (ksimbolo/s)* | *Símbolos* |
| 2450 | 2400-2483,5 | 2000 | O-QPSK | 250 | 62,5 | 16 ortogonal |

**Tabla 2.2 – Especificaciones para la transmisión en la banda de 2,4 GHz**

## 2.3. Estructura del transmisor

Antes de que se detallen todos los bloques que conforman el diseño digital del transmisor ZigBee en el siguiente capítulo, se muestra en la figura 2.1 un diagrama con la estructura que se va a implementar. Se indica también el ancho de los buses del flujo de datos (en número de bits) y las tasas de transmisión que han de cumplirse según establece el estándar 802.15.4 a lo largo de la etapa del transmisor.

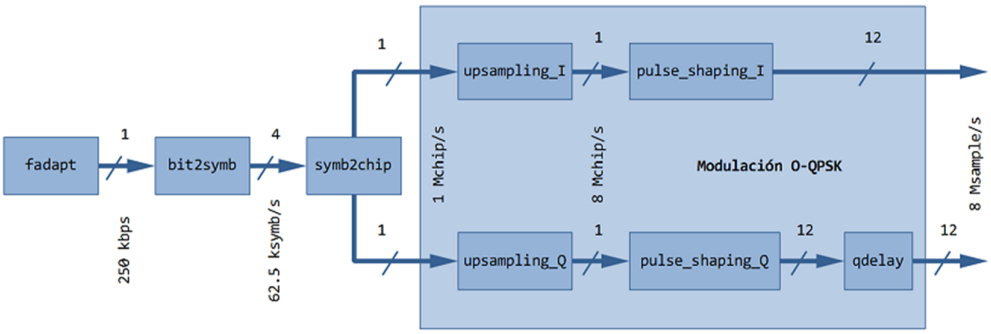



**Figura 2.1 – Esquema de bloques del transmisor ZigBee**

Queda presentada la solución propuesta para diseñar un transmisor inalámbrico preparado para funcionar siguiendo el protocolo de comunicación ZigBee. En los siguientes capítulos se detalla todo el proceso de diseño de los bloques que forman el transmisor así como las técnicas de verificación que se han empleado con el objeto de asegurar un diseño que cumpla con los requisitos de transmisión de señal de la capa física del protocolo ZigBee (capítulo 3). Seguidamente, se realizan pruebas de inyección de fallos sobre el diseño para conocer la robustez estructural del transmisor y del propio sistema inalámbrico completo (capítulo 4). Finalmente, se muestran los resultados obtenidos tras realizar las pruebas de inyección de fallos (capítulo 5) y se expresan las conclusiones que se extraen tras analizar todos los resultados así como se describen los trabajos futuros que pueden realizarse a raíz de los resultados que ofrece este trabajo (capítulo 6).

# Capítulo 3

# *Desarrollo del Transmisor*

*Este capítulo se considera el más relevante ya que en él se justifica, describe y detalla el diseño del transmisor inalámbrico, elemento principal del desarrollo del presente trabajo. Se contemplan aspectos como el diseño a alto nivel del transmisor, las principales consideraciones establecidas para su implementación y una primera prueba de verificación a nivel funcional.*

## 3.1. Diseño de la arquitectura hardware

El diseño del transmisor, en referencia al hardware que implementa la capa física del protocolo ZigBee, consta de ciertos módulos que forman la arquitectura del transmisor mostrada en el capítulo anterior y que se describen en el presente capítulo, pero para este trabajo se han diseñado además una serie de librerías propias con el fin de realizar una verificación funcional del propio transmisor. Dicha verificación, la cual se detalla en este capítulo, realiza una comparación de salidas o *crosscheck* entre el funcionamiento de cada bloque del transmisor comparando un modelo en Matlab y el diseño en VHDL.

Hay que indicar que tanto dichas librerías de verificación como el entorno de pruebas no han sido desarrolladas por el alumno exclusivamente, aunque sí ha colaborado en el estudio inicial y parte de las etapas de desarrollo y pruebas de dichas librerías junto con otros participantes del proyecto Edelweiss.

### 3.1.1. Consideraciones de implementación

Antes de comenzar a describir los diferentes módulos que conforman el diseño hardware del transmisor, es importante destacar ciertas consideraciones que se han tenido en cuenta antes de realizar el trabajo. Estos requisitos simplemente ayudan a que el código en VHDL sea más legible y aporta una mayor facilidad a la hora de realizar futuras modificaciones si fueran necesarias. Además, el primer requisito que se expone es obligatorio para tener un control total en el proceso de inyección de fallos como se verá en el capítulo 4.

- No se instanciará ningún módulo propiedad de Xilinx. El empleo de primitivas de Xilinx hace que dichos módulos sean 'cajas negras' en el diseño, impidiendo la observación de sus registros y su estructura interna. Con ello se pierde control a la hora de inyectar fallos en dichas primitivas y en este trabajo se busca tener una total transparencia en el 100% del diseño en cuanto a observación de registros principalmente. Por tanto, cada módulo se implementa con un código VHDL propio. Lo que se trata de conseguir es prototipar un futuro ASIC y por ello es incompatible el uso de cualquier primitiva de código cerrado desarrollada por terceros.

- Las máquinas de estado siempre considerarán una evolución segura a través de la sentencia '`when others =>`' en la lógica de decodificación de estados. Esto evita posibles situaciones de bloqueo y poder volver a un estado conocido.

- Todas las declaraciones de componentes, de tipos de datos y de aquellas constantes que sean comunes a varios módulos se agruparán en un mismo 'package' VHDL común a todo el proyecto. En este trabajo hay una para los bloques del diseño digital del transmisor ZigBee denominada '`edelweiss_common`', y otra para los módulos empleados en los tests de verificación denominada '`vhdl_verification`'.

- Para tener una implementación aislada de la capa física hay que generar previamente unas tramas de unidades de datos de servicio, las denominadas como '**PSDU**' (*Physical Service Data Unit*), que emulan la información proveniente de capas superiores. Con este fin, parte del equipo del proyecto ha desarrollado un bloque denominado '**datagen**', el cual recibe como parámetros de entrada el número de PSDUs a generar y la longitud en bytes la propia PSDU. Este bloque proporciona como salida una secuencia de bits que se leen de un fichero especificado mediante un *generic* en la entidad del bloque.

- El esquema del transmisor ZigBee visto en el capítulo anterior muestra una serie de tasas de transmisión de datos entre distintos bloques que hay que cumplir para que funcione el protocolo de comunicación de manera correcta. En concreto hay tres cambios de tasas dentro del esquema y por ello hay tres bloques que deben incorporar un mecanismo que sea capaz de gestionar el flujo de bits para alcanzar la tasa necesaria a la salida. Con este fin, se han implementado unos contadores (en los bloques afectados) que según un parámetro denominado *THROUGHPUT*[3], llevan la cuenta de los ciclos de reloj necesarios para ofrecer un flujo de datos a la salida con la tasa de bits que le corresponde. El hecho de que sea parametrizable, permite adaptar de una forma sencilla los bloques desarrollados, a diferentes frecuencias de reloj y tasas de datos.

### 3.1.2. Implementación de la arquitectura

Este apartado detalla el principal trabajo que ha desarrollado el alumno. Se describen todos los módulos que implementan la capa física del transmisor. Para visualizar con una mayor claridad el sistema a diseñar, se muestra a continuación el esquema de bloques completo:

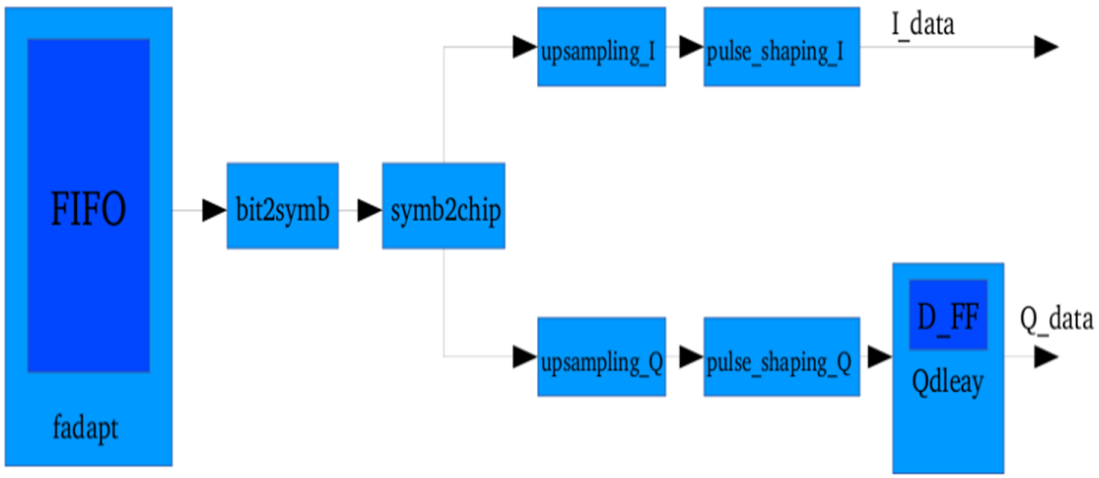



**Figura 3.1 - Esquema de la capa física del transmisor**

[3] Se define el *throughput* como el número de ciclos de reloj tras los cuales el bloque da una nueva salida.

### 3.1.2.1. FADAPT

Este bloque se encarga de la conformación de la trama de datos de la capa física, denominada como '**PPDU**' (*Physical Protocol Data Unit*). La trama sigue el siguiente formato:

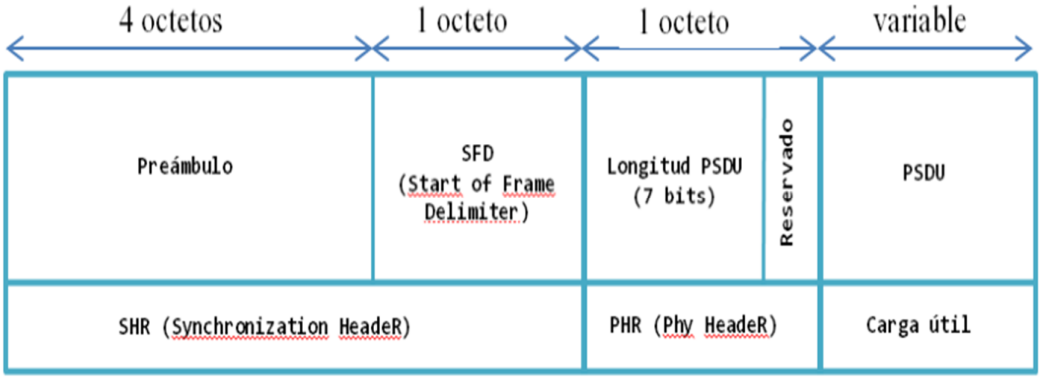



**Figura 3.2 – Trama PPDU**

La trama comienza con cuatro octetos con todos sus bits puestos a valor '0' lógico, formando el preámbulo y a continuación un octeto indicador de comienzo de trama o SFD. El valor del campo SFD se corresponde con el valor 'A7' en hexadecimal tal y como se muestra en la siguiente tabla:

| **Bit** | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| **Valor** | **1** | **0** | **1** | **0** | **0** | **1** | **1** | **1** |

**Tabla 3.1 – Valores del campo SFD**

Con esto se tiene completa la cabecera de sincronización o SHR. Posteriormente, se encuentra la cabecera de la capa física o PHR formada por un único octeto que consta de 7 bits que indican el tamaño de la trama de datos de servicio o '**PSDU**' (*Physical Service Data Unit*) en octetos, más un bit reservado. Finalmente se transmite la carga útil proveniente de las capas superiores cuya información viene integrada en la PSDU.

La manera en la que funciona este bloque es sencilla. Contiene un bloque que trabaja como una FIFO y almacena los datos de la PSDU que van llegando a la entrada del sistema. Una vez la FIFO empieza a llenarse de datos, se pone en marcha una máquina de estados que va formando la trama campo a campo de manera que a la salida ofrece al siguiente bloque tanto la trama PPDU como una señal de dato válido. Esta señal funciona como una especie de habilitación y se ha diseñado todo el sistema de manera que un bloque no comienza a funcionar hasta que recibe la habilitación del bloque que le precede.

Se detalla a continuación tanto la descripción de la entidad en VHDL del bloque FADAPT como de la FIFO:

```
entity fadapt is
  generic(
    PSDU_LENGTH: integer := 127;  -- PSDU length (bytes)
    THROUGHPUT:  integer := 416   -- clock cycles to reach valid data
  );
  port(
```

```
    clk:          in std_logic;     -- Clock input
    rst:          in std_logic;     -- Active high reset
    psdu:         in std_logic;     -- Data input
    psdu_valid:   in std_logic;     -- Data input is valid
    ppdu:         out std_logic;    -- Data output
    ppdu_valid:   out std_logic;    -- Data output is valid
    full:         out std_logic     -- FIFO full flag
  );
end fadapt;


entity fifo is
  generic(
    constant DATA_WIDTH : positive := 8;                -- FIFO data width (bits)
    constant FIFO_DEPTH : positive := 256               -- FIFO depth (mem positions)
  );
  port(
    clk:     in std_logic;                              -- Clock input
    rst:     in std_logic;                              -- Active high reset
    WriteEn: in std_logic;                              -- Write enable signal
    DataIn:  in std_logic_vector(DATA_WIDTH-1 downto 0); -- Data input bus
    ReadEn:  in std_logic;                              -- Read enable signal
    DataOut: out std_logic_vector(DATA_WIDTH-1 downto 0);-- Data output bus
    empty:   out std_logic;                             -- FIFO empty flag
    full:    out std_logic                              -- FIFO full flag
  );
end fifo;
```

### 3.1.2.2. BIT2SYMB

Este bloque funciona como un registro con entrada serie y salida paralelo. Su trabajo consiste en ir agrupando los bits provenientes de la PPDU de cuatro en cuatro y así obtener los diferentes símbolos que se utilizan en este sistema de transmisión.

Consta principalmente de un contador y una máquina de estados que comienzan a funcionar en cuanto recibe un primer dato. Se forman símbolos de cuatro bits que se envían al siguiente bloque junto con su correspondiente señal de símbolo válido. Se detalla a continuación tanto la descripción de la entidad en VHDL del bloque BIT2SYMB:

```
entity bit2symb is
  port(
    clk:          in std_logic;                     -- Clock input
    rst:          in std_logic;                     -- Active high reset
    ppdu:         in std_logic;                     -- Data input
    ppdu_valid:   in std_logic;                     -- Data input is valid
    symb:         out std_logic_vector(3 downto 0); -- Data output bus
    symb_valid:   out std_logic                     -- Data output is valid
     );
end bit2symb;
```

### 3.1.2.3. SYMB2CHIP

Como ya se ha comentado, el protocolo ZigBee emplea la técnica de espectro expandido para realizar la transmisión de información. Uno de los métodos para generar una señal con esta característica consiste en asignar una secuencia pseudoaleatoria o PN (*Pseudo Noise*) a cada uno de los símbolos a transmitir.

Para este trabajo, se tienen 16 símbolos diferentes y a cada uno de ellos se le va a aplicar una secuencia PN de 32 bits tal y como se indica en la siguiente tabla:

| Nº símbolo | Símbolo binario (b0, b1, b2, b3) | Secuencia de expansión [0, ..., 31] |
|---|---|---|
| 0 | 0000 | 11011001110000110101001000101110 |
| 1 | 1000 | 11101101100111000011010100100010 |
| 2 | 0100 | 00101110110110011100001101010010 |
| 3 | 1100 | 00100010111011011001110000110101 |
| 4 | 0010 | 01010010000101110110110011100011 |
| 5 | 1010 | 00110101001000101110110110011100 |
| 6 | 0110 | 11000011010100100010111011011001 |
| 7 | 1110 | 10011100001101010010001011101101 |
| 8 | 0001 | 10001100100101100000011101111011 |
| 9 | 1001 | 10111000110010010110000001110111 |
| 10 | 0101 | 01111011100011001001011000000111 |
| 11 | 1101 | 01110111101110001100100101100000 |
| 12 | 0011 | 00000111011110111000110010010110 |
| 13 | 1011 | 01100000011101111011100011001001 |
| 14 | 0111 | 10010110000001110111101110001100 |
| 15 | 1111 | 11001001011000000111011110111000 |

**Tabla 3.2 – Secuencias PN asociadas a cada símbolo**

Este bloque también se encarga de dividir la señal de información en dos flujos de bits, es decir; a partir de la recepción de los símbolos a la entrada, genera a la salida una componente en fase y una componente en cuadratura de la señal de información, denominadas como *chips*. El método empleado consiste en asignar los valores pares de la secuencia de expansión correspondiente (se numera al bit menos significativo con el 0 y al más significativo con el 31) a la componente en fase (I) y los impares a la componente en cuadratura (Q). El criterio que se ha establecido para el orden de la transmisión de los bits (*endianness*) es que el primer byte transmitido es el byte menos significativo y dentro de cada byte, el primer bit transmitido es el bit menos significativo.

Básicamente, el funcionamiento de este bloque está dirigido por una única máquina de estados que va asignando cada uno de los bits de la secuencia PN, que corresponde al símbolo que recibe a la entrada, a los chips de salida y así formar los flujos de bits de las componentes de fase y cuadratura. La descripción de la entidad en VHDL es la siguiente:

```
entity symb2chip is
  generic(
    THROUGHPUT:   integer := 104                    -- clock cycles to reach valid data
  );
  port(
    clk:           in std_logic;                    -- Clock input
    rst:           in std_logic;                    -- Active high reset
    symb:          in std_logic_vector(3 downto 0); -- Data input bus
    symb_valid:    in std_logic;                    -- Data input is valid
    i_data:        out std_logic;                   -- Data output (phase)
    i_data_valid:  out std_logic;                   -- Data output is valid (phase)
    q_data:        out std_logic;                   -- Data output (quadrature)
    q_data_valid:  out std_logic                    -- Data output is valid (quadrature)
  );
end symb2chip;
```

### 3.1.2.4. UPSAMPLING

Este bloque es el primero de los tres que conforman el proceso de modulación O-QPSK de la señal a transmitir. La tarea de este bloque es sobremuestrear las señales de entrada (componentes en fase y cuadratura) aplicando tantas muestras como coeficientes se empleen en el filtro de conformación de onda. Para este trabajo se ha decidido optar por una tasa de sobremuestreo de N=8, con lo que se aumenta x8 la tasa de transmisión de las señales a la salida. Como la señal de transmisión puede tener tres posibles valores [-1, 0, +1], la salida de este bloque viene dada por dos bits.

De manera gráfica, se muestra a continuación el proceso de sobremuestreo que se realiza en este bloque:


**Figura 3.3 – Sobremuestreo para N=8**

El diseño en VHDL consta de una máquina de estados que convierte un '1' lógico a la entrada como '01' (valor +1 de la señal de transmisión) y un '0' lógico como '11' (valor -1 de la señal de tansmisión). A su vez, un contador controla el número de ciclos de reloj necesarios para ir rellenando con muestras a cero ('00') con el fin de introducir el sobremuestreo que se ha definido. La entidad en VHDL de este bloque es la siguiente:

```
entity upsampling is
  generic(
    THROUGHPUT:     integer := 13;                    -- clock cycles to reach valid data
    N:              integer := 8                      -- number of samples
  );
  port(
    clk:            in std_logic;                     -- Clock input
    rst:            in std_logic;                     -- Active high reset
    upsin:          in std_logic;                     -- Data input
    upsin_valid:    in std_logic;                     -- Data input is valid
    upsout:         out std_logic_vector(1 downto 0); -- Data output bus
    upsout_valid:   out std_logic                     -- Data output is valid
  );
end upsampling;
```

### 3.1.2.5. PULSE_SHAPING

El filtro conformador de onda, diseñado por parte del equipo que se encarga del entorno en Matlab, se conoce como *half-sine*. Contiene ocho etapas (por este motivo se utiliza una tasa de sobremuestreo de N=8) y se aplica de forma paralela tanto al flujo de entrada de la componente en fase (I) como en cuadratura (Q).

Todos los coeficientes son valores positivos y para una ganancia en DC normalizada y tasa de sobremuestreo de N=8, se han obtenido los siguientes valores:

| Coeficiente | b0 | b1 | b2 | b3 | b4 | b5 | b6 | b7 |
|---|---|---|---|---|---|---|---|---|
| **Valor natural** | 0 | 0.0761 | 0.1405 | 0.1840 | 0.1988 | 0.1840 | 0.1405 | 0.0761 |
| **Valor cuantizado en 8 bits [0, ..., 255]** | 0 | 98 | 180 | 236 | 255 | 236 | 180 | 98 |

**Tabla 3.3 – Coeficientes del filtro conformador de onda**

El valor cuantizado se obtiene al asignar con el valor 255 al máximo coeficiente y el resto se calcula de forma proporcional. La estructura hardware que representa el funcionamiento de este filtro se representa en la siguiente figura:

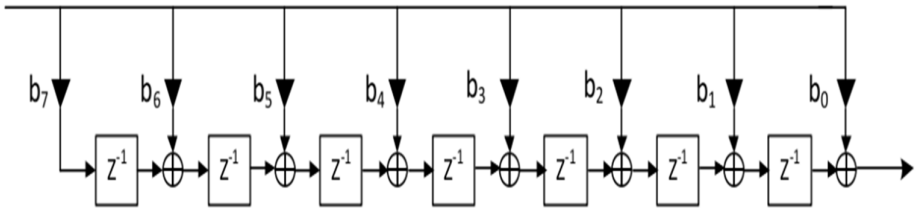



**Figura 3.4 – Estructura del filtro conformador de onda**

La finalidad de este filtro es cumplir con las especificaciones del estándar establecidas con la máscara de transmisión. Estas especificaciones definen una limitación de potencia transmitida fuera de los límites de la banda de frecuencias definida para cada canal. La conformación de la onda hace que los lóbulos laterales de la señal queden lo suficientemente atenuados para no afectar a los canales adyacentes.

Visto el diseño del filtro desde el punto de vista del procesamiento de señal mediante Matlab, en el dominio digital, el diseño en VHDL resulta más sencillo de realizar. Como los posibles valores de las entradas se encuentran en el conjunto [-1, 0, +1], los multiplicadores que aparecen en la estructura hardware del filtro (ver figura 3.4) se convierten en un sencillo bloque lógico que cambia el signo del coeficiente en el caso de que la entrada sea -1, mantiene el valor del coeficiente en el caso de que la entrada sea 1 o simplemente da un resultado nulo para el resto de los casos. De esta manera se simplifica el diseño y se optimizan recursos de la FPGA donde se vaya a implementar. El resto de elementos son sumadores y registros que retrasan las muestras un ciclo tal y como se indica en el esquema de la figura 3.4. Este es el segundo bloque que forma parte de la modulación O-QPSK cuya entidad en VHDL es la siguiente:

```
entity pulse_shaping is
  port(
    clk:             in std_logic;                     -- Clock input
    rst:             in std_logic;                     -- Active high reset
    filterin:        in std_logic_vector(1 downto 0);  -- Data input bus
    filterin_valid:  in std_logic;                     -- Data input is valid
    filterout:       out std_logic_vector(11 downto 0);-- Data output bus
    filterout_valid: out std_logic                     -- Data output is valid
    );
end pulse_shaping;
```

### 3.1.2.6. QDELAY

Se trata del tercer y último bloque que forma la modulación O-QPSK y el motivo de su uso viene dado como consecuencia del empleo de una modulación QPSK. Dicha modulación puede tomar cuatro valores de la fase a la vez para construir un símbolo de la constelación permitiendo que la fase de la señal salte hasta 180º. Estos saltos abruptos de fase en las componentes I y Q dan como resultado fuertes variaciones de su envolvente y hace que sea necesaria la inclusión de amplificadores con altas prestaciones de linealidad.

En la práctica se emplea una variante de la modulación QPSK que introduce una compensación o desviación de la sincronización entre las componentes en fase y cuadratura. Esta compensación se consigue retrasando a una de las componentes (en este trabajo se ha decidido por retrasar la componente compleja, Q) por medio periodo que equivale a un retraso de 4 muestras ya que se realiza un sobremuestreo de 8.

Gracias a esto, con la modulación O-QPSK se evitan las transiciones de 180º ya que ambas componentes nunca cambian a la vez, así se consigue una envolvente con menores variaciones que requiere unos requisitos de linealidad inferiores que en el caso de la QPSK.

El diseño en VHDL de este bloque se compone de la generación de cuatro instancias de un bloque básico de registro denominado d_ff (flip-flop tipo D). De esta manera las muestras que forman la componente en cuadratura van pasando por este bloque que consigue el retraso de 4 muestras necesario para la modulación O-QPSK tal y como se ha indicado.

Se muestra a continuación un esquemático que muestra la composición de este bloque:

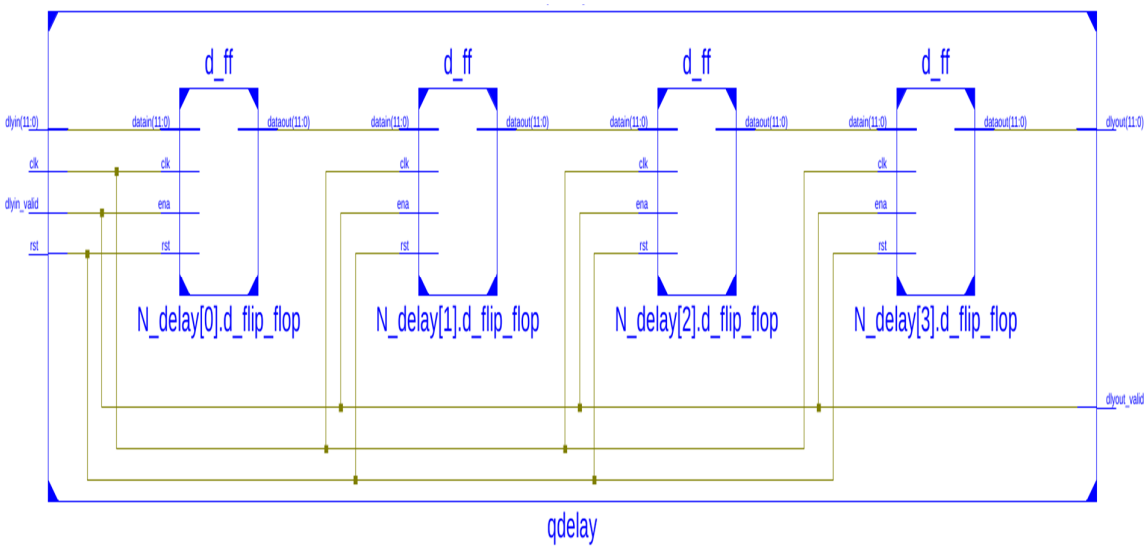



**Figura 3.5 – Estructura del bloque QDELAY**

Las entidades en VHDL que forman parte de este bloque son dos, por un lado el propio qdelay y por otro el módulo básico de registro d_ff:

```
entity qdelay is
  generic(
    INPUT_WIDTH:   integer := 12; -- Input data width
    OUTPUT_WIDTH:  integer := 12; -- Output data with
    N:             integer := 4   -- Number of samples to delay
  );
  port(
    clk:         in std_logic;                                      -- Clock input
    rst:         in std_logic;                                      -- Active high reset
    dlyin:       in std_logic_vector((INPUT_WIDTH-1) downto 0);   -- Data input bus
    dlyin_valid: in std_logic;                                      -- Data input is valid
    dlyout:      out std_logic_vector((OUTPUT_WIDTH-1) downto 0);-- Data output bus
    dlyout_valid:out std_logic                                      -- Data output is valid
  );


end qdelay;
entity d_ff is
  generic(
    DATA_WIDTH: integer:= 12 -- Input/output data width
  );
  port(
    clk:     in std_logic;                                          -- Clock input
    rst:     in std_logic;                                          -- Active high reset
    datain:  in std_logic_vector((DATA_WIDTH -1) downto 0);         -- Data input bus
    ena:     in std_logic;                                          -- Input enable
    dataout: out std_logic_vector((DATA_WIDTH -1) downto 0)         -- Data output bus
  );
end d_ff;
```

### 3.1.2.7. TOP_TX

Finalmente, el diseño del transmisor ZigBee en VHDL incluye a todos los bloques ya descritos dentro de un módulo denominado '**top_tx**', cuyo esquema general se muestra en la siguiente figura:

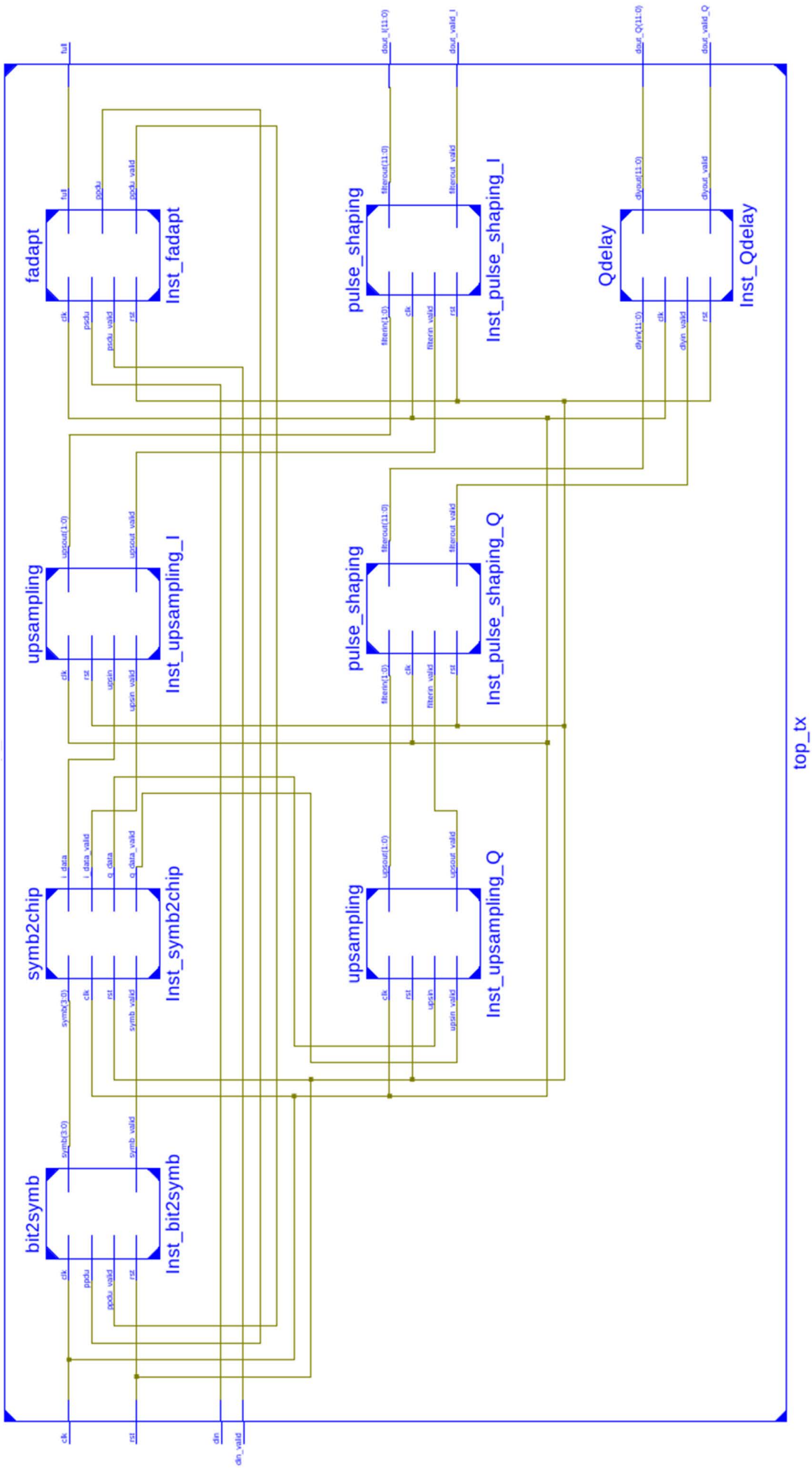



**Figura 3.6 – Diseño digital del transmisor ZigBee**

Como últimos datos a destacar sobre el diseño digital que se ha desarrollado, se muestra a continuación un resumen sobre la ocupación y frecuencia máxima de trabajo que se obtiene al implementar el diseño sobre una FGPA, en concreto es un modelo de la familia Virtex 5 (5vfx70tff1136-1) del fabricante Xilinx.

**Device utilization summary:**

```
Selected Device: 5vfx70tff1136-1

Slice Logic Utilization:
 Number of Slice Registers:            389   out of  44800     0%
 Number of Slice LUTs:                 480   out of  44800     1%
    Number used as Logic:              480   out of  44800     1%

Slice Logic Distribution:
 Number of LUT Flip Flop pairs used:   533
   Number with an unused Flip Flop:    144   out of    533    27%
   Number with an unused LUT:           53   out of    533     9%
   Number of fully used LUT-FF pairs:  336   out of    533    63%

IO Utilization:
 Number of IOs:                        31
 Number of bonded IOBs:                31    out of    640     4%

Specific Feature Utilization:
 Number of BUFG/BUFGCTRLs:             1     out of     32     3%
```

**Timing Summary:**

```
Speed Grade: -1

Minimum period: 8.122ns (Maximum Frequency: 123.122MHz)
Minimum input arrival time before clock: 4.336ns
Maximum output required time after clock: 3.389ns
Maximum combinational path delay: No path found
```

## 3.2. Verificación funcional

Una vez descrito el diseño digital del transmisor ZigBee desarrollado por el alumno, se pasa a detallar el procedimiento que se ha seguido para comprobar que cada uno de los bloques funciona de manera correcta. Como ya se ha indicado anteriormente, este trabajo forma parte de un proyecto en el que intervienen varias personas. Por un lado, un grupo dedicado al desarrollo del procesamiento de señal y modelado en Matlab del transmisor ZigBee, donde se ha analizado el espectro de la señal y comprobar que la señal O-QPSK transmitida cumple con las especificaciones del protocolo de comunicación. Por otro lado, un grupo de personas encargadas de desarrollar y testear el mismo transmisor empleando una codificación VHDL, con el objeto de tener un diseño digital al que poder realizar una prueba de inyección de fallos y analizar los resultados que se obtengan.

El alumno ha formado parte del grupo referente al ámbito digital donde su tarea principal ha sido el desarrollo del transmisor que se ha descrito en este capítulo pero también ha colaborado en el proceso de elaboración de tests con los que se ha

conseguido tener un diseño que funciona correctamente gracias al método de *crosscheck* que se ha utilizado y que se explica a continuación.

### 3.2.1. Crosscheck con modelo Matlab

El procedimiento que se ha seguido para tener un diseño digital que se comporte de igual manera que el modelo en Matlab, consiste en introducir a cada bloque por separado un mismo fichero de entradas conocidas (al modelo en Matlab y al diseño en VHDL implementado en la FPGA). El modelo en Matlab responde con unas salidas que se guardan en un archivo de texto (son valores en formato hexadecimal). Para el diseño digital se realizan una serie de *test benches*, los cuales contienen siempre a la unidad bajo test o UUT (*Unit Under Test*) y cuatro entidades no sintetizables que realizan la función de *crosscheck* según el archivo de salidas que el modelo Matlab ofrece para cada bloque.

| Entradas |
|---|
| 00 |
| 01 |
| 02 |
| 03 |

| Salidas |
|---|
| 00 |
| 00 |
| 00 |
| 00 |
| A7 |
| 04 |
| 00 |
| 01 |
| 02 |
| 03 |

**Tabla 3.4 – Ejemplo de valores de entrada y salida para el bloque FADAPT**

Las entidades no sintetizables que se han desarrollado en VHDL con objeto de poder realizar el *crosscheck* de las salidas del modelo en Matlab son las siguientes:

- **`Clkmanager:`** Este módulo simplemente se encarga de dar ciclos de reloj al simulador mientras el test se realiza y deja de dar ciclos cuando se activa una señal de entrada denominada '**`endsim`**', cuyo significado es que la librería que genera las entradas ha terminado de enviar todos los datos y por tanto, debe de finalizar el test y la simulación. También se encarga de manejar el *reset* global del sistema.

  La entidad en VHDL de este módulo es la siguiente:

```
entity clkmanager is
  generic (
    -- Period of generated clock
    CLK_PERIOD:             time      := 10 ns;
    -- Reset polarity
    RST ACTIVE VALUE:       std logic := '0';
    -- Number of cycles that reset will be asserted at the beginning of the simulation
    RST_CYCLES:             integer   := 10
  );
  port (
    -- clk stops changing when endsim='1', which effectively stops the simulation
    endsim:                 in  std_logic;
```

```
    -- Generated clock
    clk:                    out std_logic;
    -- Generated reset
    rst:                    out std_logic
  );
end clkmanager;
```

- **Datagen:** Este módulo se encarga de generar los datos de entrada a cada bloque. Realiza varias funciones, es un manejador de archivos que va leyendo los valores de entrada y los pasa al simulador para que pueda trabajar con ellos, puede comprobar que los datos de entrada se encuentran entre un conjunto de valores válidos y da error si no es así y además se encarga de activar la señal de fin de simulación al resto de módulos cuando termina de enviar el último dato. También es capaz de serializar los datos que lee del fichero de entrada. La entidad en VHDL es la siguiente:

```
entity datagen is
  generic(
    -- Print all internal details
    VERBOSE:                    boolean := false;
    -- Print debug info
    DEBUG:                      boolean := false;
    -- File where data is stored
    STIMULI_FILE:               string  := "../test/datagen_test.txt";
    -- Maximum hex chars for each input data
    STIMULI_NIBBLES:            integer := 2;
    -- Width of generated data
    DATA_WIDTH:                 integer := 8;
    -- Output one valid data each THROUGHPUT cycles
    THROUGHPUT:                 integer := 0;
    -- Output value when data is not valid
    INVALID_DATA:               datagen_invalid_data := unknown;
    -- Number of cycles between last data and assertion of endsim
    CYCLES_AFTER_LAST_VECTOR:   integer := 10
  );
  port(
    -- Align generated data to this clock
    clk:                in std_logic;
    -- Active high, tells datagen it can assert valid. Use for control-flow
    can_write:          in std_logic;
    -- Generated data
    data:               out std_logic_vector (DATA_WIDTH-1 downto 0);
    -- Active high, indicates data is valid
    valid:              out std_logic;
    -- Active high, tells the other sim processes to close their open files
    endsim:             out std_logic
  );
end datagen;
```

- **Datacompare:** Este es el módulo encargado de hacer la comparación entre las salidas de ambos modelos. Por un lado lee las salidas del archivo que se ha obtenido del modelo en Matlab y por otro lado se va comparando línea a línea con el resultado que da la simulación del diseño en VHDL. Si existe alguna discrepancia, se detiene la simulación y se da un mensaje de error indicando el valor que se ha obtenido y cuál era el valor esperado.

  Se ha desarrollado de esta manera en lugar de volcar las salidas a un fichero porque al comparar dichas salidas en tiempo de simulación, pueden ser detectados los errores mientras se ejecuta la simulación del bloque en cuestión. De esta manera se pueden obtener mensajes de error en el log del simulador justo en el instante en el que se detecta una discrepancia o error entre las salidas.

Esto permite una mayor facilidad y control de depuración del funcionamiento de los bloques, ya que se pueden ver los estados internos del circuito a la vez que se detectan los errores. Si esto se hiciera volcando las salidas a un fichero, toda la comparación es *offline*. De esta manera se han ido encontrando a lo largo de la etapa de diseño algunos errores, principalmente en el código VHDL, aunque en algunos casos también han sido detectados errores en el modelo en Matlab y así se ha llegado a obtener un transmisor ZigBee totalmente funcional en ambos modelos. La entidad en VHDL es la siguiente:

```
entity datacompare is
  generic(
    -- Allow to separate messages from different instances in SIMULATION
    SIMULATION_LABEL:     string := "datacompare";
    -- Print all internal details
    VERBOSE:              boolean := false;
    -- Print debug info
    DEBUG:                boolean := false;
    -- File where data is stored
    GOLD_OUTPUT_FILE:     string  := "../test/datacompare_test.txt";
    -- Maximum hex chars for each output data
    GOLD_OUTPUT_NIBBLES:  integer := 2;
    -- Width of generated data
    DATA_WIDTH:           integer := 8
  );
  port(
    -- Align generated data to this clock
    clk:                  in std_logic;
    -- Generated data
    data:                 in std_logic_vector (DATA_WIDTH-1 downto 0);
    -- Active high, indicates data is valid
    valid:                in std_logic;
    -- Active high, tells the process to close its open files
    endsim:               in std_logic
  );
end datacompare;
```

- **Throughputchecker:** Esta entidad es la encargada de supervisar que las señales de dato válido se activan según la tasa de transmisión de datos esperada en cada bloque. Es básicamente un contador de ciclos de reloj que compara si en el momento en el que corresponda se activa la señal de dato válida. Su entidad en VHDL es la siguiente:

```
entity throughputchecker is
  generic(
    -- To separate messages from different instances in SIMULATION
    SIMULATION_LABEL:     string  := "throughputchecker";
    -- Print debug info (developers only ;)
    DEBUG:                boolean := false;
    -- Wait cycles between valid data. Also know as throughput
    THROUGHPUT:           integer := 0
  );
  port(
    -- Align generated data to this clock
    clk:                  in std_logic;
    -- Active high, indicates data is valid
    valid:                in std_logic;
    -- Active high, tells the process that last data from datagen was sent
    endsim:               in std_logic
  );
end throughputchecker;
```

Antes de conocer el entorno de pruebas que se ha empleado, se muestra en la figura 3.7 un esquema básico de *test bench* con la idea de conocer mejor a lo que en el entorno de pruebas se define como CX (*crosscheck*).

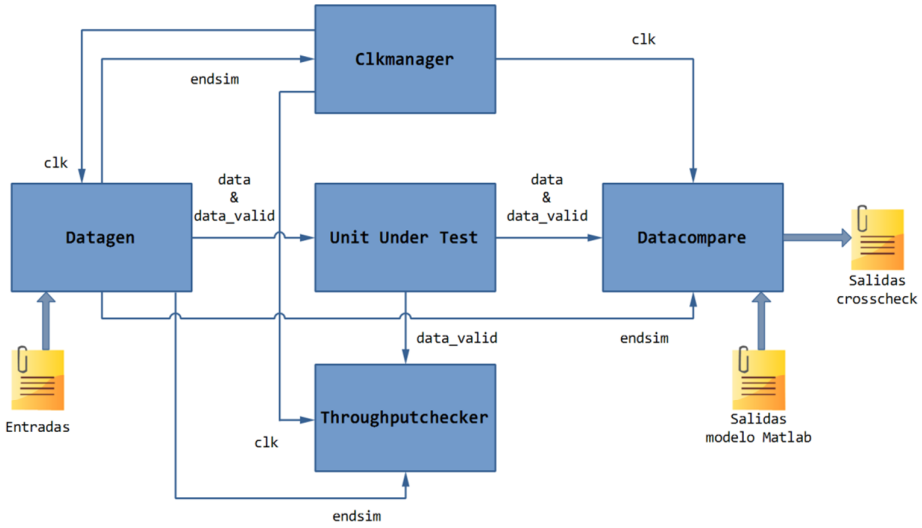



**Figura 3.7 – Esquema de un *test bench* básico**

## 3.2.2. Entorno de pruebas

De manera más gráfica, se muestra a continuación dónde se han realizado los diferentes puntos de verificación a lo largo del desarrollo del diseño digital del transmisor.

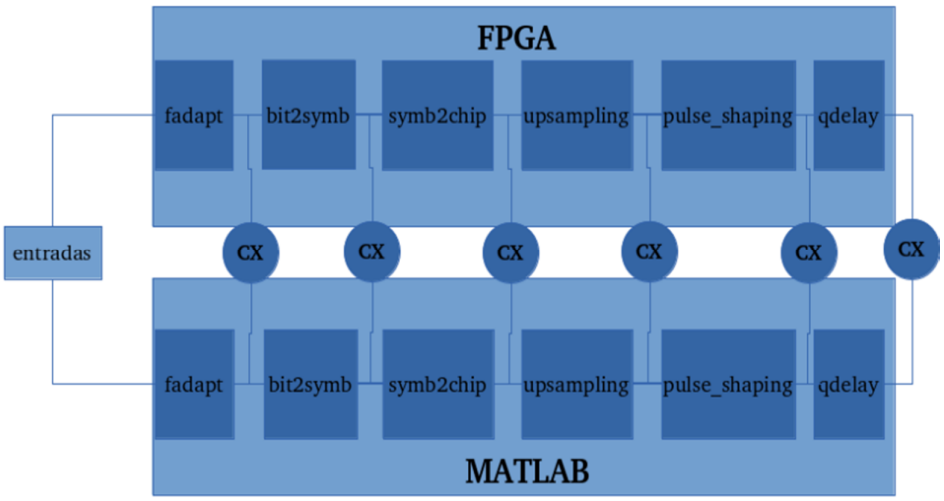



**Figura 3.8 – Esquema de *crosschecks***

Como se puede observar, se aplica un mismo fichero de entrada a ambos modelos y se realiza un *crosscheck* de las salidas de cada bloque, siendo la salida de un bloque la propia entrada al bloque siguiente. No se desarrolla un nuevo bloque hasta que el anterior ofrece un resultado de test positivo.

Este esquema es el que ha seguido el alumno durante la etapa de diseño del transmisor en VHDL, implementando el código en la FPGA ya mencionada anteriormente (Virtex 5: xc5vfx70tff1136-1) y verificando que los *test benches* para cada bloque dan los resultados esperados (no se producen errores y el número de datos correctos es mayor que cero).

Hay que destacar que todo el sistema de verificación que se ha descrito se ha desarrollado con herramientas de CMake (en un entorno Linux) para automatizar todo el proceso. Es decir, a partir de los ficheros fuente (.vhd) que forman el diseño digital, y mediante la ejecución de un comando llamado '`cmake`' se crea todo el proyecto ISE (herramienta de síntesis del código VHDL y rutado para la FPGA objetivo), se crean los archivos de simulación para cada bloque y ejecutando el comando '`make test`' se ejecutan todos los *test benches* de manera automática. Se muestra a continuación una captura de este entorno de verificación que se ha desarrollado por parte del grupo dedicado al diseño digital del transmisor.

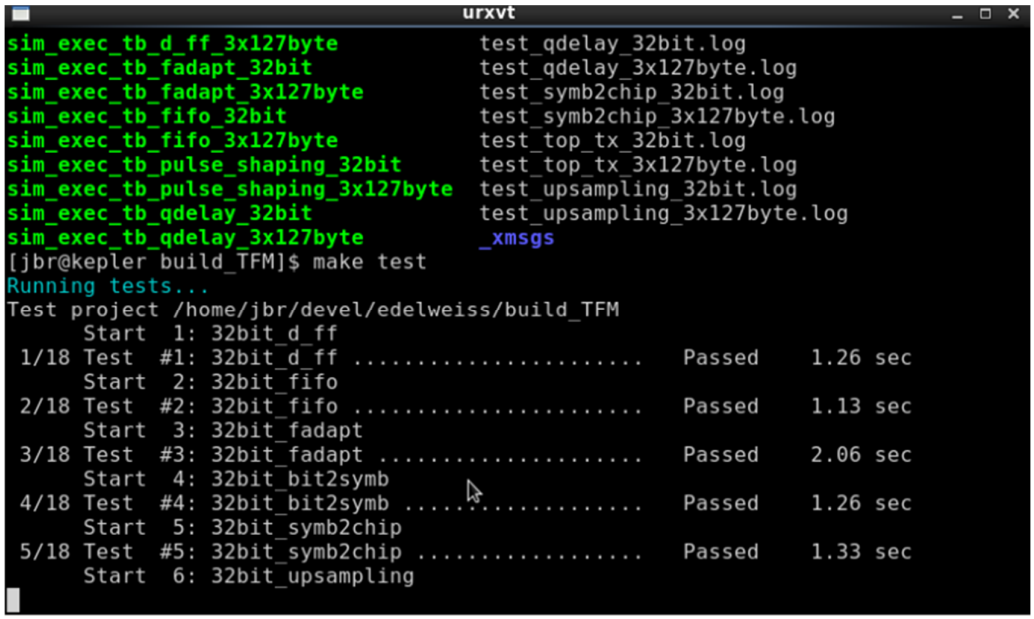



**Figura 3.9 – Entorno de verificación con la herramienta CMake**

Y el resultado final de todo el proceso de verificación que se obtiene es el siguiente:

```
[jbr@kepler build_TFM]$ make test
Running tests...
Test project /home/jbr/devel/edelweiss/build_TFM
      Start  1: 32bit_d_ff
 1/18 Test  #1: 32bit_d_ff .....................   Passed    1.26 sec
      Start  2: 32bit_fifo
 2/18 Test  #2: 32bit_fifo .....................   Passed    1.13 sec
      Start  3: 32bit_fadapt
```

```
 3/18 Test  #3: 32bit_fadapt .....................   Passed    2.06 sec
      Start  4: 32bit_bit2symb
 4/18 Test  #4: 32bit_bit2symb ...................   Passed    1.26 sec
      Start  5: 32bit_symb2chip
 5/18 Test  #5: 32bit_symb2chip ..................   Passed    1.33 sec
      Start  6: 32bit_upsampling
 6/18 Test  #6: 32bit_upsampling .................   Passed    1.27 sec
      Start  7: 32bit_pulse_shaping
 7/18 Test  #7: 32bit_pulse_shaping ..............   Passed    1.32 sec
      Start  8: 32bit_qdelay
 8/18 Test  #8: 32bit_qdelay .....................   Passed    1.26 sec
      Start  9: 32bit_top_tx
 9/18 Test  #9: 32bit_top_tx .....................   Passed    1.49 sec
      Start 10: 3x127byte_d_ff
10/18 Test #10: 3x127byte_d_ff ...................   Passed    7.36 sec
      Start 11: 3x127byte_fifo
11/18 Test #11: 3x127byte_fifo ...................   Passed    1.99 sec
      Start 12: 3x127byte_fadapt
12/18 Test #12: 3x127byte_fadapt .................   Passed    8.31 sec
      Start 13: 3x127byte_bit2symb
13/18 Test #13: 3x127byte_bit2symb ...............   Passed    6.26 sec
      Start 14: 3x127byte_symb2chip
14/18 Test #14: 3x127byte_symb2chip ..............   Passed    8.16 sec
      Start 15: 3x127byte_upsampling
15/18 Test #15: 3x127byte_upsampling .............   Passed    7.50 sec
      Start 16: 3x127byte_pulse_shaping
16/18 Test #16: 3x127byte_pulse_shaping ..........   Passed    8.53 sec
      Start 17: 3x127byte_qdelay
17/18 Test #17: 3x127byte_qdelay .................   Passed    7.63 sec
      Start 18: 3x127byte_top_tx
18/18 Test #18: 3x127byte_top_tx .................   Passed   17.89 sec

100% tests passed, 0 tests failed out of 18

Total Test time (real) =  85.93 sec
```

Como se puede observar, se han realizado dos tipos de conjuntos de test según el tamaño de la carga útil. Por un lado, se ha utilizado una única trama de datos muy pequeña (32 bits, 4 octetos de carga útil) para probar la funcionalidad básica del diseño. Por otro lado, se han empleado tres tramas de tamaño máximo (127 bytes) de la carga útil con objeto de probar un caso más cercano al funcionamiento normal del sistema. Hay que tener en cuenta que al procesar tres tramas se comprueban que al terminar de procesar la primera trama completa, las máquinas de estado permanecen en los estados correctos que les permitan estar preparadas para procesar la siguiente trama. Esto es importante ya que las condiciones iniciales de cada trama son diferentes, la primera trama viene de un *reset* inicial del sistema mientras que al procesar la segunda se viene de la primera trama procesada. El hecho de incluir una tercera trama es simplemente para hacer un test un poco más exigente.

Cabe resaltar que el tiempo esperado para realizar las 18 pruebas no llega a 90 segundos, teniendo en cuenta que el tiempo medio de simulación con el simulador ISim (que incorpora la herramienta ISE con la que se ha desarrollado el diseño digital) es de un minuto por cada bloque (al comienzo del trabajo se hizo de esta manera), se puede decir que el hecho de desarrollar una automatización mediante la herramienta CMake demuestra que es de gran utilidad y acelera en gran medida todo el proceso de verificación.

### 3.2.3. Code Coverage

Existen otros tipos de verificación que pueden realizarse a un trabajo de este tipo y de hecho se ha trabajado con otro simulador, en concreto el VSIM de Mentor Graphics, que ofrece la posibilidad de analizar el propio código VHDL que se ha elaborado y saber cuál es la cobertura real del código que ha sido probado. Este tipo de verificación se denomina Code Coverage y es el primer paso de los siete que se plantean como manera de alcanzar un *test bench* definitivo en lo referente a las capacidades de verificación sobre una FPGA [Ref. 3.1].

Existen cinco tipos de Code Coverage que se analizan dentro de un código, el conjunto de los cinco tipos busca que el 100% de líneas de código sean probadas y no permanezcan líneas de código residuales que no se estén utilizando y puedan generar algún tipo de error o simplemente que puedan dificultar el seguimiento del código en futuras revisiones.

El objetivo principal de este tipo de verificación es tratar de conseguir una cobertura tanto del código fuente del transmisor como de los *test benches*, lo más cercano al 100%, lo que indicará que se trata de un código donde cada línea está probada y tiene su función perfectamente establecida. Los tipos de Code Coverage son los siguientes:

- **`Statement:`** Trata de verificar que todas las declaraciones han sido ejecutadas durante la simulación, tomando como declaración toda línea de código que termine en un punto y coma.

- **`Branch:`** Verifica que todas las ramas alternativas que aparecen en cada sentencia 'CASE' o 'IF' del código han sido ejecutadas.

- **`Condition:`** Verifica si en cada condición de una sentencia 'CASE' o 'IF' se evalúa la sub-expresión que le acompaña.

- **`Expression:`** Se asegura de que todas las sub-expresiones que haya en las asignaciones del código se ejecuten. Ayuda a identificar sub-expresiones que se encuentren dentro de asignaciones concurrentes y no se hayan llegado a ejecutar.

- **`Finite State Machine:`** Se asegura de visitar todos los estados y seguir sus correspondientes transiciones dentro de una FSM.

Una vez finalizada la verificación funcional con el *crosscheck* correcto en todos los bloques del diseño en VHDL del transmisor ZigBee, se pasa a realizar una generación de base de datos de Code Coverage al código para analizarlo y conocer el porcentaje de cobertura que se ha alcanzado.

El resultado que ofrece el simulador VSIM es una presentación de los resultados asociados a los bloques en formato html, con lo que mediante un navegador web es posible ir seleccionando el bloque que se quiera analizar.

Para este trabajo, el resultado del Code Coverage es el siguiente:

- **Bloque FADAPT: 91.59%**

## Questa Design Coverage

**Scope: /tb_fadapt/uut**

**Coverage Summary By Instance:**

| Scope | TOTAL | Statement | Branch | FEC Condition | FSM State | FSM Trans | Assertion |
|---|---|---|---|---|---|---|---|
| TOTAL | 73.47% | 93.33% | 90.69% | 0.00% | 100.00% | 66.66% | 100.00% |
| uut | 91.59% | 91.37% | 91.66% | -- | 100.00% | 66.66% | 100.00% |
| fadapt_fifo | 63.15% | 100.00% | 89.47% | 0.00% | -- | -- | -- |

**Local Instance Coverage Details:**

| Total Coverage: | | | | | 88.67% | 91.59% |
|---|---|---|---|---|---|---|
| **Coverage Type** | **Bins** | **Hits** | **Misses** | **Weight** | **% Hit** | **Coverage** |
| Statements | 58 | 53 | 5 | 1 | 91.37% | **91.37%** |
| Branches | 24 | 22 | 2 | 1 | 91.66% | **91.66%** |
| FSMs | 20 | 15 | 5 | 1 | 75.00% | **83.33%** |
| States | 5 | 5 | 0 | 1 | 100.00% | **100.00%** |
| Transitions | 15 | 10 | 5 | 1 | 66.66% | **66.66%** |
| Assertions | 4 | 4 | 0 | 1 | 100.00% | **100.00%** |

**Recursive Hierarchical Coverage Details:**

| Total Coverage: | | | | | 87.67% | 73.47% |
|---|---|---|---|---|---|---|
| **Coverage Type** | **Bins** | **Hits** | **Misses** | **Weight** | **% Hit** | **Coverage** |
| Statements | 75 | 70 | 5 | 1 | 93.33% | **93.33%** |
| Branches | 43 | 39 | 4 | 1 | 90.69% | **90.69%** |
| FEC Conditions | 4 | 0 | 4 | 1 | 0.00% | **0.00%** |
| FSMs | 20 | 15 | 5 | 1 | 75.00% | **83.33%** |
| States | 5 | 5 | 0 | 1 | 100.00% | **100.00%** |
| Transitions | 15 | 10 | 5 | 1 | 66.66% | **66.66%** |
| Assertions | 4 | 4 | 0 | 1 | 100.00% | **100.00%** |

**Figura 3.10 – Resultado de Code Coverage para el bloque fadapt**

- **Bloque BIT2SYMB: 91.80%**

## Questa Design Coverage

**Scope: /tb_bit2symb/uut**

**Local Instance Coverage Details:**

| Total Coverage: | | | | | 86.00% | 91.80% |
|---|---|---|---|---|---|---|
| **Coverage Type** | **Bins** | **Hits** | **Misses** | **Weight** | **% Hit** | **Coverage** |
| Statements | 31 | 26 | 5 | 1 | 83.87% | **83.87%** |
| Branches | 12 | 10 | 2 | 1 | 83.33% | **83.33%** |
| FSMs | 6 | 6 | 0 | 1 | 100.00% | **100.00%** |
| States | 2 | 2 | 0 | 1 | 100.00% | **100.00%** |
| Transitions | 4 | 4 | 0 | 1 | 100.00% | **100.00%** |
| Assertions | 1 | 1 | 0 | 1 | 100.00% | **100.00%** |

**Figura 3.11 – Resultado de Code Coverage para el bloque bit2symb**

- **Bloque SYMB2CHIP: 82.25%**

**Questa Design Coverage**

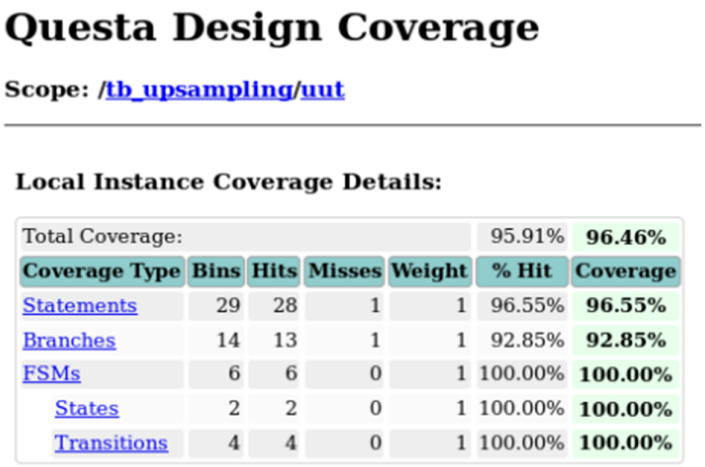
**Scope: /tb_symb2chip/uut**

**Local Instance Coverage Details:**

| Total Coverage: | | | | | 95.80% | **82.25%** |
|---|---|---|---|---|---|---|
| **Coverage Type** | **Bins** | **Hits** | **Misses** | **Weight** | **% Hit** | **Coverage** |
| Statements | 125 | 123 | 2 | 1 | 98.40% | **98.40%** |
| Branches | 29 | 27 | 2 | 1 | 93.10% | **93.10%** |
| Conditions | 2 | 1 | 1 | 1 | 50.00% | **50.00%** |
| UDP | -- | -- | -- | -- | -- | **Excluded** |
| FEC | 2 | 1 | 1 | 1 | 50.00% | **50.00%** |
| FSMs | 11 | 9 | 2 | 1 | 81.81% | **87.50%** |
| States | 3 | 3 | 0 | 1 | 100.00% | **100.00%** |
| Transitions | 8 | 6 | 2 | 1 | 75.00% | **75.00%** |

**Figura 3.12 – Resultado de Code Coverage para el bloque symb2chip**

- **Bloque UPSAMPLING: 96.46%**



**Questa Design Coverage**

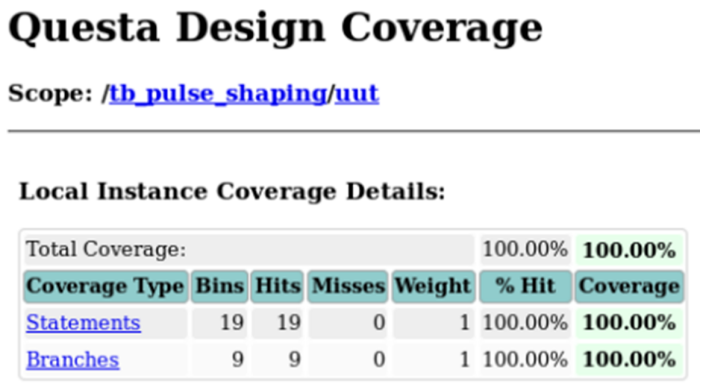
**Scope: /tb_upsampling/uut**

**Local Instance Coverage Details:**

| Total Coverage: | | | | | 95.91% | **96.46%** |
|---|---|---|---|---|---|---|
| **Coverage Type** | **Bins** | **Hits** | **Misses** | **Weight** | **% Hit** | **Coverage** |
| Statements | 29 | 28 | 1 | 1 | 96.55% | **96.55%** |
| Branches | 14 | 13 | 1 | 1 | 92.85% | **92.85%** |
| FSMs | 6 | 6 | 0 | 1 | 100.00% | **100.00%** |
| States | 2 | 2 | 0 | 1 | 100.00% | **100.00%** |
| Transitions | 4 | 4 | 0 | 1 | 100.00% | **100.00%** |

**Figura 3.13 – Resultado de Code Coverage para el bloque upsampling**

- **Bloque PULSE_SHAPING: 100%**



**Questa Design Coverage**

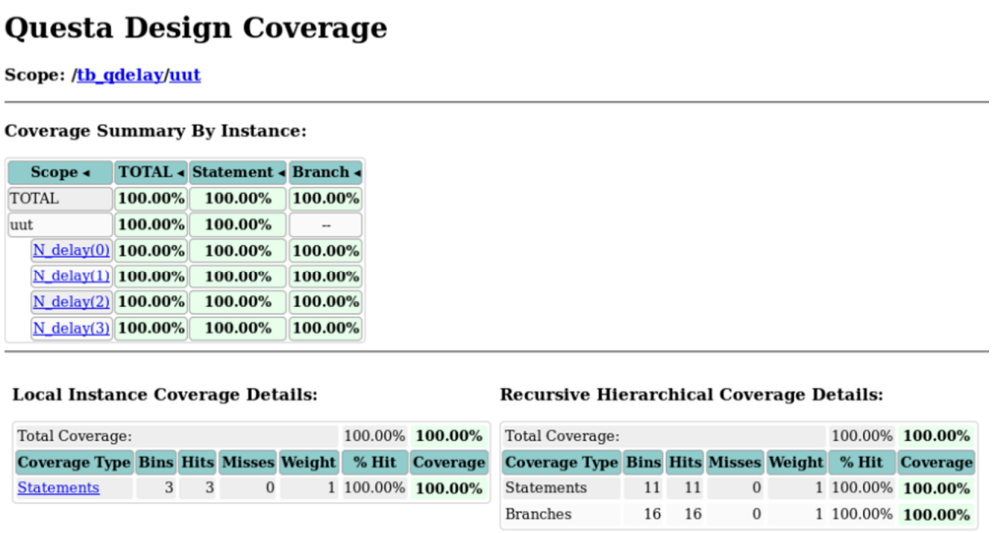
**Scope: /tb_pulse_shaping/uut**

**Local Instance Coverage Details:**

| Total Coverage: | | | | | 100.00% | **100.00%** |
|---|---|---|---|---|---|---|
| **Coverage Type** | **Bins** | **Hits** | **Misses** | **Weight** | **% Hit** | **Coverage** |
| Statements | 19 | 19 | 0 | 1 | 100.00% | **100.00%** |
| Branches | 9 | 9 | 0 | 1 | 100.00% | **100.00%** |

**Figura 3.14 – Resultado de Code Coverage para el bloque pulse_shaping**

- **Bloque QDLAY: 100%**



**Questa Design Coverage**

**Scope: /tb_qdelay/uut**

**Coverage Summary By Instance:**

| Scope | TOTAL | Statement | Branch |
|---|---|---|---|
| TOTAL | 100.00% | 100.00% | 100.00% |
| uut | 100.00% | 100.00% | -- |
| N_delay(0) | 100.00% | 100.00% | 100.00% |
| N_delay(1) | 100.00% | 100.00% | 100.00% |
| N_delay(2) | 100.00% | 100.00% | 100.00% |
| N_delay(3) | 100.00% | 100.00% | 100.00% |

**Local Instance Coverage Details:**

| Total Coverage: | | | | | 100.00% | 100.00% |
|---|---|---|---|---|---|---|
| **Coverage Type** | **Bins** | **Hits** | **Misses** | **Weight** | **% Hit** | **Coverage** |
| Statements | 3 | 3 | 0 | 1 | 100.00% | 100.00% |

**Recursive Hierarchical Coverage Details:**

| Total Coverage: | | | | | 100.00% | 100.00% |
|---|---|---|---|---|---|---|
| **Coverage Type** | **Bins** | **Hits** | **Misses** | **Weight** | **% Hit** | **Coverage** |
| Statements | 11 | 11 | 0 | 1 | 100.00% | 100.00% |
| Branches | 16 | 16 | 0 | 1 | 100.00% | 100.00% |

**Figura 3.15 – Resultado de Code Coverage para el bloque qdelay**

Queda por aclarar que el dato que aparece en color rojo en el módulo *fadapt* se debe a la FIFO, ya que en un funcionamiento normal con los valores de *throughput* nominales la FIFO no se llena, es por lo que las condiciones internas para activar esta bandera (*full*) no se están estimulando y por tanto no se prueban.

En vista de los resultados obtenidos se puede decir que en general se ha conseguido una buena cobertura en el código VHDL, destacando los bloques *pulse_shaping* y *qdelay* que por su sencillez se consigue tener un 100% de cobertura. El resto de bloques se encuentran por encima del 90% de cobertura a excepción del bloque *symb2chip* que tan sólo alcanza un 82.25%. Uno de los principales motivos por los que no se tiene un mayor porcentaje de cobertura es que no se prueban todas las ramas '`when others =>`' de las sentencias *CASE*. Estas ramas son redundantes y se han añadido por la razón que se indica en el apartado de consideraciones de implementación al comienzo de este capítulo, ya que es prioritario tener una evolución segura y que la FSM no tenga posibilidad de quedarse bloqueada si recibe un SEU en los bits que codifican el estado. El código que se ha desarrollado es por definición redundante y por ello no puede verse en el análisis de Code Coverage, ya que durante la verificación funcional no se inyectan SEUs, con lo cual el estado de una FSM siempre será uno de los estados válidos descritos en el código VHDL del bloque en cuestión.

Una de las tareas futuras que se pretende realizar consiste en profundizar acerca del Code Coverage y las técnicas que se pueden emplear para mejorar la cobertura y obtener un mayor nivel de verificación del código VHDL que se ha desarrollado para el transmisor ZigBee.

# Capítulo 4

# *Análisis de Inyección de Fallos*

*Una vez descrito el diseño digital que se ha desarrollado para el transmisor ZigBee y comprobado, mediante un proceso de verificación, que funciona de manera correcta cumpliendo los requisitos que el protocolo de comunicación establece, es momento de ir un paso más lejos y aplicarle al diseño una prueba de inyección de fallos para tratar de simular su comportamiento en un entorno hostil de radiación externa. Se describe en este capítulo la herramienta que se ha empleado para realizar dicha prueba, los parámetros del experimento y un análisis del modelo de comparación de fallos utilizado.*

## 4.1. Herramienta de análisis

Existen varias técnicas de inyección de fallos para analizar el comportamiento de un sistema ante un ataque externo, normalmente se trata de una radiación ionizante capaz de afectar al funcionamiento de un dispositivo a causa del impacto de las partículas con muy alta energía capaces de interferir sobre un circuito electrónico. Este trabajo no trata de profundizar sobre las diferentes técnicas de inyección de fallos que existen e incluso herramientas que permiten realizar este tipo de pruebas o experimentos. La razón por la que no se realiza ningún tipo de estudio sobre este tema es que desde el primer momento ya se conoce la herramienta con la cual se va a trabajar ya que se ha desarrollado en el propio Departamento de Ingeniería Electrónica de la Universidad de Sevilla.

La herramienta se llama FTU2[4][Ref. 4.1], la cual utiliza una técnica de emulación sobre FPGA no instrumentada, lo que la convierte en una herramienta no intrusiva y relativamente rápida, que además ofrece resultados jerárquicos según el árbol de registros del que disponga el diseño digital a probar. Su fundamento se basa en aprovechar las características que el fabricante Xilinx ofrece al poder emplear la reconfiguración parcial de sus dispositivos programables, como es el caso de algunas FPGAs. De esta manera es capaz de obtener mayores capacidades de depuración al aprovechar parte de la circuitería de la propia FPGA donde además se implementa el módulo bajo test o MUT, consiguiendo una mayor controlabilidad (capacidad de afectar a los estados internos del circuito) y observabilidad (capacidad de leer los estados internos del circuito) de los bits internos del circuito. Un esquema de la arquitectura que presenta esta herramienta se muestra a continuación:

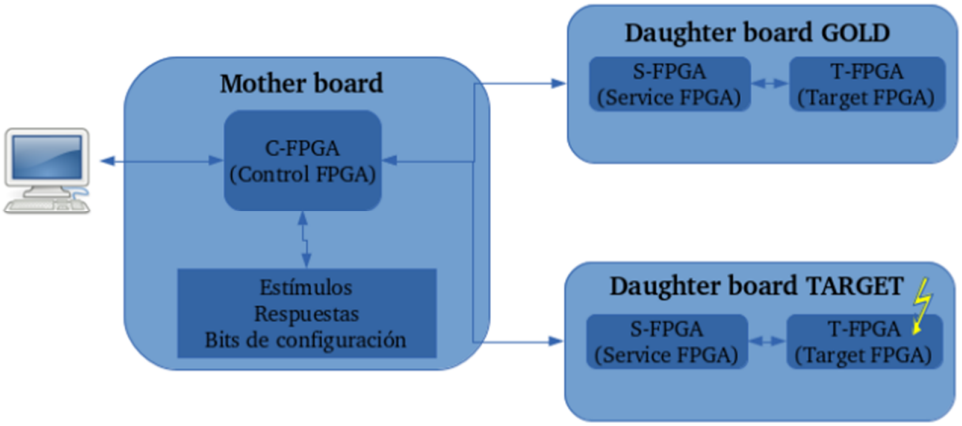



**Figura 4.1 – Arquitectura de la herramienta FTU2**

Los elementos principales de los que se compone FTU2 son 5 FPGAs integradas en tres tarjetas, una de ellas se denomina tarjeta madre (*mother board*), la cual integra a la FPGA de control o C-FPGA que se trata de un modelo ML510 XC5VFX130T de Xilinx. Además, emplea una memoria tipo DDR2 para almacenar tanto los estímulos de entrada así como las respuestas que dan las tarjetas hijas y los *bitfiles* o archivos de configuración de las FPGAs hijas.

[4] FT-Unshades 2: Fault Tolerant UNiversidad de Sevilla HArdware DEbugging System 2

Se comunica vía puerto USB con un ordenador por donde recibe los vectores de test desde una aplicación web denominada UFF[5], la cual se comunica con una herramienta software denominada TNT[6] que es la que se comunica con la tarjeta madre. A esta web es donde también se envían los resultados de las pruebas realizadas.

Por otro lado, existen dos tarjetas hijas (*daughter boards*) conectadas a la tarjeta madre por medio de unos zócalos PCI Express, ambas tarjetas hijas son idénticas y cada una de ellas contiene dos FPGAs. La primera FPGA que va directamente conectada al PCIe, se conoce como FPGA de servicio o S-FPGA del tipo XC5VFX70T de Xilinx y su función es la de establecer las comunicaciones entre la FPGA de control y la FPGA objetivo o *TARGET*. La otra FPGA que se integra en cada tarjeta hija es la T-FPGA y es en la que se implementa el diseño digital que se vaya a probar. Para la versión actual de la herramienta FTU2, para la T-FPGA puede utilizarse cualquier FPGA que sea compatible con la huella de una Virtex 5 FF1136.

Una de las tarjetas hijas se denomina GOLD, ya que contiene al MUT implementado en su T-FPGA que no va a ser atacado, al contrario de lo que ocurre en la tarjeta hija denominada TARGET, en la cual la instancia del MUT que está implementado en su T-FPGA sí va a sufrir la inyección de fallos.

Ambas tarjetas hijas comienzan a funcionar con el mismo diseño y devuelven sus respuestas ante los estímulos enviados por la tarjeta madre, a continuación se realiza una comparación de las salidas y se guardan los resultados con la salvedad de que en el diseño implementado en la T-FPGA de la tarjeta hija objetivo, se han inyectado uno o más fallos. Hasta aquí se muestra un breve resumen de la herramienta FTU2, su definición, su arquitectura y modo de funcionamiento; para una mayor información se puede acudir a [Ref. 4.2].

## 4.2. Condiciones del test

Con la prueba de inyección de fallos lo que se busca en conocer la robustez del diseño del transmisor. Dado que el objetivo de este trabajo es tratar de conseguir un transmisor inalámbrico para un entorno intrasatelital, este ensayo permite conocer la robustez estructural del diseño frente a una radiación externa de manera simulada, dando sentido al concepto de AVF visto en el primer capítulo. A la vista de los resultados que se obtengan se verá cuál es la eficiencia del diseño digital que se ha desarrollado frente a fallos originados por algunos de los tipos de efectos vistos en el capítulo 1.

Para este trabajo se van a analizar efectos tipo SEU, de hecho para evitar la aparición de posibles propagaciones de daño latente entre las diferentes inyecciones que se vayan a realizar, se ha diseñado el transmisor con una señal de *reset* global para todos los elementos de memoria del diseño del transmisor.

Las condiciones que se han establecido para realizar la prueba de inyección de fallos parten de una implementación del diseño hardware del transmisor ZigBee en una

[5] User Friendly Framework
[6] Test Analysis Tools

de las T-FPGA de la herramienta FTU2. A continuación, se realiza lo que se conoce como 'campaña' durante la cual se inyectan fallos en el diseño a partir de unos estímulos y se obtienen unas respuestas.

La campaña no es más que el proceso de inyección de fallos que se le aplica al diseño. Para este trabajo se ha realizado una campaña de 600.000 inyecciones mediante lo que se conoce como modelo básico de *bitflip*[7] en un ciclo de reloj escogido de manera aleatoria de entre todo el conjunto de estímulos. Además, dichas inyecciones se distribuyen uniformemente en todos los registros del diseño, de esta manera aquellos bloques con un mayor número de registros reciben una mayor cantidad de inyecciones.

## 4.3. Modelos de comparación

Una vez establecidas las condiciones del test de inyección de fallos es importante destacar el hecho de que las salidas que ofrece la herramienta FTU2 tras la realización de la campaña no se tratan de la manera tradicional. El modelo de comparación típico es analizar las salidas dadas por el MUT GOLD y el MUT TARGET ciclo a ciclo, tratando de encontrar todas las discrepancias del comportamiento binario y de esa manera obtener un conocimiento sobre cuáles son los registros del sistema más vulnerables a fallos, asociados como puede ser en este caso a efectos tipo SEU.

En lugar de comparar ciclo a ciclo las respuestas obtenidas entre el MUT TARGET y el MUT GOLD, lo que se hace es un post procesado de la salida completa del MUT TARGET (afectado con SEU) en Matlab tanto por el modelo de canal AWG (*Additive White Gaussian*) como por el modelo del receptor ZigBee (ambos modelos si han sido desarrollados por completo por parte del grupo del proyecto relacionado con el procesamiento de la señal), de manera que se puede determinar si el receptor es capaz de extraer información útil de las tramas de información que contienen bits erróneos a causa de la inyección de fallos.

En la figura 4.2 se puede ver un esquema del proceso de comparación que se emplea en este trabajo.

[7] Este modelo representa un cambio o inversión del valor lógico de una celda de memoria; es decir, cambios de '0' a '1', o de '1' a '0', teniendo en cuenta que dicho cambio se produce en un instante único y determinado de tiempo.

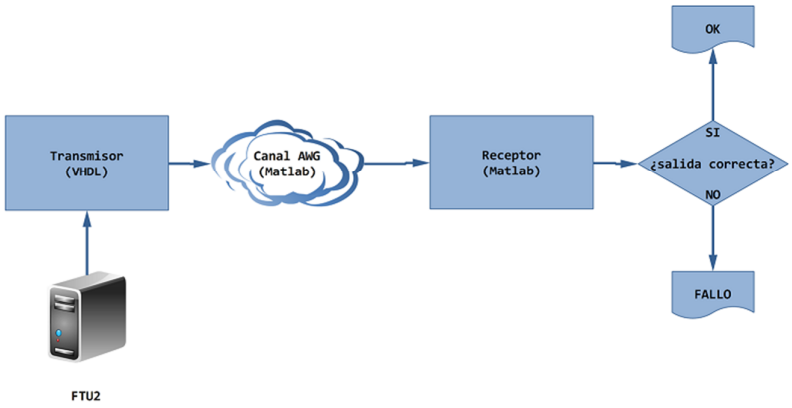



**Figura 4.2 – Nuevo modelo de comparación**

Se ha optado en este trabajo por un nuevo modelo basado en la idea de que un sistema de comunicaciones debe tener la propiedad de poder recuperar la información útil aun recibiendo tramas con errores binarios. Se sabe que en todo canal de comunicación, la señal puede verse afectada por múltiples causas que se traducen en bits erróneos cuando dicha señal llega al receptor. Una de las principales cualidades de los receptores es la de poder autorreparar la información contenida en las señales recibidas y se busca con este novedoso modelo de comparación el poder obtener resultados de AVF más precisos que con respecto al modelo tradicional ya que con la comparación ciclo a ciclo no hay posibilidad de margen a la hora de analizar las salidas.

Uno de los resultados más importantes de este experimento se observa en las medidas de BER (*Bit Error Rate*) y FER (*Frame Error Rate*) del sistema, a consecuencia de las inyecciones de fallos en el transmisor. Para obtener estas curvas, se mezclan diferentes porcentajes de tramas afectadas por SEUs con tramas no afectadas y se varían las condiciones de ruido del canal AWG. Estos resultados se detallan en el capítulo 5.

### 4.3.1. Clasificación de fallos

Tras realizar la campaña y analizar los resultados obtenidos, se puede afirmar que un gran número de tramas enviadas por el transmisor, con errores[8] debidos a los SEUs como consecuencia de la inyección de fallos, se recuperan de manera correcta por parte del receptor. Teniendo en cuenta este hecho, emplear una clasificación de fallos tradicional como puede ser *salida/latente/silencioso* se considera insuficiente y poco representativa para un sistema de comunicaciones inalámbrico como el que se tiene en este trabajo.

Gráficamente, se muestra la figura 4.3 un ejemplo de recuperación de trama errónea debido a la robustez estructural del sistema completo.

[8] Se debe aclarar que se considera como trama errónea a la salida del transmisor aquella en la que al menos existe una discrepancia ciclo a ciclo entre la salida del transmisor actual y la esperada del circuito.

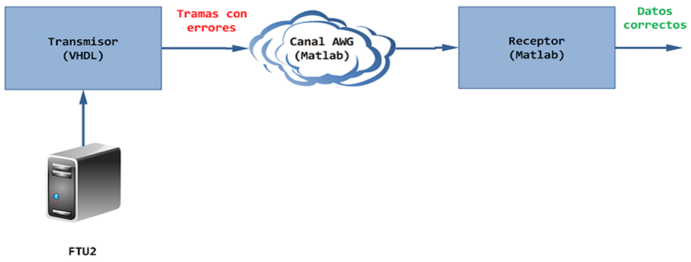



**Figura 4.3 – Ejemplo de recuperación de trama errónea**

Se opta mejor por una clasificación de fallos que separa claramente dos aspectos bien diferenciados en lo referente a la señal radio que sale del transmisor. Por un lado, se analizan los fallos que afectan al *flujo de datos* (bits de la trama de información que se quiere transmitir a lo largo de las distintas etapas del transmisor) y son aquellos que aportan errores binarios en las tramas transmitidas. Por otro lado, se identifican los fallos que afectan al *flujo de control* del circuito; es decir, a ciertos elementos que se encargan del sincronismo de la trama o mantener la tasa de transmisión de datos correcta. Estos fallos producen por un lado, desviaciones en la tasa de transmisión de datos diferentes a la tasa esperada según indica el protocolo ZigBee, este tipo de fallos se definen como ***desalineados***. Por otro lado existen fallos que provocan que una trama comience antes de lo esperado, se definen este tipo de fallos como ***precipitados***.

Con esta nueva definición de tipos de error que se han observado tras realizar la prueba de inyección de fallos, se realiza una tabla donde se puede observar la medida en que estos fallos han aparecido y cómo afectan a la recuperación o no de la información útil por parte del receptor.

| Clasificación de Fallos | Tramas | Recuperado por el RX |
|---|---|---|
| Inyecciones totales | 600000 | 539897 |
| --Binarios correctos | 168473 | 168473 |
| -- Salidas con error (TX) | 431527 | 350693 |
| ---- A tiempo y alineadas | 408385 | 350336 |
| ---- Precipitadas pero alineadas | 2411 | 337 |
| ---- A tiempo pero desalineadas | 19746 | N/A |
| ---- Precipitadas y desalineadas | 985 | N/A |

**Tabla 4.1 – Clasificación de fallos**

Es importante comentar que algunas tramas desalineadas puedan ser tal vez recuperables por el receptor, pero no se puede estudiar esta situación ya que se requiere tener el diseño del receptor en VHDL. Esto se debe a que el modelo del receptor en Matlab trabaja con muestras, que corresponden a un número determinado de ciclos de reloj en el circuito real, pero que Matlab no modela, por lo que si las muestras se desalinean no es posible analizar esta situación.

# Capítulo 5

# Resultados

*En este capítulo se pretende resaltar algunos de los resultados que se han ido señalando a lo largo de los capítulos anteriores y añadir uno de los más importantes que se han encontrado mediante la inyección de fallos en este proyecto. Se trata del estudio de la sensibilidad del sistema frente a los fallos tipo SEU, cuyos efectos se hacen notar tanto en la robustez estructural de los diferentes bloques del transmisor*

*como en las curvas de prestaciones del sistema de comunicación inalámbrico completo.*

## 5.1. Aspectos relativos a la FPGA

En el capítulo 3 se muestran datos referentes a la ocupación del diseño del transmisor sobre la FPGA donde se ha implementado. Dicha FPGA se corresponde con la que actualmente se encuentra integrada en las tarjetas hijas de la herramienta FTU2 haciendo la función de T-FPGA, que como ya se ha comentado, es donde se implementan los prototipos sobre los que se realizan la inyección de fallos. El modelo en concreto es una **Virtex 5 fx70tff1136-1** del fabricante Xilinx.

A la vista de los resultados que ofrece el informe post síntesis de la herramienta de diseño ISE (en concreto se usa la versión ISE 12.1), con la que se desarrolla el código VHDL de todos los bloques del transmisor ZigBee, cabe decir que se trata de un diseño eficiente en cuanto a utilización de recursos de la FPGA ya que al evitar el uso de multiplicadores en el filtro se ahorra gran parte del área disponible en la FPGA. Al ser un diseño donde todo el código es de elaboración propia, se consiguen encontrar formas de poder minimizar el número de celdas de memoria que utiliza la FPGA para formar los circuitos electrónicos que equivalen al código VHDL que se ha elaborado.

En dicho informe se puede observar que el porcentaje de **ocupación de lógica programable** es tan sólo del **1%** y la **ocupación de pines de entrada/salida** es de un **4%**, valores que avalan que se ha conseguido un diseño digital muy eficiente en cuanto a baja utilización de recursos de la FPGA.

Por otro lado, la **frecuencia máxima** de trabajo con la que el circuito digital puede llegar a trabajar es de **123.122 MHz**, teniendo en cuenta que todo el diseño se ha realizado para que funcione con una señal de reloj global a 100 MHz, se puede concluir con que es un diseño totalmente viable.

## 5.2. Verificación funcional

En cuanto a los procesos de verificación que se han empleado, es importante destacar la importancia que tiene el hecho de tener un plan de verificación durante la etapa de desarrollo de cualquier diseño aplicado a este tipo de proyectos.

En primer lugar, el método de verificación basado en comparar las salidas entre un modelo matemático, como es Matlab en este trabajo, y un modelo digital codificado en VHDL, ofrece muy buenos resultados en cuanto a aceleración del propio proceso de diseño y a la capacidad de resolver de manera más rápida y sencilla la aparición de errores y discrepancias entre el funcionamiento de ambos modelos.

En segundo lugar, se realiza un tipo de verificación diferente que trata de profundizar en el propio código VHDL del diseño digital mediante la generación de una base de datos que se consigue configurando el simulador con unas opciones específicas y así obtener un informe de Code Coverage. Con este informe se analiza si se cumplen todas las condiciones establecidas, si se recorren todos los estados y transiciones de las

diferentes máquinas de estado presentes en los bloques del transmisor y otras características que se detallan en el capítulo 3.

El resultado que se obtiene revela que se consigue tener un diseño con un alto porcentaje de cobertura de código probado, con 5 de los 6 bloques que componen el transmisor con un porcentaje superior al 90% de cobertura alcanzado.

## 5.3. Sensibilidad ante SEU

Este apartado pretende mostrar una serie de resultados muy importantes que incluso van a permitir el poder proponer una nueva metodología para evaluar la robustez de un sistema de comunicación inalámbrico embarcado para aplicaciones intra-satelitales. En primer lugar, para disponer de todos los datos de los cuales se extraen el resto de análisis, se muestra una tabla (ver Tabla 5.1) que representa los resultados obtenidos teniendo en cuenta todos los fallos, tanto en el flujo de datos como en el flujo de control, y donde aparecen todos los bloques del diseño junto con ciertas características que se explican a continuación.

Siguiendo el orden de las columnas, en primer lugar se tienen los diferentes bloques[9] que forman el diseño digital del transmisor ZigBee.

La segunda columna muestra el número de registros que ocupa cada bloque, es importante observar que los bloques más grandes son *fadapt* (por la FIFO que incluye) y los filtros de conformación de onda, *pulse_shaping*.

La tercera columna simplemente muestra los valores de ocupación en términos de porcentaje de cada bloque.

En la cuarta columna se indican el número de inyecciones que sufre cada bloque y como ya se dijo en el capítulo 4, el experimento realiza una inyección distribuida de manera uniforme en todos los registros del diseño por lo que se cumple que los bloques de mayor tamaño reciben un mayor número de inyecciones.

La quinta columna muestra el número de tramas que sufren algún tipo de error a la salida del transmisor (antes de llegar al receptor), mientras que en la sexta columna aparece el mismo dato pero en términos porcentuales. Es importante en este punto observar que el mayor porcentaje de tramas con daños se concentran al final de las etapas del transmisor, en los bloques *pulse_shaping* y *qdelay*.

En las dos últimas columnas se muestran los valores de las tramas con errores después de llegar al receptor, en términos numéricos y porcentuales.

[9] Señalar que el bloque *fadapt* se desglosa en dos, para diferenciar la influencia de los fallos entre la parte puramente lógica de dicho bloque (máquina de estados que forma la trama) y la FIFO que incluye.

| Bloque | Nº registros | % registros | Inyecciones totales | Tramas con daños después de transmitir | Tramas con daños después de transmitir (%) | Tramas con daños después de recibir | Tramas con daños después de recibir (%) |
|---|---|---|---|---|---|---|---|
| Total | 388 | 100 | 600000 | 431527 | 71.92 | 80834 | 13.47 |
| fadapt (lógica + fifo) | 109 | 28.09 | 168605 | 69669 | 11.61 | 52734 | **8.79** |
| fadapt_logic | 30 | 7.73 | 46395 | 24082 | 4.01 | 17181 | 2.86 |
| fadapt_fifo | 79 | 20.36 | 122210 | 45587 | 7.60 | 35553 | 5.93 |
| bit2simb | 13 | 3.35 | 20111 | 6728 | 1.12 | 6344 | 1.06 |
| simb2chip | 24 | 6.19 | 37104 | 30032 | 5.01 | 18869 | **3.14** |
| upsampling_I | 11 | 2.84 | 17006 | 13533 | 2.26 | 1323 | 0.22 |
| upsampling _Q | 11 | 2.84 | 17006 | 13533 | 2.26 | 1401 | 0.23 |
| pulse_shaping_I | 86 | 22.16 | 132956 | 111654 | 18.61 | 92 | 0.02 |
| pulse_shaping _Q | 86 | 22.16 | 132956 | 113911 | 18.99 | 0 | 0.00 |
| qdelay | 48 | 12.37 | 74256 | 72258 | 12.04 | 71 | 0.01 |

**Tabla 5.1 – Resultados de sensibilidad ante SEU**

Como primer resultado a destacar, se observa que el sistema de comunicación inalámbrico posee de manera inherente una gran robustez estructural gracias a la cual consigue extraer la información útil a pesar de recibir tramas con errores. Esto se debe principalmente al hecho de que un protocolo como ZigBee, que emplea la técnica de modulación mediante espectro expandido, consigue que todos los fallos que se producen a partir del bloque *upsampling* son prácticamente irrelevantes desde el punto de vista de la recepción, ya que como puede verse, el porcentaje de tramas erróneas es despreciable frente a otros bloques que se encuentran en una primera etapa del diseño, antes de que se realice la expansión del espectro de la señal a transmitir.

Este resultado es muy útil a la hora de pensar en un método que ayude a robustecer el sistema frente a SEUs, ya que gracias al test de inyección de fallos y al nuevo modelo de comparación empleado se ha podido medir y obtener una conclusión tan importante como que al utilizar un protocolo que trabaja con la técnica de espectro expandido no es necesario robustecer los bloques de la última etapa del diseño.

En la tabla 5.1 se marcan en rojo los bloques, que tras la recepción, afectan con un mayor porcentaje de tramas erróneas. Dichos bloques son *fadapt* y *symb2chip*, cuya explicación se puede asociar a que son bloques realmente críticos en cuanto a la función que realizan.

Por un lado, *fadapt* crea la trama binaria de la señal de información, por lo que cualquier tipo de fallo en este punto ya sea en el flujo de datos o de control puede ser desastroso para la señal que sale transmitida finalmente. Por otro lado, el bloque *symb2chip* divide el flujo de datos en dos componentes y necesita de una buena sincronización para que los bits alternados se correspondan con la señal correcta. Esto conlleva a que un fallo en este bloque puede originar una falta de sincronismo y hacer corresponder secuencias de *chips* con símbolos equivocados.

Con objeto de afianzar un poco más las conclusiones que se acaban de establecer, se muestra otra tabla (ver Tabla 5.2) en la que se tiene en cuenta que la

modulación de espectro expandido sólo afecta al flujo de datos que se transmiten y por tanto sólo se va a tener en cuenta en dicha tabla la influencia de los SEUs que interfieren en dicho flujo de datos.

Ya que los SEUs que afectan al flujo de control se sabe, a la vista de los resultados, que son más propensos a corromper un mayor número de bits en una única trama, es por ello menos probable que puedan ser corregidos gracias a la modulación de espectro expandido, y por tanto se deben tratar por separado.

En la tabla 5.2 se muestra en la segunda columna la probabilidad de que un bloque pueda sufrir el efecto de un SEU exclusivamente en el trayecto del flujo de datos. En la tercera columna se indica el porcentaje de tramas erróneas a la salida del transmisor, mientras que en la cuarta columna el porcentaje representa a las tramas erróneas después de que hayan sido recibidas. Esta última columna está directamente relacionada con la tasa de error de trama o FER (*Frame Error Rate*) ya que se han excluido aquellas tramas erróneas afectadas por el flujo de control.

| Bloque | Probabilidad de SEU (%) | Tramas con daños después de transmitir (%) | Tramas con daños después de recibir (%) |
|---|---|---|---|
| **fadapt** | 14.24 | 28.4 | 24.6 |
| **bit2symbol** | 1.57 | 17.4 | 16.9 |
| **symbol2chip** | 5.74 | 57.5 | 28.2 |
| **upsampling** | 5.95 | 3.7 | 0.01 |
| **pulse_shaping** | 54.84 | 84.5 | 0.01 |
| **qdelay** | 17.66 | 96.8 | 0 |

**Tabla 5.2 –Sensibilidad ante SEU sobre el flujo de datos**

Estos resultados confirman que los bloques a partir de los cuales se tiene la modulación de espectro expandido, son estructuralmente más robustos que el resto de bloques. El hecho de sólo tener en cuenta SEUs que afectan al flujo de datos aporta una información aún más útil, ya que los errores se traducen en un pequeño número de bits corruptos a lo largo de la trama de información que posteriormente pueden ser corregidos complementando al receptor con un código de corrección de errores adecuado. Se deduce de estos resultados que se puede implementar una protección óptima robusteciendo los bits del flujo de control utilizando técnicas tradicionales como TMR[10], así como robusteciendo los bits del flujo de datos mediante el uso de códigos detectores y correctores de errores.

Si se hubiera intentado robustecer los bloques que contribuyen con mayor porcentaje de tramas erróneas sin haber aplicado el nuevo modelo de comparación que se propone en este trabajo (que tiene en cuenta la capacidad del receptor para autoreparar fallos en los datos recibidos), se hubieran robustecido los bloques pulse_shaping de ambas ramas (I y Q) triplicando de forma innecesaria un enorme

[10] *Triple Modular Redundancy*

número de biestables y además, resultando en una mejora prácticamente nula en las prestaciones del sistema.

### 5.3.1. Curvas de ruido

Tradicionalmente, se emplean transceptores cableados con baja BER en aplicaciones intra-satelitales [Ref. 5.1], por lo que se puede entender que una baja BER tiene alguna relación con que un sistema de comunicación sea poco susceptible ante SEUs [Ref. 5.2]. En este trabajo se trata de buscar esa relación pero en un sistema de comunicación inalámbrico.

Para ello se realizan simulaciones variando la tasa de SEUs que atacan al diseño y se mide el valor de FER que ofrece el sistema. Se emplea el FER como medida de mérito porque el estándar 802.15.4, en el que se basa ZigBee, establece un máximo de un 1% de tramas erróneas como requisito de calidad para la comunicación.

Las tramas que salen del transmisor se procesan por el modelo en Matlab del receptor ZigBee y se obtienen las curvas de FER. Se muestra en la figura 5.1 que los errores tipo SEU presentan una naturaleza de ruido impulsivo y el efecto que provocan en las curvas de FER es que aparecen **suelos de ruido**. Esto significa que para una tasa de SEU dado, incluso aunque la relación señal a ruido o SNR sea alta, el porcentaje de tramas erróneas no es despreciable por lo que no se puede reducir el número de tramas erróneas todo lo que se quiera incrementando la SNR a diferencia del caso ideal (sin SEU). Se observa también que para una tasa de SEU inferior, el suelo de error también disminuye.

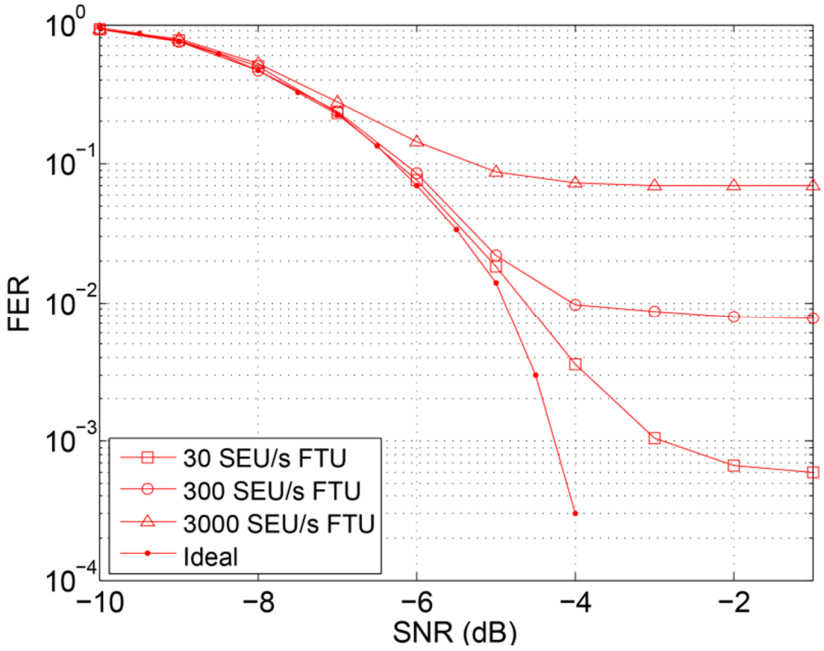



**Figura 5.1 – Curvas de FER para diferentes tasas de SEU**

En la figura 5.1 se observa que para valores altos de SNR el principal contribuyente a la curva de FER es el efecto de los SEUs, mientras que para valores bajos de SNR el principal contribuyente es el ruido inducido por el canal de

comunicación.

Para analizar el diseño del transmisor teniendo en cuenta el requisito del 1% máximo de FER de tramas erróneas, se muestra en la figura 5.2 la función de distribución de las tramas de error acumuladas. Se puede observar que un 99% de las tramas presentan menos de 25 bits erróneos y dejan de aparecer bits erróneos por debajo del 92.7% de las tramas.

Desde el punto de vista de búsqueda de estrategias de endurecimiento del sistema para aumentar la robustez ante SEU, esta gráfica es muy útil ya que para cumplir con el objetivo del FER al 1% dada una tasa de SEU, el número de bits que deben ser corregidos se observa en este tipo de gráfica.

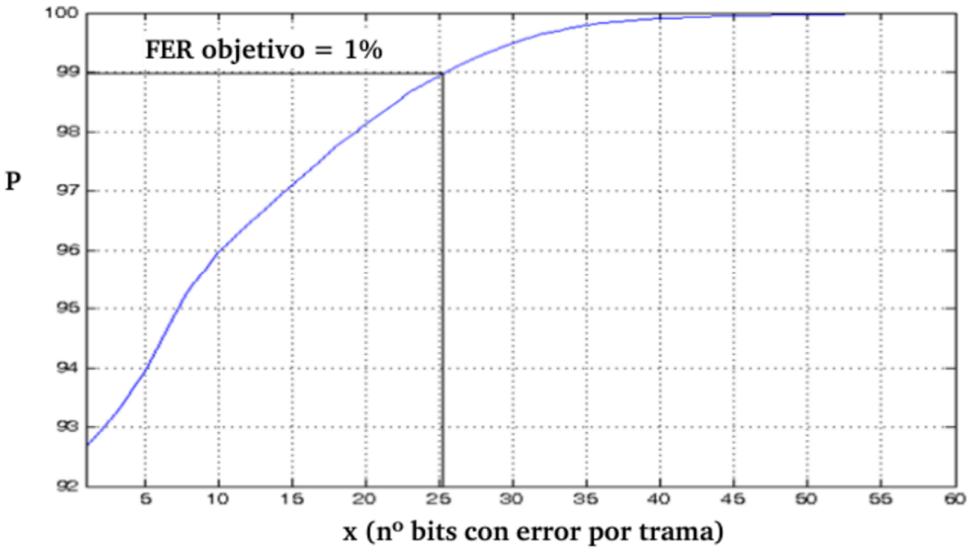



**Figura 5.2 – Función de distribución de tramas erróneas acumuladas**

Para terminar de completar el estudio de la sensibilidad del sistema ante SEU, se analizan los bits erróneos dentro de las tramas y se observa que la mayoría contiene un único bit erróneo, lo que sugiere tal y como se dijo anteriormente, que si se añaden técnicas de corrección de errores simples, se puede mejorar el rendimiento del sistema. Métodos como el BCH[11] o los códigos de Hamming son buenos candidatos de baja complejidad de implementación que ofrecen una mejor solución para aumentar la robustez del sistema ante SEU, en lugar de emplear la técnica típica de triple redundancia modular o TMR [Ref. 5.3], [Ref. 5.4].

## 5.3.2. Zonas Calientes

Por último y con objeto de tener una visión más gráfica en cuanto a los resultados que se han detallado en este capítulo, se muestran dos figuras con el esquema completo del transmisor digital que se ha diseñado, donde se representa en términos porcentuales cuánto aporta cada bloque con tramas erróneas al sistema. La manera de representarlo se realiza mediante un gradiente de color que refleja las zonas calientes para conocer la mayor o menor aportación de tramas erróneas según la intensidad del

[11] *Bose-Chaudhuri-Hocquenghem*

color en cada bloque. La diferencia entre ambas figuras radica en que una realiza la comparación de las salidas ciclo a ciclo justo a la salida del transmisor, y la otra figura muestra la aportación de tramas erróneas una vez procesadas por el receptor.

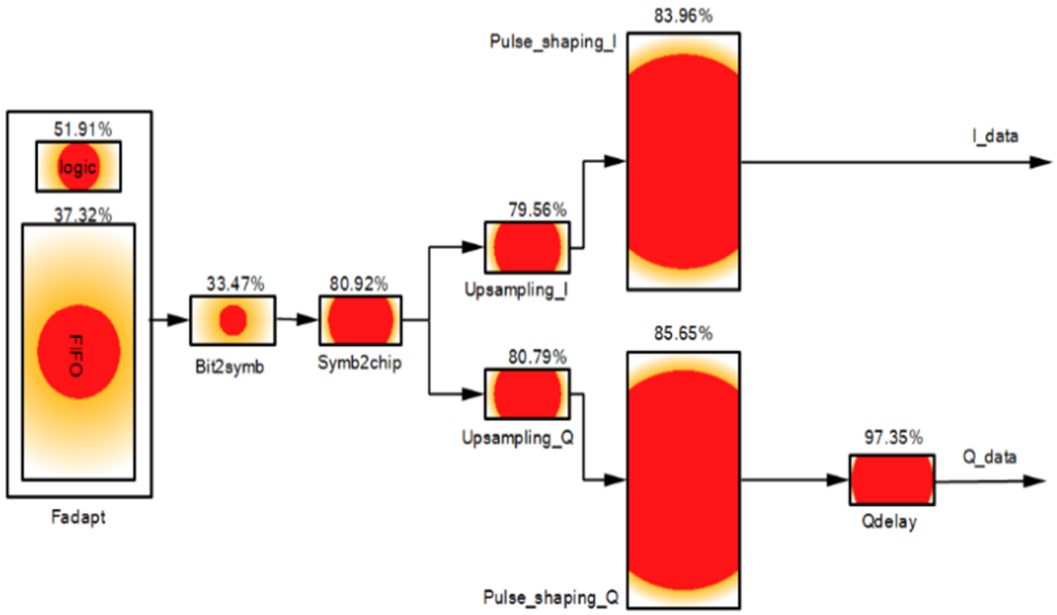



**Figura 5.3 – Aportación de tramas erróneas después de la transmisión**

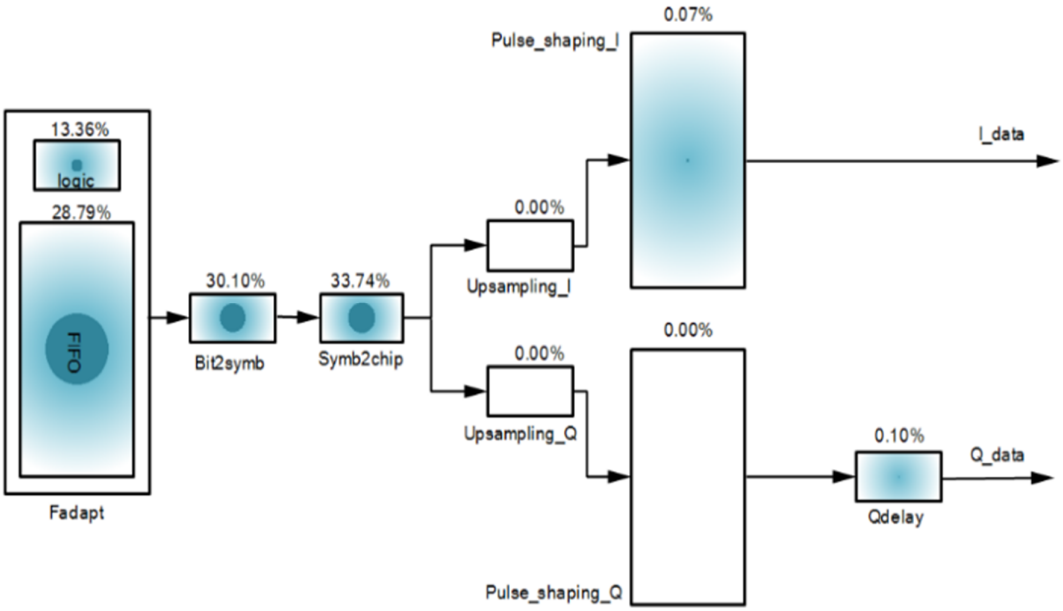



**Figura 5.4 – Aportación de tramas erróneas después de la recepción**

A la vista de lo que muestran las figuras, solo queda resaltar que gracias a la modulación de espectro expandido que emplea el protocolo ZigBee, el sistema dispone de una robustez estructural suficiente como para tener unas buenas prestaciones en

cuanto a sensibilidad ante ataques producidos por una radiación intensa, como puede ser el caso de un entorno espacial, que es el objetivo que se pretende alcanzar.

## 5.4. Metodología de Endurecimiento

Tras analizar todos y cada uno de los resultados obtenidos, se propone un método para evaluar y poder optimizar la robustez de un transceptor inalámbrico (para este trabajo se trata sólo del transmisor pero se puede aplicar al futuro receptor que se va a diseñar), para una aplicación embarcada en un satélite. Es importante destacar el hecho de que esta metodología debe de aplicarse durante la etapa de diseño.

1. Diseñar una versión del bloque transmisor/receptor sin mitigación de errores. Emplear técnicas de verificación que ayuden a obtener un diseño funcional de manera más eficiente.

2. Realizar un experimento de inyección de fallos sobre el diseño anterior.

3. Almacenar las respuestas a cada fallo obtenidas por parte del circuito y estudiar su comportamiento desde un punto de vista del sistema de comunicación. Realizar medidas de curvas de BER y FER.

4. Identificar qué bloques o regiones del diseño son más sensibles a los fallos, destacando principalmente dos tipos de fallos bien diferenciados:
    a. Fallos que producen un mayor número de tramas erróneas que son irrecuperables (contribuyen al FER).
    b. Fallos que producen un mayor número de bits erróneos por trama.

5. Realizar una mitigación de errores selectiva, endureciendo aquellas regiones más sensibles y modificando la arquitectura del diseño que sea necesaria. Es posible aplicar una estrategia mixta donde coexistan la robustez estructural y la prevista por el diseñador.

# Capítulo 6

# *Conclusiones y Trabajos Futuros*

*Este último capítulo expresa las conclusiones que se extraen tanto de la metodología que se emplea para realizar un diseño conjuntamente con un proceso de verificación, como de los resultados obtenidos tras la prueba de inyección de fallos. Finalmente, se detallan ciertos aspectos que pueden realizarse con el objeto de mejorar las prestaciones del sistema que se ha diseñado, así como exponer el camino a seguir para llegar a tener un transceptor inalámbrico completo integrado en un dispositivo físico.*

## 6.1. Acerca del diseño del transmisor

En este punto se remarca la importancia de que se ha desarrollado un transmisor completo que cumple con las especificaciones del protocolo ZigBee. Se trata de un diseño realizado en lenguaje VHDL, teniendo en cuenta ciertas consideraciones cuyo objetivo principal es conseguir un código de elaboración propia para tener el control de todos los registros del diseño. Esto es importante para analizar el comportamiento de cada bloque tras realizar las pruebas de inyección de fallos, con las que se consigue conocer la robustez estructural del diseño que se ha desarrollado.

Para poder realizar el diseño digital, se ha partido de unas especificaciones muy claras realizadas por el equipo del proyecto encargado del modelo Matlab y que ha ayudado mucho a comprender el funcionamiento de cada bloque. Destacar que durante el proceso de diseño, se han adquirido nuevos conocimientos acerca de la codificación en VHDL, como es el uso de aserciones (*assertions*) con las que se realizan, durante las simulaciones, comprobaciones específicas que establece el diseñador, con el objeto de conocer si el funcionamiento del código es correcto y si fallara, saber de manera inmediata dónde y cuál es el fallo.

Por otro lado, la colaboración en la etapa de desarrollo de las entidades VHDL para la realización de los *test benches*, ha servido para conocer la manera en la que se abren y cierran ficheros de texto en VHDL, que puede ser de mucha utilidad para posibles trabajos futuros.

En general, con estas técnicas se consigue acelerar en gran medida el proceso de diseño y lo que es más importante, conseguir que los bloques que componen el sistema cumplan con su funcionamiento correcto de manera más rápida ya que se solucionan los errores que vayan surgiendo de una forma casi inmediata al tenerlos bien localizados.

## 6.2. Acerca de la verificación

En el capítulo 3 se describen los dos procesos de verificación que se han empleado en este trabajo. Tras realizar todo el diseño del transmisor, queda patente la importancia de realizar una verificación, sea del tipo que sea dependiendo de cada proyecto, en paralelo con el proceso de desarrollo. En otras palabras, la verificación debe de formar parte de la etapa del desarrollo y no ser una mera prueba que se utilice al finalizar un diseño para analizar su comportamiento y analizar posibles errores. Una verificación a posteriori no resulta tan eficiente como que se realice a la vez que se desarrolla el diseño.

En este trabajo se ha ido verificando cada bloque diseñado a través de un *crosscheck* de salidas con un modelo matemático y no se ha pasado a diseñar el siguiente bloque de la cadena hasta que el anterior realizaba su función de manera correcta. Si en lugar de trabajar de esa forma, se hubieran diseñado todos los bloques del transmisor y al final se quisiera contrastar su salida con la del modelo en Matlab, es de suponer que el trabajo de verificación resulte mucho menos eficiente ya que los

errores pueden darse en cualquiera de los bloques, teniendo que revisar uno a uno y analizar su comportamiento por separado, lo que incrementa en gran medida el tiempo de trabajo para resolver todos los problemas que surjan.

El segundo método de verificación es algo novedoso para este grupo de trabajo, pero ha resultado ser de gran ayuda el hecho de conocer que gracias a algunas técnicas que se han empleado para el diseño en VHDL, se han obtenido valores muy altos de cobertura de código probado según los datos de Code Coverage que se han obtenido durante las simulaciones del diseño. Estos resultados indican que el código VHDL es eficiente en cuanto a número de declaraciones ejecutadas, comprobación de condiciones, ejecución de estados y transiciones entre estados en las FSM diseñadas; en general, la mayor parte del código que se ha desarrollado se ha podido probar. El reto es profundizar en este tipo de verificación y emplear algunas técnicas con las que tratar de alcanzar un 100% de cobertura en el código VHDL del transmisor.

## 6.3. Acerca del modelo de comparación

Una vez terminado el diseño digital del transmisor, se pasa a realizar una prueba de inyección de fallos con el objeto de conocer la robustez estructural del sistema. Para analizar los resultados de la prueba, el modelo de comparación a emplear es crítico. Se ha observado que realizar una comparación ciclo a ciclo, como se realiza de forma tradicional en este tipo de pruebas, no es útil para un sistema de comunicación inalámbrico, ya que tiene más sentido el hecho de conocer cómo se comporta el receptor al recibir tramas erróneas a causa de los efectos tipo SEU, que analizar las salidas del transmisor para conocer la aportación de tramas erróneas que aporta cada bloque que lo compone.

La principal contribución de este trabajo reside en que el modelo de comparación que se ha empleado considera la capacidad de autorreparación del propio sistema inalámbrico. Este nuevo modelo de comparación a nivel de sistema ha resaltado el hecho de que los bloques que según una comparación ciclo a ciclo eran más indicados a ser endurecidos por su alto número de tramas erróneas que aportan a la salida del transmisor, a nivel de sistema no resulte ser así. De hecho, desde el punto de vista del receptor dichos bloques apenas aportan tramas erróneas, siendo otros bloques los más críticos en cuanto a aportación de tramas erróneas en recepción.

## 6.4. Acerca de los resultados obtenidos

A la vista de todos los resultados analizados en el capítulo 5, se puede concluir que con este trabajo se ha realizado un estudio novedoso, donde se propone un método de endurecimiento selectivo aplicado a sistemas de comunicación inalámbricos, que considera la capacidad de autorreparación del sistema y obtiene resultados de AVF más precisos respecto a la inyección de fallos tradicional. De esta manera, se consigue una mayor eficiencia en área utilizada a la hora de endurecer el sistema ya que sólo es necesario hacerlo en aquellas regiones sensibles desde el punto de vista de trama recibida.

Las figuras de zonas calientes vistas en el capítulo 5 han dejado plasmada la importancia de la modulación de espectro expandido, ya que es muy eficiente para combatir al ruido impulsivo en comunicaciones. Se ha comprobado que los errores tipo SEU introducen un suelo de ruido sobre el rendimiento del sistema inalámbrico (curvas de FER) y que presentan justamente una naturaleza de ruido impulsivo. Es por ello, que el sistema dispone de una robustez estructural inherente con la que se ha demostrado que los bloques más sensibles a SEU a nivel de sistema se encuentran en la etapa del transmisor situados antes de que se realice la modulación de espectro expandido.

Con todo esto, se abren nuevas vías con objeto de endurecer un sistema de manera más eficiente. Se puede realizar un TMR selectivo sobre los elementos del flujo de control, se pueden implementar códigos de detección y corrección de errores sobre los elementos del flujo de datos y se puede dar por hecho que bloques situados tras una expansión de espectro expandido son prácticamente insensibles a SEU.

## 6.5. Trabajos Futuros

Para finalizar este trabajo, se propone detallar el camino que se debería seguir en este proyecto con idea de tener un sistema de comunicación inalámbrico para aplicaciones intra-satélite completo, y tratar de ponerlo en funcionamiento en un entorno de radiación para emular cómo sería su comportamiento para una aplicación embarcada en un satélite.

### 6.5.1. Desarrollo e implementación del receptor

Se está ya trabajando en el diseño digital del receptor en VHDL. Se siguen los mismos métodos de verificación que se han empleado para el desarrollo del transmisor y se van a realizar las mismas pruebas de inyección de fallos para conocer la robustez del sistema teniendo en cuenta que los bloques del propio receptor se van a ver afectados por errores tipo SEU. Para ello, se están empleando las mismas herramientas de diseño que en el caso del transmisor, se va a implementar el diseño sobre la misma FPGA utilizada en este trabajo y obviamente se va a utilizar la herramienta FTU2 para las pruebas de inyección de fallos.

### 6.5.2. Prototipado del sistema completo sobre una FPGA

Una vez se tenga el diseño digital del receptor, se ha estudiado la manera de tener un sistema inalámbrico que funcione intercambiando tramas de comunicación mediante ZigBee. Se dispone en el departamento de unas FPGAs modelo Zedboard que incorporan unos pines de expansión denominados Pmod. Existen multitud de circuitos que se conectan a los puertos Pmod para comunicarse con la FPGA y realizar diferentes acciones.

Uno de estos Pmod[12] implementa justamente la etapa radio para transmitir y recibir una señal con el protocolo ZigBee [Ref. 6.1] con lo que se puede tener a una FPGA con el diseño del transmisor implementado y otra con el receptor para conseguir

[12] En concreto es el modelo *PmodRF2 - IEEE 802.15 RF Transceiver* del fabricante Digilent

establecer un sistema completo intercambiando tramas de información de manera inalámbrica.

Un experimento muy interesante es tener o bien el transmisor o bien el receptor implementado en la FPGA TARGET de la herramienta FTU2 y realizar una inyección de fallos a la vez que se establece una comunicación inalámbrica con idea de analizar el efecto de los SEUs en tiempo real.

### 6.5.3. Implementación ASIC

El siguiente paso consistiría en fabricar un ASIC[13] del transceptor completo. Una vez que se disponga del diseño digital del transceptor probado y verificado, se puede colaborar con el grupo de microelectrónica analógica del departamento para diseñar la etapa analógica y mandar a fabricar un chip con el transceptor ZigBee completo.

El objetivo principal de fabricar un ASIC propio reside en que sería posible llevar dicho chip a un centro especializado, capaz de realizar experimentos de radiación controlada, y estudiar el funcionamiento del transceptor en un entorno hostil.

### 6.5.4. Test de radiación

Como último paso, con objeto de emular el comportamiento del transceptor en un entorno espacial, ya que el sistema está pensado para poder sustituir a los que actualmente están embarcados en satélites mediante cableado, se puede someter al ASIC a un experimento de radiación controlada.

Existe la posibilidad de poder realizar dicho experimento ya que el departamento tiene una estrecha relación gracias a varias colaboraciones con el Centro Nacional de Aceleradores (CNA) [Ref. 6.2] situado en la Isla de la Cartuja en Sevilla. De la propia página web del CNA se extrae lo siguiente:

"*El CNA es un centro mixto de la Universidad de Sevilla, Junta de Andalucía y CSIC. Se trata de una Instalación Científico-Técnica Singular, ICTS, dedicada a la investigación interdisciplinar y por tanto abierta a usuarios externos. Para ello se emplean 3 aceleradores de iones: un Tándem Van de Graaff de 3 MV, un Ciclotrón que proporciona protones de 18 MeV y deuterones de 9 MeV y un acelerador tipo Tándem Cockcroft-Walton de 1 MV, utilizado como espectrómetro de masas.*

*La aplicación de estos 3 aceleradores cubre campos tan variados como ciencias de materiales, impacto medioambiental, física nuclear y de partículas, instrumentación nuclear, tratamiento de imágenes médicas, investigación biomédica e imagen molecular preclínica o datación, entre otras.*"

[13] *Application Specific Integrated Circuit*

A la vez que se elabora el experimento de radiación, cabe la posibilidad de realizar un diccionario de fallos mediante el cual se obtiene información muy útil para conocer cómo responde el sistema según el tipo de error que haya sufrido. Esta técnica se puede realizar gracias al trabajo elaborado por J.M. Mogollón en su Tesis Doctoral [Ref. 6.3].

Por último, dada una situación en la que no fuera posible la fabricación del ASIC, se puede realizar en experimento de radiación empleando un dispositivo electrónico tipo CPLD[14] en el cual se pueda asegurar que los bits de configuración (normalmente localizados en una memoria flash) no sean sensibles a las partículas de baja energía disponible en el CNA, pero en cambio que los bits de usuario sí lo sean, para poder conocer el comportamiento del sistema a efectos de la radiación. En este caso habría que tener cuidado con las optimizaciones y redundancias añadidas por las herramientas de implementación, revisando el diseño implementado para asegurar que se mantiene la misma arquitectura y número de registros originales.

[14] *Complex Programmable Logic Device*

# Referencias


| | | |
|---|---|---|
| [Ref 1.1] | Optos - http://www.inta.es/NoticiaActualidad.aspx?Id=1960 | 15 |
| [Ref 1.2] | D. Heynderickx, "Review on modeling of the radiation belts", Int. J. Mod. Phys.A, vol. 17, nos. 12&13, pp. 1675-1684, 2002 | 16 |
| [Ref 1.3] | J. Barth, "Modeling space radiation environments", *IEEE NSREC Short Course Notes*, Ch. 1, Snowmass, CO, USA, 1997 | 16 |
| [Ref 1.4] | "Solar effects on communications", Power Engineering Review, IEEE, Volume 11, Issue9, pp. 6-11, Sept. 1991 | 16 |
| [Ref 1.5] | C.T. Russell, "The solar wind interaction with the Earth's magnetosphere: a tutorial", *IEEE Transactions on Plasma Science*, Volume 28, Issue 6, pp. 1818-1830, Dec. 2000 | 16 |
| [Ref 1.6] | D. Binder, E.C. Smith, and A.B. Colman, "Satellite anomalies from galactic cosmic rays", *IEEE Trans. Nuclear Sci.*, vol. NS-22, pp. 2675-2680, Dec. 1975 | 16 |
| [Ref 1.7] | http://www.inta.es/noticias/documentos/FOTON-M3_OWLS_2007-09-13.pdf | 20 |
| [Ref 1.8] | Santamaría, A., López-Hernández, F., Guerrero, H., Arruego, I., & Rodriguez, S., "Wireless Infra-Red Links for Intra-Satellite Communications", *Proceedings of DASIA 2003 (ESA SP-532)*. 2-6 June 2003, Prague, Czech Republic. Editor: R.A. Harris. Published on CDROM., id.49.1 | 20 |

| | | |
|---|---|---|
| [Ref 1.9] | ESTEC/ESA contract 13784/99/NL/MV, “Study on optical wireless links for intra-satellite communications”, Dec. 2000 | 21 |
| [Ref 1.10] | http://www.ieee802.org/15/pub/TG4.html | 21 |
| [Ref.1.11] | http://www.ieee802.org/15/pub/TG1.html | 21 |
| [Ref 1.12] | http://www.ieee802.org/11/ | 21 |
| [Ref 1.13] | http://www.ieee802.org/16/ | 21 |
| [Ref 3.1] | https://verificationacademy.com/courses/evolving-fpga-verification-capabilities | 46 |
| [Ref. 4.1] | Mogollon, J.M.; Guzman-Miranda, H.; Napoles, J.; Barrientos, J.; Aguirre, M.A., "FTUNSHADES2: A novel platform for early evaluation of robustness against SEE," *Radiation and Its Effects on Components and Systems (RADECS),12th European Conference* on , vol., no., pp.169,174, 19-*23 Sept*. 2011 | 51 |
| [Ref. 4.2] | http://ftu.us.es/ | 52 |
| [Ref. 5.1] | M. Pignol, “System Hardening and Real Applications,” *Int. School on the Effects of Radiation on Embedded Systems for Space Applications, (SERESSA)*, Moscow, Oct. 2013 | 61 |
| [Ref. 5.2] | M.A. Cosgrove, "Using a system-level bit-error-rate model to predict on-orbit performance," *IEEE Trans. Nuclear Science*, vol. 50, no. 6, pp.2352-2357, Dec. 2003 | 61 |

| | | |
|---|---|---|
| [Ref. 5.3] | V.P. Mahadevaswamy *et al*., "Implementation of Fault Tolerant Method Using BCH Code on FPGA," *Int. J. of Soft Computing and Engineering (IJSCE),* vol. 2, Sep. 2012 | 62 |
| [Ref. 5.4] | C. Yuanyuan and Z. Xunying, "Research and implemention of interleaving grouping Hamming code algorithm," *IEEE Int. Conf. on Signal Processing, Communication and Computing (ICSPCC)*, pp. 1-4, Aug. 2013 | 62 |
| [Ref. 6.1] | https://digilentinc.com/Products/Detail.cfm?NavPath=2,401,927&Prod=PMOD-RF2 | 68 |
| [Ref. 6.2] | http://acdc.sav.us.es/cna/ | 69 |
| [Ref. 6.3] | JM Mogollon, J Nápoles, H Guzman-Miranda, MA Aguirre, "Real time SEU detection and diagnosis for safety or mission-critical ICs using hash library-based fault dictionaries," *Radiation and Its Effects on Components and Systems (RADECS)*, Sept. 2011 | 70 |