Escuela Superior de Ingenieros
Departamento de Ingeniería Electrónica
Universidad de Sevilla

Interfaces de datos de alta velocidad para sistemas de inyección de fallos de segunda generación

Proyecto Fin de Carrera

Javier Barrientos Rojas
Ingeniero de Telecomunicación
Tutor: Hipólito Guzmán Miranda

Sevilla, Julio 2013

A Rocío por sus ánimos hasta llegar aquí,
A mis padres y a mi hermano

NOTA: *Este trabajo ha sido desarrollado en el contexto del proyecto FT-UNSHADES2 financiado por la Agencia Espacial Europea (ESA) con nº de contrato 25988/NL/JK en conjunto con el Ministerio de Ciencia e Innovación de España dentro del marco del proyecto RENASER +, numerado TEC2010-22095-C03-01.*

Índice general

1. Capítulo 1: Introducción….....6
1.1. Problemática.....7
1.2. Efectos de la radiación.....7
1.2.1. Mecanismos físicos de daños.....9
1.3. Eventos de Efectos Únicos (SEE).....9
1.3.1. Errores Físicos.....10
1.3.1.1. Enclavamiento por Evento Único.....10
1.3.1.2. Ruptura de Puerta por Evento Único.....11
1.3.1.3. Ruptura por Evento Único.....11
1.3.2. Errores Lógicos.....12
1.3.2.1. Conmutación por Evento Único.....12
1.3.2.2. Conmutación Múltiple de Bits.....13
1.3.2.3. Transitorio por Evento Único.....13
1.3.2.4. Múltiples Conmutaciones por Evento Único.....13
1.4. Sistemas de Inyección de Fallos.....14
1.4.1. Requisitos básicos y tipos de técnicas.....16
1.4.2. Sistema de 1ª Generación: FT-UNSHADES.....17
1.4.3. Sistema de 2ª Generación: FTU2.....19

2. Capítulo 2: Descripción de la plataforma de trabajo.....21
2.1. Arquitectura de FTU2.....22
2.1.1. Parte Hardware.....22
2.1.2. Parte Software.....24
2.1.3. Parte Firmware.....25

3. Capítulo 3: Solución Propuesta.....28
3.1. Descripción General.....29
3.2. Controlador de interfaz con memoria DDR2 SDRAM.....29
3.2.1. Estrategia de uso de un microprocesador Power PC 440.....31
3.2.2. Estrategia de manejo directo de la interfaz.....36
3.3. Controlador de interfaz PCI Express.....44

4. Capítulo 4: Pruebas y resultados.....48
4.1. Introducción.....49
4.2. Test de funcionalidad de la interfaz con la memoria DDR2: estrategia de uso del microprocesador Power PC.....49
4.2.1. Resultados de velocidad.....49
4.3. Test de funcionalidad de la interfaz PCIe.....50
4.3.1. Resultados de velocidad.....50
4.4. Test de funcionalidad de la interfaz con la memoria DDR2: estrategia de uso del MIG.....51
4.4.1. Resultados de velocidad.....52

5. Capítulo 5: Conclusiones.....53
5.1. Conclusiones técnicas.....54
5.1. Conclusiones personales.....55
5.1. Trabajos futuros.....56

6. Referencias / Bibliografía.....58

CAPÍTULO 1

Introducción

En este primer capítulo se van a describir brevemente dos de los tres conceptos que se pueden extraer del propio título del presente proyecto. En primer lugar, ¿qué es un sistema de inyección de fallos?, ¿cuál es la principal motivación para estudiar estos sistemas? Se darán respuestas, que servirán para ubicar la situación en la que se encuentran hoy día las técnicas que intentan suprimir los efectos negativos de un problema que se conoce desde hace años. Y en segundo lugar, ¿a qué nos referimos con segunda generación? Se verá la evolución de un sistema de inyección de fallos y así en posteriores capítulos se podrá comprobar la mejora de los resultados mediante el desarrollo de las nuevas interfaces realizadas en este proyecto.

1.1. Problemática

La evolución tecnológica en términos de la electrónica, se mide entre otros factores, en la reducción del tamaño en los procesos de fabricación de los diferentes dispositivos y en un empleo de fuentes de alimentación de menores voltajes. El principal motivo de dicha evolución es la competencia existente entre una multitud de empresas implicadas, para ofrecer al público diferentes productos electrónicos, cada vez más inteligentes, con menores tamaños, baterías más duraderas, etc.

Por otro lado, los problemas de la radiación en circuitos electrónicos se estudian desde los años 60, cuando a raíz de la carrera espacial se comenzaron a encontrar una serie de fallos en los sistemas electrónicos desconocidos hasta entonces. Por aquellos años, la tecnología era de un tamaño muy superior al de hoy día, y por ello los daños causados por rayos cósmicos o por protones de alta energía no afectaban a los dispositivos a nivel de la superficie terrestre; mientras que fuera de la atmósfera, las altas energías de las partículas que interactúan con la electrónica si podían llegar a provocar errores.

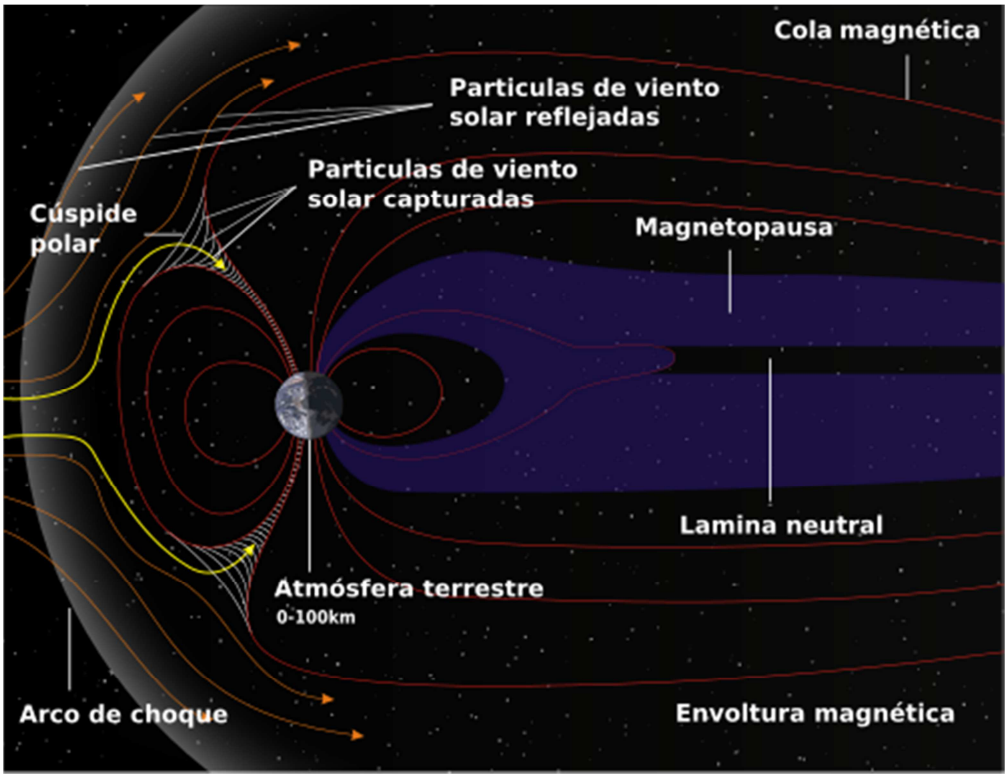
Si se unen ambos planteamientos, el resultado que se obtiene es que el escalado que se realiza en la tecnología electrónica, provoca que cada vez más se puedan ver afectados dispositivos electrónicos a nivel de superficie terrestre a causa de la radiación. Por todo esto, aplicaciones que requieran una alta fiabilidad ya sea en un entorno hostil como el sector aeroespacial o sistemas a nivel del mar (automoción, telecomunicaciones, etc) cada vez tienen más en cuenta los fenómenos producidos por los efectos de la radiación. Este es el principal motivo y punto de partida del estudio y despliegue de sistemas de inyección de fallos.

Descrita la situación en la que se encuentra el desarrollo del presente proyecto, en los próximos apartados se van a describir por un lado cuáles son los fallos o errores que pueden aparecer a causa de la radiación y por otro lado, conocer un sistema de inyección de fallos desarrollado por profesores del Departamento de Ingeniería Electrónica de la Universidad de Sevilla, a cuya evolución se le han aplicado, entre otras mejoras, los estudios realizados en este proyecto fin de carrera.

1.2. Efectos de la Radiación

Las mayores fuentes que producen daños por radiación en los circuitos electrónicos son las siguientes:

- Protones y electrones atrapados en los cinturones de Van Allen
- Iones pesados atrapados en la magnetosfera
- Rayos cósmicos, cuya composición es aproximadamente de un 85% protones, 14% partículas alfa y un 1% de iones pesados
- Iones pesados y protones procedentes de llamaradas solares



Partes de la Magnetosfera
(Imagen cedida por la NASA en dominio público)

El estudio de cómo afectan los daños producidos por la radiación, ha dado como resultado el diferenciar entre dos grandes categorías. La principal diferencia se basa en que los errores al fin y al cabo, son producidos por fenómenos físicos muy complejos y siempre se intentan modelar con parámetros circuitales, ya que resulta más sencillo a la hora de plantear diseños de circuitos electrónicos y ver de qué manera afectan dichos modelos de error, ya sean valores de tensión, corriente o cargas en nodos concretos. Pues bien, estos modelos son preferiblemente realizados en el dominio digital ya que ayuda a simplificarlos y además poder aprovechar las capacidades de los simuladores de lógica digital y emuladores hardware mediante el uso de las FPGA. Aún así, no todos los errores se pueden modelar de esta manera ya que producen daños físicos en la estructura del propio circuito o son capaces de dejarlos inoperativos.

Este es el principal motivo que diferencia ambas categorías en el estudio de los efectos causados por la radiación. Por un lado aparecen los **errores físicos** (del inglés *hard errors*), son aquellos que pueden llegar a destruir físicamente partes del circuito y es imposible poder solventarlos de manera digital. Por otro lado, se encuentran los **errores lógicos** (del inglés *soft errors*), que sí se pueden modelar de manera digital y así ser capaces de analizarlos y encontrar estrategias para mitigarlos. Sobre estos últimos, se centran las técnicas que se van a presentar para la mejora de un sistema de inyección de fallos.

1.2.1. Mecanismos Físicos de Daños

Los mecanismos físicos que producen daños sobre los circuitos electrónicos pueden describirse de forma general en tres grupos:

1) 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.

2) Daño por Desplazamiento (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.

3) 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.

Dentro de este mecanismo de daños, se encuentran algunos de los efectos que más nos interesan en el estudio del presente proyecto.

1.3. Efectos de Eventos Únicos (SEE)

Como se ha comentado anteriormente, los dispositivos digitales son los que más se ven afectados por estos mecanismos de daños, y ese hecho permite emplear emuladores hardware usando FPGAs para simular estos ataques sobre diferentes diseños digitales y así analizar la robustez de dicho diseño, esta técnica es la que más adelante se explicará y es conocida como inyección de fallos.

Los principales daños que se pueden llegar a producir son: degradación de parámetros del dispositivo, aumento del consumo, pérdida de funcionalidad y sobre todo bits erróneos en memorias y lógica digital. En este punto nos disponemos a describir los principales efectos de eventos únicos, divididos en las dos categorías ya

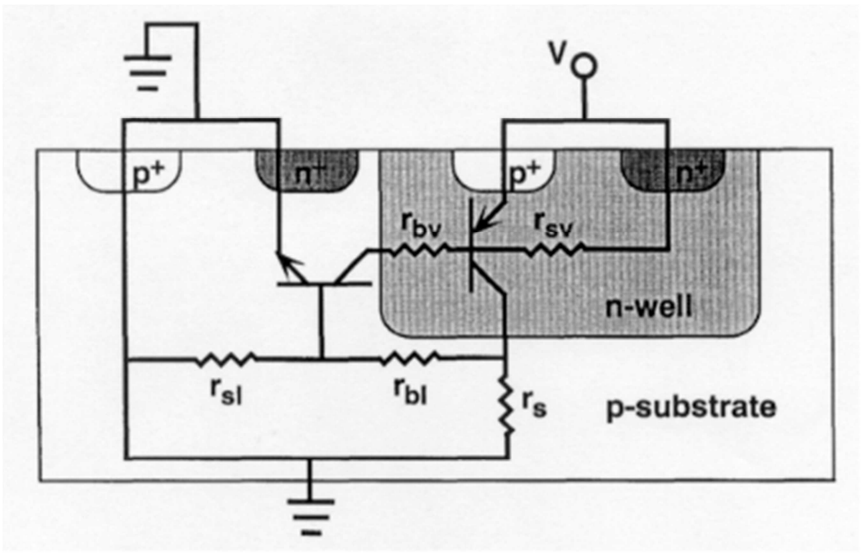
descritas según sus efectos sean imposibles de poder modelar de forma digital, o en caso contrario si se pueda obtener dicho modelo.

1.3.1. Errores Físicos

Como se ha comentado, son errores destructivos capaces de dejar inoperativo un circuito electrónico. La forma de protegerse ante este tipo de errores se realiza durante el proceso de fabricación de los dispositivos, empleando una tecnología denominada *rad-hard*, que implementa una serie de protecciones físicas sobre el circuito. En esta categoría se encuentran los siguientes efectos:

1.3.1.1. Enclavamiento por Evento Único

O también **SEL** (del inglés *Single Event Latchup*)**,** se da en dispositivos que contengan alguna estructura PNPN parásita. Cuando a causa de la radiación, una partícula de alta energía (iones pesados o protones) atraviesa una de dichas uniones puede activar alguno de los transistores bipolares parásitos generando un cortocircuito entre alimentación y tierra. Este efecto puede estar limitado a una pequeña región local o bien propagarse y afectar en mayor medida al dispositivo. Las grandes corrientes causadas por este efecto de cortocircuito pueden causar daños permanentes en los componentes si no están protegidas externamente contra dicha corriente de cortocircuito y la disipación de potencia correspondiente. Los dispositivos implementados directamente sobre un substrato de silicio, tipo 'Bulk CMOS', son más propensos a sufrir estos efectos que aquellos que emplean una capa de dieléctrico, como se realiza en las tecnologías SOI (del inglés *Silicon On Insulator*).

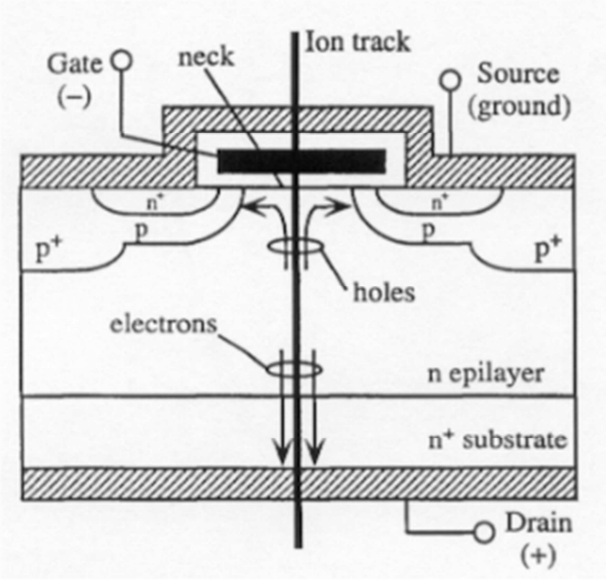


Transistores bipolares parásitos
(Imagen extraída de A.H.Johnston et al., IEEE Trans. Nucl. Science, Vol.43, N.6,Dec.1996)

Existe otra variante denominada Enclavamiento en el Sustrato por Evento Único (SES, del inglés *Single Event Snapback*), que no necesita contener una estructura PNPN. En este caso se da a causa de la activación de un transistor BJT parásito en la estructura NPN, que pueden darse en transistores MOS de canal N.

1.3.1.2. Ruptura de Puerta por Evento Único

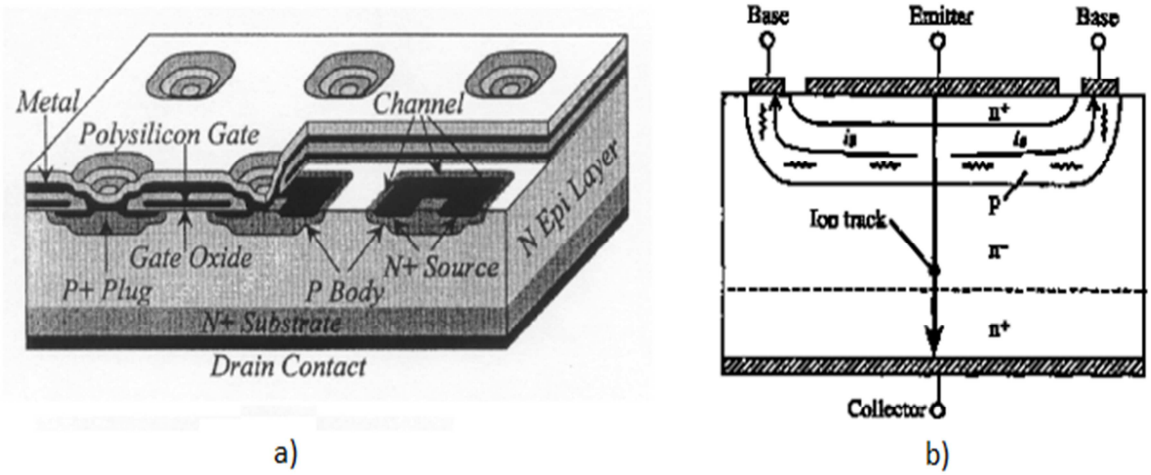
O también **SEGR** (del inglés *Single Event Gate Rupture*), este efecto se produce cuando un ión pesado impacta en la región de puerta de un transistor MOSFET de potencia. Este impacto junto con un alto valor de tensión aplicada en la puerta del transistor puede provocar una ruptura del dieléctrico en la capa aislante del dióxido de silicio, dando como resultado un sobrecalentamiento en la región de puerta y llegando a destruirla.



Detalle de la región de puerta
(Imagen extraída de J.H.Johnson & K.F.Galloway, IEEE NSREC short course, 1996)

1.3.1.3. Ruptura por Evento Único

O también **SEB** (del inglés *Single Even Burnout*), al igual que en el caso anterior se da en transistores MOSFET de potencia y también en transistores BJT de potencia, ya que ambos pueden llegar a conducir elevados valores de corrientes en sus conmutaciones. Sus estructuras circuitales están preparadas hasta ciertos valores de corriente, valores que a causa de la radiación pueden aumentar debido a que la tensión drenador-fuente es mayor que la tensión de ruptura de las estructuras parásitas que contienen ambos dispositivos. De nuevo, se produce un sobrecalentamiento que puede destruir completamente el transistor.



Estructuras de transistores de potencia: a) MOSFET b) BJT
(Imagen extraída de J.H.Johnson & K.F.Galloway, IEEE NSREC short course, 1996)

1.3.2. Errores Lógicos

A diferencia de los efectos anteriormente mencionados, los que aquí se describen producen efectos transitorios que no llegan a destruir físicamente un dispositivo o circuito electrónico. En este caso, las partículas que atacan a un diseño digital por medio de la radiación, provocan pulsos de tensión debidos a pequeñas corrientes instantáneas generadas al impactar con el circuito. Estos pulsos pueden ser capturados según ciertas condiciones y provocar así conmutaciones en registros o memorias que alteren el funcionamiento esperado de un sistema digital.

Los efectos que se encuentran dentro de esta categoría sí pueden ser modelados de forma digital, con idea de poder analizar los cambios que producen en cada diseño específico y así poder obtener información útil referente a la vulnerabilidad del diseño, con el objetivo de aumentar la robustez y por ende la fiabilidad del sistema completo.

Existe un modelo básico para estudiar este tipo de efectos, denominado ***Bit-Flip***. Se trata de un modelo asociado a la consecuencia de los fenómenos físicos producidos sobre un dispositivo digital cuando sufre alguno de los siguientes ataques que se van a describir. 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.

En esta categoría se encuentran los siguientes efectos:

1.3.2.1. Conmutación por Evento Único

O también **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.

Este tipo de efecto afecta de igual manera a tecnologías bipolares como tecnologías CMOS, y ambas tecnologías van reduciendo el tamaño de sus transistores a medida que evolucionan las técnicas de fabricación de dichos dispositivos. Esto conlleva a reducir el valor del parámetro Q_c necesario para producir un SEU, por lo tanto los dispositivos se vuelven cada vez más sensibles a partículas ionizantes de menor energía, provocando que cada vez sea más usual tener este efecto en dispositivos situados en torno a alturas cercanas al nivel del mar.

1.3.2.2. Conmutación Múltiple de Bits

O también **MBU** (del inglés *Multi Bit Upset*), es una variante del efecto anterior, produce el mismo efecto de cambiar el estado lógico pero en este caso no se trata de un sólo biestable si no de un conjunto de ellos al mismo tiempo.

Se dan en circuitos que tengan en ciertas regiones una alta densidad de transistores; por ejemplo las memorias RAMs se ven muy afectadas ya que tienen varios biestables en regiones de área muy pequeñas. A medida que la tecnología evoluciona, a una mayor densidad de transistores, mayor es la posibilidad de tener este tipo de efecto.

1.3.2.3. Transitorio por Evento Único

O también **SET** (del inglés *Single Event Transient*), este efecto se da cuando la salida de una puerta lógica sufre un cambio temporalmente debido a que un transistor de dicha puerta que debería de estar cortado según su entrada, en cambio conduce a causa de la generación de carga producida por el impacto de alguna partícula ionizante. Si este efecto es capturado por un biestable, entonces aparece un fenómeno equivalente al SEU.

Para que este efecto tenga consecuencias graves deben de darse varias condiciones ya que dependiendo de la lógica en la que se haya producido, este efecto puede propagarse por el circuito y afectar al funcionamiento esperado del sistema, o bien puede enmascararse y mitigarse sin que el circuito se vea afectado. Además del estado de la lógica, este efecto tiene mayor o menor posibilidad de aparecer dependiendo de factores como la frecuencia del reloj y del ancho del pulso transitorio que se genera a causa de la radiación.

1.3.2.4. Múltiples Conmutaciones por Evento Único

O también **SEMU** (del inglés *Single Event Multi Upset*), se trata de un efecto parecido al MBU pero con la diferencia esencial de que los biestables no tienen que estar próximos entre sí. Se da cuando el pulso transitorio generado por un SET tiene un retraso de propagación similar hasta cada uno de los biestables implicados, así cuando llega el flanco de reloj, se captura el transitorio en cada uno de los biestables afectados a la vez.

De manera resumida, podemos clasificar todos los efectos vistos anteriormente en la siguiente tabla:

Efecto	Categoría de Error	Modelo Digital
TiD	Físico	No
DD	Físico	No
SEL	Físico	No
SEGR	Físico	No
SEB	Físico	No
SEU	Lógico	Bit-Flip en biestable único
MBU	Lógico	Bit-Flip en conjunto de biestables
SET	Lógico	Bit.Flip en biestable único que captura el transitorio
SEMU	Lógico	Bit-Flip en varios biestables que capturan el transitorio a la vez

Clasificación de Efectos por Eventos Únicos

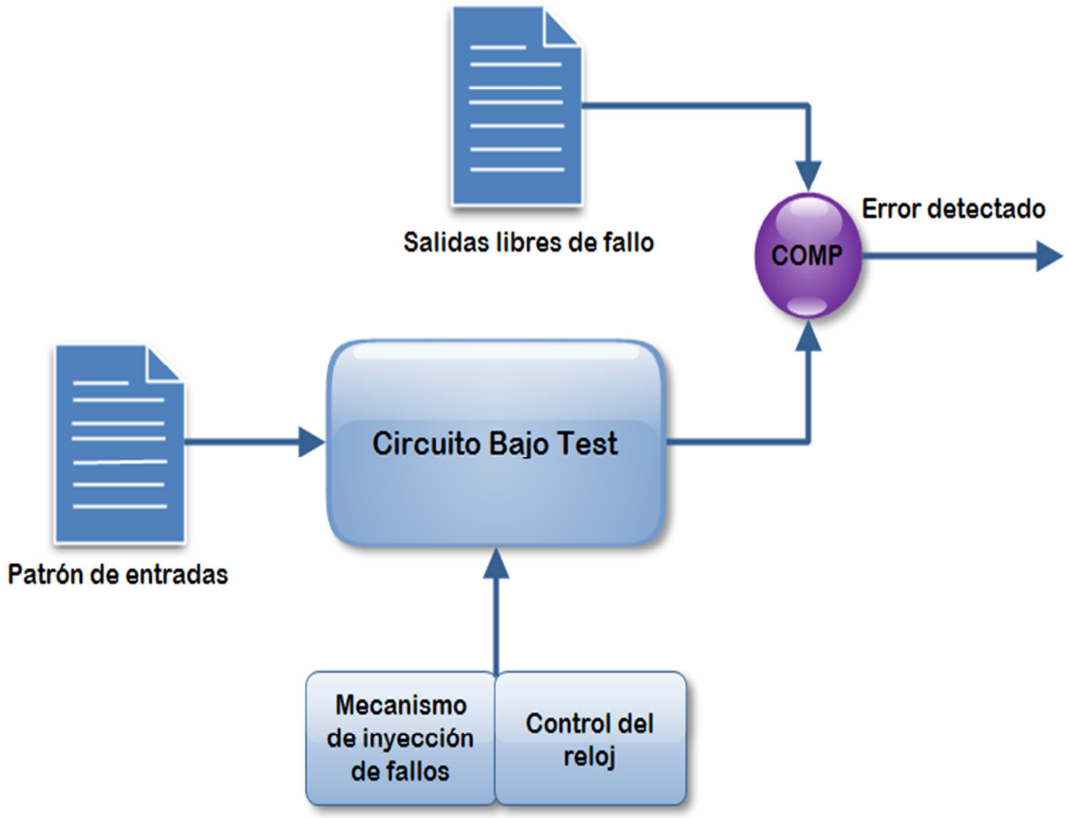
Como ya se ha comentado, para el presente proyecto se han desarrollado unas interfaces de datos de alta velocidad que ayudan a mejorar el funcionamiento de un sistema de inyección de fallos que se describe más adelante. Ya se conocen los motivos principales y los efectos que la radiación provoca sobre un circuito electrónico, que dan sentido al estudio y desarrollo de sistemas de inyección de fallos. Ahora queda aclarar en qué consisten estos sistemas, qué pueden aportar y cómo se utilizan.

1.4. Sistemas de Inyección de Fallos

La problemática asociada a los efectos negativos que la radiación produce sobre sistemas electrónicos, ya sean aplicaciones espaciales, telecomunicaciones, automoción, etc. y cada vez más influyente en sistemas establecidos a nivel del mar, ha hecho que se requieran de técnicas que ayuden a estudiar los fenómenos físicos que degradan las prestaciones de un circuito electrónico y encontrar la manera de mitigarlos. Una de las técnicas más empleadas es la Inyección de Fallos con la cual es posible analizar la tolerancia de un circuito electrónico a ataques sufridos por la radiación.

La idea que hay tras este tipo de técnicas es recrear las condiciones de fallo de un circuito a causa de la radiación, sobre una representación del circuito en cuestión realizada a un nivel de abstracción concreto. Con esto, se puede analizar la sensibilidad frente a errores lógicos que presenta dicho circuito antes de la fase de fabricación del mismo, obteniendo resultados de posibles nodos sensibles y críticos que ayuden a mejorar el diseño final.

De forma genérica, un sistema de inyección de fallos consta de un circuito bajo test o MUT (del inglés *Module Under Test*) y un patrón de entradas que requiere un cierto número de ciclos de reloj conocido, para poder ser recorrido por completo. El funcionamiento consiste en poner en funcionamiento al circuito desde un estado conocido de reset hasta un ciclo determinado, a partir del cual se pone en marcha un mecanismo de inserción de fallos o bit-flips en uno o varios registros de memoria del circuito. Este proceso continúa hasta encontrar alguna discrepancia en las salidas con respecto a las salidas del circuito libre de fallos conocidas de antemano.



Esquema de un Sistema de Inyección de Fallos genérico

Para este tipo de técnica se emplea una dupla del tipo: [BIT, TIEMPO], con la cual se modela un fallo representando el dónde y cuándo se inyecta cada uno de ellos dentro del MUT. Los fallos pueden clasificarse de la siguiente manera:

- **Silencioso** - cuando no se detecta discrepancia alguna en las salidas ni en el estado interno del MUT. Es un caso en el cual el daño ha sido compensado por la propia estructura del circuito.
- **Latente** - cuando no se detectan diferencias en las salidas pero sí en el estado interno del MUT.
- **Daño** - cuando se obtienen diferencias en las salidas con respecto a aquellas esperadas por el sistema sin fallos.

La técnica de inyección de fallos es determinista, ya que como se acaba de comentar, se conoce en todo momento el dónde y cuándo se inyecta cada fallo; es decir, el usuario tiene en su mano información sobre en qué parte del circuito se inyecta y en qué instante de tiempo.

Existen otras técnicas que ayudan a conocer los efectos que la radiación provoca sobre dispositivos electrónicos, como por ejemplo los test de radiación. Este tipo de test requiere de instalaciones capaces de recrear un ambiente similar al que se encontraría un circuito electrónico, sufriendo ataques de partículas de alta energía. Son técnicas de mayor coste que la inyección de fallos y no es posible manejar información del registro o el instante de tiempo en el que se inyectan los fallos. De hecho, las técnicas de inyección de fallos tratan de emular a los test de radiación de manera que puedan obtener resultados fiables y con menores costes y necesidad de menos recursos.

1.4.1. Requisitos básicos y tipos de técnicas

Todo sistema de inyección de fallos que pretenda emular un test de radiación aplicando errores lógicos a un circuito electrónico, debe de poder tener la capacidad de *insertar bit-flips*. Esto es así porque como ya se ha comentado anteriormente, los errores lógicos pueden modelarse digitalmente y es precisamente el modelo de bit-flip el más utilizado, proporcionándole al usuario la posibilidad de leer el valor de ciertos elementos de memoria pertenecientes al circuito prototipo, y reescribir dicho valor lógico por su opuesto.

Un parámetro fundamental para este tipo de técnicas es la *velocidad*, normalmente representada por número de fallos inyectados por unidad de tiempo. El *alcance* es importante cuando se trata de inyectar en circuitos de alta complejidad, como es el caso de los microprocesadores, por ejemplo existen técnicas que no pueden de por sí inyectar en una caché de un microprocesador y otras que lo hacen de forma innata al tratarla como una memoria más del diseño completo. Un parámetro que no se puede nunca olvidar es el *coste*, tanto económico como temporal, en referencia al ciclo completo del test desde que se prepara hasta que se obtienen los resultados; y también hay que resaltar la importancia de la *precisión* con la que se llega a representar el diseño final.

Además se pueden citar otras características de tipo secundario que aportan algo más de información a la hora de poder realizar una elección más afinada sobre qué tipo de técnica se puede emplear sobre un circuito prototipo específico. Por un lado la *información necesaria* (descripción HDL, ipcores, layout) sobre el circuito a testear, otra característica define la precisión de la herramienta software que corre el circuito (denominada *precisión SW*), otra como el *alcance temporal* que indica la posibilidad de poder inyectar en todos los ciclos de reloj pertenecientes al ciclo de trabajo, y finalmente si es posible emplear cualquier *tipo de circuito* o sólo es posible inyectar en dispositivos concretos (como un sistema microprocesador).

Existen diferentes clasificaciones dentro de las técnicas de inyección, en el presente trabajo se va a representar mediante una tabla, que pretende mostrar de manera rápida y clara las prestaciones más significativas de cada tipo de técnica sin entrar en detalle del funcionamiento de cada una de ellas:

Técnica		Velocidad	Alcance	Coste	Precisión	Info necesaria	Precisión SW	Alcance temporal	Tipos de circuito
Simulada		✗	✓	✓✓✓	✓	✗	✗	✓✓✓	✓
Emulada por FPGA	Instrumentada	✓✓✓	✓	✓✓	✗✗	✗	✓✓	✓✓✓	✓
	No Instrumentada	✓✓	✓✓✓	✓✓	✓✓✓	✓	✓✓	✓✓✓	✓✓✓
Emulada sobre Prototipo	Pin-Level	✓✓	✗✗	✓✓	✓✓✓	✓✓✓	✓✓✓	✗	✓✓✓
	SWIFI	✓✓	✗	✓✓	✓✓✓	✓✓✓	✓	✗	✗✗
	Láser	✓✓	✓✓✓	✗	✓✓✓	✗	✓✓	✓	✓✓✓

Comparativa de las distintas técnicas de inyección de fallos
Cortesía de Hipólito Guzmán Miranda (H.Guzmán[1])

El presente proyecto se centra en el empleo de la técnica de emulación por FPGA No Instrumentada. Lo que se tiene es un sistema que aprovecha las características que Xilinx ofrece con la posibilidad de emplear la reconfiguración parcial de sus dispositivos programables, como es el caso de algunas FPGAs de dicho fabricante. Con esta característica se obtiene una mayor depuración aprovechando circuitería de la FPGA donde además se implementa el prototipo, por lo que se tiene una mejor controlabilidad y observabilidad de los bits del circuito.

Esta técnica tiene como ventaja que no se modifica la netlist original del prototipo; es decir, se realiza la inyección al MUT sin verse afectado por ninguna circuitería externa, y además se llega a mantener una alta velocidad de test. Otra de las principales ventajas es que resulta más sencillo preparar los diseños para los diferentes test que quieran realizarse sobre el MUT, con lo que se acelera el proceso al ser posible una automatización más o menos sencilla del proceso. También es importante en el uso de estas técnicas, el hecho de que una empresa no esté dispuesta a prestar el código de su circuito prototipo para que se le pueda aplicar la inyección sobre una FPGA. En estos casos, que por motivos de propiedad intelectual no se dispone del código fuente del MUT, esta técnica permite que pueda realizarse el test con un fichero EDIF o NGC post-síntesis del mismo.

1.4.2. Sistema de 1ª Generación: FT-UNSHADES

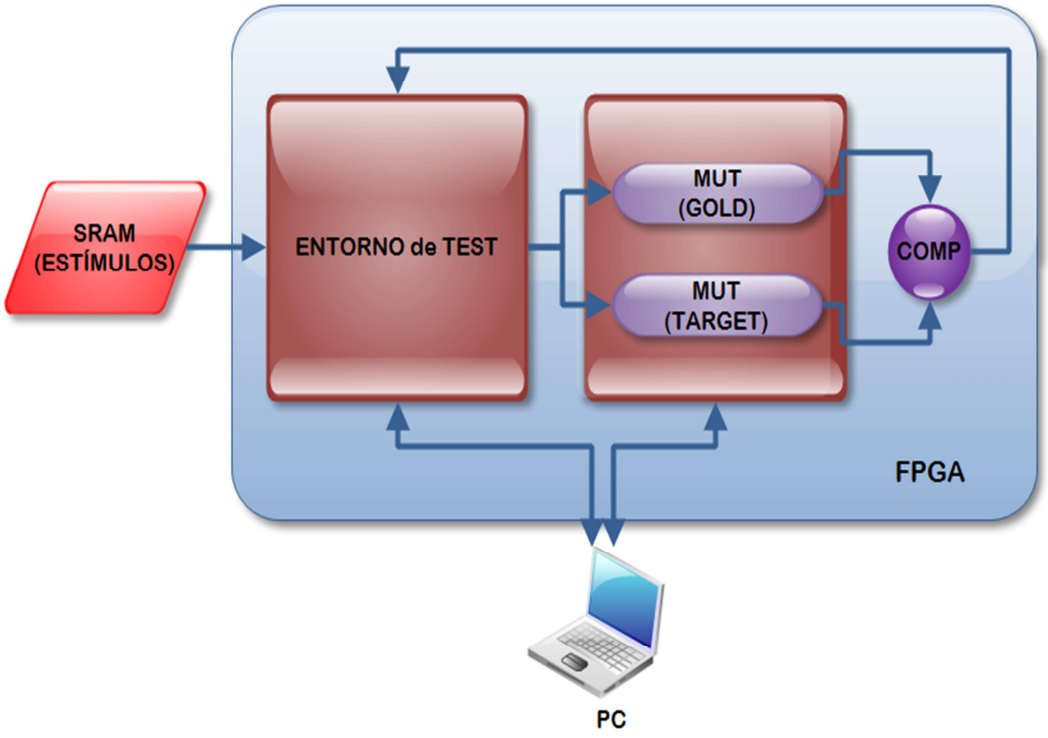
En este apartado se describe un sistema de inyección de fallos desarrollado[1] por el departamento de Ingeniería Electrónica de la Universidad de Sevilla, cuyo nombre

[1] Puede obtenerse más información sobre FT-UNSHADES en los siguientes documentos: M.Aguirre[2] y en el enlace web M.Aguirre[3]

proviene de la siguiente descripción en inglés: **F**ault **T**olerant - **UN**iversity of **S**eville **HA**rdware **DE**bugging **S**ystem.

Se trata de un sistema que se encuentra dentro de los Emulados sobre FPGAs y No Instrumentados. Mediante el empleo de la reconfiguración parcial de elementos programables dentro de una FPGA del fabricante Xilinx, es capaz de realizar el estudio de fiabilidad de netlists sobre un diseño de un circuito digital. El sistema completo consta de dos partes bien diferenciadas, por un lado se encuentra la plataforma hardware dedicada, que emplea una FPGA de la familia Virtex-II de Xilinx. Por otro lado, dispone de un conjunto de herramientas software que sirven de interfaz al usuario tanto para preparar un test como para visualizar los resultados.

De manera general, la arquitectura de la plataforma FT-UNSHADES consta de una doble instancia del diseño prototipo, una denominada SEU (o TARGET) MUT que es la atacada con los bit-flips y otra denominada GOLD MUT a la que no se le inyecta ningún fallo, que corre en paralelo con la anterior y se emplea para comparar las salidas de ambas instancias. Ambas instancias del diseño a testear están gobernadas por el controlador de reloj del sistema, con el cual se determina cuándo se inyecta un bit-flip. Ambas salidas van a parar a un comparador donde se analizan las posibles discrepancias y además las entradas (o estímulos) llegan a las instancias desde una memoria externa donde están almacenadas. Para completar el modelo de inyección de fallos falta nombrar el software que emplea este sistema, denominado TNT (del inglés Test aNalysis Tools) que hace el papel de interfaz entre la plataforma hardware y el ordenador de usuario.



Arquitectura de FT-UNSHADES

Desde el punto de vista hardware, se inyectan fallos de tipo SEU mediante modelos de bit-flip en uno o varios registros de usuario pertenecientes al MUT TARGET. La

inyección se realiza por medio de la lectura de paquetes de bits de configuración que contienen el valor actual de un registro de usuario, luego se procesan (momento en el que se le aplica un bit-flip mediante una escritura), y finalmente son cargados en la plataforma hardware con los nuevos valores. Este proceso se denomina **ciclo de inyección** (para este sistema también se emplea el término *run*), que no es más que una emulación completa del MUT en la FPGA, durante la cual se ha tenido que insertar al menos un bit-flip.

El programa TNT constituye la parte software del sistema, su función principal consiste en leer el fichero de asignación lógica (.ll del inglés *logic location*), de donde se extrae la posición física de cada registro del MUT instanciado dentro de la FPGA y así poder establecer la conexión con el nombre lógico de cada registro que se obtiene tras el procesos de síntesis a alto nivel. El fichero .ll se obtiene del propio flujo de diseño estándar que proporciona Xilinx.

Esta característica de poder relacionar cada elemento de memoria interno del circuito, que se encuentra reflejado en la descripción a alto nivel del mismo, con el bit concreto al que se corresponde en la FPGA, soluciona tanto el problema de controlabilidad como el de observabilidad. Ya que es posible atacar cualquier bit de la FPGA a través de la información obtenida de la lectura del archivo de asignación lógica.

Una vez demostrada la técnica de inyección de fallos que se puede realizar con la plataforma FT-UNSHADES, queda resaltar sus principales características:

- Al ser un sistema No Instrumentado, no introduce circuitería extra que modifique al MUT, se trata de una técnica **no intrusiva**.
- Puede alcanzar **velocidades** de test de hasta 100 fallos/seg, donde el cuello de botella se encuentra en el enlace entre la FPGA y el PC (se trata de un enlace de comunicaciones USB 2.0 o bien un puerto paralelo EPP1.9 a 1.5 MB/s).
- Capaz de realizar **análisis en profundidad** a cada prototipo de diseño digital.
- Minimiza los **costes** para la preparación y realización de un test.
- Es **flexible** ya que el sistema de inyección permite hacerlo en cualquier instante de tiempo y en cualquier Flip-Flop del diseño.
- Permite testear diseños de hasta 101 entradas (en número de salidas viene limitado por la capacidad de la FPGA).

1.4.3. Sistema de 2ª Generación: FTU2

En este punto se comienza a describir el sistema para el cual se han estudiado e implementado las dos interfaces de datos de alta velocidad que son el fundamento principal del presente proyecto. FTU2 es una plataforma desarrollada por el Departamento de Ingeniería Electrónica de la Universidad de Sevilla, y se trata de una evolución natural de su antecesor FT-UNSHADES.

Esta nueva generación busca ante todo cumplir con 5 objetivos:

1. Mantener la esencia de FT-UNSHADES como técnica no intrusiva y analizar tanto fallos tipo SEU, como SET e incluso MBU.
2. Permitir la instancia de diseños con circuitos más complejos y de mayores tamaños.

3. Poder almacenar vectores de test de mayor tamaño así como disponer de memorias más grandes para almacenar los estímulos y resultados.
4. Aumentar la tasa de inyecciones por segundo y en definitiva acelerar el proceso del ciclo de inyección completo (al menos superar en un orden de magnitud los valores alcanzados en FT-UNSHADES).
5. Crear una nueva interfaz de usuario con la plataforma, más intuitiva y con acceso web.

En lo que atañe al presente proyecto, se van a analizar dos técnicas que van a ayudar en parte a resolver los puntos 3 y 4 anteriormente citados. En concreto, se trata por un lado de una interfaz de almacenamiento de datos sobre una memoria DDR2, para cuyo manejo se han estudiado dos vías: uso de un microprocesador integrado en una FPGA tipo PowerPC440, o la implementación de un MIG (del inglés *Memory Interface Generator* de Xilinx). Por otro lado, se implementa un enlace de comunicaciones usando el protocolo PCIExpress a través del cual se establece la comunicación y transvase de datos útiles entre varias FPGAs que compone la plataforma. El desarrollo de las dos técnicas que se acaban de comentar se verán explicadas en el capítulo 3.

Este trabajo no pretende abarcar una descripción completa sobre la plataforma FTU2, de hecho por motivos de propiedad intelectual, se trata de un proyecto perteneciente a la Agencia Espacial Europea (ESTEC Contract no. **22981/09/NL/JK**) y por ello no podrá mostrarse código fuente sobre las implementaciones realizadas o detalles muy concretos sobre las mismas. Aún así, en el siguiente capítulo se utiliza FTU2 como modelo para explicar y dar sentido al estudio y desarrollo de las interfaces anteriormente mencionadas, ya que es la plataforma con la que se ha trabajado para el desarrollo del presente proyecto. Se harán descripciones de la funcionalidad y se mostrarán esquemas que ayuden a comprender el trabajo que se ha realizado, así como indicar los resultados que se han obtenido.

CAPÍTULO 2

Descripción de la plataforma de trabajo

Durante este capítulo se pretende describir la plataforma sobre la cual se han implementado las interfaces de datos de alta velocidad que ayudan a mejorar las prestaciones del sistema de inyección de fallos de primera generación. En concreto, se emplea la plataforma FTU2, gracias a la cual se podrá comprobar si efectivamente se han conseguido mejoras tanto en almacenamiento como en velocidad del ciclo de inyección en relación con los resultados que se tienen de su antecesor, el sistema FT-UNSHADES. El objetivo de este capítulo es por tanto, presentar la arquitectura de la plataforma y localizar la necesidad y uso de ambas interfaces que dan sentido al presente proyecto.

2.1. Arquitectura de FTU2

Esta plataforma de inyección de fallos que se clasifica dentro de las emuladas sobre FPGA y No Instrumentada, sigue a grandes rasgos el esquema que se vio en el apartado 1.4.2. del capítulo anterior, incluyendo cambios en la arquitectura que buscan un aumento y mejora de las capacidades del sistema FT-UNSHADES.

Dichos cambios se pueden dividir en tres apartados: cambios de hardware, software y firmware. Durante este capítulo se describen los principales cambios referidos al hardware y software de la plataforma con respecto a su antecesor, también se menciona la estructura general del firmware del sistema para localizar así las dos interfaces de interés desarrolladas para este proyecto.

2.1.1. Parte Hardware

Respecto al nivel hardware, a diferencia de FT-UNSHADES donde se tienen dos instancias idénticas del MUT dentro de una única FPGA de la familia Virtex II (XC2V8000) de Xilinx, en FTU2 se dispone de dos tarjetas compuestas por dos FPGAs cada una (denominadas tarjetas *hijas* o **DB** del inglés *Daughter Board*), que permite separar físicamente ambos diseños prototipo, interconectadas con otra FPGA (denominada tarjeta *madre* o **MB** del inglés *Mother Board*) que realiza funciones de control.

Las tarjetas hijas reciben un nombre según su función, una se denomina GOLD para indicar que incluye una instancia del MUT que no va a ser atacada, y la otra se llama TARGET ya que incluye el MUT sobre el cual se realizan las inyecciones. Por tanto, en FTU2 se disponen de cinco FPGAs en total:

- **Tarjeta Madre** - 1 FPGA Virtex V
    - o denominada FPGA de Control: C-FPGA (del inglés *Control FPGA*)
        - Modelo ML510 XC5VFX130T de Xilinx

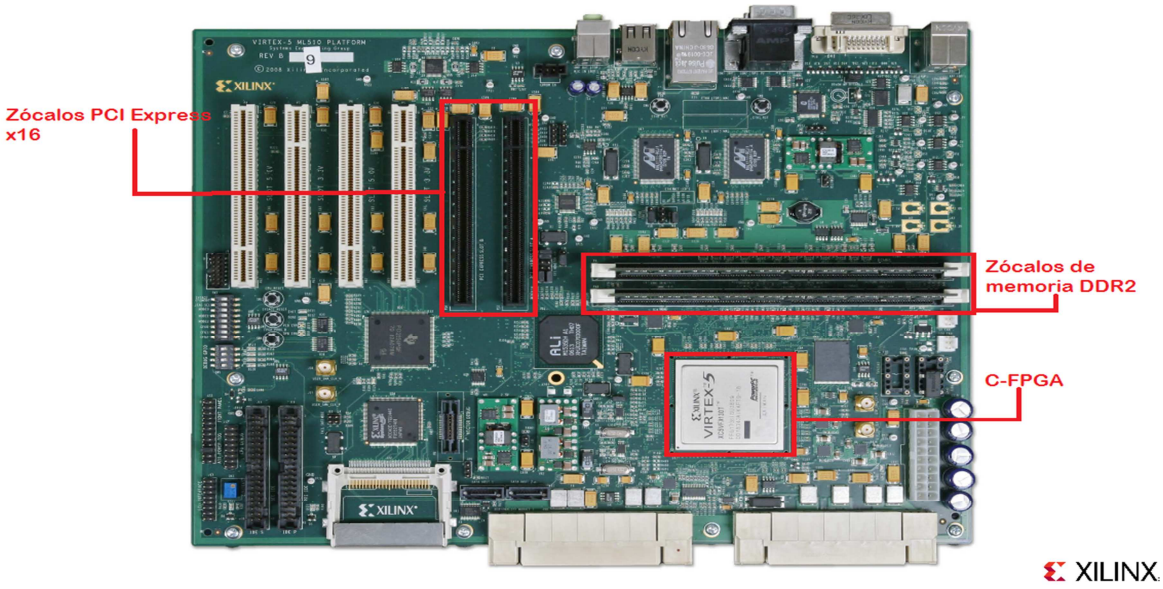


Imagen de la tarjeta ML510

- **Tarjeta Hija (GOLD)** - 2 FPGAs

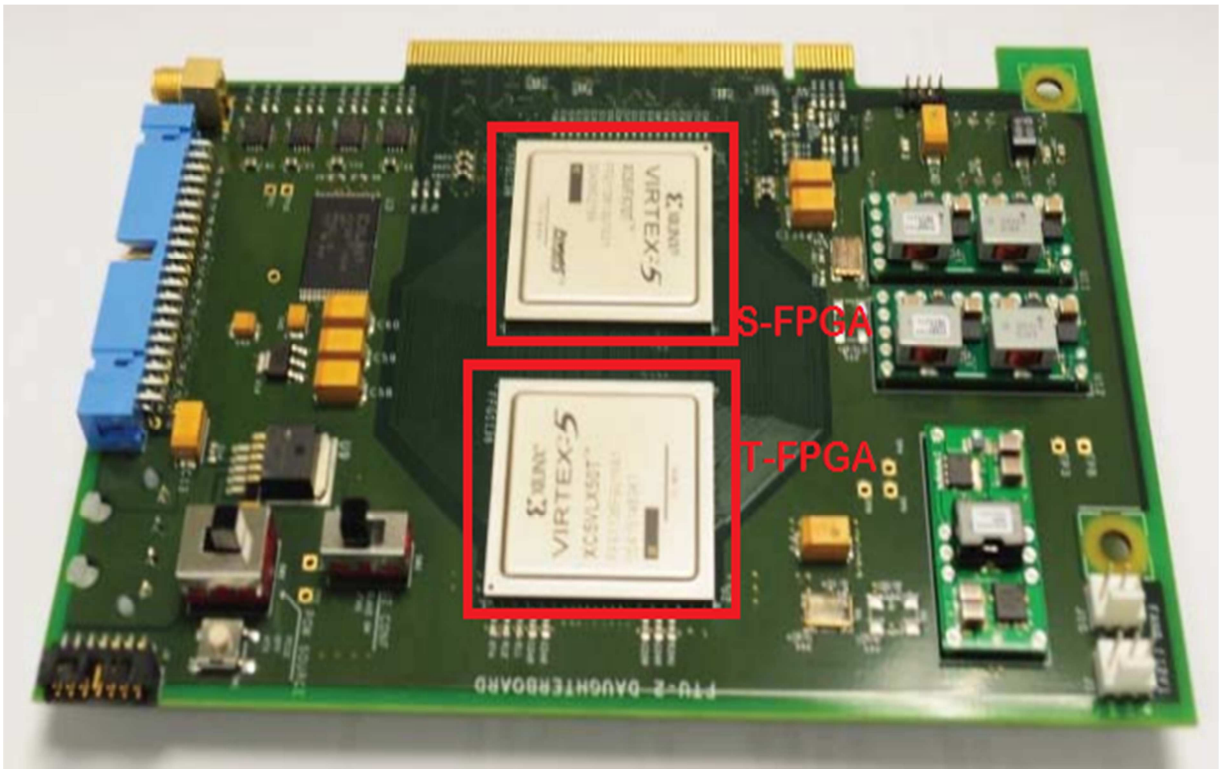
    - 1 denominada FPGA Objetivo: T-FPGA (del inglés *Target FPGA*)
        - Modelo: Depende del usuario[2]
        - Función: Implementación del MUT

    - 1 denominada FPGA de Servicio: S-FPGA (del inglés *Service FPGA*)
        - Modelo: XC5VFX70T de Xilinx
        - Función: Establecer las comunicaciones entre la T-FPGA y la C-FPGA

- **Tarjeta Hija (TARGET)** - 2 FPGAs
-

    - 1 denominada FPGA Objetivo: T-FPGA (del inglés *Target FPGA*)
        - Modelo: Depende del usuario
        - Función: Implementación del MUT

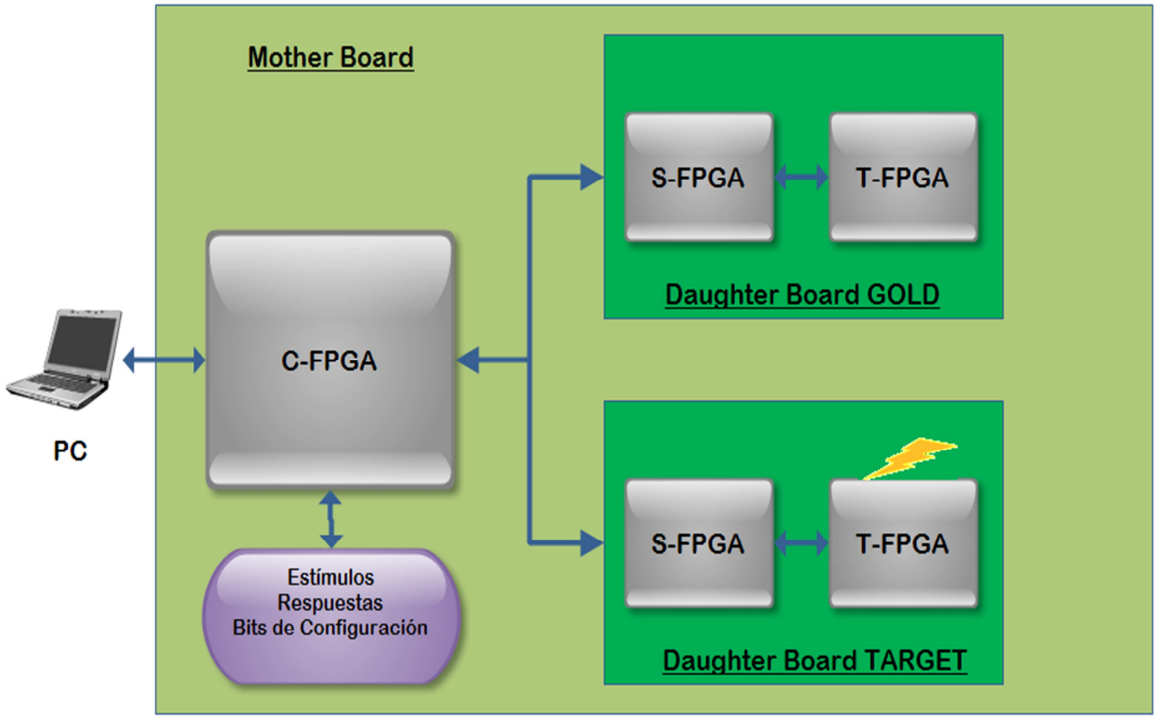
    - 1 denominada FPGA de Servicio: S-FPGA (del inglés *Service FPGA*)
        - Modelo: XC5VFX70T de Xilinx
        - Función: Establecer las comunicaciones entre la T-FPGA y la C-FPGA



Fotografía de una de las tarjetas hijas

[2] La tarjeta hija ha sido diseñada con idea de que el usuario pueda seleccionar entre una serie de FPGAs que puedan ser compatibles con el rutado hacia la S-FPGA y que se adecúe a su diseño prototipo. El primer prototipo de FTU2 ha sido diseñado para adaptarse a todas las huellas compatibles de FPGAs Virtex 5 FF1136. Para más información acudir a (J.M.Mogollón [4]).

De forma resumida, la MB se encarga de proporcionar los estímulos a las DB y recibir las salidas que dan ambas tarjetas. Luego, compara las salidas y entrega los resultados al software para que el usuario pueda analizarlo.



Arquitectura Hardware de FTU2

Cabe destacar las principales funciones que realiza la MB:

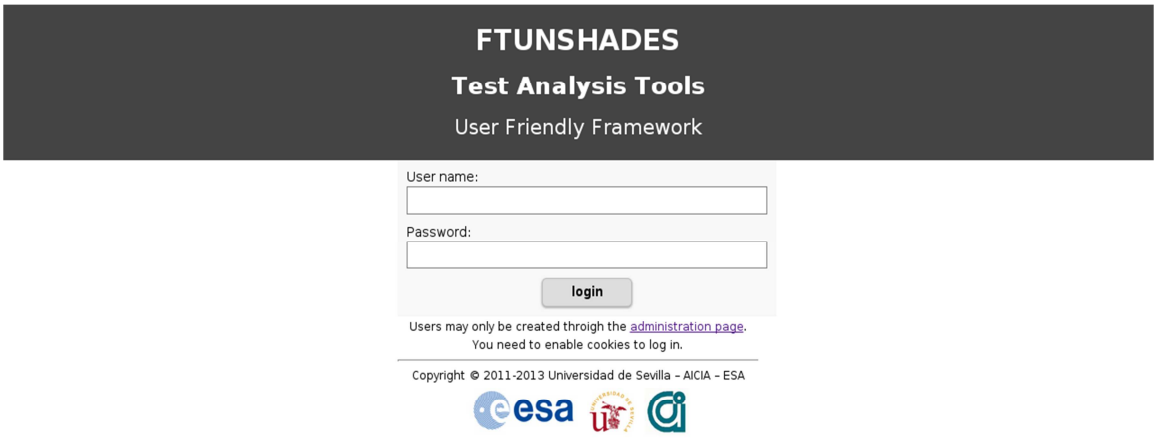
- Los fallos que van a ser inyectados se transmiten a la tarjeta en un sólo archivo a través de un puerto USB 2.0 o bien por Ethernet y el resto de la operación se realiza a nivel hardware, devolviendo más tarde el resultado del test hacia el PC del usuario. Con esto se minimiza el tráfico de datos desde el PC hacia la MB y viceversa, que resultaba ser el auténtico cuello de botella en la plataforma FT-UNSHADES.
- Contiene dos módulos de memoria DDR2 DIMM que pueden llegar a almacenar hasta 1 GB en total. En dichos módulos se almacenan los estímulos de entrada, los vectores de error, los archivos de configuración para las T-FPGAs y los resultados del test.
- Control del Reset y Reloj del sistema completo.
- Realiza la comparación de las salidas de ambos MUT, en todos y cada uno de los ciclos de test, así como almacenar los resultados de cada test.

2.1.2. Parte Software

A nivel Software también hay modificaciones muy importantes tanto de cara a la interfaz que la plataforma ofrece al usuario, como a las nuevas funcionalidades que aporta FTU2. El programa TNT (del inglés *Test aNalysis Tools*) se mejora en el sentido de que el usuario sólo debe preocuparse de preparar las acciones a alto nivel; es decir, configurar los parámetros de las inyecciones (el dónde y el cuándo), el diseño del MUT

y los estímulos de entrada. A diferencia de FT-UNSHADES, el resto de acciones de bajo nivel se realiza a nivel hardware, con lo que se gana en reducción de tráfico de datos desde el PC de usuario hasta la MB y se aumenta la velocidad del sistema.

Además se ha desarrollado una aplicación web, denominada UFF (del inglés *User Friendly Framework*), a través de la cual el usuario puede realizar los test por internet mediante cualquier navegador comercial e incluso desde un dispositivo móvil (teléfono, tablet, etc). Esta nueva forma de acceder al sistema hace que sea más sencilla e intuitiva la manera de trabajar con FTU2 además de permitir un acceso remoto a la plataforma, cosa que no era posible en FT-UNSHADES.



Inicio de sesión de UFF-TNT

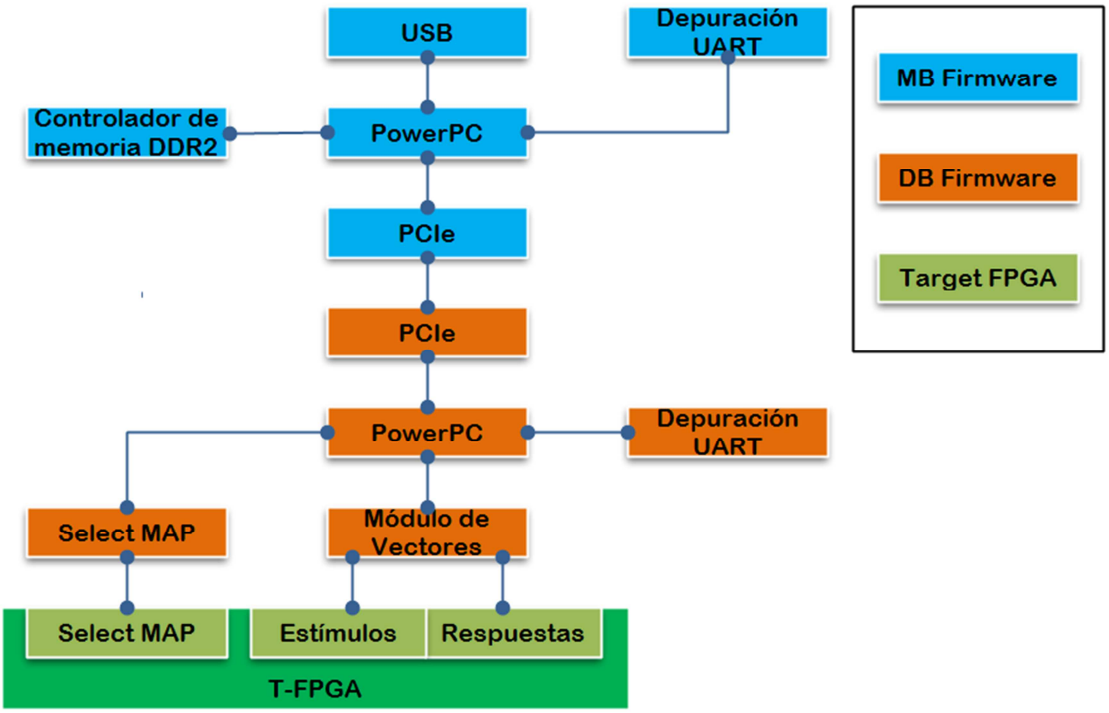
2.1.3. Parte Firmware

Este punto es el de mayor interés ya que se van a describir los principales módulos que hacen posible el funcionamiento interno del sistema FTU2. Dentro de los cuales se encuentran las dos interfaces desarrolladas para el presente proyecto, como ya se indicó en el capítulo anterior se trata de un sistema de manejo de una memoria DDR2 y un módulo para el protocolo de comunicaciones PCI Express.

Las unidades que componen el firmware son aquellas que constan de alguno (o más de uno) de los siguientes elementos:

- Código VHDL de elaboración propia
- Primitivas de Xilinx
- Software embebido

El siguiente esquema trata de mostrar el firmware que forma la plataforma completa de FTU2:



Estructura del Firmware de la plataforma FTU2

Como puede observarse en el gráfico, ambas tarjetas madre e hija se ayudan de un microprocesador tipo PowerPC 440 que viene integrado en cada una de las FPGAs empleadas; en concreto se emplean dichos microprocesadores en las dos S-FPGA de las tarjetas hijas y en la C-FPGA de la tarjeta madre. Estos elementos son los encargados de recibir los diferentes comandos que envía el usuario a través del puerto USB, al igual que reciben los datos (ya sean estímulos, vectores de test, respuestas, etc.) y organizar el flujo de información de la manera adecuada según cada comando recibido.

El puerto USB es la raíz del todo el tráfico de datos que se introduce o sale de la plataforma y es esencial en la interfaz con el usuario. También se observan puntos de observación y depuración a través de puertos serie (UART), que a pesar de que no influyen en el funcionamiento de los diferentes modos de operación de FTU2, son básicos para visualizar los posibles errores en la transmisión y recepción de datos durante las etapas de desarrollo.

Las FPGAs objetivo (T-FPGA) implementan los módulos necesarios para poder realizar la inyección de fallos sobre el MUT instanciado en ellas. Cabe destacar el módulo de SelectMAP ya que es el puerto por el cual se llega a configurar el diseño implementado en la T-FPGA; es un proceso crítico para el funcionamiento de FTU y sigue los siguientes pasos:

1. El usuario envía la configuración del MUT desde su PC hacia la plataforma FTU2.
2. A través del USB llega al PowerPC y éste almacena dicho archivo en la memoria DDR2.
3. Antes de comenzar una campaña, la MB se encarga de configurar ambas T-FPGA de forma simultánea, recogiendo el archivo guardado en la memoria DDR2 y enviándolo a cada una de las S-FPGA a través del puerto PCIExpress.

4. Cada S-FPGA puede configurar de forma automática a la T-FPGA gracias al puerto SelectMAP.

Por último, se observa en el gráfico que la conexión física entre la MB y las dos DB se realiza por medio de unos puertos PCI Express de hasta x16 líneas ya que implementan un protocolo de comunicaciones PCI Express 2.0.

Se tiene en este momento, totalmente descrita la localización y necesidad de dos elementos básicos para la funcionalidad del firmware. Por un lado, es necesario un módulo para controlar la memoria DDR2, ya que es la encargada de almacenar los ficheros que ponen en marcha los principales modos de operación de la plataforma FTU2, así como los ficheros generados como resultado de dichos procedimientos. Y por otro lado, la comunicación entre la tarjeta madre e hijas se realiza a través de puertos que manejan protocolos de comunicación PCI Express 2.0, por lo que es necesario la implementación de módulos que gestionan dichas comunicaciones para tener un control de datos eficiente y sin errores.

En el siguiente capítulo se describe de forma más precisa cómo se resuelven estas dos interfaces, sus principales bloques y funcionamiento, características hardware e incluso algunos detalles acerca del código desarrollado en ciertos módulos que han surgido a raíz de errores y problemas encontrados durante la etapa de diseño y depuración del sistema.

CAPÍTULO 3

Solución Propuesta

A lo largo del presente capítulo se van a describir las dos soluciones aportadas con el objeto de solventar dos de los diferentes módulos necesarios para el funcionamiento del firmware completo sobre el que se basa la plataforma FTU2. Se trata de conocer cada una de estas interfaces desarrolladas, sus principales características, sus elementos hardware, la problemática encontrada durante la etapa de estudio y desarrollo. Por tanto, en este capítulo se da respuesta al último concepto que reza en el título del presente proyecto; tras tener claro los conceptos de un Sistema de Inyección de Fallos de 2ª Generación, ahora se explica cuáles son esas interfaces de datos de alta velocidad. No se debe olvidar que el objetivo es mejorar las prestaciones del sistema anterior, conocido como FT-UNSHADES, y en el capítulo siguiente se presentarán los resultados obtenidos con el trabajo realizado sobre ambas soluciones propuestas.

3.1. Descripción General

La principal aportación de este proyecto fin de carrera se basa en el trabajo desarrollado sobre el estudio de soluciones a dos tipos de interfaces muy diferentes. Por un lado, resolver la manera de manejar una memoria DDR2 incluida en una tarjeta de desarrollo ML510 de Xilinx. El control del acceso a los datos debe de hacerse de forma eficiente, sin pérdida de datos ni obtención de duplicados para así obtener un flujo de comunicación rápido y continuo cuando se realice cualquier tipo de acceso ya sea de lectura o escritura. Por otro lado, existe un punto esencial en la arquitectura firmware de la plataforma que se trata de las comunicaciones entre la tarjeta madre y las hijas. Para ello se opta por emplear un bus serie de alta velocidad mediante el uso de un protocolo PCI Express 2.0 (o bus PCIe), donde el desafío recae en el manejo del tráfico de datos y al igual que ocurre con el controlador de la memoria, debe de ser un flujo fiable y eficaz ya que cualquier error en los datos puede acarrear fallos en los comandos recibidos, resultados del test erróneos, malas configuraciones de las T-FPGAs, etc.

3.2. Controlador de interfaz con memoria DDR2 SDRAM

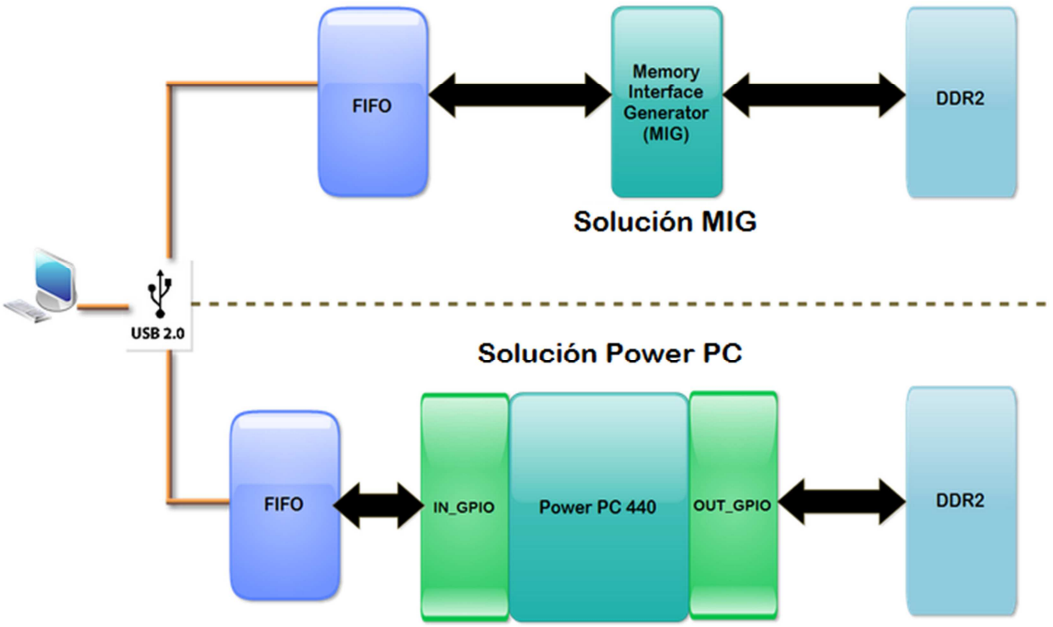
Queda claro que el uso de una memoria masiva es esencial para la arquitectura firmware de la plataforma, ya que en ella se almacenan los elementos básicos para realizar las inyecciones: archivos de configuración (bitstreams) de las T-FPGAs, estímulos de entrada que pueden llegar a ser archivos de gran tamaño (depende del objetivo del usuario), respuestas de las campañas, etc.

Para este trabajo se emplea una tarjeta de desarrollo tipo ML510 que incluye una FPGA Virtex V XC5VFX130T de Xilinx (la denominada C-FPGA) que actúa como *tarjeta Madre*. En ella encontramos dos zócalos de memorias DDR2 SDRAM Registered DIMM de hasta 1 GByte de capacidad cada una de ellas, aunque existe una anomalía en el pinout de la tarjeta ML510 que hace que sólo sea accesible físicamente la mitad de cada memoria; es decir, las memorias disponen de tres bits para indicar a qué dirección de banco acceder (ddr2_ba0, ddr2_ba1, ddr2_ba2), se da la circunstancia de que el pin que le corresponde a la señal ddr2_ba2 no se encuentra implementada en la FPGA, por lo que sólo tenemos accesible 512 MBytes para cada una de las memorias DDR2, con lo cual el sistema dispone de hasta 1 GBytes de capacidad total de almacenamiento. Las memorias se corresponden con un modelo del fabricante Wintec D2RE01GX809 (Wintec[5]), cuyas principales características son:

- 240 pines
- Tasa de datos de 667MHz y reloj de 3ns
- Posibilidad de detección y corrección de errores ECC
- Latencia programable (CAS 3 y 4)
- Longitud de ráfaga programable (4 u 8 palabras)
- Auto refresco

Para el manejo de la memoria DDR2 se han estudiado dos estrategias diferentes, una de ellas se basa en el control directo mediante una interfaz proporcionada por Xilinx denominada MIG (del inglés *Memory Interface Generator*), la otra emplea el

microprocesador Power PC 440 que viene integrado dentro de la C-FPGA. En el siguiente diagrama se muestra la arquitectura funcional de ambas estrategias:



Diferentes estrategias de control de la memoria DDR2

Como puede apreciarse a priori, la estrategia de emplear el MIG necesita de menos elementos a diseñar para obtener el control de la memoria DDR2; sin embargo, el primer prototipo de la plataforma FTU2 implementa la estrategia del control de la memoria por medio del microprocesador Power PC. Esto es así porque el diseño y manejo del MIG no es sencillo de resolver y trae varios problemas que al final no asegura su correcta integración en el sistema ya que se pierden o duplican datos en las transmisiones de pruebas que se realizan durante la etapa de depuración. Se trata de un diseño más complejo y aún sabiendo que es mucho más óptimo en términos de velocidad con respecto al uso del Power PC, se decide continuar con la estrategia del microprocesador ya que es mucho más sencilla de implementar, tal y como puede verse a continuación.

Aún así, una vez finalizado el diseño completo del firmware de FTU2 y probado su correcto funcionamiento, se vuelve al estudio del MIG y se consigue resolver todos los problemas de integración, obteniendo resultados teóricos en pruebas de envío de datos que ofrecen mucha mayor velocidad de transmisión de los que se tienen mediante el Power PC. Más adelante se detallan los problemas encontrados con el MIG y la manera de resolverlos.

Cabe señalar que para que ambas estrategias funcionen de manera correcta, es necesaria la implementación de unas FIFOs cuya función es desacoplar diferentes dominios de reloj, ya que los datos de entrada y salida pertenecen a distintas interfaces con tasas de transmisión diferentes. En cada una de las dos estrategias descritas a continuación se especifican las señales con las que trabajan dichas FIFOs.

3.2.1. Estrategia de uso de un microprocesador Power Pc 440

Este método es el que se emplea en el primer prototipo de la plataforma FTU2, su sencilla implementación y fácil depuración hace que su desarrollo se realice en poco tiempo teniendo una funcionalidad básica del sistema de manera más o menos rápida. Lo que se hace es emplear las interfaces de propósito general de entrada/salida (GPIO) ofrecidas por el microprocesador Power PC que viene integrado en la C-FPGA. Gracias a ellos se accede de manera asíncrona a la memoria DDR2 por medio de la interfaz del controlador de memoria que implementa el microprocesador (o bien MCI del inglés *Memory Controller Interface*). A continuación se muestra un esquema general con los principales módulos y señales que entran en juego para configurar esta estrategia:

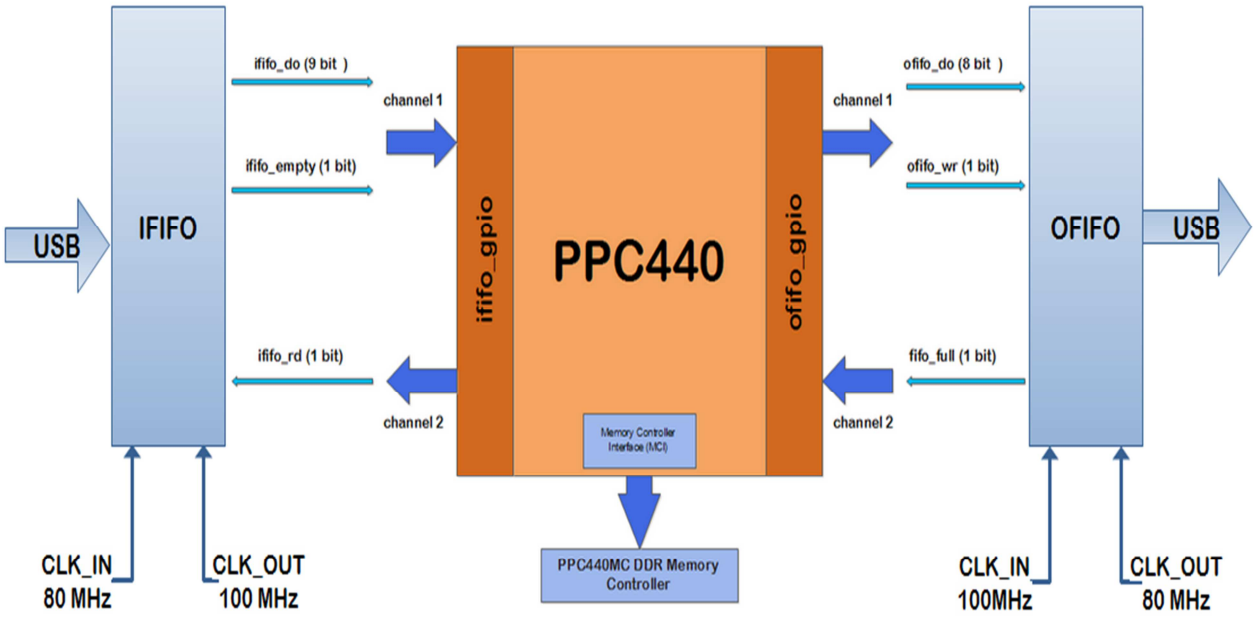


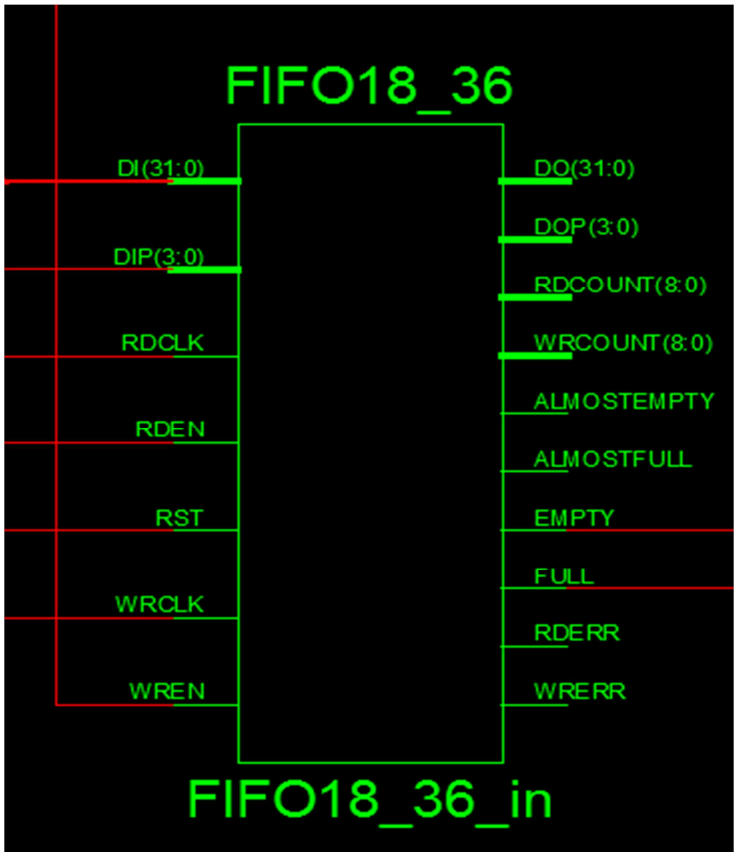
Diagrama de bloques de manejo de la memoria DDR2 mediante Power PC

Los elementos necesarios para realizar esta configuración son los siguientes:

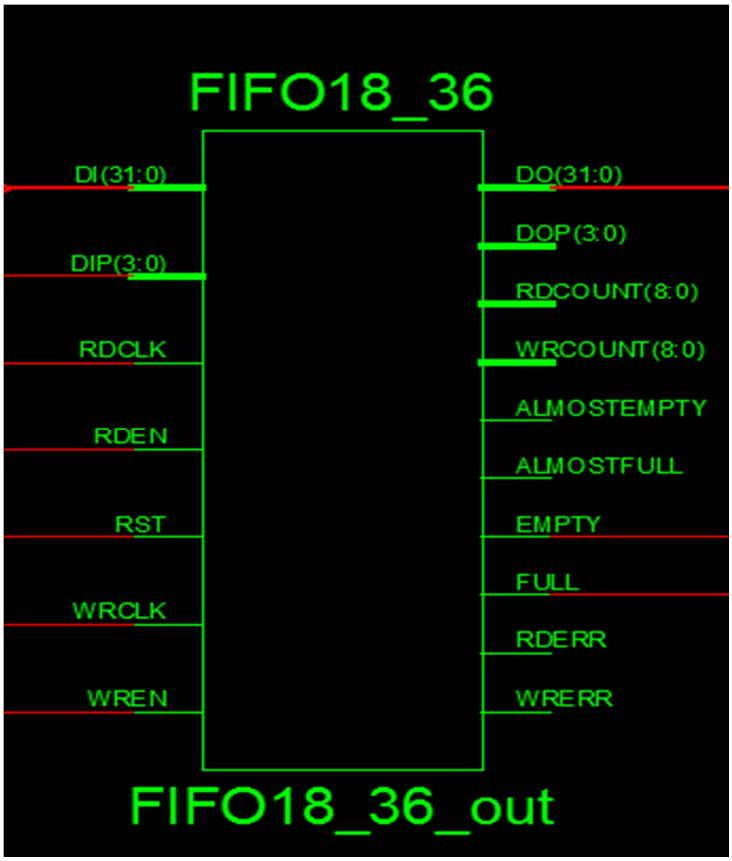
- **IFIFO, OFIFO**
    - Función

      Ambas FIFOs resuelven el problema de desacoplar los relojes de las interfaces que pretenden comunicarse, en este caso entre el USB y el microprocesador Power PC. Por un lado el USB trabaja con un reloj que ofrece un máximo de 80 MHz para tráfico de datos y por otro lado, el microprocesador emplea el reloj del sistema (reloj de la C-FPGA) de 100 MHz para comunicarse con las señales de sus puertos GPIOs, aunque internamente el Power PC trabaja a una frecuencia de 400 MHz a la hora de llevar a cabo sus micro instrucciones internas.

    - Esquemáticos



Módulo FIFO de entrada



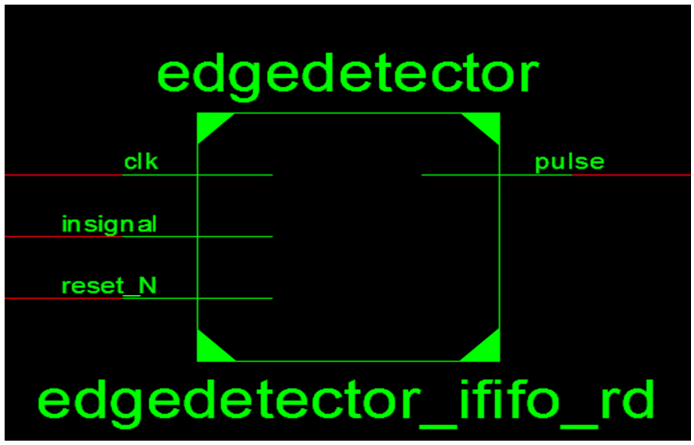
Módulo FIFO de salida

- A tener en cuenta

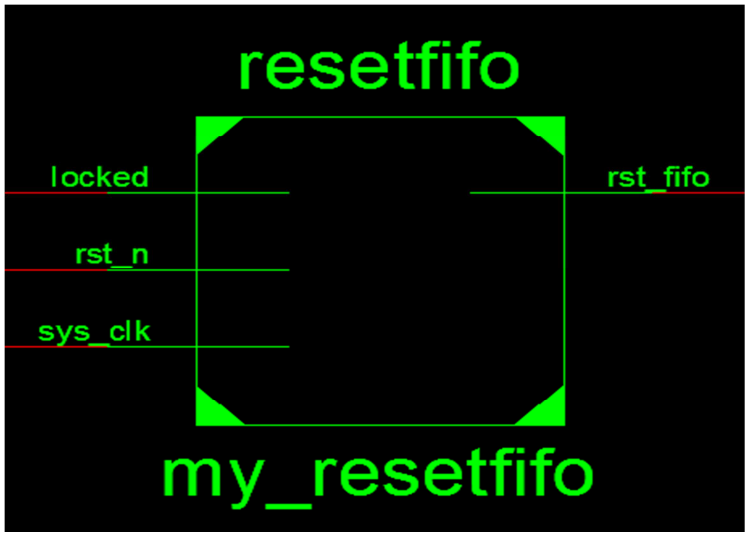
Uno de los primeros problemas encontrados es la pérdida del primer dato enviado, error que se debe a una mala gestión del reset de las FIFOs, ese problema junto a la diferencia de relojes en cuanto a escritura y lectura dentro de la mismas, hace que justo el primer dato se vea afectado o bien perdiéndose o bien duplicándose. Este hecho hace que tengamos que preparar dos módulos para controlar el funcionamiento de las FIFOs, que resultan básicas para todo el firmware.

Por un lado, se diseña un pequeño módulo de control en las señales de lectura/escritura en cada FIFO implementada en el diseño, normalmente estas señales vienen definidas como *nombre_identificativo_rd_pulse* o bien *nombre_identificativo_wr_pulse* y son la salida de un módulo llamado *edgedetector* que simplemente se encarga de detectar un cambio en el flanco de subida de la verdadera señal de lectura/escritura que se recibe, haciendo que dicha captura sea más estable justo en el ciclo de reloj en el cual debe de realizarse la operación.

Por otro lado, las primitivas FIFO de Xilinx necesitan una secuencia de inicialización algo compleja; para que sea correcta es imprescindible que reciban un determinado número de flancos activos de reloj por cada una de sus dos entradas de reloj (son FIFOs asimétricas) y esto debe ocurrir mientras el reset de la FIFO está activo. Este motivo provoca que si alguno de sus relojes depende del otro, ya sea a través de una DCM (del inglés *Digital Clock Manager*) o un PLL, no se puede emplear para la FIFO el mismo reset global que usa la DCM o PLL, ya que puede darse el caso de que mientras la DCM o PLL se encuentran en reset es imposible que la FIFO reciba ciclos de reloj por su segunda entrada de reloj por lo que la inicialización no se realiza de forma correcta. Esto lleva a la conclusión de que las FIFOs deben de tener un reset diferente y para ello se diseña un módulo llamado *resetfifo*. Es el encargado de gestionar un reset, manteniéndolo activo un cierto número de ciclos de reloj tras la sincronización de la DCM o PLL (se da cuando activan una señal denominada *locked*).



Módulo edgedetector de acceso a lectura



Módulo resetfifo

- **GPIOs**

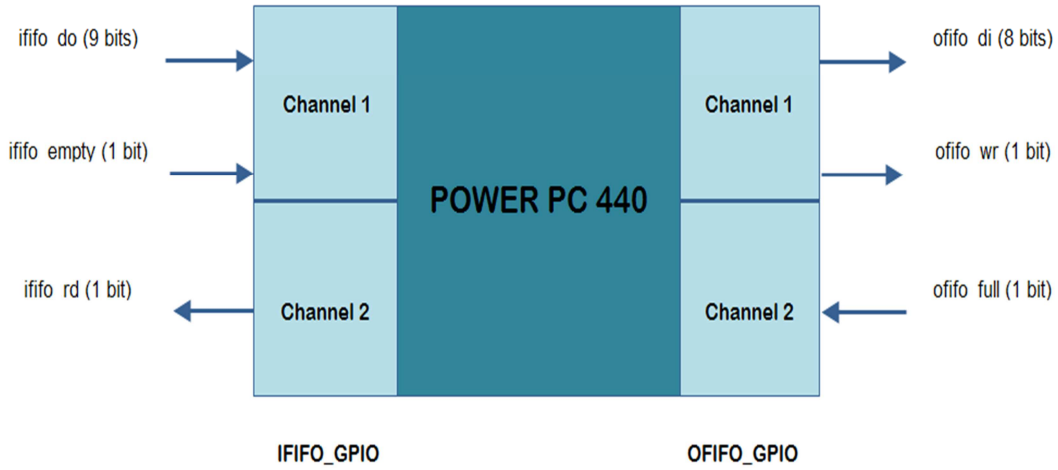
  - o Función

    Estos módulos de entrada/salida de propósito general son la puerta por la que es posible mantener comunicación con el microprocesador Power PC. Son configurables por el usuario, disponen de dos canales por cada GPIO que pueden ser seleccionados como puertos de entrada o salida, de 1 a 32 bits cada canal. Para realizar esta estrategia, es necesaria la siguiente configuración:

    1 GPIO que se comunica con IFIFO
    1 canal de entrada de 10 bits (channel1): 8 bits datos, 1 bit control, 1 bit bandera
    1 canal de salida de 1 bit (channel2): 1 bit control

    1 GPIO que se comunica con OFIFO
    1 canal de salida de 9 bits (channel1): 8 bits datos, 1 bit control
    1 canal de entrada de 1 bit (channel2): 1 bit bandera

  - o Esquemático



Configuración de los GPIO para la comunicación USB-PPC-DDR2

- A tener en cuenta

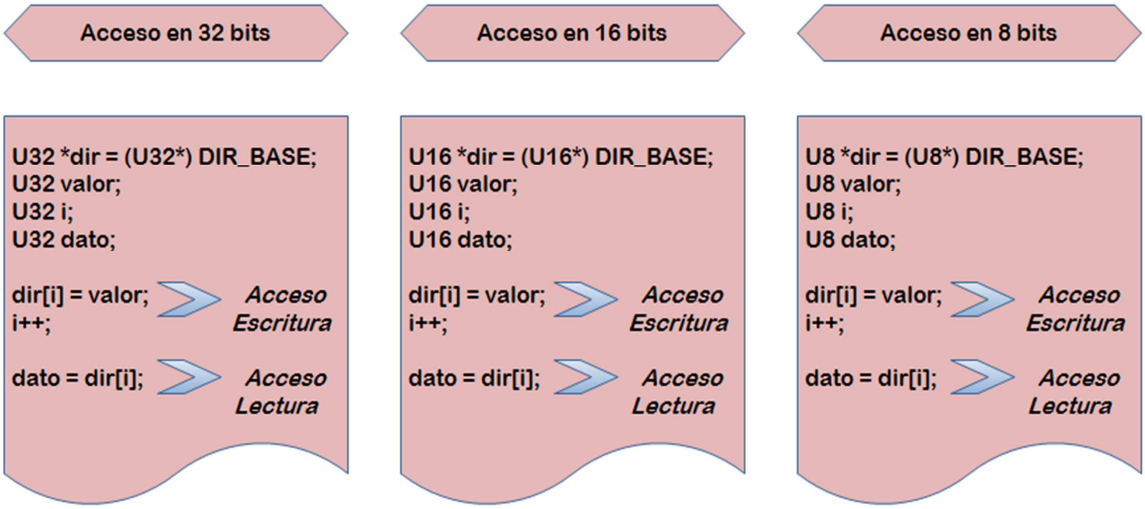
  Cada canal dispone de un máximo de 32 bits, capacidad que no se está aprovechando pero pueden emplearse en cualquier momento para añadir nuevas funcionalidades al sistema.

- **Memory Controller Interface (MCI)**

  - Función

    Se trata del módulo encargado del manejo propiamente dicho de la memoria DDR2. Trabaja bajo las órdenes del microprocesador y es transparente al diseñador; es decir, como diseñador conectamos nuestros datos a la señal correspondiente en el Power PC y mandamos una petición de lectura o escritura según se explica a continuación, accediendo a su mapa de memoria, y es el MCI el encargado de comunicar todas las señales físicas de la memoria DDR2 con las lógicas que describimos en el software embebido mediante la herramienta SDK.

La ventaja de este sistema es que el control de la memoria se hace de forma transparente al diseñador, al contrario que en el diseño del MIG donde hay que realizar módulos para el manejo de las posiciones de memoria, control de accesos, control de ráfagas, etc. La memoria DDR2 está implementada en el mapa de memoria del microprocesador y por ello resulta sencillo manejar el acceso a su contenido ya que puede realizarse mediante un software embebido dentro del microprocesador y que se diseña con la herramienta SDK de Xilinx. Dicha herramienta permite una rápida implementación y compilación de nuevos comandos que requiera el sistema, además de una cómoda depuración para poder observar el comportamiento del código y solucionar posibles errores. Por tanto, el acceso a la memoria DDR2 es tan sencillo como acceder a una posición del mapa de memoria del microprocesador, ya sea en modo escritura o lectura, también nos permite realizar este acceso con tres tipos de ancho de datos.



 # DEFINE DIR_BASE XPAR_DDR2_SDRAM_DIMM0_MEM_BASEADDR

Interfaz de usuario a la memoria DDR2 mediante Power PC

La principal desventaja de esta estrategia es que la velocidad de las inyecciones se ve afectada por el número de accesos a memoria que realiza el microprocesador para cada operación de lectura/escritura, lo que acarrea en conjunto una pérdida de velocidad en el sistema completo aunque permite tener una solución válida para poder construir todo el firmware y tener una primera aproximación de la plataforma FTU2 que funcione de manera correcta. En el capítulo cuarto se podrán comparar los resultados obtenidos sobre las pruebas de validación y así observar si el hecho de emplear el Power PC degrada el parámetro de velocidad de inyección tanto como para no suponer una mejora con respecto a FT-UNSHADES.

3.2.2. Estrategia de manejo directo de la interfaz

Este método se puede realizar gracias a un módulo desarrollado por Xilinx denominado MIG[3] el cual hace el papel de intermediario entre el usuario y la memoria DDR2, ofrece una serie de señales con las que acceder a las posiciones de memoria disponibles así como la posibilidad de configurar manualmente todos y cada uno de los parámetros de los que dispone cada módulo de la memoria. A continuación se muestra de manera general las principales señales y módulos que implementa el MIG:

[3] Para una mayor información, consultar el capítulo 9 de la sección IV del siguiente documento (Xilinx[6]).

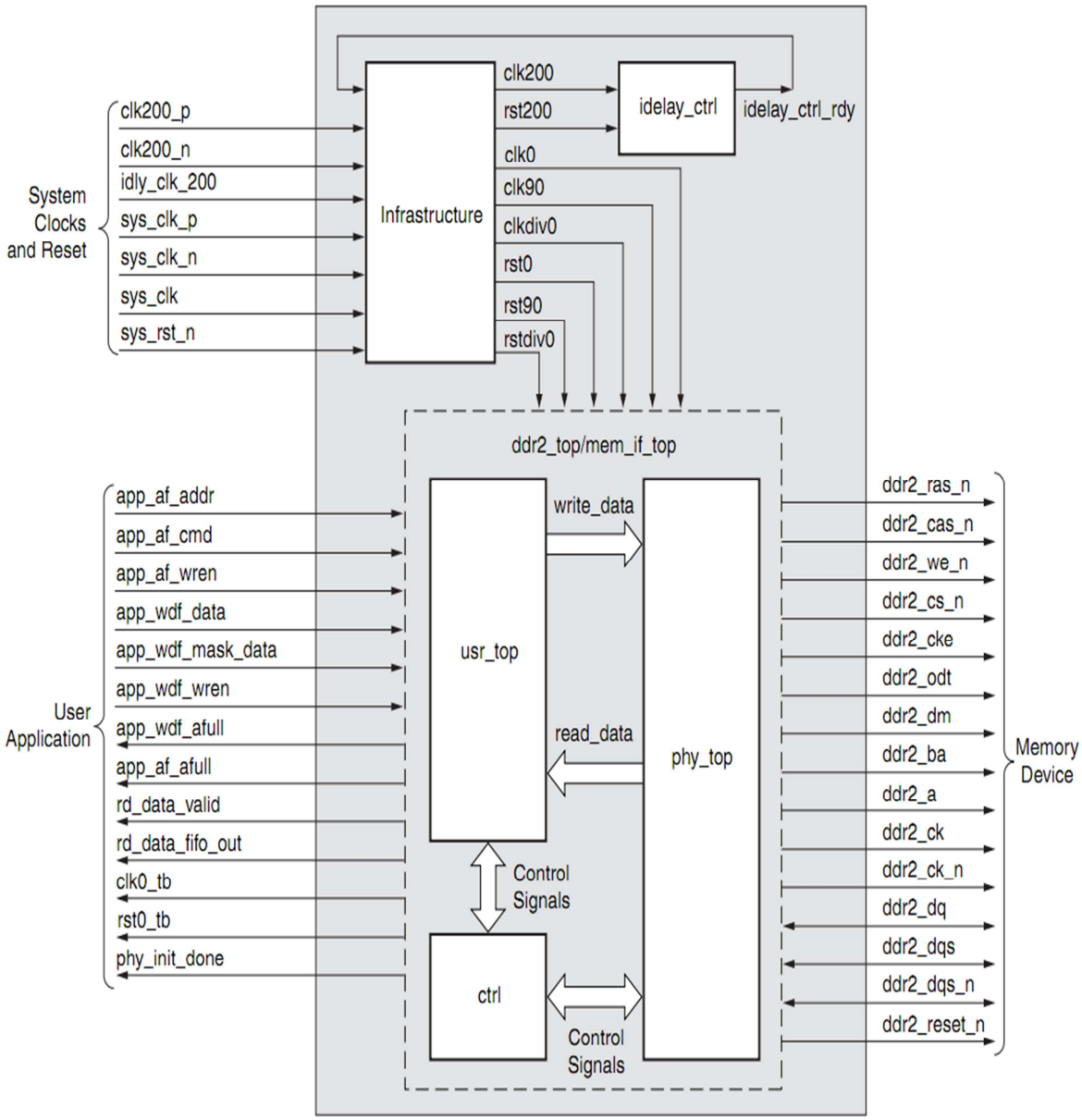


Diagrama de bloques del MIG
(imagen obtenida del documento ug086 de Xilinx)

Se emplea una configuración en la que el usuario debe proveer al MIG de todas las señales de reloj necesarias para el correcto funcionamiento, además de un reset. Mediante una primitiva de Xilinx se implementa un PLL con el cual se obtienen todos los relojes a partir del reloj del sistema de 100 MHz. Como puede observarse, todas las señales bajo la denominación de *User Application* son aquellas que el usuario puede manejar para la lectura/escritura de datos en la memoria DDR2, así como controlar el flujo de información y poder comprobar la correcta calibración del MIG. En cambio, las señales denominadas *Memory Device* son transparentes al usuario ya que es el MIG el encargado de su gestión directa con la memoria DDR2.

El módulo *Infrastructure* genera principalmente todos los relojes y reset necesarios para el sistema. El reloj **clk200** de 200 MHz necesario para el módulo de retrasos, el reloj principal **clk0** de 100 MHz (copia con 0º de desviación de fase del reloj de entrada del sistema), **clk90** (ídem que el anterior pero con una desviación de fase de 90º),

clkdiv0 (mitad de la frecuencia del reloj de entrada del sistema y con una desviación de fase de 0º con respecto a clk0). También se encarga de convertir cualquier entrada diferencial de reloj en una salida single-ended, además de generar todas las señales de reset requeridas por el diseño.

Uno de los módulos más importantes para el funcionamiento del MIG es el denominado *idelay_ctrl*, el cual instancia una primitiva IDELAYCTRL de Virtex-5 de Xilinx que se emplea para calibrar de forma continua los retrasos de los diferentes elementos que componen el MIG (cabe señalar que el MIG trabaja con la DDR2 a una frecuencia de 300 MHz y son críticos todos y cada uno de los tiempos que intervienen en el sistema), así como reducir los efectos que puedan ocasionar las variaciones de voltajes y temperatura en los procesos activos. Para más información sobre esta primitiva consultar en el capítulo 14 de (Xilinx[7]).

En el módulo *ctrl* se generan todas las señales de control requeridas tanto para la interfaz hacia la memoria DDR2 como para la interfaz de usuario. Una de sus funciones principales es decodificar el comando de usuario y así notificar a la memoria que acción se va a realizar, mediante la señal *app_af_cmd* se sabe si el usuario pretende escribir o leer en la memoria, siguiendo la siguiente codificación:

Si app_af_cmd = “000” -> comando de escritura
Si app_af_cmd = “001” -> comando de lectura

El módulo denominado *phy_top* implementa el nivel más alto de la interfaz física del diseño. Aquí se incluyen todos los bloques de entrada/salida (IOBs) y otras primitivas como IDDR, ODDR, IODELAY. Uno de los aspectos más importantes es que este módulo instancia tanto al elemento encargado de inicializar el MIG, como al elemento encargado de calibrarlo; dos operaciones básicas sin las que el MIG no puede funcionar si no se resuelven de forma correcta.

Por último, el módulo *usr_top* es el que ofrece al usuario la interfaz a través de la cual puede comunicarse con el MIG. Se encarga de recibir y almacenar los datos del usuario, comandos y direcciones, en unas FIFOs que tiene implementadas. Mediante las señales generadas en el módulo *ctrl* estas FIFOSs intercambian los datos con la memoria DDR2 a través de la capa física.

Una vez explicado el diseño y principales funciones del MIG, queda describir la manera en la que se tratan los datos desde el lado del usuario. Para el usuario, el MIG es una “caja negra” con la cual se puede acceder al contenido de la memoria DDR2, por tanto, la estrategia a seguir es diseñar ciertos módulos capaces de organizar los datos de manera que concuerden tal y como lo espera el MIG, también manejar el correcto direccionamiento, el uso de ráfagas de datos, y por último implementar una máquina de estados general que supervise las operaciones de lectura/escritura según corresponda.

Hay que señalar que la siguiente estrategia que se describe no ha sido implementada aún dentro de la plataforma firmware al completo, tan sólo se ha comprobado su correcto funcionamiento y parte de su integración dentro del diseño, con el objeto de realizar mediciones en la transmisión de los datos y poder compararlos a los que se obtienen con el uso del Power PC, dicha información de resultados se detalla en el

siguiente capítulo. Se muestra a continuación un esquema general de cómo implementar una lógica para manejar la DDR2 gracias al MIG.

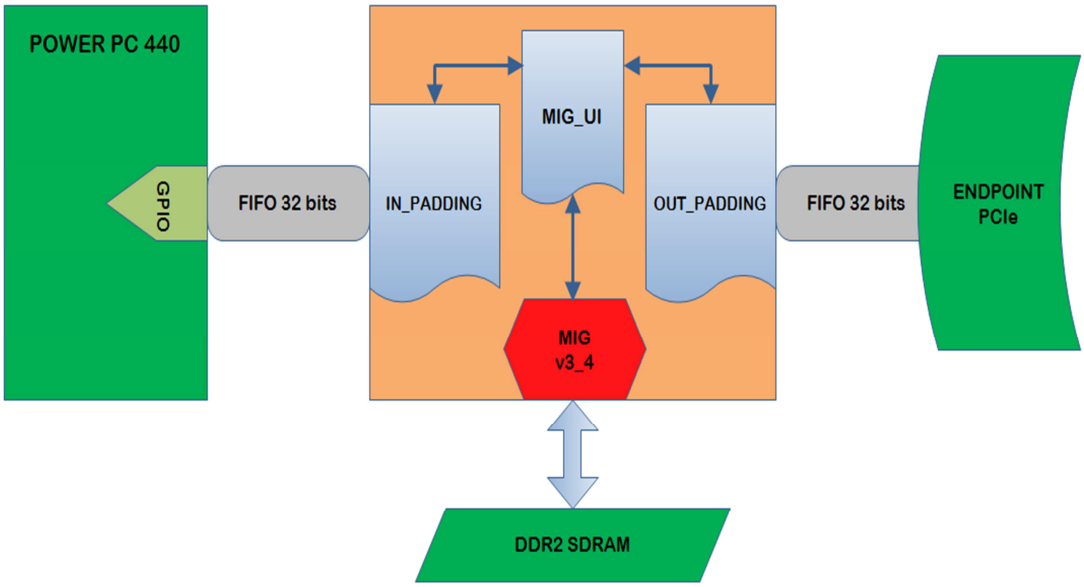


Diagrama de bloques de manejo de la memoria DDR2 mediante interfaz directa

Como puede observarse, lo que se hace es conectar mediante un GPIO del Power PC una FIFO de 32 bits de ancho de datos a un módulo denominado IN_PADDING, esto se hace para aprovechar el código que ya se ha desarrollado desde el USB hasta el Power PC y así tenemos la seguridad de que los datos llegan hasta este punto sin ningún error. De la misma manera se conecta otra FIFO de 32 bits de ancho de datos al endpoint del PCI Express y a un módulo denominado OUT_PADDING, se reutiliza así el código de conexión al endpoint que se describe en el siguiente punto. Por tanto se integran tres nuevos módulos, que se detallan a continuación, entre el Power PC (que ya no tiene acceso directo a la memoria DDR2) y el bus PCIe.

- **IN_PADDING**
    - Función

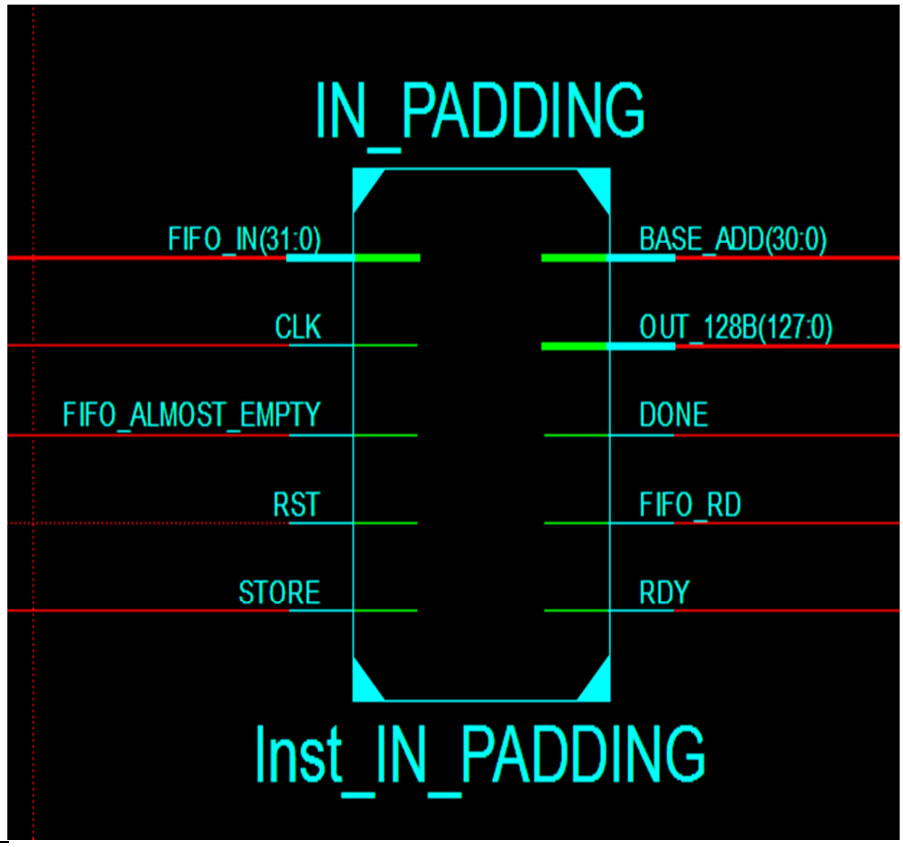
      Se trata de un módulo que prepara los datos recibidos a través de la FIFO para entregarlos de manera correcta a la primitiva que implementa el MIG. Recibe los datos en flujos de 32 bits de la siguiente manera:

      1) la dirección a la que acceder
      2) el número de datos que se envían
      3) todos los datos

      Mediante una máquina de estados interna, va preparando la dirección de memoria a la que se accede y se la envía al módulo MIG_UI. Luego prepara las ráfagas de datos (está configurado para trabajar con 4 ráfagas de 128 bits cada una), va almacenando los datos que recibe en una FIFO interna, contando hasta formar una palabra de 128 bits, cuando recibe habilitación por la señal denominada *store* entonces envía una ráfaga

hacia la memoria DDR2, cuando el envío se ha realizado correctamente activa la señal *done* para indicar que ya puede enviar la siguiente ráfaga.

- Esquemático



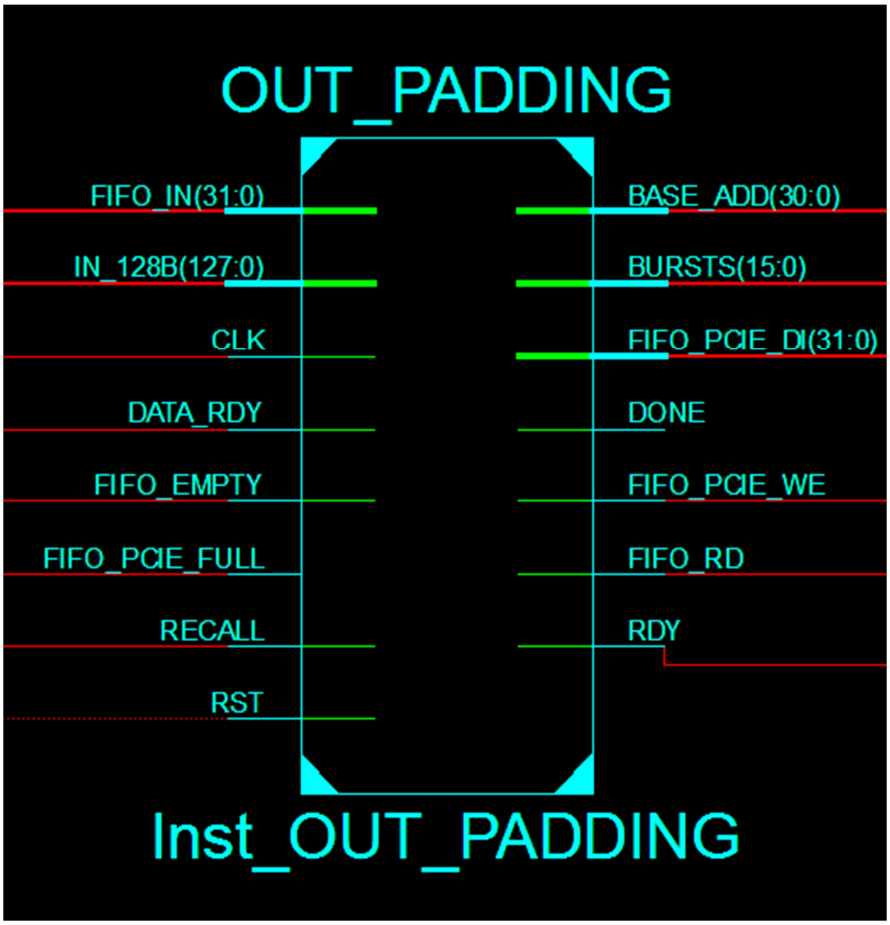
Módulo IN_PADDING

- **OUT_PADDING**

  - Función

    Este módulo funciona de manera contraria al anterior, maneja la dirección de la que leer y recibe las ráfagas de datos en flujos de 128 bits que va partiendo en trozos de 32 bits para poder enviarlo al bus PCIe a través de una FIFO ubicada entre ambos módulos. El uso de las señales *done* y *recall* son semejantes a las anteriormente mencionadas *done* y *store* respectivamente.

  - Esquemático



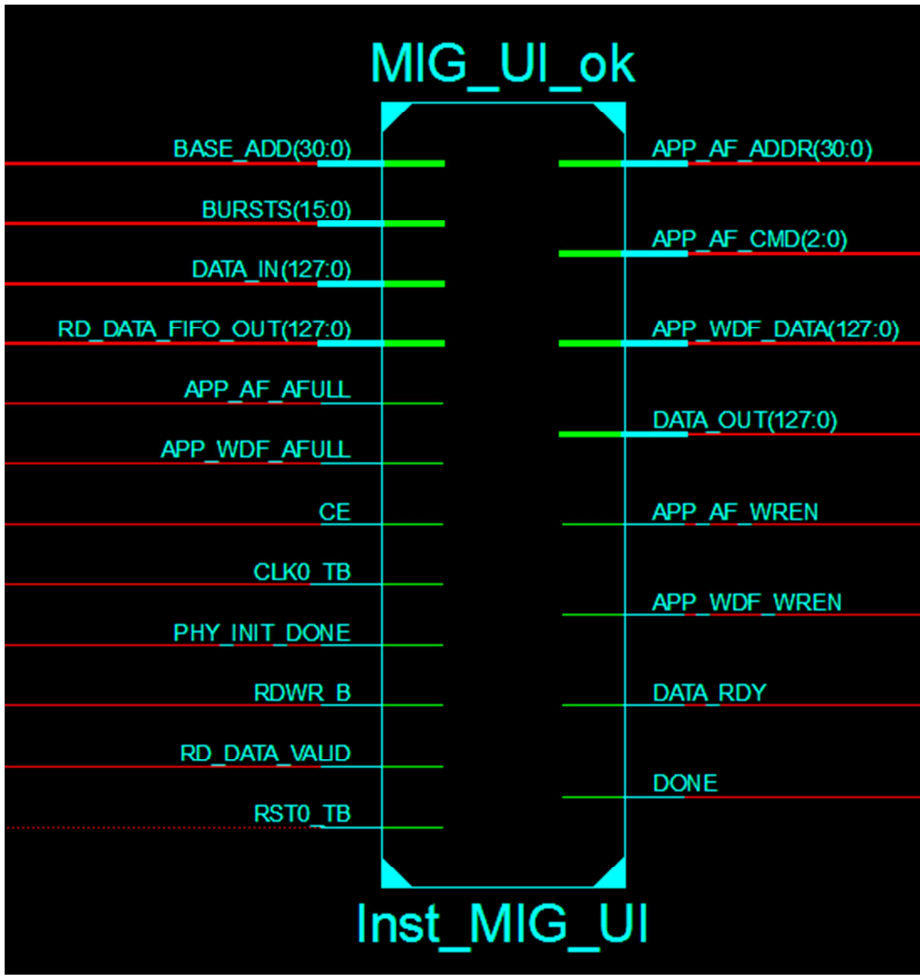
Módulo OUT_PADDING

- **MIG_UI**

  - Función

    Integra la máquina de estados principal para las operaciones de lectura y escritura sobre la memoria DDR2. Está comunicado directamente con la primitiva del MIG y es el encargado de recibir los datos y la dirección por parte del módulo IN_PADDING y enviar los datos leídos al módulo OUT_PADDING. También gestiona la escritura y lectura mediante ráfagas, cuando envía el número de ráfagas indicado en la señal *BURSTS* activa la señal *DONE* para indicar que la ráfaga es completa.

  - Esquemático



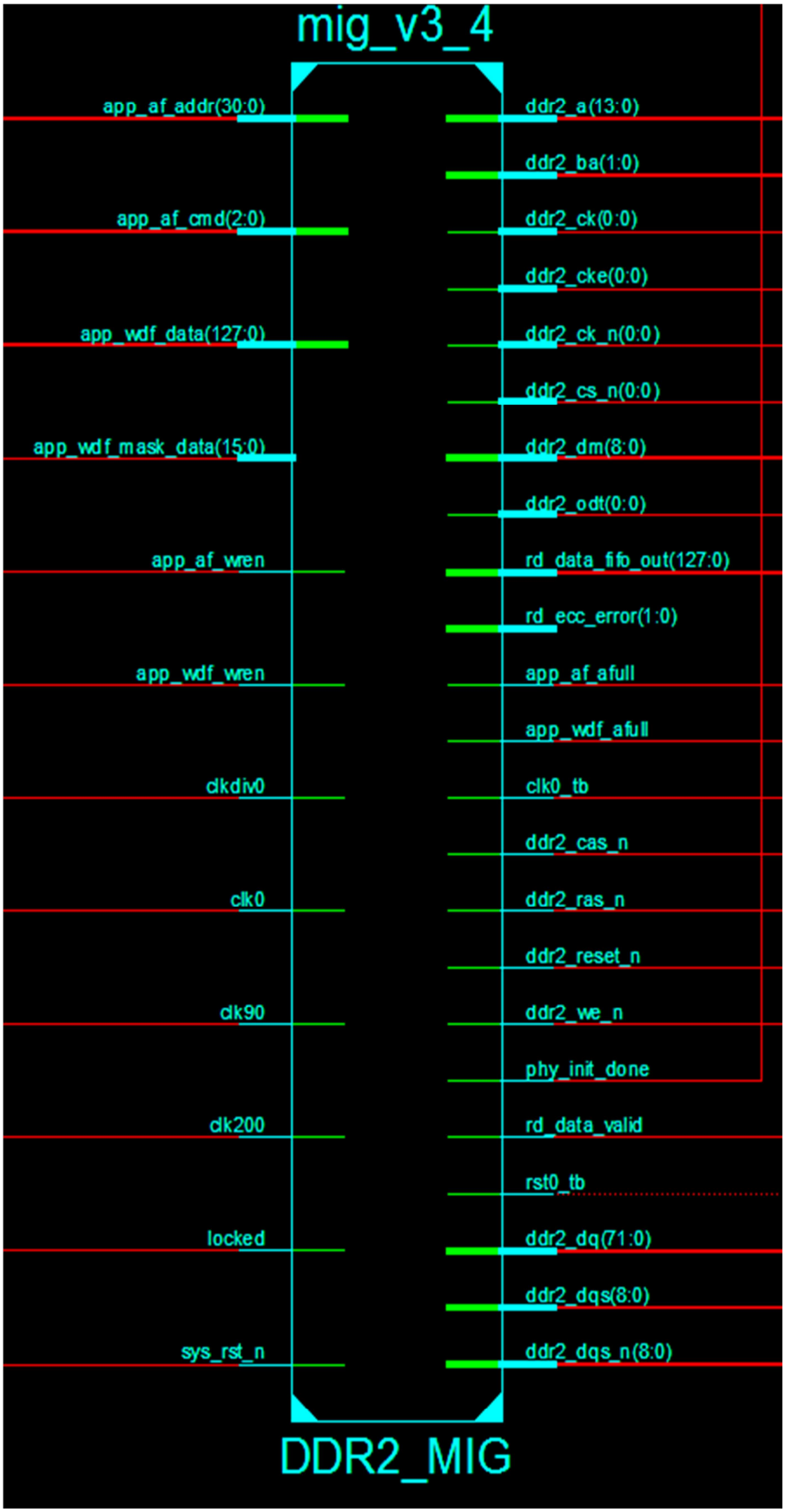
Módulo MIG_UI

- **MIG v3_4**

  - Función

    Este módulo es la implementación de la primitiva dada por Xilinx para comunicarse el usuario con la memoria DDR2, el término v3_4 indica la versión empleada del IP Core. Se encarga desde el proceso fundamental de calibración de la memoria, inicialización, escritura en los diferentes bancos que se compone la memoria y lectura de los mismos, hasta gestiona el auto refresco de la propia DDR2 entre otras funciones. Básicamente se compone de los módulos vistos en el diagrama de la página 8 y descritos posteriormente.

  - Esquemático



Módulo del MIG v3_4

- A tener en cuenta

  Cuando se comentaba al final del apartado 3.2 que no se ha implementado la estrategia del MIG en el primer prototipo de FTU2 debido a ciertos errores que aparecieron en la etapa de diseño, dichos errores se corresponden justamente con esta primitiva.
  El IP Core dado por Xilinx trae varias erratas, entre ellas el modelo de memoria empleado, donde Xilinx trabaja con una memoria del fabricante Micron modelo MT9HTF12872PY – 1GB que prácticamente es equivalente a la DDR2 de Wintec que incorpora la tarjeta ML510, salvo un valor del tiempo de activación y precarga de los retrasos que para Micron tiene un valor de 40 ns y para Wintec de 45 ns, si no se cambia ese detalle, el MIG no calibra y no se inicializa.
  Toda la primitiva completa, compuesta de 19 archivos en formato Verilog, hizo que fuera bastante complejo entender la manera de cómo el usuario puede comunicarse con el MIG enviando sus propias tramas de datos. Pasaron varios meses hasta conseguir enviar y recibir nuestras propias tramas, leyendo los datos prefijados de una memoria RAM y aún así a veces aparecían datos erróneos por lo que no se llegaba a tener asegurada una comunicación fiable. Este hecho hizo que se desarrollara la estrategia del uso del Power PC y al final se volvió a partir de un diseño básico donde la primitiva del MIG calibraba y se inicializaba de forma correcta y se desarrollaron los módulos que acaban de describirse con los cuales sí se ha conseguido establecer una comunicación totalmente fiable.

3.3. Controlador de interfaz PCI Express

Se presenta en este apartado la segunda solución que se aporta en este proyecto, a una de las interfaces de datos de alta velocidad necesarias para el desarrollo del firmware de la plataforma FTU2. En esta ocasión se trata de establecer una comunicación fundamental entre las tarjetas involucradas en el sistema; es decir, ambas tarjetas hijas se comunican con la tarjeta madre a través de unos zócalos que implementan un bus de comunicaciones PCI Express 2.0 (PCIe en adelante).

PCIe es un bus serie de alta velocidad y fiable que viene a sustituir al antiguo bus paralelo PCI. Aunque a efectos prácticos, el bus PCIe se comporta como un conversor paralelo-serie-paralelo, ya que dispone de líneas (o *lanes* en inglés) de comunicación full dúplex que realizan una comunicación punto a punto entre dispositivos conectados entre sí. Estas líneas pueden tener diferentes configuraciones: x1, x2, x4, x8, x16 y hasta x32, nomenclatura que especifican el número de líneas del que dispone el zócalo PCIe de un dispositivo, para nuestro caso son zócalos de PCIe x16. Para el bus PCIe versión 2.0 que tenemos en la tarjeta ML510 se tiene un caudal de información por línea de 500 MB/s, esto significa que se puede alcanzar hasta 500 MB/s x16 líneas = 8 GB/s de ancho de banda de transmisión teórica. Sin embargo, una de las técnicas que hacen de este bus que sea tan rápido es el sistema de codificación/decodificación 8b/10b que transforma símbolos de 8 bits en palabras de 10 bits (ayuda a mantener un equilibrio en las tensiones de las líneas y recuperación del reloj), por lo que se incluye una sobrecarga de información que hace perder aproximadamente un 20% de la capacidad total del ancho de banda disponible.

Para esta implementación tenemos una limitación hardware a causa de la FPGA empleada, la Virtex V 130T, ya que de los dos zócalos que tiene disponibles, uno de ellos alcanza una velocidad de x4 pero el otro sólo a x1, lo que implica que para que exista simetría de velocidades entre ambas tarjetas hijas, el sistema debe de usar velocidad a x1 en ambos zócalos. Eso significa que se pierde una gran capacidad de ancho de banda de transmisión del PCIe y para solventar esta limitación lo que se hace es eliminar la mayor parte de capas del protocolo PCIe y simplemente se desarrollan las capas física y de enlace. Para ello se usan los transceptores (o transceivers en inglés) que se conectan a los puertos ROCKET IO localizados en los endpoints que se implementan como primitivas de Xilinx para controlar la interfaz del PCIe. Con esta técnica se llega a alcanzar un ancho de banda teórico de hasta 2.5 Gbps, al cual si le quitamos el 20% que se pierde por la codificación 8b/10b, da como resultado de unos 2 Gbps disponibles para el usuario.

En los siguientes gráficos se muestra la arquitectura general del diseño empleado para manejar la interfaz del PCIe:

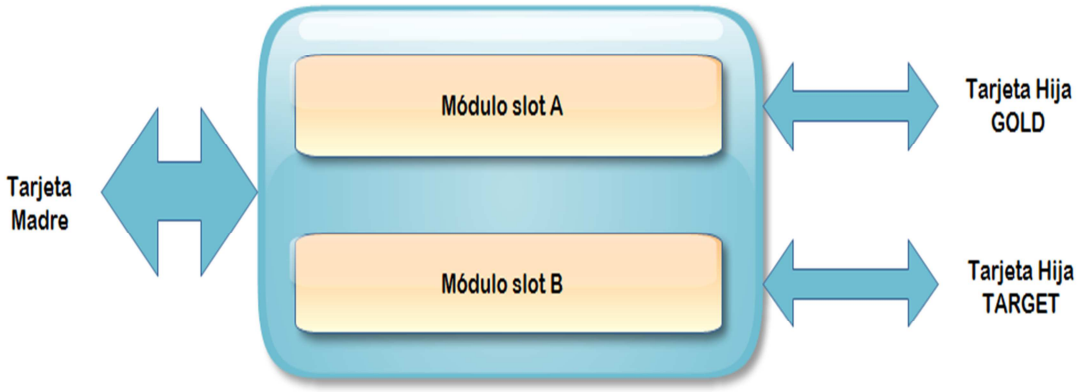
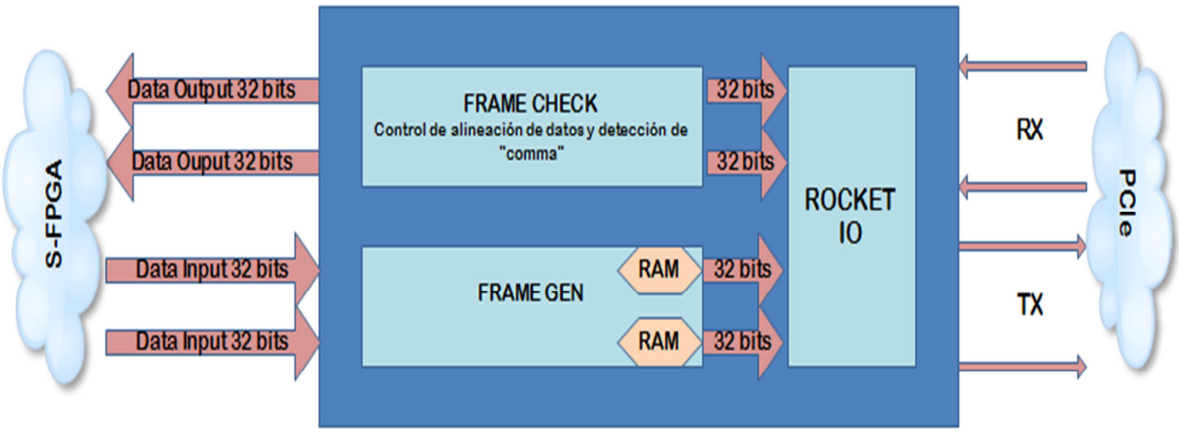


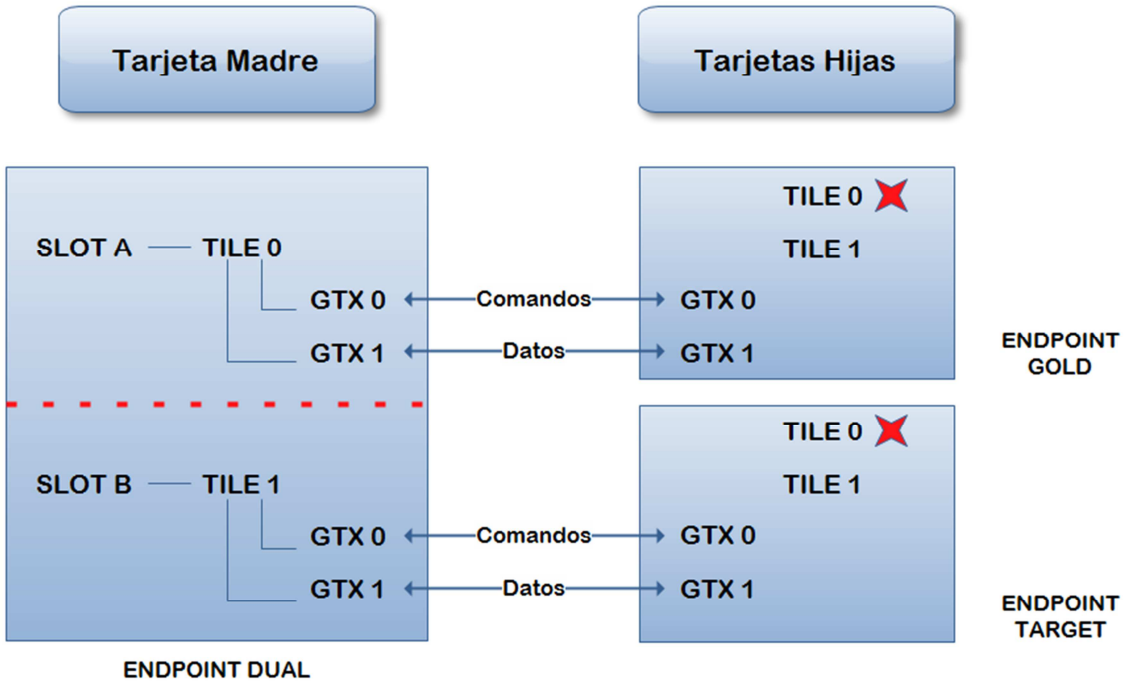
Diagrama completo del PCIe

Cada uno de los módulos que se indican están compuestos por una serie de instancias codificadas en lenguaje VHDL que realizan varias funciones para que la transmisión y recepción de datos funcione correctamente. Al haber eliminado capas superiores del protocolo, se ha tenido que realizar chequeos para controlar el flujo de la información, detección de caracteres especiales (conocidos como caracteres “K”), alineación correcta de los datos e inicialización y mantenimiento del enlace (el bus PCIe siempre debe de estar enviando información). El siguiente diagrama muestra el contenido de cada uno de los módulos implementados ya sea para el zócalo (o slot) A o B.



Interfaz ofrecida por una única tarjeta hija

La conexión lógica entre los endpoints implementados en cada una de las tarjetas se realiza punto a punto, tal y como se especifica en el protocolo de PCIe 2.0, y para ello se emplean los puertos ROCKET IO localizados en los transceptores (GTX) que ofrece cada una de las primitivas de PCIe empleadas. El siguiente gráfico muestra cómo se ha decidido utilizar uno de los enlaces para la transmisión de los comandos y el otro enlace para la transmisión de los datos.

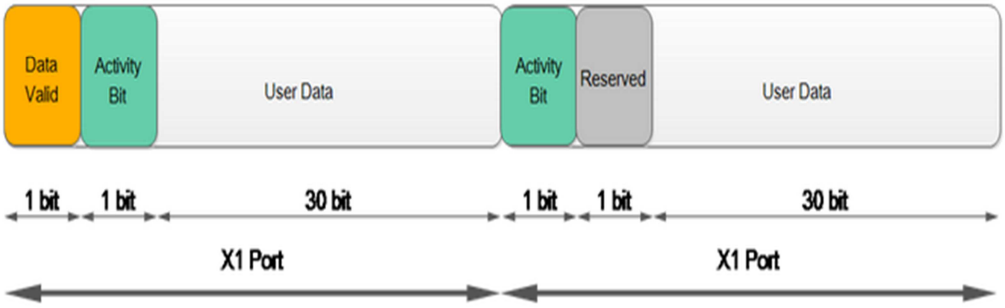


Conexiones lógicas entre endpoints del PCIe

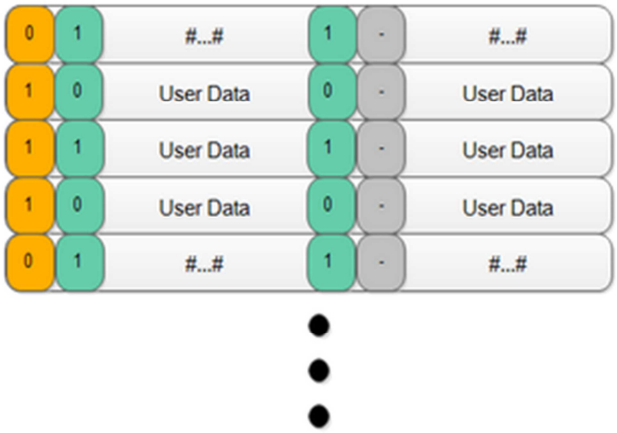
Para finalizar, se muestra el formato del canal de comunicaciones que se diseña para el tráfico de datos a través de esta interfaz junto con un ejemplo de transmisión de datos. Cabe señalar que se emplea un bit denominado "*Activity Bit*", cuya alternancia implica que existen nuevos datos a enviar o bien hacia la tarjeta madre o bien hacia las tarjetas hijas. También es importante el concepto de los caracteres "K" antes mencionados, se trata de unos caracteres que tienen un significado especial dentro del protocolo PCIe, el

más importante es el conocido como "*comma*" ya que sirve para mantener el alineamiento de las tramas de datos. Son símbolos de control disponibles en la codificación de línea 8B/10B empleada en la capa física del protocolo PCIe, algunos[4] se muestran a continuación:

Si el Byte más significativo = K.28.0 la palabra de 32 bits recibida no es relevante.
El símbolo K.28.5 es la "comma" de alineación.
El símbolo K.28.7 no debe ser utilizado.



Canal de comunicaciones



Ejemplo de transmisión de datos

[4] Para una mayor información consultar (Wiki[7]).

CAPÍTULO 4

Pruebas y resultados

Este capítulo trata de resaltar la correcta funcionalidad de las soluciones propuestas en el capítulo anterior, así como comprobar que cumplen con los objetivos buscados. Como interfaces de datos de alta velocidad, es necesario responder a dos cuestiones fundamentales: ¿es fiable la transmisión de los datos? y ¿se obtiene una velocidad de transmisión aceptable a las necesidades?. A lo largo del presente capítulo se detallan las pruebas realizadas con el objetivo de responder a ambas preguntas.

4.1. Introducción

La manera de probar la correcta funcionalidad de las diferentes interfaces, puede asimilarse de igual manera a las tres, teniendo en cuenta la siguiente estrategia:

- Envío de datos conocidos por el usuario o bien desde una RAM instanciada en la FPGA de la tarjeta madre, o bien introducidos por una consola desde un PC que llegan a la tarjeta madre a través del puerto USB.
- Recibir los datos y visualizarlos por una consola gracias a la conexión serie UART que dispone la tarjeta ML510.
- Comparar los resultados.

Cronológicamente hablando, el primer test de depuración se le hizo a la estrategia que emplea el microprocesador Power PC ya que no necesita del PCIe para poder comprobar su funcionalidad. A continuación se consigue probar el funcionamiento del bus PCIe y por último, se depura la estrategia del MIG una vez que se ha comprobado la fiabilidad del PCIe ya que como se pudo observar en el capítulo anterior referente al MIG, sí se necesita del bus PCIe para tener un sistema completo con el que depurar el funcionamiento de dicho módulo.

4.2. Test de funcionalidad de la interfaz con la memoria DDR2: estrategia de uso del microprocesador Power PC

Antes de probar esta estrategia, se asegura mediante varios ejemplos diseñados en VHDL, el correcto envío de datos a través del puerto USB y transferencia a través de las FIFOs que sirven para desacoplar relojes. Gracias a estos diseños se van colocando las bases de una correcta comunicación antes de implementar los diferentes GPIOs y conectar el microprocesador Power PC.

La máxima prueba de que el diseño funciona correctamente es el hecho de que actualmente se encuentra integrado en el primer prototipo de la plataforma FTU2, de cuya versión existen dos plataformas completamente funcionales, una en las instalaciones de la Agencia Espacial Europea donde se han empezado a realizar test de inyección en diseños propios, y otra unidad que se encuentra en el Departamento de Ingeniería Electrónica de la Universidad de Sevilla, cuya intención es seguir mejorándola con la idea de integrar la opción del MIG con el que se puede llegar a obtener mejores resultados en cuanto a velocidad de inyección.

4.2.1. Resultados de velocidad

Esta estrategia no era la diseñada originalmente como la opción preferida para manejar la memoria DDR2, ya que se sabía con antelación que los accesos que emplea el microprocesador para cada operación de lectura o escritura eran perjudiciales para

términos de velocidad ya que se incrementan los tiempos a la hora de procesar los diferentes comandos que envía el usuario por medio de la plataforma software. Aún así era de esperar obtener mejores resultados que el sistema FT-UNSHADES ya que se evita el cuello de botella de tener que acceder al PC de usuario a través del puerto USB para ir recogiendo los estímulos de las inyecciones.

El resultado obtenido es que se mejora la velocidad de inyecciones por segundo pero se mantiene en el mismo orden de magnitud que se tenía en la plataforma anterior. Es cierto que se consiguen mejorar ciertas características como la capacidad de implementar diseños más complejos, con un mayor número de puertos de entrada/salida, con una mayor capacidad de almacenamiento y una mejor interfaz de uso de cara al usuario, pero en términos de velocidad, aunque se obtienen mejores resultados, se quedan muy por debajo de las expectativas.

4.3. Test de funcionalidad de la interfaz PCIe

Es el segundo elemento que se consigue depurar una vez implementados todos los endpoints necesarios para crear las dos primeras capas del protocolo PCIe. Una vez comprobada la fiabilidad del transporte de datos desde el USB hasta la DDR2, nos sirve para certificar que el uso de los GPIOs es correcto. Y dado que el PCIe se conecta al Power PC de igual manera que lo hace el USB; es decir, mediante un GPIO y teniendo entre ambos módulos una FIFO que desacopla los diferentes dominios de reloj, podemos asegurar que de aparecer errores en la transmisión de datos, se debe a errores de implementación del propio PCIe.

Al igual que ocurre con el caso anterior, la prueba de que la interfaz funciona es el hecho de que el diseño completo se encuentra operativo en las plataformas de FTU2 que ya están siendo utilizadas.

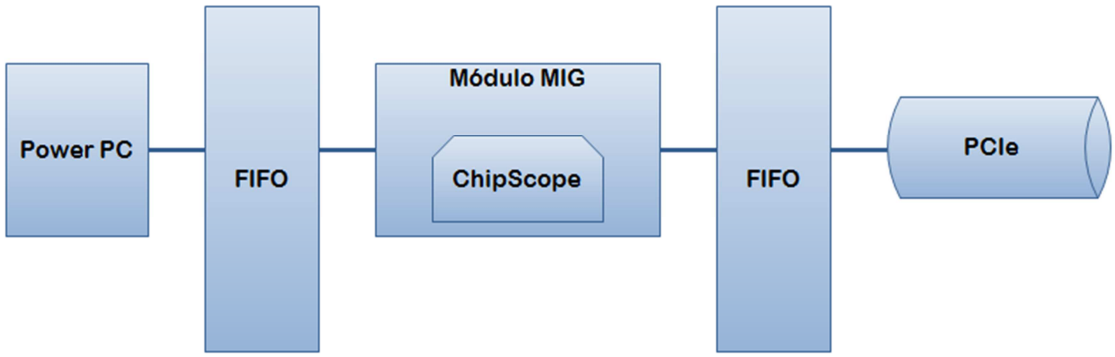
4.3.1. Resultados de velocidad

Como se pudo comprobar en el capítulo anterior, debida a la limitación hardware que se tiene por el uso de la FPGA V5130T, se desaprovecha una gran cantidad de ancho de banda de transmisión del bus PCIe. Gracias a eliminar las capas superiores del protocolo del propio bus se consigue ahorrar el envío de varias cabeceras que aportan un overhead extra y reducen aún más dicha capacidad. Con esta técnica empleada de implementar únicamente las capas física y de enlace se puede conseguir un ancho de banda máximo de 2.5 Gbps en cada sentido (transmisión y recepción), lo cual asegura que esta interfaz no es motivo de cuello de botella para el sistema ya que se trata de una transmisión suficiente para los objetivos buscados.

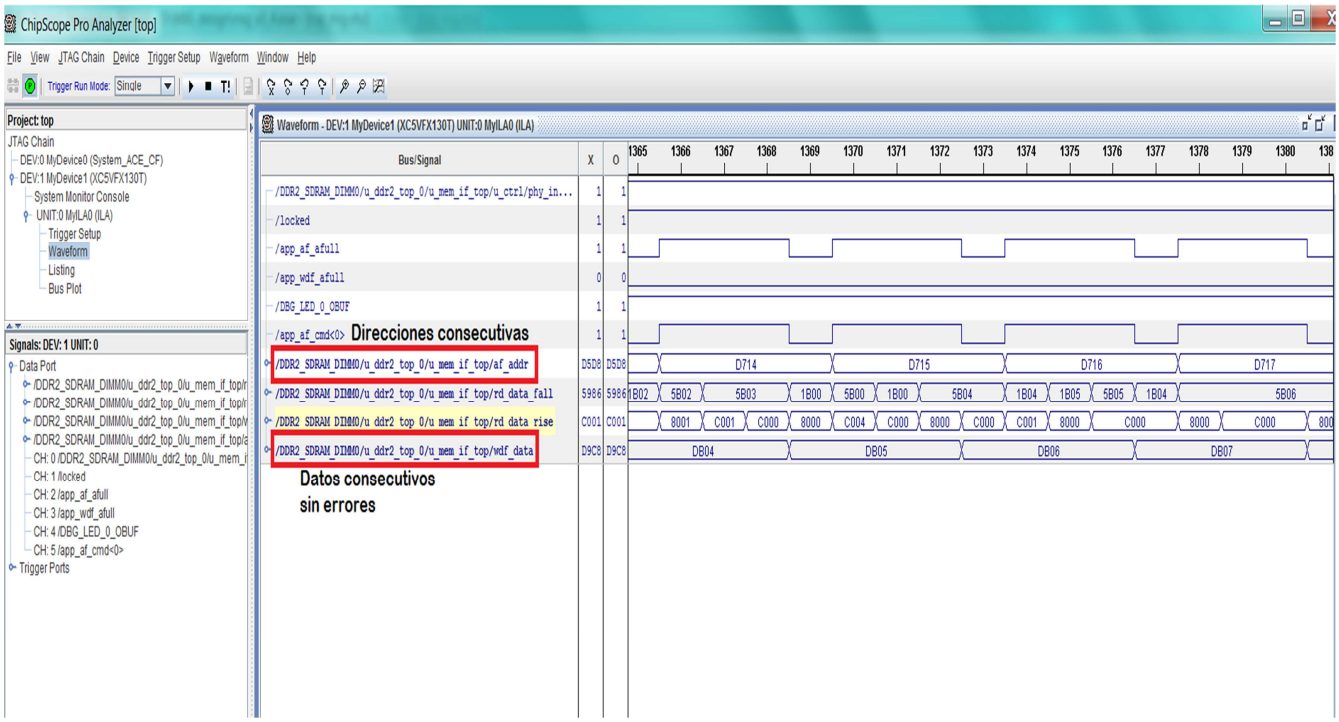
4.4. Test de funcionalidad de la interfaz con la memoria DDR2: estrategia de uso del MIG

Este es el último elemento en ser depurado, ya que se consigue poner en marcha después de tener una primera versión del firmware completo funcionando. Como ya se ha comentado en el capítulo anterior, esta estrategia no está aún implementada en la estructura firmware completa, aunque sí emplea ciertos elementos básicos como las conexiones con los GPIOs mediante FIFOs y el PCIe. La manera de comprobar que los datos se obtienen sin error fue a diferencia de los dos casos anteriores (que se ayudaban de la conexión UART para ver por consola los datos), introduciendo un módulo de ChipScope dentro del diseño con el cual poder observar la transferencia correcta de la información que se transmite.

El esquema que se emplea y el resultado obtenido se muestra a continuación:



Test de funcionalidad del Módulo MIG completo



Recepción de datos correcta

4.4.1. Resultados de velocidad

En este caso se realiza un cálculo aproximado de un caso de ejemplo, con el cual se puede hacer una idea de la capacidad de transmisión de este diseño. Es de esperar que sabiendo que el MIG trabaja interiormente con una señal de reloj a 300 MHz, y teniendo implementado el sistema de ráfagas, esta estrategia va a aportar una mayor velocidad en el acceso a la memoria DDR2 que la ofrecida por medio del Power PC.

El ejemplo empleado para realizar los cálculos es el siguiente:

- Se envían 64 datos de 128 bits cada uno (comprende tanto la escritura como la lectura de los mismos)
- Se mide mediante un elemento diseñado en VHDL, el número de ciclos que se tarda en realizar la operación completa. Resultado = 159 ciclos
- El reloj es de 3 ns/ciclo
- Se tiene un ancho de banda de transmisión de unos **17 Gbps**

Como puede comprobarse, esta estrategia ofrece, al menos a la vista de las pruebas realizadas, una capacidad muy superior a la que se tiene con el empleo del microprocesador Power PC en lo que al manejo de la memoria DDR2 se refiere. Queda ver su funcionamiento integrado dentro de la arquitectura del firmware completo, pero todo hace pensar que se trata de la mejor solución para obtener resultados de velocidad que alcancen los objetivos buscados.

CAPÍTULO 5

Conclusiones

En este capítulo final se exponen las principales conclusiones obtenidas tras la realización del presente trabajo. Las conclusiones se expresan tanto a nivel técnico en cuanto a lo que aporta a un primer prototipo de plataforma de inyección de fallos totalmente funcional, como a título personal de lo que le ha aportado al alumno el desarrollo del proyecto. Para finalizar, se comentan futuras líneas de estudio para encontrar posibles mejoras.

5.1. Conclusiones técnicas

El presente trabajo se ha desarrollado para formar parte de un proyecto que está integrado en el ámbito de los errores en circuitos electrónicos a causa de la radiación. Se han comentado las principales causas y efectos dañinos que provoca la radiación sobre dispositivos o sistemas electrónicos, ya sean en un entorno espacial, sistema de telecomunicaciones, automoción e incluso cada vez más, llegando a dispositivos que se encuentran en la superficie terrestre. Se han introducido las principales técnicas de estudio y mitigación de este tipo de errores, llegando a los sistemas de inyección de fallos.

Estos sistemas están teniendo una mayor relevancia en el mundo de la industria ya que ayudan a resolver la manera de cómo proteger los circuitos electrónicos con un menor coste que realizar tests de radiación en laboratorios especializados. Además, es justamente para un tipo de sistema de inyección de fallos, para el cual ha hecho falta el estudio de ambas interfaces de datos de alta velocidad desarrolladas en este trabajo. Primeramente se ha descrito una plataforma de inyección de fallos desarrollada por el Departamento de Ingeniería Electrónica de la Universidad de Sevilla, con la cual se han podido desarrollar los principales conceptos de un sistema emulado sobre FPGAs. Y a continuación, se ha descrito la plataforma con la cual se ha trabajado en este proyecto, que no es más que una evolución del sistema anterior, en la que se conservan los principales conceptos acerca de cómo realizar las inyecciones sin modificar el diseño del MUT, pero obviamente buscando la mejora de ciertas características como la capacidad de almacenamiento, posibilidad de tener diseños del MUT de FPGAs más modernas, menores tiempos de ejecución de un test, mejoras en la interfaz del usuario y la plataforma, etc.

El presente proyecto resuelve dos problemas que aparecen en la etapa de diseño de un sistema de inyección de fallos basado en una plataforma hardware empleando diferentes FPGAs, denominado FTU2. Cada uno de esos problemas se enmarcan dentro de la arquitectura firmware del sistema, en la cual se desarrollan todas y cada una de las operaciones funcionales de la plataforma. Como ya se ha explicado, las soluciones propuestas tratan de resolver dos de las interfaces que son necesarias controlar para el correcto funcionamiento del diseño completo, y además se busca conseguir mejorar las prestaciones de una plataforma anterior denominada FT-UNSHADES.

Por una parte, el tema del almacenamiento es clave en la mayoría de los diseños digitales de cierta complejidad, y para este trabajo es algo fundamental en dos aspectos. En primer lugar, se busca tener mayor capacidad para almacenar diferentes archivos, y en segundo lugar minimizar los tiempos de procesamiento y de acceso a la memoria con el objetivo de maximizar la velocidad de trabajo del sistema completo.

En este trabajo se han estudiado y desarrollado dos estrategias para el control de la interfaz con la memoria DDR2 SDRAM que disponemos en la arquitectura hardware del sistema, para ambas estrategias la capacidad de almacenamiento total es idéntica, teniendo hasta 1 GByte disponible para almacenar los archivos que forman parte del funcionamiento de la plataforma FTU2. Comparando este dato con los 32 MBytes de memoria externa que se disponen en la plataforma FT-UNSHADES, podemos afirmar que se consigue aumentar de forma considerable la capacidad de almacenamiento del nuevo sistema de 2ª generación.

Por otra parte, el otro objetivo que se busca mejorar con el uso de estas interfaces es el de aumentar la velocidad de trabajo de la plataforma FTU2, o lo que es lo mismo, minimizar los tiempos de ejecución de una campaña (test completo). Sobre el tema de velocidad hay que comentar ciertos aspectos; el estudio de las dos estrategias de manejo de interfaces con la memoria DDR2 han dado dos resultados muy distintos.

Primeramente hemos visto que el uso del microprocesador Power PC ha permitido desarrollar el firmware por completo teniendo a punto un primer prototipo de la plataforma FTU2, pero los resultados obtenidos no han superado las expectativas tal y como se deseaba.

En cambio, se ha visto que el desarrollo de la estrategia empleando la interfaz directa mediante el MIG ofrece, de manera teórica, una transferencia de datos muy superior a la alcanzada con el microprocesador, obteniendo tiempos de acceso a la memoria DDR2 muy cortos y mejorando con ello el parámetro de velocidad de la manera que se estaba buscando.

Por último, la idea de usar un protocolo de comunicaciones tipo PCI Express para comunicar diferentes FPGAs vemos que es un acierto, tanto por el ancho de banda que puede llegar a obtenerse, como por la fiabilidad que ofrece en el tráfico de datos y las diferentes maneras de controlar y gestionar todo el flujo de información que viaja a través del bus.

5.2. Conclusiones personales

Este trabajo me ha introducido en un mundo desconocido para la gran mayoría de estudiantes de una Ingeniería de Telecomunicaciones, y en cambio sirve para ver otra de las muchísimas aplicaciones que tienen esos dispositivos tan versátiles llamados FPGAs. El mundo de los errores de radiación, sistemas de inyección de fallos, etc, tiene una gran base teórica ya que hay mucha física detrás de cada una de las anomalías que se han descrito en este trabajo, existen muchísimos estudios de daños a nivel de semiconductor, estudios de cálculos de niveles de energía de las partículas ionizantes para los cuales un dispositivo electrónico puede ser dañado o no, técnicas de endurecimiento y protección de circuitos ante los efectos de la radiación, y una infinidad de temas que surgen de las diferentes ramas de los efectos de la radiación. Y de toda esa teoría aparecen estudios y desarrollos de técnicas para seguir estudiando y mejorar los diseños digitales ayudando a los diseñadores a tener circuitos más robustos ante ataques externos de radiación, técnicas entre las cuales se encuentra FTU2 que emplea las características de reconfiguración parcial en ciertas FPGAs del fabricante Xilinx, con el que se ha obtenido un sistema de inyección de fallos totalmente funcional.

Durante este trabajo tuve la inmensa fortuna de poder asistir y colaborar en una de las conferencias más importantes a nivel mundial referente al tema de efectos de la radiación sobre componentes y sistemas, el conocido congreso RADECS, que en septiembre de 2011 se desarrolló en la ETSI y asistieron más de 450 expertos de todo el mundo, donde pude conocer a varias personas que básicamente me hicieron ver que se trata de un problema de ingeniería que cada vez va a más ya que se están encontrando con mayores desafíos según evolucionan las tecnologías de fabricación de componentes.

Personalmente, en referencia a los dos años en los que he estado trabajando en el Departamento de Ingeniería electrónica, puedo decir que he podido participar en casi la

mayoría de los aspectos del desarrollo de la plataforma, no sólo en el estudio de las dos interfaces descritas para este proyecto, si no también he adquirido conocimientos acerca de la reconfiguración parcial, manejo de diferentes ipcores de Xilinx, tester de la interfaz de usuario, he realizado pruebas de depuración y validación de diferentes elementos del firmware (empleando otras FPGAs como una ML507 o una Spartan 3), y ante todo me quedo con cada detalle, cada "pelea" superada ante las dificultades y cada consejo que me han dado todos los participantes del proyecto.

5.3. Trabajos futuros

Está claro que las soluciones aquí propuestas han sido desarrolladas para una aplicación muy específica, ya que incluso las versiones de las herramientas del paquete ISE de Xilinx con las que se han trabajado, van cambiando la manera de implementar sus primitivas según van evolucionando sus FPGAs. Esto no quita que se ha desarrollado un conocimiento básico sobre el uso de las FIFOs para desacoplar relojes, el empleo del MIG, el desarrollo de un software embebido dentro de un microprocesador Power PC e incluso el desarrollo de la capa física para un protocolo de PCI Express 2.0.

Todo ello ha servido para tener un primer prototipo de la plataforma FTU2 donde el parámetro de velocidad de test no ha alcanzado aún los valores esperados. Por tanto, los trabajos que pueden realizarse con idea de tener una plataforma FTU2 capaz de realizar inyecciones y disponer de sus resultados de forma más rápida son los siguientes:

- **Integrar el diseño del MIG en el firmware completo**

  Ahora se dispone de un diseño capaz de recibir datos desde el USB, almacenarlo en la memoria DDR2 y enviar dicha información por el bus PCI Express hacia una tarjeta hija. Pero no se ha probado el traspaso de los diferentes comandos de usuario y de control para poder realizar una inyección y obtener los resultados.
  Está previsto desarrollar este camino en un futuro a corto plazo ya que las posibilidades de mejora son muy altas. Se sabe que el elevado número de accesos a memoria que debe realizar el Power PC es el cuello de botella que tiene actualmente la plataforma FTU2, sin embargo el empleo del MIG elimina todo este tipo de retrasos y por ello ofrece una transferencia de datos teórica muy superior (unos 40 Gbps) comparada con la ofrecida por el Power PC.

- **Desarrollar el bus PCI Express**

  Actualmente el bus PCI Express tiene una limitación hardware debida a la FPGA V5 130T de la tarjeta ML510. La tarjeta dispone de dos zócalos PCIe para conectarse con ambas tarjetas hijas que están preparadas para trabajar a una velocidad full dúplex de hasta PCIe x8. Sin embargo, la FPGA 130T dispone en uno de los zócalos una capacidad de hasta a x4 líneas pero el otro sólo tiene disponible a x1. Este hecho conlleva que para mantener la simetría de velocidad entre ambas tarjetas hijas tan sólo se puede trabajar a velocidad x1, esto aporta finalmente un ancho de banda de sólo 2 Gbps por enlace disponibles para el usuario, cuando sería posible llegar con este protocolo hasta 16 Gbps por enlace.

Se observa que la pérdida de capacidad es abrumadora, aunque la única manera de solventar este problema es integrando una FPGA de mayor capacidad que la 130T y que tenga huella compatible con la tarjeta ML510, como puede ser una FPGA tipo 200T que dispone de un zócalo a x8 y otro a x4 lo que puede multiplicar la capacidad de transmisión por un factor de cuatro. Otra opción es elegir otra tarjeta madre que reúna las características de la ML510 y teniendo en cuenta que mejore las prestaciones de las conexiones habilitadas para el uso del PCI Express.

Referencias / Bibliografía

(H.Guzmán[1]) Hipólito Guzmán Miranda. Tesis Doctoral. *Aportaciones a las técnicas de emulación y protección de sistemas microelectrónicos complejos bajo efectos de la radiación.*

(M.Aguirre[2]) Miguel A. Aguirre and J. N. Tombs. *FT-UNSHADES user manual,* Oct 2004. GTE-AICIA. Proyecto FT-UNSHADES. B

(M.Aguirre[3]) Miguel A. Aguirre. Página web del Proyecto Unshades. *http://www.gte.us.es/~aguirre/Web_unshades/index*

(J. M. Mogollón[4]) J.M. Mogollón, H. Guzmán-Miranda, J. Nápoles, J. Barrientos, M.A. Aguirre. RADECS Proceedings Sept 2011. *FTUNSHADES2: A novel Platform for Early evaluation of Robustness against SEE*

(Wintec[5]) *http://www.wintecind.com/OEM/DRAM/DDR2/240P_REG_ECC.html*

(Xilinx[6]) *http://www.xilinx.com/support/documentation/ip_documentation/ug086.pdf*

(Wiki[7]) *http://en.wikipedia.org/wiki/8b/10b_encoding*