Capítulo 9

Implementación en VHDL y síntesis en FPGA

El objetivo final del proyecto es implementar una selección de los métodos de estimación espectral descritos en el equipo final de resonancia magnética, que será un equipo autónomo y transportable. La selección de los métodos a implementar se hace en base a las pruebas realizadas en *MATLAB*. Los métodos que obtienen mejores resultados son los que se implementan en el equipo final.

El equipo estará provisto de un microprocesador de la familia *Colibri*, fabricado por *Toradex*. Este microprocesador se utiliza para las tareas de control descritas en la tercera parte del proyecto, y que tiene por título *Sistema de control para un equipo de análisis RMN basado en un microporcesador de 32 bits*.

En el equipo final se incorporará una *FPGA*[1] que se usará para hacer el procesado inmediatamente posterior a la conversión A/D, descrito en el capítulo 4, y parte del análisis espectral, cuya carga computacional se repartirá entre la *FPGA* y el microprocesador.

Para programar la FPGA, se utiliza el lenguaje *VHDL*, siglas en inglés de *VHSIC*[2] *Hardware Description Language*, que consiste en un lenguaje de descripción de hardware usado ampliamente, junto con *Verilog*, para el diseño digital de circuitos integrados tales como *ASIC*[3] y FPGAs.

[1] Field Programmable Gate Array
[2] *Very High Scale Integrated Circuit.*
[3] Application-specific integrated circuit

9.1. Adquisición de señal y promediado

Para realizar las pruebas en laboratorio, se ha utilizado una placa de circuito integrado desarrollada en la empresa *AD Telecom*. Esta placa integra una *FPGA* de la familia *Virtex-II*, modelo *XC2V2000*, fabricada por *Xilinx*, así como dos conversores A/D de 12 bits y una serie de puertos de expansión; todo ello útil para la realización de las pruebas necesarias.

La adquisición de la señal se realiza mediante los conversores A/D. Éstos proporcionan a la *FPGA* las muestras de la señal que hay a la entrada, que se almacenan en la memoria *RAM*[4] interna que tiene la *FPGA*, que permite un ancho de bus de direcciones de hasta 15 *bits* para un ancho de bus de datos de 12 *bits*. Esto corresponde a una memoria interna de 128 *kbytes* (2^{15} 12 *bits*) y permite almacenar $2^{15} = 32768$ muestras, que a una frecuencia de muestreo de 1 *MHz* corresponden a 32 *ms* de señal.

El promediado de señales consecutivas puede realizarse sin necesidad de consumir más memoria. A medida que se adquieren señales, se suman a la que ya se tiene en memoria, con lo que al finalizar la recepcion se tiene en memoria la suma de todas las señales recibidas. De este modo se tiene la suma de todas las señales y se facilita el promediado. El único problema es el posible desbordamiento de valores en la memoria. Esto se puede solucionar de diferentes maneras:

- Promediando un número reducido de señales, de modo que no de lugar a desbordamiento. Ésta opción es poco deseable, ya que interesa minimizar el nivel de ruido, promediando un número elevado de señales.

- Aumentando el número de bits de cada palabra de la memoria *RAM*. Esto puede no ser posible sin el uso de una memoria externa, ya que la memoria interna de las *FPGA* es típicamente muy reducida.

- Controlando internamente cuando se produce el desbordamiento y desechando los bits menos significativos, que son los más ruidosos y los que menos información transportan. Esto puede ser peligroso, ya que, en presencia de mucho ruido, es posible que la excursión de la señal útil comprenda un número reducido de bits, y el resto de la señal sea ruido. En este caso, al realizar los promediados, se observa cómo los bits más significativos tienen siempre el valor 0.

[4] *Random Access Memory.*

- En la práctica, se adopta un compromismo entre las tres soluciones anteriores. Se usa tanta memoria como sea posible, controlando que no ocurra desbordamiento, y realizando un número adecuado de promediados, de modo que no se pierda información.

Para finalizar el proceso de promediado, hace falta dividir por el número total de señales que se han sumado. Las operaciones con valores en coma flotante, especialmente las multiplicaciones y divisiones, necesitan de un número muy elevado de recursos físicos cuando se compara con los necesitados por las operaciones con valores enteros, recursos que son muy limitados en dispositivos como las *FPGA*, por lo que se intenta evitar la realización de operaciones en coma flotante en la medida de lo posible. Por este motivo se trabaja con la señal sin haberla dividido por este valor, teniendo siempre en cuenta que se está trabajando con una señal temporal escalada por un cierto factor. En caso que sea absolutamente imprescindible encontrar el valor promediado real, se trabaja con un número de señales que sea potencia de 2, ya que para dividir un número por 2^n basta con desplazar la representación binaria entera de dicho número n bits a la derecha, y no ers necesario implementar ninguna operación con valores en coma flotante.

El resto de operaciones descritas en el capítulo 4 consiste en diezmar la señal, interpolarla o usar la técnica de sobremuestro. Todas estas operaciones son factibles de ser implementadas con relativa facilidad.

9.2. Procesado posterior

El resto del procesado que se aplica a la señal después de realizar el procesado a frecuencia intermedia descrito en el capítulo 4 consiste en el análisis espectral. Para llevar a cabo estas técnicas es necesario implementar diversos algoritmos, como por ejemplo el de la transformada rápida de Fourier o uno para resolver sistemas de ecuaciones lineales[5] para poder aplicar los estimadores espectrales paramétricos. Estos y otros algoritmos se implementan con mayor facilidad usando lenguajes como *C* que describiéndolos directamente en hardware, ya que una implementación de estas rutinas en *VHDL* requiere trabajar con números en coma flotante y realizar un número elevado operaciones aritméticas y lógicas con ellos, lo que supone el uso de una gran cantidad de lógica electrónica.

[5]Como Jacobi, Gauss-Seidel, etc.

Una posible solución consiste en la utilización de los denominados *IP Core*[6], que consisten en unidades lógicas reutilizables que realizan una función determinada. Estos núcleos suelen realizar operaciones que tienen un elevado coste de diseño y desarrollo, y son propiedad de la empresa que los ha desarrollado para licenciarlos. El precio de un IP core es muy elevado, pero ahorra el coste de desarrollo por parte de la empresa que requiere de su uso. Son módulos que tienen una interfaz que permite su integración en el resto de un diseño, y se tratan como cajas negras, ya que no se tiene acceso al código de desarrollo, y no se posee información sobre ellos más allá de las que proporciona el distribuidor. En el presente caso, se podrían usar IP cores que realizaran tareas como la transformada de Fourier, por ejemplo.

Si se tiene en cuenta que una vez adquiridas las muestras de la señal a analizar ya no es crítico realizar estas operaciones en tiempo real, motivo por el que se suelen usar las *FPGA*; otra posible solución sería aprovechar la interfaz de comunicación entre la *FPGA* y la presencia del microcontrolador en el sistema de control del equipo para trasladar el control sobre el procesado de señal al microcontrolador, en el que resulta mucho menos complicada la implementación de los diversos algoritmos utilizados. Esta solución eliminaría la desventaja que supondría la dependencia de los distribuidores de *IP Cores*, y resultaría más económica a largo plazo.

9.3. Detalles de implementación

En la figura 9.1 se representa un esquema de la integración del presente proyecto con los otros dos proyectos finales de carrera que han sido desarrollados paralelamente, comentados en el capítulo 1. La parte correspondiente a este proyecto es el cuadro azul con título *Procesado de señal*. Cada una de las tres interfaces representadas se implementa como un módulo en la FPGA, y se describe en lenguaje *VHDL*.

La figura 9.2 esquematiza el funcionamiento de la adquisición de una señal durante un experimento. Por una parte, la *FPGA* espera una señal por la interfaz de control que indique en qué momento exacto se debe iniciar la captura. El equipo de control sincroniza todo el sistema para que la captura se haga en el momento adecuado. Para realizar la captura se ha desarrollado un módulo en *VHDL* que aprovecha la memoria *RAM* interna de la *FPGA*. Este módulo permite adquirir diversas señales secuencialmente

[6]Literalmente, núcleo de propiedad intelectual.

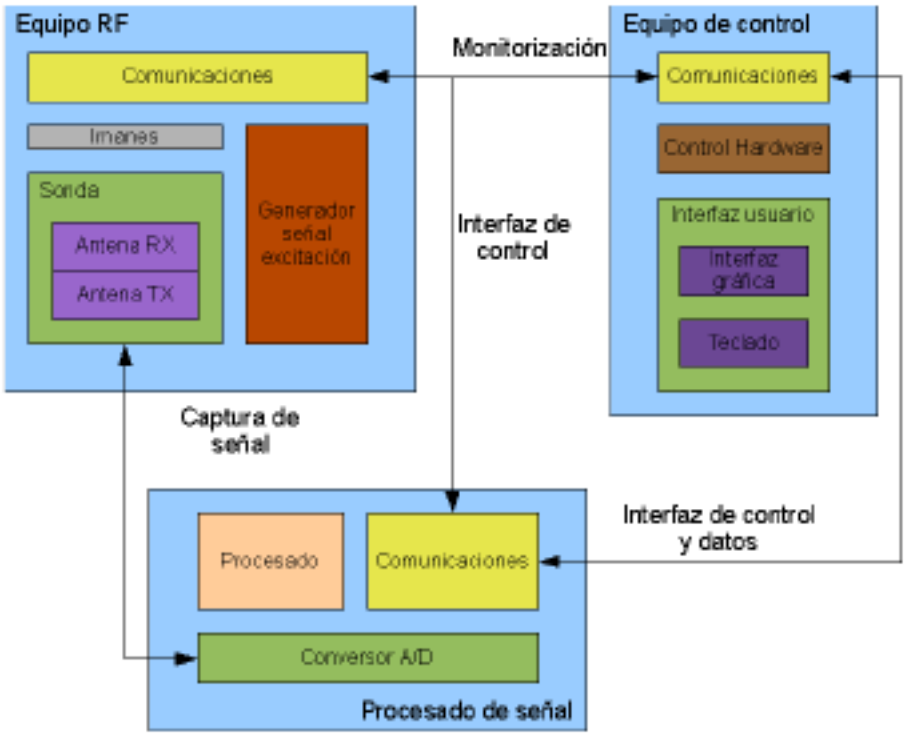

Figura 9.1: Diagrama de integración de los tres componentes del equipo de análisis RMN.

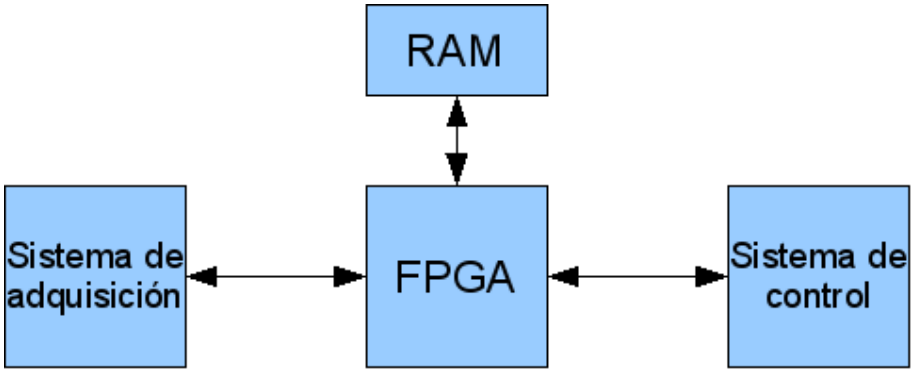

Figura 9.2: Diagrama de funcionamiento de la adquisición de señal.

para realizar un promediado que reduzca la potencia de ruido, como se describe en la sección 4.6. La fase de adquisición de la señal proveniente del equipo RF es crítica, ya que debe hacerse en tiempo real y con una sincronización perfecta, motivo por el cual se utiliza una *FPGA*, ya que el microprocesador del equipo de control debe ejecutar a diversos procesos y no puede dedicarse exclusivamente a la captura. Cualquier pequeño retardo introducido por el sistema operativo durante la adquisición de la señal invalidaría el experimento.

Una vez la *FPGA* ha recogido la señal del experimento, el procesado posterior no tiene restricciones tan rígidas de tiempo, y puede hacerse tanto en la *FPGA* como en el procesador del sistema de control. En un primer prototipo, el procesado se realiza en el procesador, por lo que la *FPGA* transmite la señal recibida al sistema de control mediante la interfaz de datos que conecta ambos subsistemas. Una vez se han transmitido los datos, el sistema de control procesa los datos y presenta el resultado al usuario final. La posibilidad de implementar y realizar parte del procesado en la *FPGA* aligeraría la carga del procesador en caso necesario.

9.4. Conclusiones

Durante la realización del presente proyecto, se han implementado los procesos descritos en la sección 9.1 en lenguaje *VHDL* y se han programado en la *FPGA*, y los descritos en la sección 9.2 se han implementado en lenguaje *C* y los ejecuta el microprocesador *Colibri*. Se han conectado las placas de

ambos proyectos, haciendo posible su comunicación y se ha comprobado el correcto funcionamiento del sistema completo.