Sistema de Adquisición y Tratamiento de Datos del RICH-1 del Experimento COMPASS

Maria Liz Crespo

Director

Dr. Alberto A. Colavita

Tesis para optar al grado de

Doctor en Ciencias de la Computación

Universidad Nacional de San Luis

San Luis, Argentina

2004

A la memoria del Dr. Raúl H. Gallard

Agradecimientos

En primer lugar quiero agradecer a mi director de tesis *Dr. Alberto A. Colavita* por haberme dado la oportunidad de colaborar en un experimento desafiante como COMPASS, y sobre todo por haberme permitido desarrollar y llevar a cabo esta tesis.

En segundo lugar quiero expresar mi gratitud al *Lic. Andres Cicuttin* por haber cumplido el rol de codirector de esta tesis, sin serlo formalmente. Agradezco su disponibilidad permanente, sus valiosos aportes en la escritura de la tesis, y su amistad.

Este trabajo se desarrolló en el Laboratorio de Microprocesadores del Centro Internacional de Física Teórica (ICTP) y del Instituto Nacional de Física Nuclear (INFN) de Trieste, Italia. Expreso mi agradecimiento al Prof. Franco Bradamante, al ICTP y al INFN. Agradezco también a todos mis colegas del Laboratorio de Microprocesadores y de la Sección de Trieste del INFN quienes colaboraron en el desarrollo y ejecución de todo el sistema de adquisición de datos del RICH de COMPASS. En particular expreso mi gratitud a Stanka Tanascovich, Alexandre Chapiro, Veronica Diaz, Razak Ijaduola, Fabio Fratnik, Anna Martin, Silvia Dalla Torre, Fulvio Tessarotto, Andrea Bressan, Benigno Gobbo y Pietro Cristaudo.

Mi agradecimiento al Departamento de Informática de la Facultad de Ciencias Físico, Matemáticas y Naturales dependientes de la Universidad Nacional de San Luis por haberme apoyado en la realización de mis estudios de posgrado en el exterior. En

especial quiero expresar mi gratitud a mis amigos Norma Herrera y Guillermo Leguizamon, y también a Susana Esquivel, Cecilia Montoya y Marcelo Errecalde, por haberme ayudado a resolver los trámites administrativos relacionados con la tesis, y sobre todo por darme la certeza de poder siempre contar con ellos.

Quiero también agradecer a mi hermana Silvia, a mi cuñado Kiko, a mis sobrinos Matias y Martina, y a todos mis amigos quienes de una manera u otra me soportaron y alentaron durante el desarrollo de este trabajo.

Pero esta tesis no hubiera sido posible sin el apoyo, motivación y cariño incondicional de mi esposo Marco y de mis padres Silvia y Jorge. Para ellos, mi amor y gratitud.

Indice General

Indice de Figuras

Indice de Tablas

Introducción

Los experimentos de física de las altas energías de última generación se caracterizan, desde el punto de vista instrumental, por la enorme cantidad y velocidad de los datos generados desafiando los más modernos sistemas existentes de adquisición de datos. Tales experimentos requieren la creación de soluciones inéditas que están condicionadas por restricciones de índole práctica. Existe una diferencia substancial entre la creación totalmente académica libre y la creación con restricciones impuestas por el propósito del sistema a diseñar, el tiempo de realización y hasta los fondos disponibles para el proyecto.

El tema de esta tesis está encuadrado dentro de un ambiente con restricciones que ataca un problema que requiere soluciones tecnológicas originales e innovativas de importancia para la física de las altas energías y las ciencias de la computación. Muchas innovaciones de hardware, arquitectura y software han nacido de esta problemática. Quizás la contribución más conocida es la *World Wide Web* creada en el CERN (*European Organization for Nuclear Research*) para resolver el problema práctico de comunicación de los físicos de las altas energías.

Así como ocurrió con la *Web*, los experimentos actuales de física de las altas energías desafían el "estado del arte" en ciencias de la computación, teniendo exigencias extremas de potencia computacional y capacidad de comunicación y

almacenamiento. Sus demandas en campos tan variados como cálculos complejos, reconstrucción y procesamiento de imágenes, sistemas veloces de adquisición de datos, sistemas distribuidos de tiempo real, sistemas de redes y comunicación, y técnicas de simulación proveen grandes estímulos para el desarrollo de sistemas complejos nuevos y originales.

En particular, las necesidades de los sistemas de adquisición de datos de la nueva generación de experimentos constituyen un verdadero desafío. Ellos tienen que ser capaces de manejar la información proveniente de los miles de canales que conforman los detectores de partículas, y altos *trigger rates* que implican que los datos tengan que ser adquiridos y registrados en muy cortos tiempos. La exigencia de resolución espacial y temporal de los detectores de partículas, tanto como la necesidad de acumular una gran cantidad de datos para su elaboración estadística, imponen un elevado número de canales y una alta frecuencia de adquisición de los mismos. Esta exigencia cuantitativa determina nuevos problemas cualitativos, algunos de los cuales son descriptos a continuación:

- La cantidad de datos adquiridos hace necesaria la reducción de los mismos antes de ser almacenados. Por medio de algún tipo de pre-procesamiento *in situ* tales datos pueden ser reducidos antes de incorporarlos al flujo principal. Por ejemplo, datos con señales por debajo de cierto nivel de ruido podrían ser descartados inmediatamente o varios datos parciales podrían ser combinados en datos simples más significativos (*clusterization, median filtering, correlation, pattern recognition*, etc.). Esto requiere que el sistema sea más sofisticado, incluyendo una cierta capacidad de elaboración *on line*.
- El número y la distribución física de los canales hace necesaria la división del sistema de adquisición en subsistemas más pequeños que trabajen en paralelo, y

en consecuencia aparece el problema de la sincronización y el control de los mismos.

- Las características asíncronas del sistema de adquisición así como el alto *trigger rate* con mínimo tiempo muerto requieren la implementación de métodos de tiempo real.
- Cuando un instrumento debe durar años y las condiciones experimentales pueden variar, se hace necesario que el sistema de adquisición sea suficientemente flexible para poder adaptarse a los cambios requeridos. Por lo tanto es fundamental el uso de dispositivos reprogramables y/o reconfigurables.
- La complejidad de ciertos instrumentos requiere un estudio intensivo de los mismos en modo de poder usarlos eficientemente. Para este propósito el sistema de adquisición debe proveer las funciones necesarias que permitan la realización de tal estudio, por ejemplo el análisis del ruido y de las señales.
- Durante el tiempo de ejecución del experimento, la zona experimental se transforma en radiactiva y no es posible el acceso a la misma. Por lo tanto, el sistema de adquisición debe poder ser programado y controlado remotamente.

Tales requerimientos determinan el desarrollo de sistemas de adquisición de datos nuevos, altamente especializados y optimizados en velocidad. Esta tesis describe la solución arquitectural, haciendo hincapié en el software, de un sistema masivamente paralelo de adquisición y tratamiento de datos que satisface las exigencias descriptas anteriormente. Tal sistema fue desarrollado y está siendo usado por el detector de partículas RICH del experimento COMPASS que se encuentra en funcionamiento en el CERN desde el año 2002. COMPASS es en la actualidad el experimento adquiriendo ampliamente la mayor cantidad de eventos por unidad de

tiempo comparado con otros experimentos de física de las altas energías, y su funcionamiento está previsto hasta el año 2010.

Para dar una idea de la complejidad del detector y en consecuencia de su sistema de adquisición y tratamiento de datos, el RICH cuenta con aproximadamente 83000 canales, distribuidos físicamente en una superficie de 5,3 m^2, que deben ser adquiridos a un *trigger rate* que puede alcanzar los 100 kHz con un tiempo muerto en el orden del microsegundo. El valor de cada canal es digitalizado en 10 bits a los cuales se le suma su identificación representada en 18 bits, resultando en un total de aproximadamente 29 GBytes/s de datos generados.

La complejidad del detector RICH de COMPASS, el elevado número y distribución física de los canales y el alto *trigger rate* con mínimo tiempo muerto, definieron al sistema de adquisición y elaboración de datos del detector como un problema único para el cual fue necesario la creación de una solución nueva y original que provea paralelismo masivo, alta performance, respuesta en tiempo real y flexibilidad. A tal efecto se desarrolló un sistema reconfigurable de adquisición y elaboración de datos de tiempo real duro, basado en Procesadores de Señales Digitales (DSP) conectados en red y combinados con dispositivos de lógica programable (FPGA). El sistema está conformado por 200 DSP y cada DSP cuenta con una FPGA actuando como coprocesador.

Una programación adecuada del DSP permite explotar la potencialidad de tales procesadores, conectarlos en red y combinarlos con dispositivos FPGA. Los DSP de última generación están físicamente capacitados para formar parte de una red y cuentan con una memoria interna suficiente para implementar en software el protocolo y la arquitectura de comunicación de la red. La combinación DSP-FPGA es

adecuada para implementar sistemas reconfigurables de tiempo real duros, conjugando el paralelismo del hardware con la flexibilidad del software.

Esta tesis conjuga aspectos de varias áreas de Ciencias de la Computación tales como: Sistemas de Tiempo Real, Redes de Computadoras, Procesamiento de Señales Digitales y Computación Reconfigurable.

Organización de la Tesis

En el *capítulo 1* se introduce el experimento COMPASS. Se describen las características generales de COMPASS desde el punto de vista instrumental, y en particular se describe el detector RICH. Este capítulo y el próximo tienen como objetivo dar una visión global del contexto en el cual se desarrolló la tesis, proveyendo además las correspondientes referencias para quienes estén interesados en los detalles del experimento.

En el *capítulo 2* se describe la cadena de adquisición de datos global de COMPASS, de la cual forma parte el sistema de adquisición y elaboración de datos del detector RICH. Aproximadamente un tercio de los canales que deben ser adquiridos por todo el experimento pertenecen al RICH.

En el *capítulo 3* se presentan, en primer lugar, las características y los requerimientos generales asociados con el sistema de adquisición de datos del detector. En segundo lugar se describe la arquitectura global del sistema de adquisición de datos del RICH, y se dan las justificaciones de las elecciones realizadas para su implementación. Por último se describe el protocolo y la arquitectura de comunicación de la red de DSP, y se describen los mecanismos que permiten la reprogramabilidad y reconfigurabilidad del sistema.

En el *capítulo 4* se presenta la tarjeta MultiDSP (Dolina) que principalmente gestiona la red de DSP y la comunicación de la red con la PC de control del sistema. Desde Dolina se distribuyen también señales de sincronización con restricciones temporales duras. Se describe en detalle el software de los DSP de Dolina, responsable de la funcionalidad de la tarjeta.

En el *capítulo 5* se presenta la tarjeta principal de adquisición, elaboración y transmisión de datos (Bora) describiendo en forma detallada su funcionalidad y arquitectura. El cerebro de Bora es un DSP actuando conjuntamente con una FPGA.

En el *capítulo 6* se describe en detalle el software del DSP de Bora: servicios ofrecidos, tareas de tiempo real, gestión de la red, e interacción con la FPGA y con otros dispositivos involucrados en la funcionalidad de Bora.

En el *capítulo 7* se describe, en primer lugar, la arquitectura global implementada en la FPGA y su funcionalidad como coprocesador del DSP. En segundo lugar se describe el software de configuración de la FPGA.

En el *capítulo 8* se presenta una discusión y generalización de las aproximaciones usadas y de los resultados obtenidos.

Finalmente en el capítulo de *conclusiones* se presenta la performance obtenida por el sistema desarrollado y descrito en este trabajo, y se resumen los aspectos sobresalientes de las conclusiones de la tesis, algunos de los cuales se enuncian con más detalle en el capítulo 8.

A lo largo de la tesis se usarán abreviaturas, las cuales se resumen en un glosario al final de los capítulos. Términos técnicos usualmente usados en inglés no fueron traducidos al español y se incluyen en estilo itálico.

Publicaciones Relacionadas con la Tesis

Durante el desarrollo de la tesis se publicaron (como coautor) los siguientes artículos describiendo parte de la misma y algunos de los resultados obtenidos:

1. Baum, G. et al. "*BORA: a Front End Board, with local intelligence, for the RICH Detector of the COMPASS Collaboration*". Nuclear Instruments & Methods in Physics Research (NIM-A). Section A 433 (1999) 426-431.

2. Albrecht E. et al. "*COMPASS RICH-1*". Nuclear Instruments & Methods in Physics Research (NIM-A). Section A 478 (2002) 340-343.

3. Baum, G. et al. "*The COMPASS RICH-1 read-out system*". Nuclear Instruments & Methods in Physics Research (NIM-A). Section A 502 (2003) 246-250.

4. Baum, G. et al. "*RICHONE: a software package for the analysis of COMPASS RICH-1 data*"**.** Nuclear Instruments & Methods in Physics Research (NIM-A). Section A 502 (2003) 315-317.

5. Albrecht E. et al. "*COMPASS RICH-1*". Nuclear Instruments & Methods in Physics Research (NIM-A). Section A 502 (2003) 112-116.

6. Albrecht E. et al. "*First performances of COMPASS RICH-1*". Nuclear Instruments & Methods in Physics Research (NIM-A). Section A 518 (2004) 586-589.

Capítulo 1

El Experimento COMPASS al CERN

El CERN es una organización internacional de investigación científica compuesta por 20 estados europeos, que nace basado en la premisa que la ciencia es un esfuerzo comunitario y depende del libre acceso a la información e intercambio de ideas. En la actualidad alrededor de 6500 científicos provenientes de aproximadamente 80 países de todo el mundo participan de los experimentos del CERN. Sin embargo típicamente tales científicos trabajan la mayor parte del tiempo en sus propias universidades e institutos. El modo de organización es formando "colaboraciones" que tienen como objetivo el desarrollo de un experimento. Actualmente tales colaboraciones están conformadas por cientos de físicos, ingenieros, *computer scientists* y otros expertos. Esto refleja el tamaño y la complejidad de los aparatos necesitados para llevar a cabo un experimento. De este modo grupos de investigación distribuidos en distintos paises diseñan y construyen instrumentos altamente complejos, y operan en conjunto con otros grupos durante el tiempo de ejecución del experimento.

La colaboración COMPASS al CERN comprende 220 científicos provenientes de 26 universidades e institutos de todo el mundo. La tarea fundamental de COMPASS es mejorar nuestro entendimiento de la estructura del hadron[1], para este

[1] Partículas hechas de *quarks*, incluyendo los núcleos (protones y neutrones) de la materia ordinaria.

propósito se construyó un detector de 50 metros de largo en la línea de partículas del acelerador *Super Proton Synchrotron* (SPS) del CERN.

Este capítulo presenta una descripción general del experimento COMPASS a los efectos de introducir globalmente el contexto en el cual se desarrolló esta tesis. Se describen el ciclo del acelerador SPS, en el cual se producen las partículas que deben ser detectadas e identificadas por la estructura instrumental del experimento, y las características generales de dicha estructura, poniendo énfasis en uno de los detectores de partículas claves del experimento: el *Ring Imaging Cherenkov* RICH-1.

1.1 El Experimento COMPASS

COMPASS (*COmmon Muon Proton Apparatus for Structure and Spectroscopy*) es un experimento de física de las altas energías que se desarrolla al CERN. El propósito del experimento es el estudio de la estructura y espectroscopia del hadron, y de la estructura espín del nucleon, usando haz de muones y haz de hadrons de alta intensidad. Detalles de los objetivos de la física del experimento pueden ser encontrados en [1, 2, 3, 4, 5].

COMPASS es un experimento de blanco fijo localizado en el área norte del CERN. La Figura 1.1 muestra los aceleradores del CERN relevantes a COMPASS (SPS y *Proton Synchrotron* (PS)) y la posición del área experimental. Protones acelerados por el SPS a una energía de 400 *GeV* son emitidos en el llamado punto de extracción y guiados hasta el blanco de producción T6. Las colisiones de protones y el núcleo del blanco, producen principalmente piones y kaones decayendo a muones polarizados longitudinalmente. Atravesando la línea de partículas M2, los muones son dirigidos al blanco polarizado del experimento COMPASS, donde la interacción toma lugar.

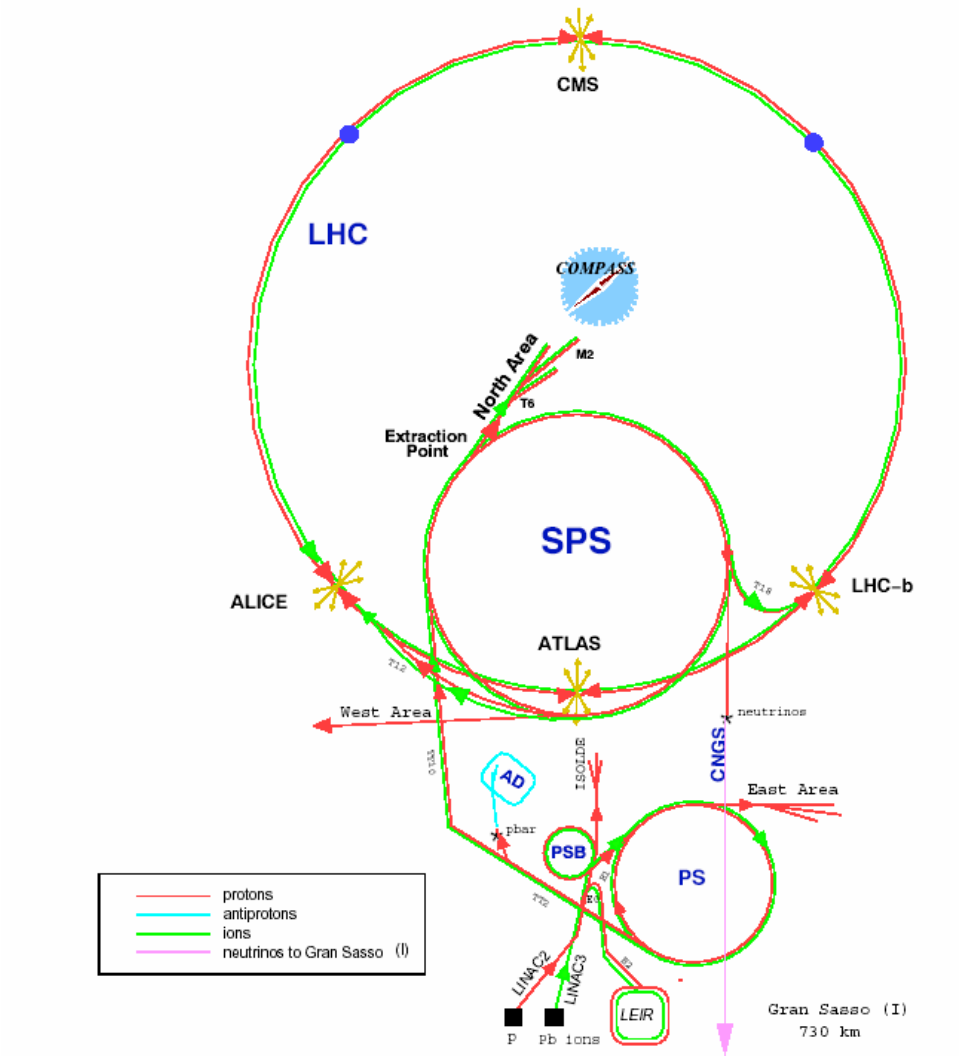

Figura 1.1 Aceleradores relevantes a COMPASS (SPS y PS) y posición del área experimental.

1.2 La Estructura de *spill* del SPS

El haz de muones para el experimento COMPASS se produce haciendo colisionar protones de alta energía contra un blanco de producción. En un cierto punto especial de extracción, los protones provenientes del anillo del acelerador se arrojan contra el blanco de berilio de producción T6 con una intensidad incidente en el rango de 2×10^{12} a 1.2×10^{13} protones por cada ciclo del SPS. Debido al ciclo de trabajo del SPS, los muones útiles para COMPASS se producen durante 5,1 segundos (*on-spill time*) seguido por un periodo sin partículas de 11,7 segundos (*off-spill time*),

determinando así un ciclo total de 16,8 segundos. La Figura 1.2 muestra esquemáticamente el ciclo del SPS donde la línea continua representa la intensidad de los protones en el SPS.

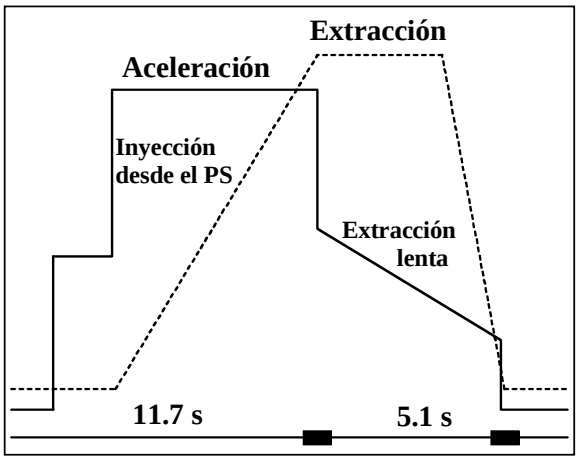

Figura 1.2 Estructura de *spill* del SPS.

La inyección de protones preacelerados por el PS en el SPS produce el escalón visible en la izquierda de la figura. Durante la aceleración la intensidad permanece constante hasta que los protones son extraídos hacia el blanco T6 (extracción lenta). Piones y kaones seleccionados por sus momentos, emergen de la colisión protón-berilio decayendo en muones en la línea de partículas M2. Esta línea de partículas puede ser convertida relativamente rápido en una línea hadrónica para los programas de física que así lo requieren.

1.3 El Espectrómetro de COMPASS

El programa de la física de COMPASS determina el diseño del experimento desde el punto de vista instrumental. COMPASS usa un espectrómetro con excelente detección e identificación de partículas, capaz de soportar flujos de hasta 2.8×10^8 partículas por *spill*. La Figura 1.3 muestra esquemáticamente el espectrómetro usado durante el año 2002. La granularidad de los detectores usados es adecuada para el

flujo de partículas esperado, el cual alcanza los 30 MHz/cm^2 en la línea de haz central. La gran aceptación del espectrómetro de COMPASS se logra por medio de un sistema de rastreo subdividido en un conjunto de detectores en sucesión con creciente capacidad de lectura de datos. *Triggers* dedicados y sistemas de adquisición de datos veloces y complejos complementan la performance del espectrómetro.

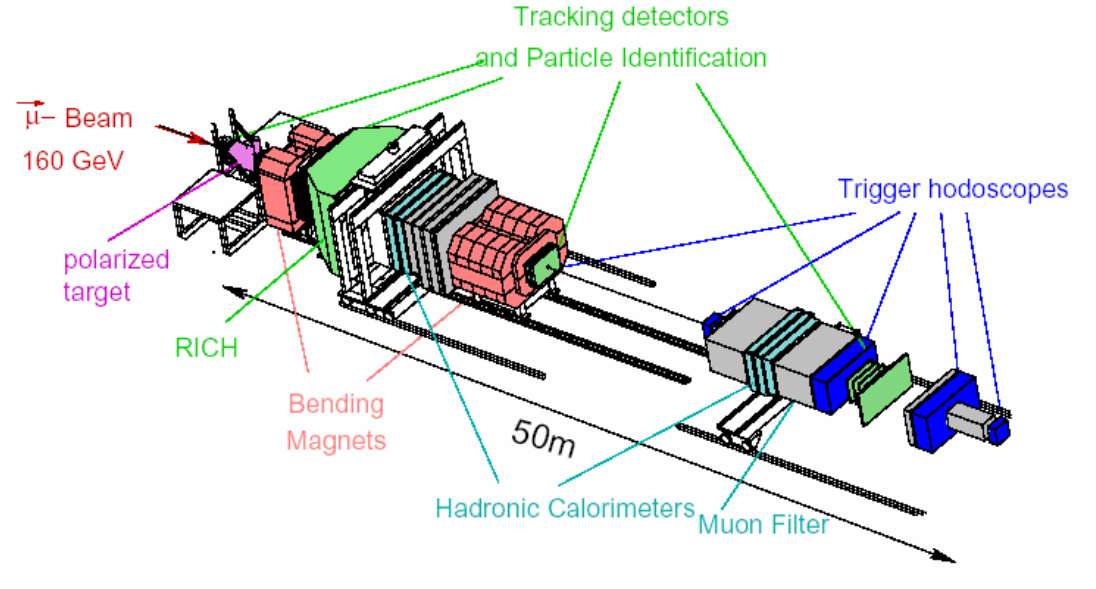

Figura 1.3 El espectrómetro del experimento COMPASS

La primer sección del espectrómetro detecta las partículas emergentes a ángulos grandes, y la segunda detecta partículas con momento alto emergiendo a ángulos más pequeños. Ambas secciones contienen detectores de rastreo de área grande y pequeña. Detectores rápidos se posicionan en el haz y cerca de él, determinando así la necesidad de una alta granularidad de estos detectores. Los detectores de área grande detectan partículas en regiones distantes del haz. Con esta estructura se obtiene aproximadamente una aceptación vertical de 180 *mrad* y una horizontal de 250 *mrad*.

Para medir el momento de las partículas se utilizan dos magnetos a campo abierto. Puesto que el ángulo de deflexión en un campo magnético depende del

cociente entre la intensidad del campo magnético y el momento de la partícula, el primer magneto del espectrómetro se usa para desviar y separar la trayectoria de las partículas en la región de bajo momento. Las partículas con momento alto pasan casi sin deflexión por el primer magneto y el segundo magneto es responsable de la curvatura de las trayectorias.

Detectores RICH, calorímetros hadrónicos y electromagnéticos, y filtros de muones son usados para la identificación de partículas. Los filtros de *muones* distinguen muones de otras partículas cargadas. Los calorímetros son principalmente usados para medir la energía de las partículas. Descripciones detalladas de las diferentes componentes del espectrómetro pueden ser encontradas en [5, 6, 9]. Puesto que esta tesis propone un sistema de adquisición y procesamiento de datos masivo que fue desarrollado y esta siendo usado por el RICH, uno de los detectores claves de COMPASS, la siguiente sección da una descripción más detallada del mismo.

1.4 El Detector RICH de COMPASS

COMPASS usa un detector RICH, llamado RICH-1, para la detección e identificación de partículas. Los detectores RICH son usados para medir la velocidad de partículas cargadas de alta energía [7, 8]. La Figura 1.4 muestra esquemáticamente las principales componentes de un detector RICH. La cámara de hilos con *pixels* detectores (*photodetector*) es sensible a los fotones *Cerenkov* emitidos en un cono por una partícula que atraviesa un medio denso a una velocidad mayor a la de la luz en ese mismo medio. El ángulo del cono de emisión o el radio del círculo proyectado como así también el número de fotones producidos son una medida directa de la velocidad de la partícula. Esta medida junto con una medida independiente del momento permite identificar la partícula y su energía.

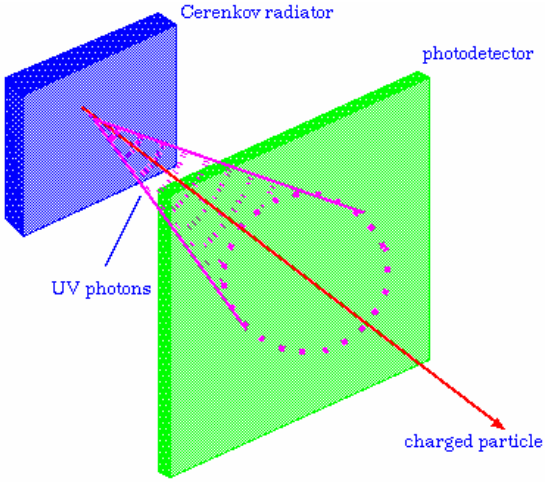

Figura 1.4 Esquema de un detector RICH

Casi todas las mediciones físicas de COMPASS necesitan una buena identificación hadrónica, y este es uno de los requerimientos más desafiantes del experimento. El objetivo principal del detector RICH de COMPASS es la identificación del hadron. De acuerdo a los requerimientos del experimento, el RICH debe ser capaz de separar piones, kaones y protones con un momento de hasta 50 *GeV/c* en un ambiente de gran intensidad. El RICH-1 debe cubrir toda la aceptación prevista a gran apertura angular (horizontal: 250 *mrad* y vertical: 180 *mrad*). Estas consideraciones definieron los parámetros de diseño del RICH-1 de COMPASS [10, 11, 12, 13]. La Figura 1.5 muestra el RICH-1 en forma esquemática. Las partículas que atraviesan el radiador de gas (*gas radiator*) inducen la emisión de fotones *Cerenkov* los cuales son enfocados por un conjunto de espejos (*UV mirror*) hacia detectores de posición de fotones (*photon detectors*) como se muestra en la parte izquierda de la Figura 1.5.

El RICH-1 de COMPASS es un detector gaseoso cuyo radiador de 3 m de longitud contiene C_4F_{10}/N_2 a presión atmosférica [14]. El conjunto de espejos está formado por espejos esféricos de 6,6 m de radio, segmentados en 116 piezas hexagonales cubriendo un área total de 21 m^2 [15]. Estos espejos forman dos

superficies esféricas con diferentes centros de curvatura para poder focalizar los fotones *Cerenkov* sobre los dos detectores de fotones ubicados arriba y abajo de la región de aceptación de partículas. Los fotones emitidos con un ángulo de *Cerenkov* producen en la superficie de los detectores de fotones un círculo de radio *r*, a partir del cual se puede calcular la velocidad (β) de la partícula:

$$\beta = \left(n \cdot \cos \frac{2r}{R_s} \right)^{-1}$$

donde *n* es el índice de refracción del radiador (C_4F_{10}: n = 1.00153, N_2: n = 1.000297) y R_S = 6,6 m es el radio de curvatura de los espejos.

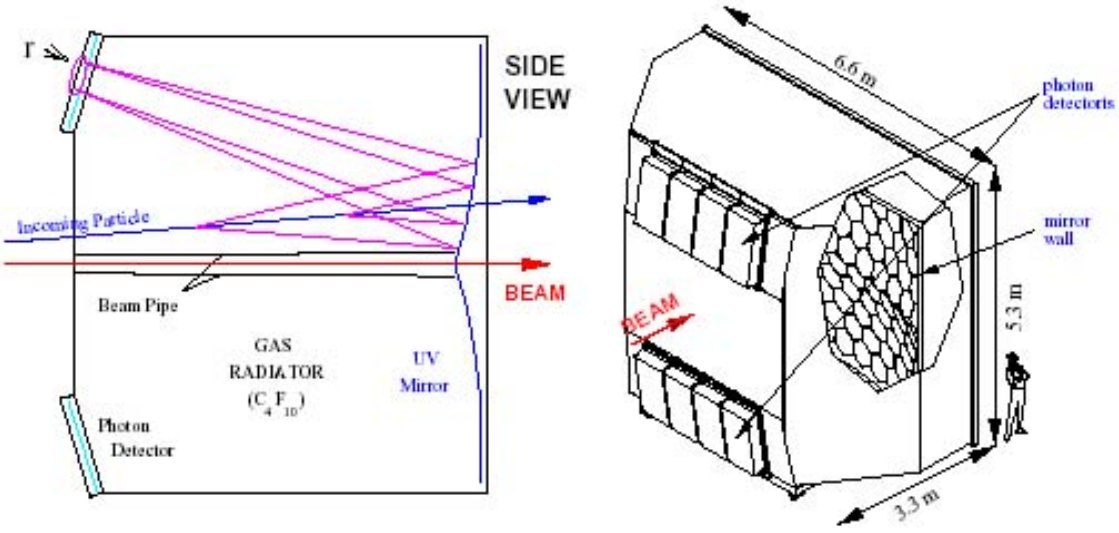

Figura 1.5 Esquema del RICH-1 de COMPASS.

Los detectores de fotones son cámaras multi-hilos proporcionales (*Multi-Wire Proportional Chambers* (MWPC)), equipadas con fotocátodos de *CsI* (Ioduro de Cesio), con una área activa total de 5,3 m^2. Una multitud de señales rápidas con alta eficiencia y bajo nivel de ruido son provistas por los detectores de fotones que cuentan con pocos milímetros de resolución espacial. La segmentación del área activa total en *pixels* de 8,8 mm^2 resulta en un total de 82944 canales, cada uno de los cuales

cuenta con su propia electrónica analógica de lectura. Ocho cámaras idénticas organizadas en dos grupos de cuatro ubicados arriba y abajo de la zona de aceptación de partículas, tal como muestra la parte derecha de la Figura 1.5, conforman los detectores de fotones. Cada cámara consta de una superficie activa de 60 x 120 cm^2. Los fotocátodos están formados por dos PCBs (*printed circuit boards*) de doble capa que cuentan con un área de 60 x 60 cm^2.

Las tarjetas para la adquisición de los datos generados por el detector, las cuales constituyen uno de los principales temas de esta tesis, serán descriptas en detalle en los siguientes capítulos. Estas tarjetas son directamente colocadas en la parte externa de los fotocátodos. Desde el punto de la adquisición de datos, el RICH-1 es el detector que genera la mayor parte (30%) de los datos producidos por el experimento. Los primeros círculos han sido obtenidos durante el *run* del año 2002 [16, 17], tal como muestra la Figura 1.6 en la que se puede apreciar un evento reconstruido.

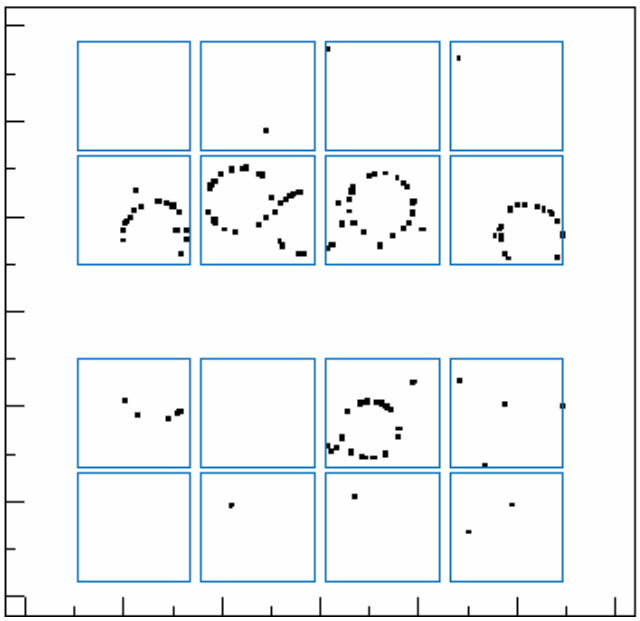

Figura 1.6 Evento "multi-círculo" reconstruido a partir de los datos generados por el detector RICH-1 durante el *run* del año 2002.

Capítulo 2

La Cadena de Adquisición de Datos de COMPASS

El enorme número de aproximadamente 250000 canales conformando los distintos detectores de la estructura instrumental de COMPASS, junto con las dimensiones de su espectrómetro (4 x 5 x 50 m^3) y el alto *data rate* comparado con el de otros experimentos, exigieron la aplicación de nuevas y más sofisticadas aproximaciones para el sistema de adquisición de datos global (DAQ) de COMPASS. Una característica fundamental es la flexibilidad y escalabilidad en modo de poder adaptarse a posibles cambios o futuras implementaciones de nuevos componentes. Dependiendo de los programas de física, el sistema tiene que ser capaz de soportar un *trigger rate* en el rango de 5 kHz a 100 kHz, dando por resultado un caudal pico de varios GBytes de datos por segundo y hasta 300 TBytes almacenados en cinta por año.

La comparación con otros experimentos de física de las altas energías que manipulan también *data rates* altos, tal como se puede observar esquemáticamente en la Figura 2.1, demuestra que aún el experimento *Large Hadron Collider* (LHC) al CERN, que comenzará a funcionar en el 2007, almacenará en cinta menos eventos por segundo. Sin embargo, el número de canales del detector LHC causará que la cantidad de datos registrados sea mayor a la de COMPASS. El sistema DAQ de COMPASS no

solo tiene que ser capaz de manejar *trigger rates* altos, conduciendo a una alta cantidad de eventos por segundo a ser almacenada en cinta, sino también una correspondiente cantidad grande de datos que supera la del experimento *BaBar*, que producía los más altos *data rates* antes de COMPASS. En el 2002 COMPASS fue el experimento generando ampliamente el mayor número de eventos, constituyendo un gran desafío para el desarrollo de sistemas de adquisición de datos nuevos y originales.

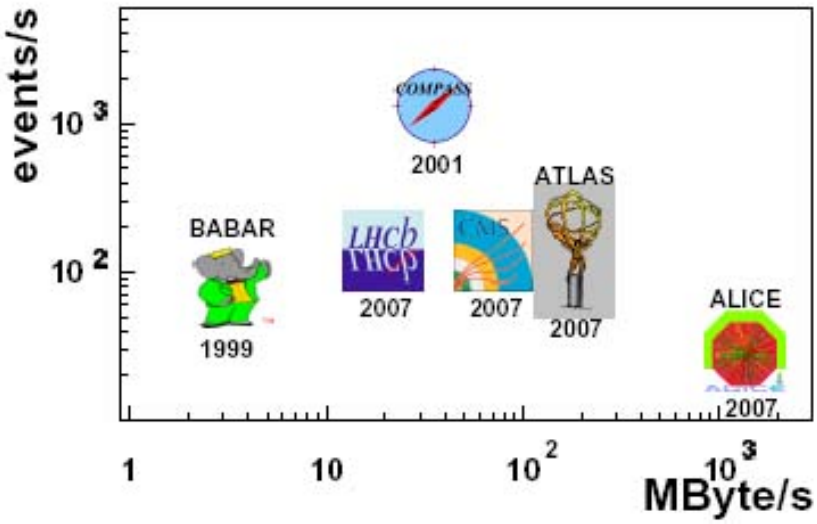

Figura 2.1 *Data rates* de experimentos de física de las altas energías actuales y futuros.

En los sistemas DAQ tradicionales se montan pre-amplificadores sobre los detectores y típicamente un gran número de cables de par trenzado o coaxial transmiten las señales a amplificadores o discriminadores. Desde allí se transfieren a convertidores digitales (*Analog to Digital* ***C****onverter* (ADC) o *Time to Digital Converter* (TDC)) montados en *Camac*, *VME* o *Fastbus Crates*. En este caso la adquisición se desarrolla a través de *backplane* y la reconstrucción local y global de eventos se efectúa en una etapa posterior sobre *VME CPUs* o *Workstations*.

En COMPASS se adoptó una aproximación diferente [18]. Las señales provenientes de los detectores se digitalizan y preprocesan en tarjetas de adquisición

montadas sobre los mismos, y se transfieren a módulos de lectura central (*Readout* D*river*) vía cables *Ethernet* o fibras ópticas comerciales. La Figura 2.2 muestra gráficamente la comparación entre los sistemas DAQ tradicionales (parte superior) y el sistema DAQ elegido por COMPASS (parte inferior) [9].

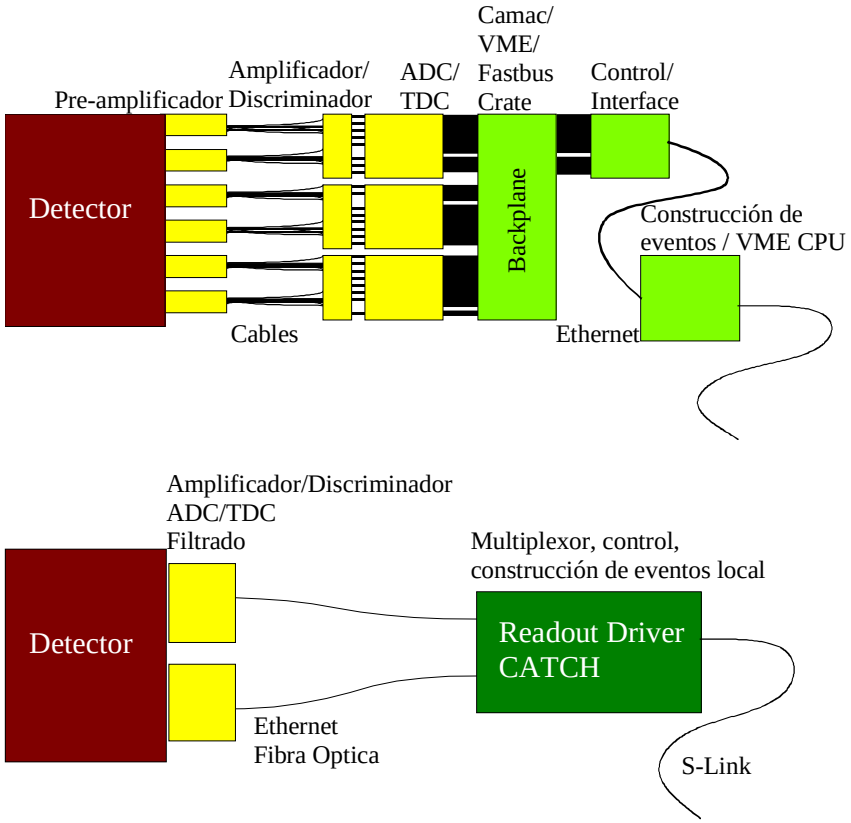

Figura 2.2 Comparación entre un sistema DAQ tradicional (parte superior) y el sistema DAQ de COMPASS (parte inferior).

El flujo de datos comienza en los detectores que cuentan con sus propios sistemas de adquisición. Los módulos de lectura (CATCH) concentran los datos tan cerca a las tarjetas de adquisición de los detectores como sea posible, en pocos aunque más grandes flujos de datos. Esto optimiza la cantidad de datos transmitidos por unidad de tiempo comparado con la adquisición vía *backplane* usada en los sistemas tradicionales, puesto que COMPASS usa conexiones punto a punto especializadas (S-

link). Este sistema también economiza la cantidad de cables como se puede ver en la Figura 2.2.

Los objetivos previamente mencionados, es decir, la manipulación de altos *data* y *trigger rates*, grandes eventos y tiempos muertos variables, son alcanzados adoptando un sistema totalmente paralelo y *pipelined*, en el cual los datos de cada *trigger* son guiados a través de toda la cadena de adquisición mientras otros *triggers* pueden ser aceptados. En distintas etapas los datos son colocados en *buffers* para minimizar el tiempo muerto y evitar la pérdida de información. Con el objetivo de proveer máxima flexibilidad, el sistema se realizó con un diseño modular.

En este capítulo se describen, en términos generales, cada una de las partes del sistema de adquisición de datos global de COMPASS siguiendo la trayectoria de los datos a través de la cadena de adquisición, esquemáticamente representada en la Figura 2.3. Información detallada de las partes individuales de este sistema puede ser encontrada en [18, 19, 20].

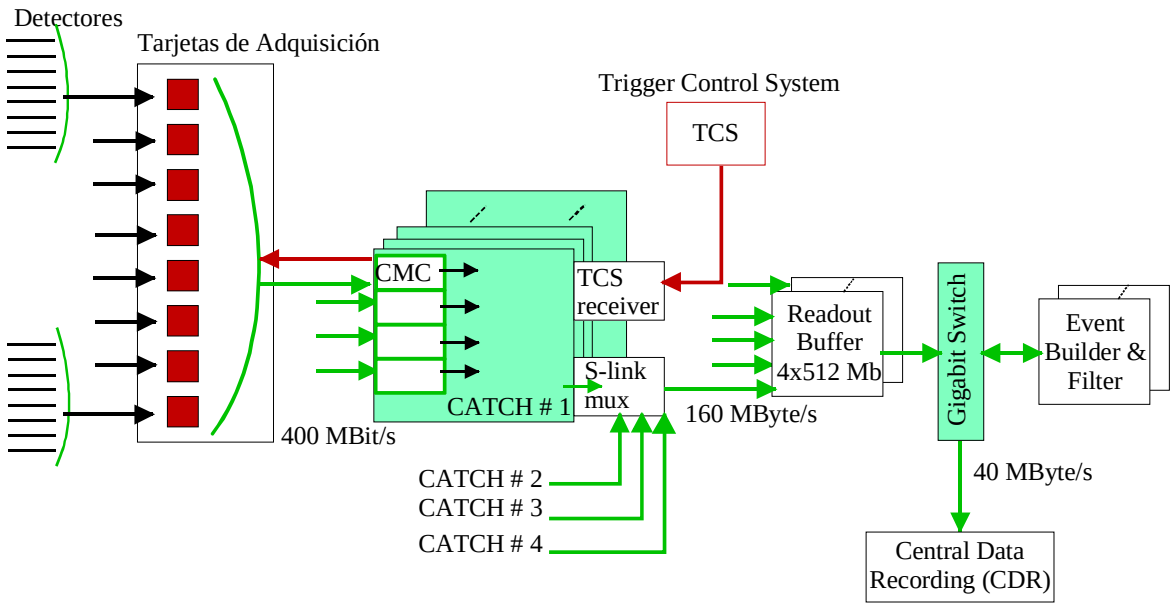

Figura 2.3 La cadena de adquisición de datos del experimento COMPASS.

2.1 Trayectoria del Flujo de Datos en la Cadena de Adquisición

El flujo de datos comienza en los detectores que generan señales analógicas. La digitalización de tales señales se lleva a cabo en los sistemas de adquisición de datos de los detectores, por medio de dispositivos de digitalización (ADCs y TDCs) incorporados en tarjetas de adquisición montadas directamente sobre los detectores. Altos *trigger* rates de hasta 100 kHz sólo se pueden manipular con la aplicación en tiempo real de mecanismos de selección, generalmente desarrollados en los sistemas de adquisición de los detectores, que permiten seleccionar sólo datos significativos para ser incorporados al flujo principal. Existen sistemas de adquisición especializados para cada tipo de detector, cada uno con características particulares.

Desde las tarjetas de adquisición de los detectores los datos formateados de acuerdo a un formato previamente establecido, se transmiten a módulos de lectura (CATCH) distribuidos, una parte integral del sistema DAQ de COMPASS, vía fibras ópticas o cables Ethernet estándar. Las principales tareas del CATCH son la rápida lectura de los datos provenientes de las tarjetas de adquisición y la reconstrucción local de sub-eventos a una velocidad de procesamiento de hasta 160 MBytes/s. En paralelo, el CATCH también distribuye las señales de *trigger* y de sincronización desde el Sistema de Control del *Trigger* (TCS) a las tarjetas de adquisición.

Desde el CATCH los datos formateados de acuerdo al formato de datos general de COMPASS, se transmiten hacia PCs de alta performance denominadas ROB (*ReadOut Buffer*) vía fibras ópticas. Se usa el protocolo S-Link desarrollado en el CERN para conexiones punto a punto veloces y toma lugar a través de un único S-Link o combinando hasta cuatro módulos CATCH por medio de un multiplexor S-Link. El sistema principal de reconstrucción de eventos consiste en PCs ROB y PCs constructoras de eventos. En contraste a los componentes previos, los cuales se

distribuyen sobre toda la extensión del experimento COMPASS, el sistema de reconstrucción de eventos se localiza en una habitación DAQ dedicada para mantener las conexiones tan cortas como sea posible.

Cada ROB contiene cuatro módulos RAM de 512 MByte cada uno, denominados *spillbuffers*, que pueden almacenar los datos de más de un *spill*. Un software específico combina la información de los *spillbuffers* y la transfiere vía un *Gigabit Switch* a las PCs constructoras de eventos para su posterior procesamiento. En estas computadoras, PCs comerciales como los ROBs, la información de cada evento proveniente de todos los diferentes ROBs se combina en un evento único, que ahora incluye los datos de todos los detectores. Por lo tanto, los datos que pertenecen a un mismo evento se transfieren a un mismo constructor de evento.

Desde los constructores de eventos, los eventos finales se transfieren al registro de datos central (CDR) donde se escriben en medios de almacenamiento masivo. En una primera etapa, se acumulan temporalmente en alguno de los discos locales de la COMPASS *Computing Farm* (CCF) antes de ser almacenados en cinta. El sistema completo es altamente escalable y fácil de actualizar. Mayores *data rates*, por ejemplo debido a un incremento en el número de canales de los detectores, puede ser fácilmente adaptado agregando nuevas ramas de adquisición.

En general, la adquisición de los datos sólo toma lugar durante el ciclo de *spill* del acelerador SPS (sección 1.1). Los datos son recibidos, procesados y transferidos por los sistemas de adquisición de los detectores, a través de los dispositivos CATCH, durante el tiempo de *spill* (5,1 s). Los *spillbuffer* se diseñaron en forma tal que puedan almacenar datos de más de un *spill*. En consecuencia se puede hacer uso de la pausa entre *spills* transmitiendo datos continuamente a las PCs constructoras de eventos. Varios de estos *spills* son agrupados formando un así llamado *run*. Un *run* contiene un

número preseleccionado de *spills* en el rango de 1 a 2047. Normalmente un *run* dura entre 20 y 30 minutos produciendo datos que se agrupan en varios archivos de 1 GByte cada uno. A cada *run* se le asigna un único número, que en combinación con el número de *spill* y el número de evento caracterizan unívocamente a cada evento durante el tiempo de vida del experimento.

Puesto que los datos de los detectores son parcialmente combinados al nivel de los módulos CATCH y la concentración final de los eventos se desarrolla en etapas posteriores, el número de evento tiene que ser conocido por el CATCH. Por esta razón el Sistema de Control del *Trigger* (TCS) le provee al CATCH la señal de *trigger* más un encabezamiento de evento que contiene la información necesaria. Las señales de *trigger* se distribuyen a todos los sistemas de adquisición de los detectores mientras que los encabezamientos de eventos se almacenan en los CATCH, quienes los utilizan para controlar, combinar y dar formato a los datos provenientes de las tarjetas de adquisición de los detectores, que también incluyen el número de evento e información adicional.

2.2 Sistema de Control del *Trigger*

La Figura 2.4 muestra un esquema de la arquitectura del sistema que genera los *triggers*. Por cada evento, la lógica del *trigger* genera un *trigger* de primer nivel (FLT). Por un lado, es una ventaja que las señales del detector se digitalicen directamente en los sistemas de adquisición de los detectores, ya que los datos digitalizados son mucho más fáciles de manejar que los analógicos. Por otro lado, en un sistema de adquisición pipelined como el del experimento COMPASS, cada evento tiene que ser identificado sin ambigüedad en las etapas iniciales del sistema de modo de poder identificar y manipular correctamente todos los datos de los detectores en

etapas posteriores. Esto significa que las señales de los *triggers* tienen que ser distribuidas a los distintos sistemas de adquisición, donde los datos digitalizados se acumulan y agrupan, tarea que desarrolla el sistema de control del *trigger* (TCS) [23, 24].

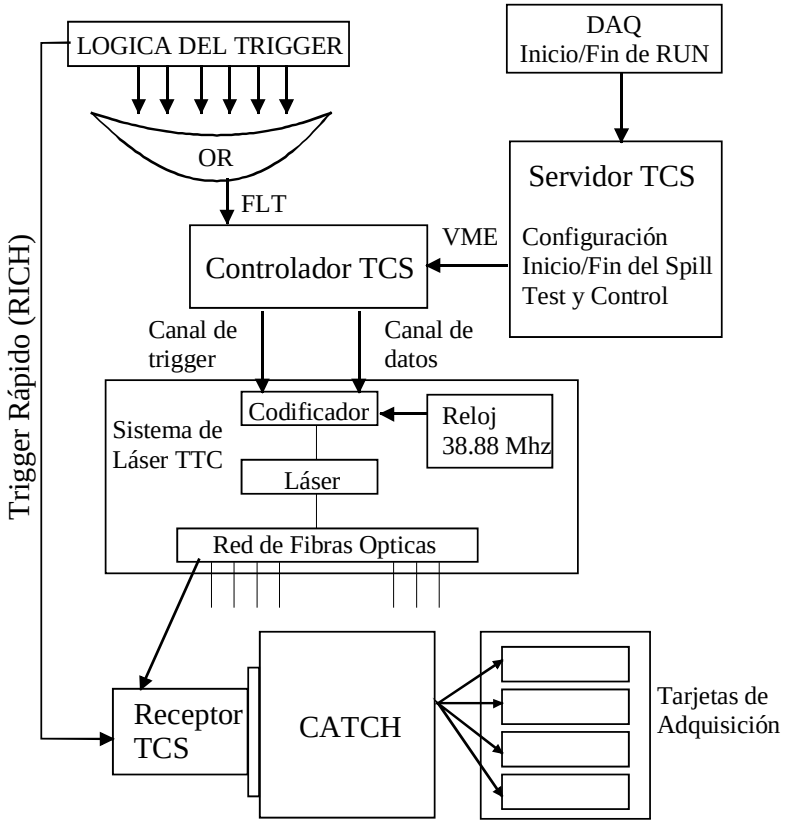

Figura 2.4 Arquitectura del sistema que genera los *triggers*.

El TCS consiste de un controlador TCS que genera un encabezamiento (*header*) de evento que contiene básicamente el número de evento, el número de *spill* y el tipo del evento. Cada *trigger* esta acompañado por el *header* de evento y ambos se transmiten a todos los módulos receptores TCS vía una red de fibras ópticas. La distribución se basa en una red de fibras ópticas pasiva con una única poderosa fuente de láser, modulada con la frecuencia del *Asynchronous Transfer Mode* (ATM, [25]) de 155,52 MHz correspondiente a cuatro veces la frecuencia del reloj del TCS de

38,88 MHz. El reloj del TCS se utiliza por una gran parte del experimento y es por lo tanto provisto a varias de las tarjetas de adquisición de los detectores. La información distribuida se recibe vía módulos receptores TCS conectados a los CATCH. Los receptores TCS preparan la información para los módulos CATCH, desde donde la señal de *trigger* se distribuye a las tarjetas de adquisición de los detectores.

2.2.1 El Controlador TCS

El controlador TCS es la parte central del sistema de control del *trigger*, su tarea principal es la de proveer las señales de *trigger* y de sincronización a todas las componentes. A través de sus conectores de entrada recibe el reloj desde el sistema de láser TTC [26], el *trigger* de primer nivel desde la lógica del *trigger*, y la información del *spill* desde el SPS. El controlador TCS sigue la estructura de *spill* SPS tal como muestra la Figura 2.5, donde las abreviaturas BOS y EOS representan las señales *comienzo de spill* y *final de spill*.

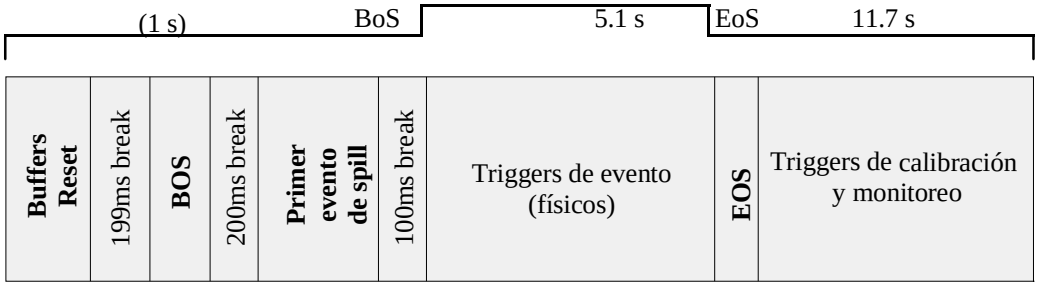

Figura 2.5 Estructura de *spill* del TCS.

La información del *spill* es usada por el controlador TCS para generar *triggers* artificiales que indiquen la estructura de *spill* a los equipos subsiguientes. El número de evento es re-iniciado al comienzo de un nuevo *spill* y simultáneamente el número de *spill* es incrementado en uno. Diferentes tipos de *triggers* son marcados por el tipo

del evento, que permite la distinción entre *triggers* físicos y artificiales tales como *primer-evento-de-run* o *triggers* de calibración. Antes del comienzo de un nuevo *spill*, se limpian todos los *buffers* involucrados en los sistemas de adquisición de los detectores y en los CATCH para garantizar la correcta adquisición de datos en el siguiente *spill*, especialmente si ocurrieron errores en el anterior.

Para distribuir el *header* de evento y la señal de *trigger* se usan dos canales independientes (canal de *trigger* y canal de datos), cada uno con un ancho de banda de 38,88 Mbaud. El controlador TCS sincroniza los *triggers* de primer nivel con el reloj de 38,88 MHz. Esto significa que las señales de *trigger* se transmiten con una precisión de 25,72 ns y por lo tanto deben medirse separadamente. Ellas se codifican exclusivamente sobre un canal de prioridad optimizado para mínima latencia. La información del *header* del evento se transmite vía el segundo canal. Los dos canales se operan independientemente, haciendo la latencia del *trigger* tan pequeña como sea posible distribuyendo las señales de *trigger* inmediatamente. En contraste, la generación del *header* del evento necesita más tiempo y se transmite después vía el segundo canal. El TCS codifica las señales de entrada a unos 77,76 Mbaud, las transfiere a la red de fibras ópticas y desde allí a los módulos receptores TCS.

Además de la distribución del *trigger* de primer nivel, el controlador TCS también determina el número medio de *triggers* generados por unidad de tiempo. Después de cada *trigger* existe una ventana de tiempo, en la cual no se permite procesar ningún nuevo *trigger*. Este “tiempo muerto fijo” se puede ajustar en el rango de 175 ns y 13,175 μs. Por otro lado, en algunas tarjetas de adquisición el tamaño de los *buffers* es el factor limitador y no pueden manejar más de un cierto número de *triggers* dentro de un período de tiempo dado. Este “tiempo muerto variable” puede estar en un rango de hasta 15 *triggers* en una ventana de tiempo seleccionable entre

25,72 ns y 1685,6 μs. El software del servidor TCS especifica, vía VME, ambos tiempos muertos (fijo y variable). Los valores estándar para los tiempos muertos del 2002 fueron 3 μs para el "tiempo muerto fijo" y cuatro *triggers* en 40 μs para el "tiempo muerto variable".

El servidor TCS también es responsable de la comunicación entre el TCS y el sistema DAQ, donde se define la estructura de *run*, y distribuye las señales de inicio y fin de *spill* recibidas desde el SPS.

2.2.2 El Receptor TCS

Cada uno de los módulos CATCH alberga un receptor TCS que decodifica y re-formatea la información proveniente, a través de fibra óptica, del controlador TCS. En el receptor TCS se almacena el *header* de evento, el cual se usa por el CATCH que lo recupera de una memoria del receptor administrada como cola (FIFO).

La mayor parte del experimento usa el *trigger* de primer nivel proveniente del controlador TCS. Una excepción es el sistema de adquisición del detector RICH-1, el más grande del experimento en cuanto a la dimensión y número de canales, el cual requiere que la señal de *trigger* llegue antes. Por lo tanto para el RICH-1 se utilizan *triggers* dedicados que llegan al receptor TCS directamente desde la lógica del *trigger*. Los mismos nos son procesados por el controlador TCS, por lo tanto no están sincronizados con el reloj del controlador.

2.3 Sistemas de Adquisición de Datos de los Detectores

La cadena de adquisición de datos del experimento comienza en los detectores. COMPASS cuenta con una variedad de detectores con características

diferentes. Por ejemplo, una propiedad que los distingue es el número de canales y en consecuencia la cantidad de datos generada. Por lo tanto, cada detector tiene su propio sistema de adquisición de datos, implementado a través de tarjetas de adquisición montadas sobre los mismos.

Típicamente la digitalización de las señales analógicas generadas por los detectores de COMPASS se lleva a cabo en las tarjetas de adquisición que albergan amplificadores, discriminadores y dispositivos de digitalización (ADCs o TDCs). La diferencia entre los sistemas de adquisición de los distintos detectores radica en los mecanismos usados para la digitalización de las señales, en el pre-procesamiento aplicado a los datos digitalizados, en los servicios provistos los cuales permiten analizar la calidad de los datos generados por el detector, en el tiempo muerto requerido entre *triggers* y en el medio usado para la comunicación entre las tarjetas de adquisición y los módulos CATCH.

El sistema de adquisición, tratamiento y control de datos del detector RICH-1, el cual cuenta con aproximadamente un tercio del conjunto total de canales, representa la parte principal de esta tesis y se describe detalladamente en los siguientes capítulos. Información detallada de los sistemas de adquisición de datos del resto de los detectores de COMPASS puede ser encontrada en [9, 18, 19, 20].

2.4 Los Módulos CATCH

Las principales tareas de los módulos CATCH (*Compass Accumulate Transfer and Control Hardware*) son la lectura de los datos provenientes de las tarjetas de adquisición de los detectores y la reconstrucción local de sub-eventos [9].

Para la adaptación con los distintos detectores se diseñaron cuatro tipos de tarjetas intercambiables denominadas *Common Mezanine Card* (CMC). Las CMC

actúan como interfase entre las tarjetas de adquisición y los dispositivos CATCH, y se encargan de proveer los datos al CATCH, a través de FIFOs, en un formato unificado. Información detallada de los cuatro tipos diferentes de CMC (*HOTLink*, *HOTFiber*, *TDC* y *Scaler*) puede ser encontrada en [9, 27, 28]. Para la comunicación con el sistema de adquisición de datos del detector RICH-1, se utilizan las *HOTFiber-CMC*. Los datos generados en cada tarjeta de adquisición se transmiten por medio de un chip transmisor *Hotlink* [29] y en cada CMC existen cuatro chips receptores *Hotlink* que reciben los flujos de datos provenientes de cuatro tarjetas de adquisición. Las salidas de cada receptor *Hotlink* (8 bits paralelos) se combinan en palabras de 32 bits y se almacenan en un FIFO desde donde el CATCH lee los datos.

La comunicación física entre la *HOTFiber-CMC* y la tarjeta de adquisición de datos del RICH-1 se realiza utilizando dos fibras ópticas, una para la recepción de datos y otra para la transmisión del *trigger*. La tarjeta de adquisición determina el *rate* de transmisión a 400 Mbaud.

Cada CATCH elabora los datos de cuatro CMC. La Figura 2.6 muestra un esquemático de la trayectoria del flujo de datos a través del CATCH. Por cada CMC existe una FPGA, denominada *Merger*, que combina los datos de todos los canales de entrada que pertenecen al mismo evento y realiza chequeos de consistencia de los datos recibidos. Los datos de los cuatro *Mergers*, correspondientes a 16 tarjetas de adquisición de alguno de los detectores, se combinan en una FPGA, denominada *Formatter*, donde se formatean de acuerdo al formato general de datos de COMPASS [21]. Errores de transmisión identificados por los *Mergers* o errores en los datos detectados por el *Formatter* se marcan con un código de error especial con el objetivo de reconstruir el problema desde los datos. El *Formatter* realiza reconstrucción de

sub-evento local de acuerdo al número de evento, y almacena los datos combinados de un evento en un FIFO de datos.

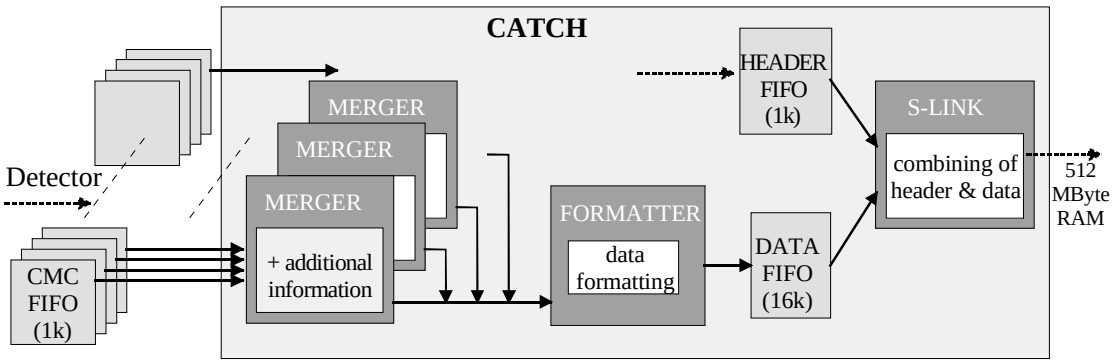

Figura 2.6 Trayectoria del flujo de datos a través del módulo CATCH.

Los datos generados por el CATCH se transmiten a las PCs ROB vía una conexión S-Link. El protocolo S-Link requiere un *header* y un *trailer* que identifiquen el inicio y el final del bloque de datos. El *header* contiene información específica del evento y el *trailer* consiste de una única palabra especial de 32 bits. El *header* S-Link, proveniente del receptor TCS se almacena en un FIFO (*Header FIFO*). Los datos de ambos FIFOs (*Header and Data FIFOs*) se combinan, en un único bloque, en una FPGA S-Link desde donde se transmiten hacia los ROBs.

Los eventos son procesados por cada señal de *trigger* proveniente del TCS. El CATCH transfiere directamente los *triggers* a las tarjetas de adquisición de datos de los detectores.

2.5 La Conexión S-Link

El protocolo S-Link [22] fue desarrollado para el experimento ATLAS del CERN como una conexión punto a punto veloz. En COMPASS, el S-Link se usa como conexión entre el módulo CATCH y las PCs ROB, como muestra

esquemáticamente la Figura 2.7. Existen dos versiones diferentes, la versión más simple sólo provee un canal de datos. En el experimento COMPASS, el enlace físico es una fibra óptica doble contando con un canal de ida y otro de retorno. Esta versión provee control del flujo a través de un bit especial (*lff*) que indica cuando el enlace está lleno y no se pueden manipular más datos. El protocolo también provee información de error. Los tres valores de ancho de banda disponibles (100, 128 y 160 MBytes/s) son usados. La frecuencia de reloj máxima del S-Link es de 40 MHz y es la utilizada puesto que coincide con el reloj interno del módulo CATCH. El protocolo S-Link maneja los datos desde FIFOs localizados en ambos extremos de la conexión.

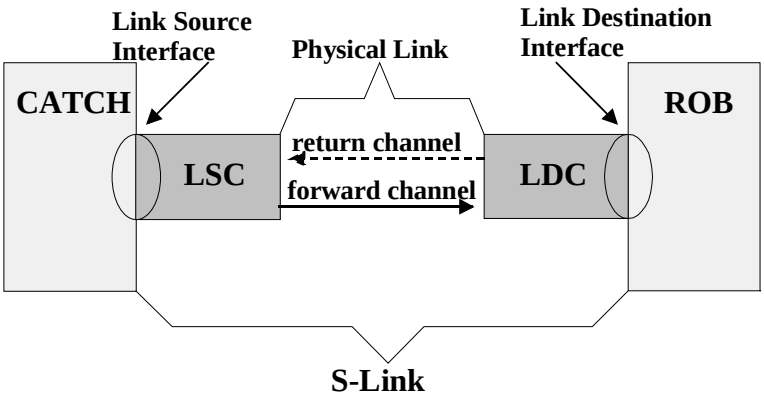

Figura 2.7 Conexión S-link entre el CATCH y el ROB.

La conexión física es una fibra óptica que transmite 32 bits de datos en paralelo acompañada de un bit de control (*ldown*) que indica una posible desconexión. Dado que los detectores y en consecuencia los módulos CATCH se distribuyen en todo el área experimental, los cables de fibra tienen una longitud de 50 a 100 m. Las tarjetas transmisoras S-Link (LSC) se conectan a los CATCH y las tarjetas receptoras S-Link (LDC) se instalan en las PCs ROB, donde los datos de entrada se almacenan en *spillbuffers*.

Debido a que a los módulos CATCH se comunican con tarjetas de adquisición de detectores diferentes, cada uno con características particulares, los flujos de datos generados son también diferentes. Para proveer una carga uniforme a las conexiones S-Link particulares y hacer un uso óptimo del ancho de banda disponible, los datos de hasta cuatro módulos CATCH se combinan en un único flujo de datos por medio de un multiplexor S-Link [9, 21]. La Figura 2.8 muestra un esquema de tal multiplexor.

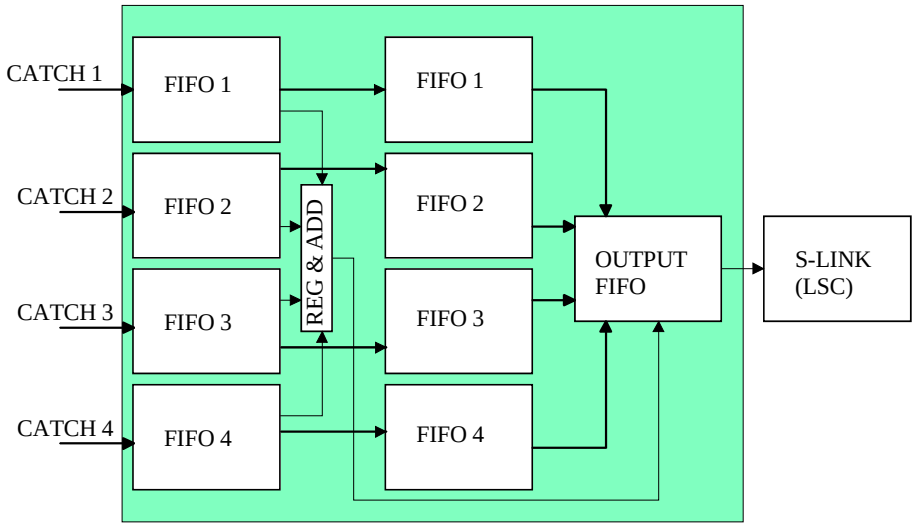

Figura 2.8 Esquemático del Multiplexor S-Link.

Los datos recibidos desde el CATCH se almacenan en FIFOs de entrada y posteriormente se escriben en un segundo conjunto de FIFOs tan pronto como éstos estén vacíos. Paralelamente, las dos primeras palabras de cada evento se almacenan en registros para extraer el número de evento y el tamaño de la información. La suma de los tamaños de eventos de todos los módulos CATCH conectados se incrementa en dos incorporando las dos palabras de encabezamiento generadas por el multiplexor, como muestra la Tabla 2.1.

La primer palabra del encabezamiento contiene el tamaño del evento total, el identificador de la fuente del multiplexor y el tipo del evento. La segunda palabra del

encabezamiento se toma de la segunda palabra del *header* S-Link almacenado en el CATCH, y contiene el número de evento y de *spill*. Esto se usa para controles de consistencia con la información de evento de los otros módulos CATCH conectados con este multiplexor. La primera y última palabra de cada evento se marcan como palabras de control y se necesitan para sincronizar el S-Link. La última palabra *CFED120* es una palabra única definida para indicar el fin de las transmisiones de eventos vía S-Link.

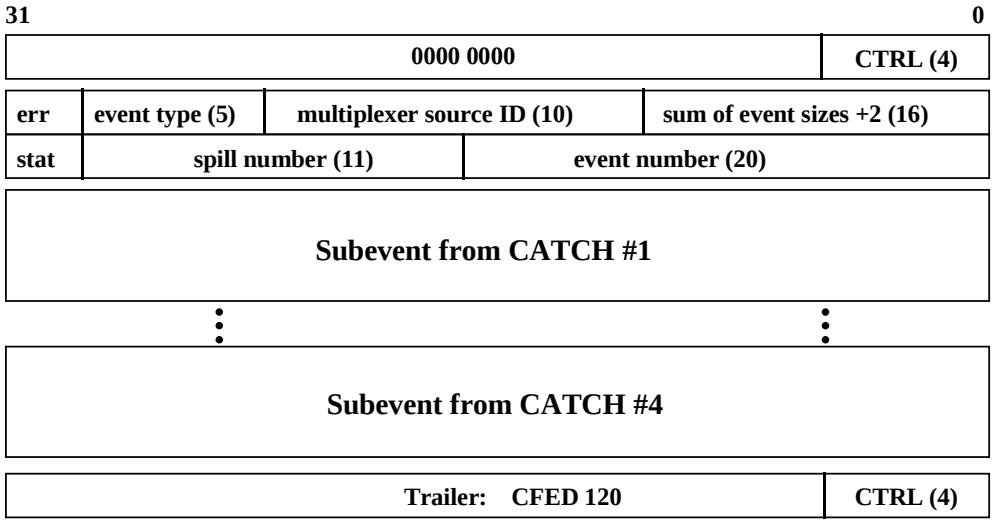

Tabla 2.1 Estructura de evento del Multiplexor S-link.

Dado que el protocolo de entrada del multiplexor S-Link es exactamente el mismo que el de salida, este sistema es transparente y flexible. Como el S-Link individual, el multiplexor genera la señal *lff* en el caso que sus FIFOs de entrada alcancen un estado de casi lleno y no sean capaces de manipular más datos. En el caso de una conexión faltante o un CATCH desprogramado, el multiplexor no espera por los datos ingresados a través de esas líneas. Si un CATCH activo no transmite datos por alguna razón, el multiplexor lo desactiva utilizando la señal *ldown*. La parte de salida del multiplexor cuenta con un FIFO a través del cual el multiplexor se comunica con un LSC como el utilizado en la conexión S-Link individual.

2.6 Las PCs ROB

Los ROBs (*ReadOut Buffer*) [9] son PCs comerciales que cuentan con dos procesadores con sistema operativo Linux. Cada ROB almacena cuatro tarjetas *spillbuffer* conectadas a sus *slots* PCI y cada tarjeta *spillbuffer* tiene una conexión *piggiback* a la tarjeta S-Link por donde ingresan los datos. Los datos de entrada son temporalmente almacenados en un FIFO antes de ser almacenados en un módulo de memoria RAM de 512 MByte, como ilustra la Figura 2.9. Los *spillbuffers* pueden almacenar datos provenientes de más de un *spill*. El almacenamiento temporal de los datos en una cola permite usar eficientemente la pausa entre *spills* originada por el ciclo del acelerador SPS. Esto reduce el elevado *data rate* en un factor de más de tres.

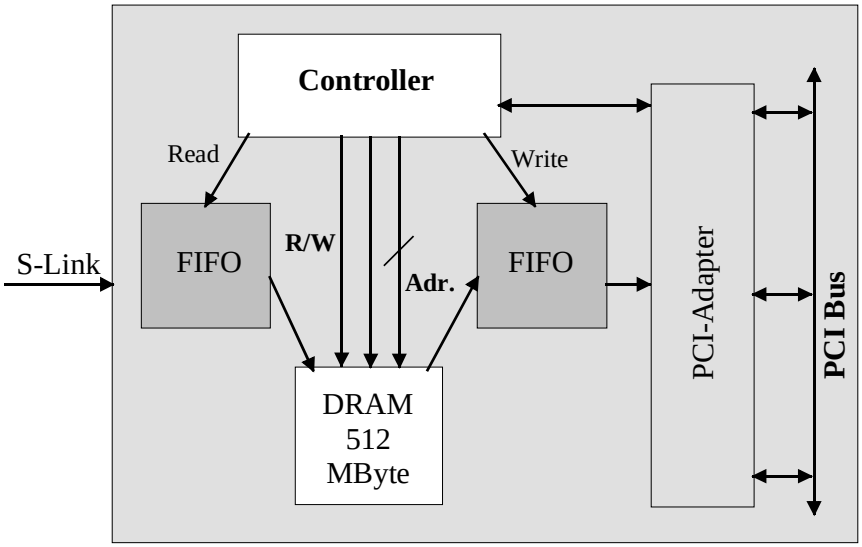

Figura 2.9 Esquemático de la tarjeta *spillbuffer*.

Los *spillbuffer* se diseñaron como tarjetas PCI [30]. La lectura de los datos desde la memoria RAM se realiza a través de un segundo FIFO. Los módulos de memoria son *dual port* lo que implica que pueden ser leídas y escritas simultáneamente. El software “Unificador de Datos Locales” que se ejecuta en todos

los ROBs es responsable de combinar los datos que pertenecen a un mismo evento desde los cuatro *spillbuffer* [21]. Una vez que se verifica la consistencia de los datos procesados, éstos se transfieren a las PCs constructoras de eventos vía *Gigabit Ethernet*.

2.7 Los Constructores de Eventos

Los constructores de eventos son PC similares a los ROB. Cada PC constructora de evento [9] toma todos los fragmentos de datos correspondientes a un evento desde los diferentes ROBs y los une en un único bloque de datos de evento. Los ROBs transmiten los eventos a todos los constructores de eventos en turno. En el 2002 se usaron 16 ROBs y 12 constructores de eventos como muestra la Figura 2.10.

Las PCs constructoras de eventos realizan la construcción del evento final, dando a los datos el formato mostrado en la parte derecha de la Figura 2.11 [21]. Los eventos finales se combinan en archivos de 1 Gbyte, los cuales se almacenan en los discos locales de los constructores de eventos, desde donde se transfieren a la *COMPASS Computing Farm* vía los mismos *Gigabit Switches* a través de los cuales los datos llegaron usando el protocolo *Remote File I/O (RFIO)* [31] *on top of TCP/IP*. Las conexiones entre los ROBs, las PCs constructoras de eventos y el registro de datos central son *Network Switches*.

2.8 La Estructura de Datos de DATE

Los datos registrados por el experimento COMPASS tienen el formato establecido por la estructura general del *Data Acquisition Test Environment* (DATE). DATE es un software de adquisición de datos desarrollado para el experimento

ALICE [32]. Su estructura y el formato de los datos se optimizaron para un sistema de adquisición que concentra y almacena los datos utilizando PCs de adquisición como los ROBs y las PCs constructoras de eventos interconectadas por una *switching network*. Por lo tanto, COMPASS adoptó este software y lo modificó para sus necesidades específicas. En la Figura 2.11 se esboza el formato de los datos [21].

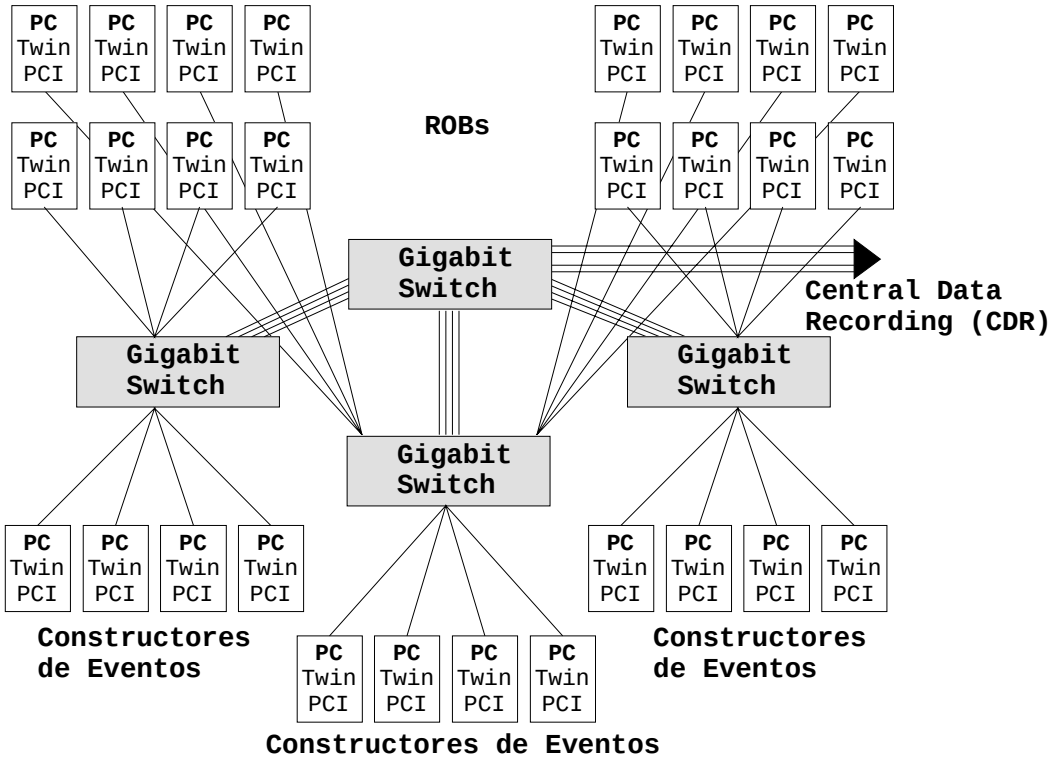

Figura 2.10 Sistema de construcción de eventos correspondiente al año 2002 que consiste de 16 ROBs y 12 constructores de evento.

Los datos de entrada son los datos pre-procesados transferidos desde los módulos CATCH a las tarjetas *spillbuffer*. Los datos de todas las tarjetas *spillbuffer* se combinan en los ROBs formando bloques de datos correspondientes a sub-detectores precedidos por un encabezamiento que contiene información acerca del tipo del evento, usualmente eventos físicos, el número de palabras del encabezamiento y del sub-evento completo, y un identificador (ID) de detector, el cual es único para cada ROB. Esto lo hace el software “Unificador de Datos Locales” que se ejecuta en los

ROBs. La longitud del sub-evento permite saltar directamente a la información correspondiente a cualquiera de los sub-detectores en el evento final, como lo indican las flechas en la parte derecha de la Figura 2.11.

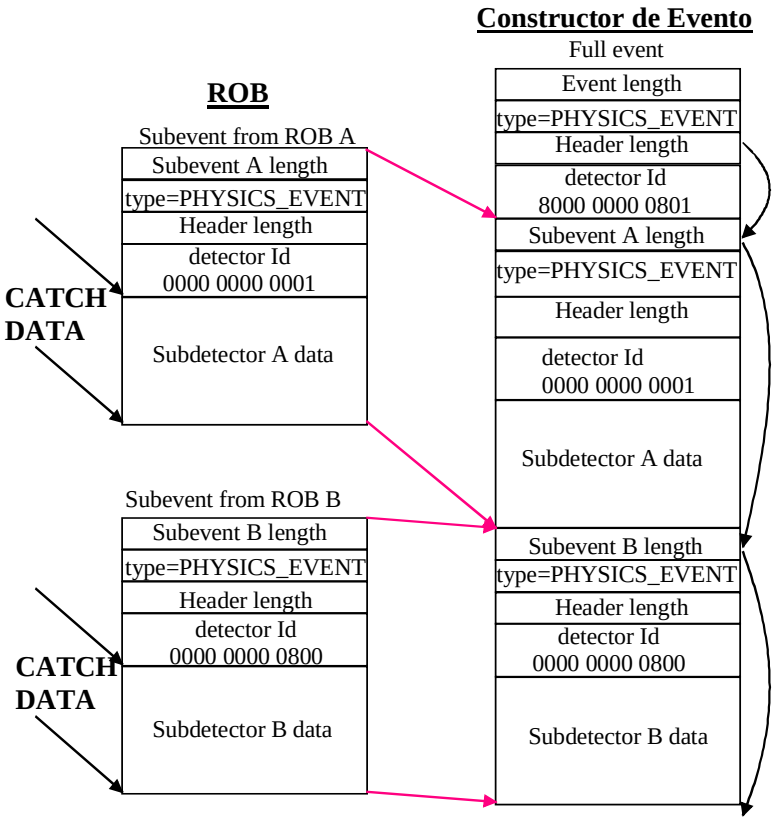

Figura 2.11 Estructura de datos de un evento DATE.

Normalmente, varios ROBs transmiten sus datos a las PCs constructoras de eventos, como se muestra en la Figura 2.10. Los datos pertenecientes al mismo evento son transferidos al mismo constructor. Esto se hace distribuyendo equitativamente los eventos entre todos los constructores de eventos conectados, de acuerdo a un modelo *round-robin*. La construcción del evento final se realiza generando un encabezamiento global y combinando los sub-eventos de todos los ROBs. El identificador (ID) del detector en el encabezamiento global es un *or* a nivel de bits entre todos los IDs de detector de los sub-eventos.

El software DATE no solo es responsable del flujo explicado y del control del *run*, también provee la contabilidad de la cantidad de datos registrados y el número de *run*, mostrando además mensajes de error y de estado. DATE soporta también testeos *on-line* de una fracción de los eventos desde el flujo de datos principal, permitiendo verificar la calidad de los datos adquiridos; procedimiento denominado monitoreo.

2.9 Transferencia de Datos al Registro de Datos Central

COMPASS usa el servicio del registro de datos central (CDR) del CERN para registrar los datos [33, 34, 35, 36]. El sistema DAQ transmite los eventos, para ser almacenados en cinta, al centro de cálculo del CERN situado a pocos kilómetros del área experimental, vía una red dedicada. Allí se localizan la *COMPASS Computing Farm* (CCF) y los servidores de cinta, como muestra la Figura 2.12.

Los archivos de datos que arriban desde las PCs constructoras de eventos se almacenan localmente en los discos de los servidores de datos (*Data Server*), desde donde se transfieren a las cintas. Los 260 TBytes almacenados durante el año 2002, cantidad que será incrementada hasta alcanzar los 300 TBytes, no pueden estar disponibles en disco al mismo tiempo. La limitación del espacio en disco se minimiza por un administrador de almacenamiento jerárquico, el cual se denomina *Cern Advanced STORage* (CASTOR) [37]. La reconstrucción de los datos se realiza usando el programa de reconstrucción y análisis CORAL de COMPASS [38]. La CCF provee la potencia computacional para realizar esta tarea a través de 200 CPUs.

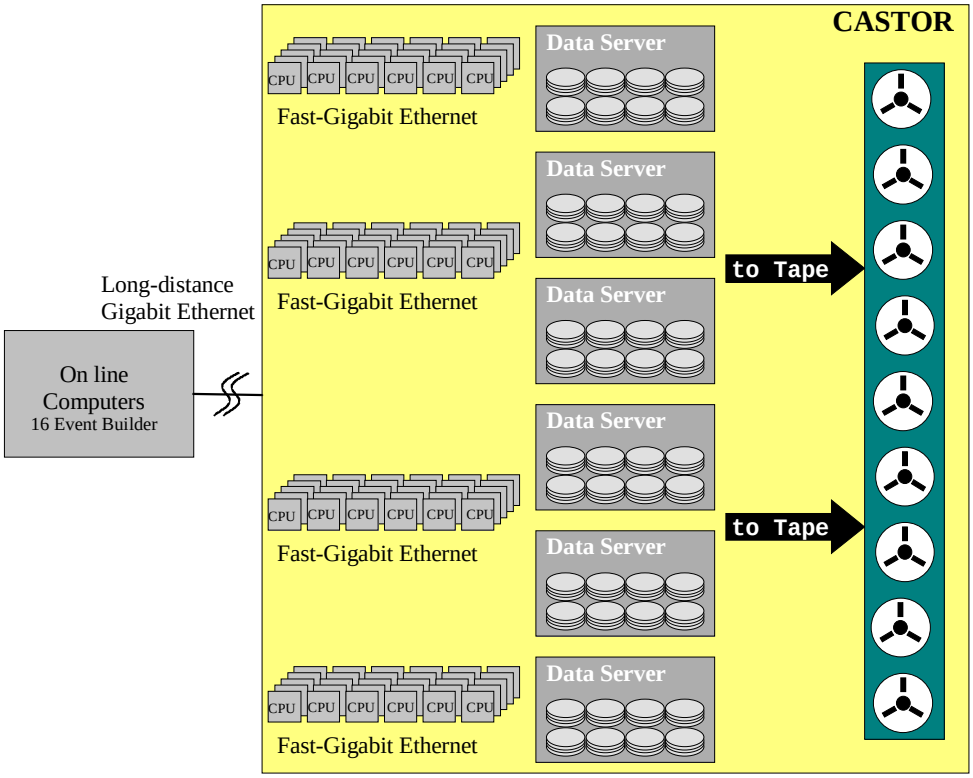

Figura 2.12 Estructura de la *COMPASS Computing Farm* (CCF).

Capítulo 3

El Sistema de Adquisición de Datos del Detector RICH-1

En la cadena de adquisición de datos de COMPASS, cada uno de los diferentes detectores que conforma la estructura instrumental del experimento cuenta con su propio sistema de adquisición. Aproximadamente un tercio de los canales que deben ser adquiridos por todo el experimento pertenecen al detector RICH-1. La enorme cantidad de datos que debe ser adquirida, procesada y transmitida por el sistema de adquisición del detector, a un ritmo de *triggers* asíncronos que pueden alcanzar los 100 kHz, exigieron el diseño y la implementación de una solución nueva y original para el mismo.

Considerando que el detector cuenta con 82944 canales distribuidos en una superficie de 5,3 m^2, y que sus valores se digitalizan en 10 bits, el volumen de datos (*Vol1*) que deben ser adquiridos y tratados puede alcanzar los 10,368 GBytes/s.

$$Vol1 = (canales\,adquiridos \times bytes\,por\,canal) \times triggers/s$$

La gran cantidad de datos adquiridos hace necesaria la reducción de los mismos antes de incorporarlos al flujo principal. Por lo tanto, a la adquisición se le suma una pre-elaboración y filtrado de los datos en tiempo real. Resultados de simulaciones en las que se estudió estadísticamente la aplicación de métodos de

selección sobre los datos digitalizados, determinaron que por cada *trigger* hasta el 20% de los canales adquiridos pueden resultar significativos. Considerando además que de acuerdo con el formato de datos general del experimento [21], la información asociada con cada canal significativo del RICH-1 debe ser codificada en 32 bits (los cuales incluyen el valor y el identificador del canal) y empaquetada en paquetes de evento delimitados por un *header* y un *trailer*, el volumen de información (*Vol2*) a ser transmitida hacia el resto de la cadena de adquisición del experimento puede alcanzar los 6,636 GBytes/s.

$$Vol2 = \left(canales\,transmitidos \times \left(\text{bytes por canal} + \text{header} + \text{trailer}\right)\right) \times triggers/s$$

Manipular tales cantidades de información por unidad de tiempo representa, al presente estado del arte, un desafío para el área de ciencias de la computación y es el origen de esta tesis. En este capítulo se describen los problemas asociados con las exigencias del sistema de adquisición de datos del detector RICH-1 de COMPASS y las soluciones que se adoptaron para satisfacerlas.

3.1 Requerimientos y Características Generales

De acuerdo con los requerimientos y características de la cadena de adquisición global del experimento, el sistema de adquisición de datos del detector RICH-1 debe cumplir con ciertos requisitos básicos que se describen a continuación.

La adquisición analógica de cada uno de los 82944 canales de los detectores de fotones del RICH-1 así como también su digitalización en 10 bits implica la necesidad de proveer y controlar las señales requeridas por los dispositivos que realizan tales funciones.

El volumen de datos adquiridos (*Vol1*) hace necesaria la aplicación de algún criterio de selección a los datos digitalizados de modo de transmitir al sistema de adquisición de datos global sólo los canales que contengan datos significativos. Los datos significativos son aquellos que presumiblemente corresponden a los canales alcanzados por fotones *Cerenkov* emitidos por el pasaje de las partículas incidentes a través del radiador gaseoso del detector (Sección 1.4). Una pequeña fracción del total de los canales contiene información significativa y sólo estos datos deben ser transmitidos por el sistema de adquisición de datos del detector. A estos canales se suman aquellos que aún no conteniendo información significativa, superan un cierto umbral debido a fluctuaciones estadísticas del ruido y que no pueden descartarse a priori por ser indistinguibles de aquellos datos significativos. Sólo una compleja elaboración *offline* podrá distinguir datos significativos del ruido, basándose en algoritmos de *pattern recognition*.

Desde el punto de vista de la adquisición de datos, los detectores de fotones del RICH-1 pueden verse como una matriz cuadrada de 288 x 288 *pixels* donde a cada *pixel* le corresponde una posición geográfica (x,y). Por lo tanto, al valor digital del canal se le debe incorporar un identificador que permita determinar unívocamente la posición geográfica del canal. De este modo, en la fase posterior de análisis de los datos, podrán ser identificados por cada evento[1] los canales conteniendo información significativa. La información asociada con cada canal significativo (valor digital e identificador) debe ser codificada en 32 bits y empaquetada en paquetes de evento. Para la transmisión de los paquetes de eventos al sistema global de adquisición de datos, se utiliza el protocolo de comunicación *HOTFiber-CMC* (Sección 2.4).

[1] Denominamos evento al conjunto de canales transmitidos al sistema global de adquisición de datos correspondientes a cada *trigger*.

Además de las actividades específicas relacionadas con la adquisición de datos durante el *run*, la complejidad del detector RICH-1 el cual es un instrumento único diseñado específicamente para el experimento COMPASS, requiere de mecanismos que permitan estudiar y analizar el comportamiento de cada una de las señales que genera en manera individual. Dicho estudio se debe realizar autónomamente respecto de la cadena de adquisición global del experimento así como debe efectuarse en modo autónomo la preparación del sistema para la adquisición de datos (inicialización, configuración y programación de sus componentes) y el control del funcionamiento del sistema durante el tiempo de vida del experimento. Tales actividades deben además ser efectuadas en forma remota puesto que la radioactividad producida por el haz de partículas hace inaccesible el área experimental donde se encuentran los detectores y sus tarjetas de adquisición de datos.

Uno de los requerimientos fundamentales es la flexibilidad en cuanto a que el sistema debe poder adaptarse a posibles nuevas condiciones experimentales que puedan surgir durante los años de vida del experimento.

El gran número de canales del detector y la velocidad en la que los mismos deber ser adquiridos, elaborados y transmitidos exigen la distribución del sistema de adquisición en subsistemas más pequeños que trabajen en paralelo. En consecuencia aparece el problema de la sincronización y el control de los mismos. Cada uno de tales subsistemas debe recibir y responder contemporáneamente y en tiempo real a las señales de *trigger* y a las señales de inicio y fin de *run* y de *spill*, las cuales delimitan las etapas de la adquisición de datos.

Considerando los requisitos mencionados anteriormente, el sistema de adquisición de datos del detector RICH-1 se puede dividir, desde el punto de vista de su funcionalidad, en los siguientes cuatro grandes aspectos:

- adquisición, tratamiento y transmisión de datos,
- estudio y calibración,
- control y monitoreo, y
- programación y configuración de sus componentes.

3.1.1 Adquisición, Tratamiento y Transmisión de Datos

La adquisición, tratamiento y transmisión de datos efectuada durante el *spill* es sumamente crítica en cuanto a la velocidad de procesamiento. En esta etapa el sistema debe responder en tiempo real a cada uno de los *triggers* generados asincrónicamente por el TCS, teniendo en cuenta que el *trigger rate* puede alcanzar los 100 kHz con un tiempo muerto mínimo entre *triggers* del orden del microsegundo.

Las restricciones temporales impuestas por el *trigger rate* definen al sistema de adquisición, tratamiento y transmisión de datos del RICH-1 como de "tiempo real duro" [39, 40, 43, 44]. En sistemas de "tiempo real duro", el incumplimiento de las restricciones de tiempo afecta directamente la utilidad de los resultados producidos y en consecuencia la performance total del sistema. En nuestro caso, la pérdida de un *trigger* o de algunas de las señales de sincronización (inicio y fin de *run* y de *spill*), o la demora en los tiempos de respuesta a un *trigger*, implicaría la desincronización con el sistema global de adquisición y la pérdida de los datos generados por el detector, haciendo inútiles, en consecuencia, los datos adquiridos por todo el experimento. La información proveniente de todos los detectores, y en particular del RICH-1 que cuenta con un tercio de los canales que deben ser adquiridos por el experimento, es esencial para los objetivos de la física de COMPASS.

Durante el *spill* en el *run*, el sistema responde a cada *trigger* adquiriendo y digitalizando en 10 bits las señales analógicas provenientes de cada uno de los *pixels*

del detector. Las señales provenientes del detector son amplificadas y filtradas analógicamente por un chip de *front-end* específicamente diseñado para el RICH-1 (Gassiplex [86, 87, 88]). Nos referiremos a la salida analógica de este chip como al valor del canal. Todos los canales se digitalizan en una escala única absoluta de 0 a 1023.

En ausencia de señal cada canal presenta ruido debido a múltiples fuentes (*termal noise*, *fliker noise*, *pickup noise*, etc.) y se asume que este ruido es gaussiano, blanco y estacionario [45, 46, 47, 48, 49, 50, 51]. Por ruido gaussiano se entiende que la amplitud del ruido es una variable aleatoria cuya función de densidad de probabilidad es gaussiana, por ruido blanco se entiende que todas las componentes de frecuencia están presentes con la misma intensidad (o bien que no hay ninguna componente de frecuencia que prevalece), y por ruido estacionario se entiende que sus características no varían en el tiempo.

Un ruido así descripto viene completamente caracterizado por dos parámetros: el valor medio μ y la desviación estándar σ de su distribución gaussiana. La señal de cada canal debe ser evaluada con respecto al ruido del canal correspondiente (ver Figura 3.1). La amplitud de la señal debe referirse al valor medio del canal correspondiente y el significado estadístico de esta señal debe confrontarse con la desviación estándar del ruido correspondiente. Así, por ejemplo, si tenemos una señal relativa mayor a un σ entonces la probabilidad aproximada de que se trate de una señal verdadera es mayor del 84%, y la probabilidad aproximada de que se trate de una fluctuación estocástica es menor del 16%.

Visto que pocos canales por evento tendrán señales significativas y que solo los datos de estos canales merecen ser transmitidos y almacenados, se impone analizar cada canal a fin de decidir si transmitir su valor o descartarlo. El análisis de cada canal

consiste en tomar el valor absoluto digitalizado y confrontarlo con un umbral elegido según algún criterio. Si el valor resultante no supera el umbral se descarta, caso contrario se transmite el resultado de la diferencia entre el valor del canal y el umbral. Normalmente el criterio de elección del umbral es un criterio estadístico. Si se quiere asegurar que la probabilidad de transmitir datos correspondientes a ruido sea menor que un valor dado entonces se fijará el umbral (T) como el valor medio (μ) más un factor (α) de la desviación estándar (σ) correspondiente:

$$T = \mu + \alpha\sigma$$

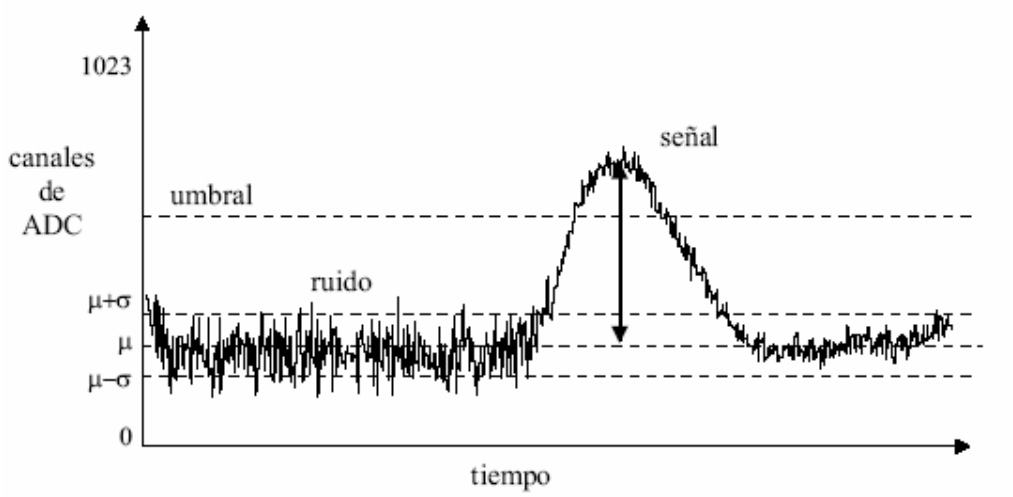

Figura 3.1 Señal característica a la salida del *Gassiplex* en función del tiempo. El valor medio estacionario de la salida del *Gassiplex* es μ, y su desviación estándar es σ. La amplitud relativa de la señal se mide con respecto a μ. El dato transmitido es la diferencia entre el valor absoluto de la señal y su umbral correspondiente.

Para nuestro modelo de ruido un factor cero implica que en media la mitad de los canales serán aceptados. El criterio de elección del umbral puede ser distinto para cada canal. Por ejemplo, si se quiere eliminar un canal de la transmisión, entonces se fijará su umbral al máximo de la escala, y si por el contrario se quiere incluirlo siempre, se fijará su umbral al mínimo. Con respecto al criterio estadístico es importante notar que también la señal esperada es una variable estocástica y tiene su

función de densidad de probabilidad, así un umbral alto tendrá una baja probabilidad de aceptar ruido pero también será baja la probabilidad de aceptar una señal autentica. La elección final será un compromiso entre diferentes factores y dependerá de las particulares condiciones y planes experimentales.

Por cada *trigger* durante la adquisición de datos, los valores (10 bits) de los canales que superan su umbral junto con el identificador del canal (18 bits) se codifican en palabras de datos de 32 bits, los cuatro bits restantes se utilizan para sincronización con el CATCH. Con tales datos se arma un paquete de evento y se transmite hacia los módulos CATCH. Puesto que la longitud del paquete de evento es variable, dos palabras especiales de 32 bits cada una: *header* y *trailer*, indican el inicio y el fin del paquete respectivamente, y contienen el número del evento (20 bits) y la fuente desde donde los datos son transmitidos (8 bits); de nuevo los cuatro bits restantes se utilizan para sincronización con el CATCH.

Las actividades involucradas en la adquisición de datos deben ser efectuadas a altísima velocidad por lo que requieren algún tipo de procesamiento paralelo y *pipelined*, el uso de métodos de software de tiempo real y lenguajes de programación de bajo nivel.

3.1.2 Estudio y Calibración

A fin de determinar el umbral para cada canal es necesario caracterizar todos y cada uno de los canales. Esencialmente se requiere conocer el valor medio y la desviación estándar de cada canal en ausencia de señal. Para esto se procede a tomar un cierto número de muestras independientes cuando el experimento no está en *run* y hacer la estadística correspondiente. El valor medio y la desviación estándar de las muestras serán las estimaciones del valor medio y desviación estándar

correspondientes a cada canal en ausencia de señal. Cuanto más muestras se tomen menor será el error estadístico de estimación de estos parámetros.

La característica de los canales en ausencia de señal puede variar como consecuencia de diversos factores (temperatura, ruido electromagnético ambiental, etc.), por lo tanto es importante poder recaracterizar los canales y redefinir sus umbrales en cualquier momento que se considere necesario.

3.1.3 Control y Monitoreo

Durante el tiempo de vida del experimento se deben proveer mecanismos para controlar y monitorear el funcionamiento del sistema de adquisición del detector en manera independiente del sistema de adquisición global del experimento. Tales mecanismos incluyen el control y monitoreo del funcionamiento de cada uno de los dispositivos involucrados en el sistema, de la información que permite determinar si el sistema está correctamente sincronizado con el sistema de adquisición global (número de *run*, número de *spills* por *run* y número de eventos por *spill*), de las características de un evento tomado como muestra de cada *spill*, de los valores medios y desviación estándar de cada canal generalmente utilizados para determinar los valores de umbral, y de los valores corrientes de umbral para cada canal.

3.1.4 Programación y Configuración

La programación y configuración de los dispositivos involucrados así como también el establecimiento de los valores iniciales, se efectúa cada vez después del *power-up* del sistema y puede llevarse a cabo en cualquier momento que sea solicitado.

La flexibilidad requerida, en modo que el sistema pueda adaptarse a eventuales cambios que puedan surgir durante el tiempo de vida del experimento, implica el uso de dispositivos reconfigurables o reprogramables por software.

3.2 Arquitectura Global del Sistema

En esta sección se presenta una descripción general de la arquitectura global del sistema de adquisición, tratamiento y control de los datos generados por el detector RICH-1. El servicio de *trigger* a la velocidad requerida es posible adoptando un sistema masivamente paralelo y *pipelined*, en el cual los datos de cada *trigger* son procesados y transmitidos al resto de la cadena de adquisición mientras otros *triggers* pueden ser aceptados. En distintas etapas los datos son almacenados temporalmente en memorias administradas como colas (FIFOs) para minimizar el tiempo muerto y evitar la pérdida de información.

La distribución de los 82944 canales del detector en una superficie de 5,3 m^2, la cantidad de información generada y la velocidad en la que los datos generados deben ser tratados, hace necesaria la división del sistema de adquisición en subsistemas que adquieran y procesen en paralelo un subconjunto del total de canales del detector. Siguiendo entonces el concepto de "divide y vencerás", se diseño la tarjeta Bora[2] [82] para la ejecución de las principales tarea de adquisición de los datos generados por 432 de los canales pertenecientes al RICH-1. Por lo tanto, para adquirir el total de los canales del detector se utilizan 192 tarjetas Bora. Bora fue diseñada en el Laboratorio de Microprocesadores del INFN-ICTP de Trieste (Italia), donde esta tesis se desarrolló.

[2] Bora es el nombre de un fuerte viento característico de la ciudad de Trieste (Italia), que alcanza velocidades de hasta 140 km por hora.

Las tarjetas de adquisición se montan en la parte externa de los fotocátodos del detector. Por cada una de las ocho cámaras de fotones del RICH-1, existen dos fotocátodos compuestos externamente por PCBs (Sección 1.4). Cada PCB cuenta con conectores a través de los cuales se pueden leer las señales analógicas provenientes de 48 canales del detector. La tarjeta Bora tiene nueve conectores y permite entonces la adquisición de las señales correspondientes a 432 canales. La Figura 3.2 muestra un PCB en manera esquemática, en la que se puede apreciar que 24 tarjetas Bora conforman una cámara del detector.

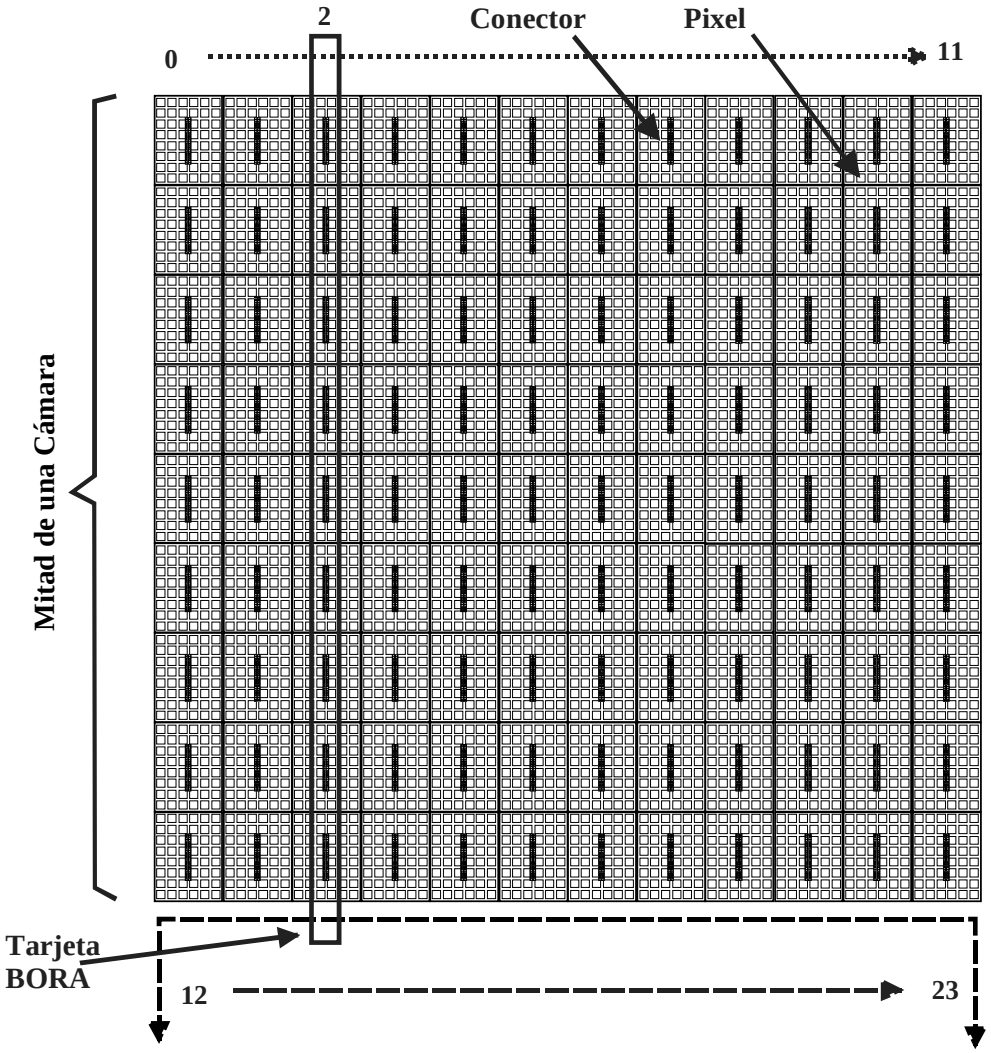

Figura 3.2 Parte externa de un fotocátodo correspondiente a la mitad de una cámara del RICH-1.

El sistema de adquisición, tratamiento, control y transmisión de datos del detector RICH-1 se basa en 192 tarjetas Bora conectadas a los canales del detector [83]. La Figura 3.3 muestra un esquema de la arquitectura global del sistema.

Las funciones principales de la adquisición de datos se realizan en las tarjetas Bora. Cada Bora cuenta con un Procesador de Señales Digitales (DSP), responsable del funcionamiento de la tarjeta, y un dispositivo de lógica programable *Field Programmable Gate Array* (FPGA) que actúa como coprocesador paralelo del DSP. El capítulo 5 describe en forma detallada la arquitectura de la tarjeta Bora.

Los DSP [52, 53, 54, 55, 56, 57, 58, 59, 60] son procesadores optimizados para el procesamiento de señales digitales en tiempo real. Los DSP de nueva generación cuentan además con el hardware para conectarlos en red y con una memoria interna suficiente para implementar en software el protocolo de administración y comunicación de la red. Interrupciones externas con alta prioridad prefijada, *buffers* circulares implementados en hardware y controladores de DMA paralelos permiten la adecuada manipulación de las señales externas (*trigger*, inicio de *spill*, fin de *spill*) y de los datos que deben tratarse en tiempo real a alta velocidad.

Durante la adquisición, elaboración y transmisión de datos, el uso de una FPGA como coprocesador paralelo del DSP permite el servicio de los *triggers* a la velocidad requerida. Bajo la supervisión del DSP, la FPGA responde a nuevos *triggers* controlando la adquisición, digitalización y almacenamiento temporal en memorias FIFO de los datos provenientes del detector, mientras se procesan y transmiten a la cadena de adquisición global datos previamente almacenados que fueron generados por *triggers* recibidos anteriormente.

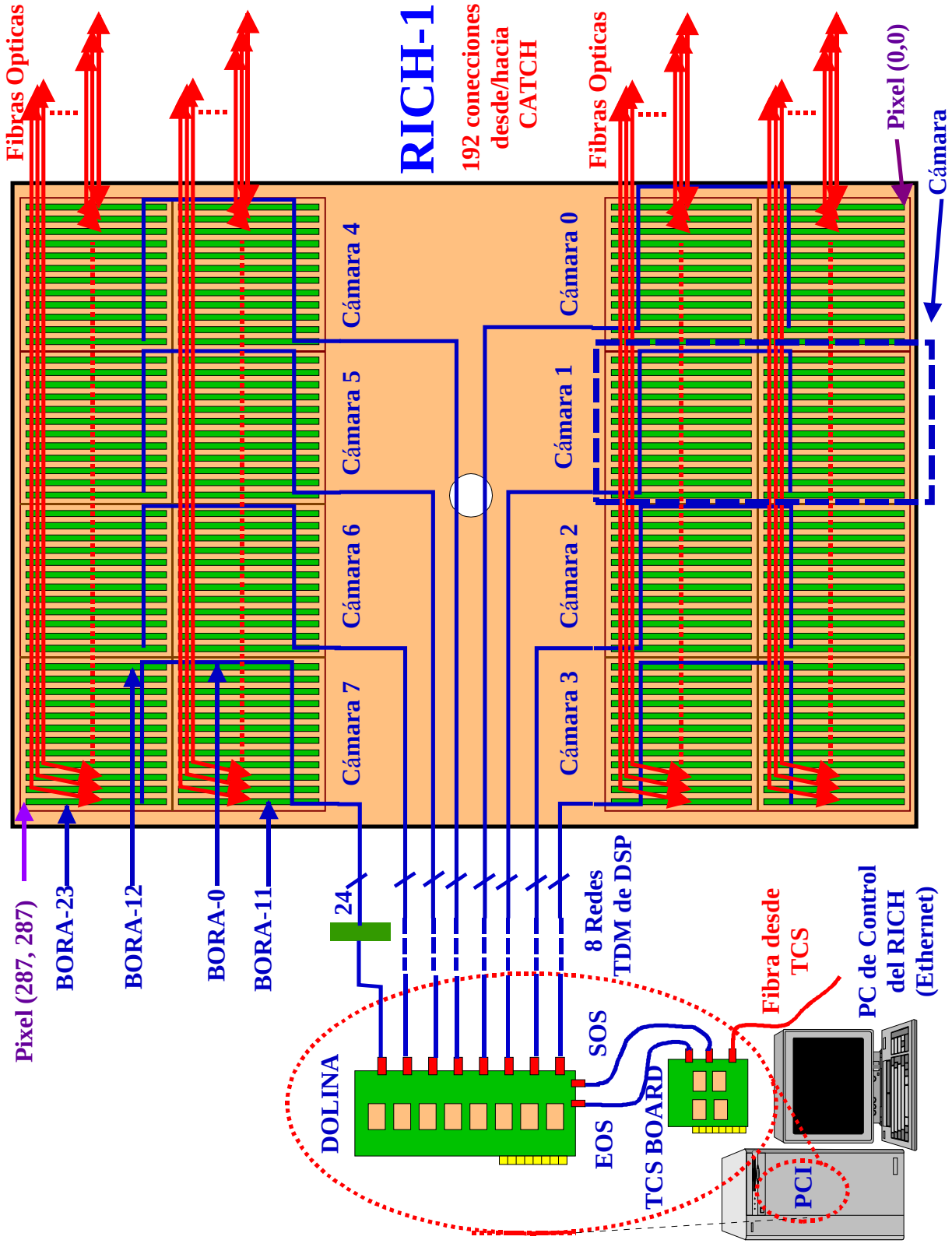

Figura 3.3 Arquitectura global del sistema de adquisición, tratamiento y control de datos del detector RICH-1 de COMPASS.

Las FPGA [61, 62, 63, 64, 65, 66, 67] son dispositivos de lógica programable formados por arreglos de bloques lógicos configurables e interconectables. Además de su inherente reconfigurabilidad, que las hace ideales para ser usadas en sistemas de computación reconfigurables [68, 69, 70, 71, 72, 73, 74, 75], las FPGA se programan utilizando lenguajes específicos de descripción de hardware (VHDL [76, 77, 78, 79, 80], Verilog [81]), y su uso es adecuado en "sistemas de tiempo real duros" por su inherente paralelismo en hardware.

La comunicación de la tarjeta Bora con el exterior se efectúa por medio de dos fibras ópticas y una red dedicada de DSP. Las fibras ópticas son usadas para la recepción del *trigger* proveniente del CATCH, y para la transmisión de los paquetes de evento el resto de la cadena de adquisición global del experimento. La red de DSP se usa principalmente en las actividades en las cuales la velocidad de transmisión no constituye un factor crítico.

A través de la red de DSP se lleva a cabo la programación, configuración e inicialización del sistema, el control y monitoreo del funcionamiento del mismo, y el estudio y calibración de las señales generadas por el detector. La red de DSP permite también la reprogramación y reconfiguración del sistema en cualquier momento que sea requerido.

El sistema cuenta con ocho redes de DSP, una por cada cámara del detector. Cada red está formada por 24 DSP correspondientes a las tarjetas Bora de una cámara y un DSP extra ubicado en una tarjeta multiprocesador, llamada Dolina[3], que contiene ocho DSP del mismo tipo que los de Bora. Como se puede apreciar en la Figura 3.3, Dolina se conecta al *bus* PCI de la PC de control del detector. Como Bora, la tarjeta Dolina fue diseñada en el Laboratorio de Microprocesadores del INFN-ICTP de

[3] Se denomina Dolina a una concavidad del suelo, formada por la acción del agua pluvial, típica de los terrenos cársicos.

Trieste (Italia), donde esta tesis se desarrolló. El capítulo 4 describe en forma detallada la funcionalidad y arquitectura de Dolina.

Las principales funciones de la tarjeta Dolina son la administración y control de las redes de DSP, y la comunicación entre la PC de control y tales redes. En la PC corre una aplicación de alto nivel, denominada "Controlador del RICH", desde la cual se puede controlar y monitorear en todo momento el funcionamiento del sistema. Durante el proceso de programación y configuración del sistema se envían desde la PC los programas correspondientes a cada uno de los DSPs y FPGAs que conforman el sistema.

Las señales de sincronización que determinan el inicio y fin del *run*, y el inicio y fin de cada *spill* en el *run* arriban a las Bora a través de la red de DSP. Las señales de inicio de *run* (SOR) y de fin de *run* (EOR) llegan a la PC directamente desde la sala "DAQ Control" por medio de la red de área local del CERN, y desde la PC son pasadas a Dolina que las transmite hacia todas las Bora. Las señales de inicio de *spill* (SOS) y de fin de *spill* (EOS) llegan directamente a Dolina desde una tarjeta TCS, Dolina cuenta con 200 milisegundos para transmitirlas hacia todas las Bora.

Durante la adquisición de datos en tiempo de *run*, las Bora envían hacia la PC información acerca del estado del sistema por cada *spill*. Datos de un evento por *spill*, número del *spill* corriente en el *run*, número de *triggers* por *spill* e información acerca de voltajes y temperaturas de las tarjetas de adquisición pueden ser chequeados en todo momento desde el "Controlador del RICH".

Antes del inicio del primer *spill* del *run* y cada vez que sea requerido desde la PC fuera del *run*, las Bora calculan los valores medios y desviación estándar de cada canal en ausencia de señal. Tales datos se transmiten hacia la PC en los casos que se haya solicitado, y hacia el sistema de adquisición global, junto con los valores

de umbrales corrientes e información del *run* precedente (número de *spills* en el *run* y número de *triggers* por *spill*), sincrónicamente con el primer *trigger* del primer *spill* del *run*. Desde la PC también pueden establecerse, modificarse y leerse los valores de umbrales de cada Bora.

3.3 Redes de DSP

El sistema cuenta con ocho redes de DSP dedicadas funcionando en paralelo. Cada red está conformada por 24 DSP correspondientes a las tarjetas Bora de una cámara y uno de los ocho DSP de Dolina.

Los DSP contienen el hardware para implementar, a través de uno de los puertos seriales del procesador, una red *Time Division Multiplexed* (TDM). En una red TDM [84, 85] los procesadores que conforman la red comparten la misma línea de comunicación y cada uno de ellos dispone de la capacidad total del medio de transmisión durante periodos de tiempo fijos, denominados *slots*. Los datos transmitidos se organizan en bloques, denominados *frames*, que contienen una secuencia de *slots*, como muestra esquemáticamente la Figura 3.4.

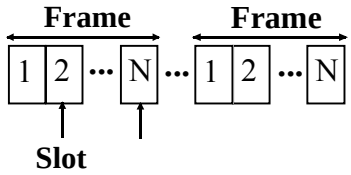

Figura 3.4 Organización de los datos transmitidos en una red TDM.

A cada procesador que forma parte de la red se le asigna uno o más *slots*, y durante el tiempo correspondiente a tales *slots* el procesador dispone de todo el medio de transmisión independientemente de que el *slot* contenga o no datos. En

consecuencia existe un desperdicio de la capacidad de la línea de comunicación, pero permite simplicidad de implementación y tiempos determinísticos de servicio.

Cada una de las ocho redes del RICH-1 está conformada por 25 DSP donde a cada DSP se le asigna uno o más *slots*. Durante el tiempo de *slot*, el procesador correspondiente puede transmitir y/o recibir una palabra de 32 bits. La comunicación sigue el modelo *master/slave* donde el DSP de Dolina es el *master* de la red y los DSP de Bora son los *slave*s. El *master* provee las señales que determinan el inicio del *frame* y el reloj a través de los cuales el receptor y el transmisor se sincronizan. En el *master* también se determina la velocidad de la red TDM que puede ir desde 1 Mbit/s hasta 60 Mbits/s dependiendo de la longitud del cable de comunicación. En nuestro caso, la PC que alberga la tarjeta Dolina se encuentra fuera del área experimental por lo que la longitud del cable que comunica Dolina con las tarjetas Bora, montadas en el detector, es de aproximadamente 30 metros y la velocidad de la red TDM se definió en 1,33 Mbits/s. La Figura 3.5 muestra la conexión física entre los DSP de una red que se establece a través de uno de los puertos seriales del procesador denominado SPORT0.

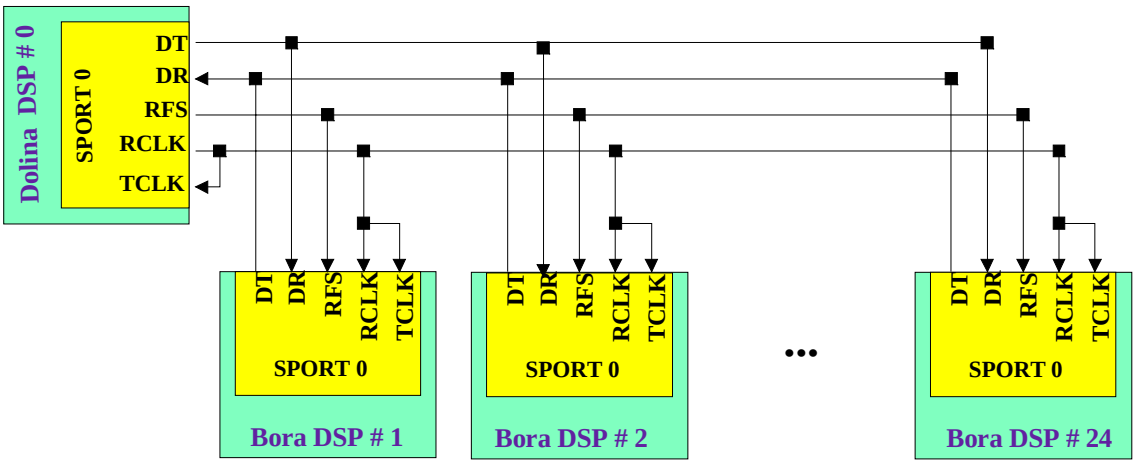

Figura 3.5 Conexión física entre los DSP de una red.

La señal RFS (*Reference Frame Sync*) provee la referencia de tiempo indicando el inicio de un *frame*. La misma señal es usada por el transmisor y el receptor, y puede ser generada internamente o externamente al procesador. El DSP de Dolina genera la señal internamente y el puerto RFS se vuelve una salida. Existe un registro al interno del procesador donde se establece por software la frecuencia de la señal RFS (*RFS_frec*). Cada uno de los DSP de Bora recibe la señal, generada externamente, a través del puerto RFS que en este caso funciona como entrada.

*RFS_frec = Bit Rate / (Número de slots por frame * Longitud del slot en bits)*

El puerto serial del procesador tiene dos señales de reloj, una para la transmisión (TCLK) y otra para la recepción (RCLK) de los datos. Cuando el modo de operación de la red es TDM, se utiliza un mismo y único reloj serial de referencia tanto para el receptor como para el transmisor. Por lo tanto el puerto TCLK funciona siempre como entrada y se conecta a su correspondiente puerto RCLK, y la señal RCLK se utiliza para transmitir el reloj de referencia de la red. El DSP de Dolina genera la señal a través del puerto RCLK que funciona como salida. Existe un registro al interno del procesador donde se establece por software la frecuencia del reloj (*CLK_frec*). Cada uno de los DSP de Bora recibe la señal de reloj, generada externamente, a través del puerto RCLK que en este caso funciona como entrada.

CLK_frec = Reloj interno del DSP / divisor

donde el reloj interno del DSP es de 60 MHz y divisor es un número entero entre 1 y 60.

Cada uno de los DSP transmite los datos a través del puerto DT (*Data Transmit*) y los recibe por medio del puerto DR (*Data Receive*).

Durante los tiempos correspondientes a los *slots* asignados al procesador, se puede transmitir datos, recibir datos, recibir y transmitir datos, o no efectuar alguna

operación. Por software se establece la asignación de los *slots* al procesador y el modo de uso de los mismos. Es posible también habilitar o deshabilitar individualmente *slots* específicos durante el uso de la red. El procesador transmite y recibe palabras de datos únicamente a través de los *slots* habilitados, ignorándolos en otro caso. De acuerdo a la configuración inicial especificada por el software de los DSP, que será explicado en los siguientes capítulos, el número de *slots* se estableció en 25 coincidiendo con el número de DSP de la red.

Tal como muestra la Figura 3.5, cada uno de los DSP de red se identifica con un número entre 0 y 24, donde el DSP de Dolina es el DSP número 0 en la red. De este modo, el DSP de Bora identificado como el DSP número 1 en la red puede transmitir y recibir datos utilizando el *slot* número 1, el DSP de Bora identificado como el DSP número 2 en la red puede transmitir y recibir datos utilizando el *slot* número 2, y así siguiendo hasta el DSP de Bora número 24 que puede transmitir y recibir datos utilizando el *slot* número 24. El DSP de Dolina puede entonces transmitir y recibir datos, hacia y desde los DSP de Bora, por medio de los *slots* número 1 hasta 24. La Tabla 3.1 resume dicha relación. El *slot* 0 se reserva para transmisión en Dolina y para recepción en cada una de las Bora.

	SLOT 0	**SLOT 1**	**SLOT 2**	**SLOT 3**	**…**	**SLOT 23**	**SLOT 24**
DSP #0	T	R/T	R/T	R/T	…	R/T	R/T
DSP #1	R	R/T	-	-	…	-	-
DSP #2	R	-	R/T	-	…	-	-
DSP #3	R	-	-	R/T	…	-	-
. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .
DSP #23	R	-	-	-	…	R/T	-
DSP #24	R	-	-	-	…	-	R/T

Tabla 3.1 *Slots* a través de los cuales se establece la comunicación entre los DSP ("R" significa recibe y "T" significa transmite).

Para la recepción de datos, el DSP provee una propiedad opcional que compara el dato en entrada, a través de alguno de los *slots* habilitados para ese procesador, con una palabra de 32 bits preestablecida por software denominada "palabra clave". Dependiendo del resultado de la comparación, el DSP acepta o ignora el dato proveniente de la red.

Cada uno de los DSP de Dolina administra y controla la red de DSP, y gestiona la comunicación entre dicha red y la PC donde se ejecuta el "Controlador del RICH". Memorias *dual-port* residentes en Dolina son usadas como medio físico a través del cual cada uno de los DSP de Dolina intercambia datos con la PC (Sección 4.1 y 4.2).

Cuando se habla de red de computadoras [85], es decir, de dos o más computadoras interconectadas que intercambian información con fines cooperativos; los siguientes conceptos surgen naturalmente: protocolo de comunicación y arquitectura de comunicación entre tales computadoras.

3.3.1 Protocolo de Comunicación

Para establecer la comunicación entre dos entidades pertenecientes a sistemas diferentes, ellas deben hablar el mismo lenguaje. Qué se comunica, cómo se comunica, y cuándo se comunica deben conformar un conjunto de convenciones mutuamente aceptadas por las entidades involucradas. Tales convenciones son referidas como "protocolo de comunicación", el cual puede definirse como el conjunto de reglas que gobiernan el intercambio de datos entre dos entidades. En nuestro caso, el intercambio de datos entre la PC y cada uno de los DSP que conforman las redes, se efectúa utilizando paquetes de red. Para la transmisión de comandos sin parámetros y códigos de error se utilizan paquetes de red "cortos" cuya

longitud es de dos palabras de 32 bits, y para la transmisión de programas, datos y comandos con parámetros se utilizan paquetes de red "largos" cuya longitud es de 128 palabras de 32 bits. La Tabla 3.2 muestra el contenido de las dos palabras de 32 bits que conforman un paquete de red "corto".

byte 3		*byte 2*	*byte 1*	*byte 0*
H		**O**	**L**	**A**
1	**Código de Comando / Código de Error**	**00000000**	**Fuente**	**Destino**

Tabla 3.2 Paquete de red "corto" (2 palabras de 32 bits).

La primer palabra del paquete es fija y corresponde a la representación en código ASCII de la palabra: **"HOLA",** la misma identifica el inicio del paquete y se utiliza como "palabra clave" para la recepción de datos en la red de DSP. La segunda palabra está conformada por los siguientes campos:

a) **Fuente (bits 0-7)** y **Destino (bits 8-15)** especifican el identificador del procesador que generó el paquete y el identificador del procesador a quien va dirigido el paquete respectivamente. La PC y los DSP de Dolina y Bora se identifican con una palabra de 8 bits, el identificador de la PC es "11111111b". La Figura 3.6 muestra el formato del identificador del DSP (xxxyyyyy), donde los 3 bits más significativos (xxx: bits 7-5) identifican la red y los 5 bits menos significativos (yyyyy: bits 4-0) identifican el DSP dentro de la red correspondiente. El número de redes de DSP en el sistema es 8 por lo tanto el identificador de la red toma un valor en el rango 0-7, y el número de DSP que conforman cada red es 25 por lo que el identificador del DSP en la red toma un valor en el rango 0-24. Por ejemplo, el identificador del DSP de Bora número 2 en la red número 0 es 00000010b, el identificador del DSP de Bora número 23 en

la red número 7 es 11110111b y el identificador del DSP de Dolina en la red número 3 es 01100000b.

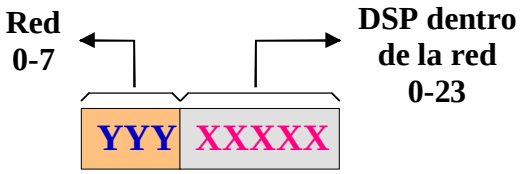

Figura 3.6 Identificador del DSP

b) **Código de Comando o de Error (bits 24-30)** especifica el código del comando o del error que se está transmitiendo, conocido también por el receptor que es el que lo interpreta.

c) El ***bit* 31** especifica el tipo de paquete ("corto" o "largo") y su contenido es siempre 1 cuando se trata de paquetes "cortos".

En el caso de los comandos "inicio de *spill*" (SOS) y "fin de *spill*" (EOS), que llegan directamente desde el TCS a la tarjeta Dolina y para los cuales los DSP de Dolina cuentan con sólo 200 milisegundos para transmitirlos a todas las tarjetas Bora, se utiliza una única palabra de 32 bits. La representación en código ASCII de la palabra **"HOLJ"** identifica el comando SOS y la representación en código ASCII de la palabra **"HOLE"** identifica el comando EOS.

Los paquetes "largos" (128 palabras de 32 bits) están conformados por dos palabras de encabezamiento seguidas de 125 palabras que contienen el cuerpo del paquete, y un *trailer*. La Tabla 3.3 muestra el formato de los paquetes de red "largos".

La primer palabra del encabezamiento y el *trailer* son fijos y contiene la representación en código ASCII de las palabras **"HOLA"** y **"CHAU"** respectivamente.

byte 3		*byte 2*	*byte 1*	*byte 0*
H		**O**	**L**	**A**
0	**Tipo de Información**	**Número de Paquete**	**Fuente**	**Destino**
Cuerpo del Paquete (125 palabras)				
C		**H**	**A**	**U**

Tabla 3.3 Paquete de red "largo" (128 palabras de 32 bits).

La segunda palabra del encabezamiento está conformada por los siguientes campos:

a) **Fuente (bits 0-7)** y **Destino (bits 8-15)** tienen en el mismo significado que en los paquetes "cortos".

b) **Número de Paquete (bits 16-23)** especifica el número del paquete y es generado por el transmisor en el caso que la información transmitida (datos o programas) involucre una secuencia de más de un paquete, en otro caso es cero.

c) **Tipo de Información (bit 24-30)** especifica el tipo de la información transmitida: programa de DSP, configuración de FPGA, umbrales, datos de los canales del detector (identificador del canal, valores medios y desviación estándar), datos de las tarjetas Bora (voltajes y temperaturas), datos del *run* (número de *spill* y número de *triggers* por *spill*) y código de comando con parámetros (los parámetros se transmiten en el cuerpo del paquete).

d) El **bit 31** es siempre 0 estableciendo que el paquete es "largo".

El **Cuerpo del Paquete** contiene la información transmitida y su formato varía dependiendo del tipo de información, especificado en el campo "Tipo de Información". Tales formatos serán descriptos en los siguientes capítulos.

La primer palabra del paquete cumple la misma función que en los paquetes "cortos". El *trailer* identifica el fin del paquete y se utiliza por el receptor, como

medida de seguridad adicional, que controla que la última palabra de un paquete "largo" sea efectivamente la palabra **"CHAU"**, enviando al transmisor del paquete un código de error en caso contrario.

3.3.2 Arquitectura de Comunicación

En términos generales, la arquitectura de comunicación involucra tres conceptos básicos: aplicaciones, computadoras y red. Las aplicaciones se ejecutan en las computadoras, las computadoras están conectadas a la red, y los datos, generados por las aplicaciones, se transfieren por la red desde una computadora hacia otra. De acuerdo a tales conceptos, la tarea de comunicación se puede organizar en la siguiente jerarquía de tres capas relativamente independientes: la capa de acceso a la red, la capa de transporte y la capa de aplicación. La Figura 3.7 muestra esquemáticamente la arquitectura de comunicación basada en tres capas, en la que se refleja que las capas que están al mismo nivel en distintas computadoras se comunican por medio de un protocolo.

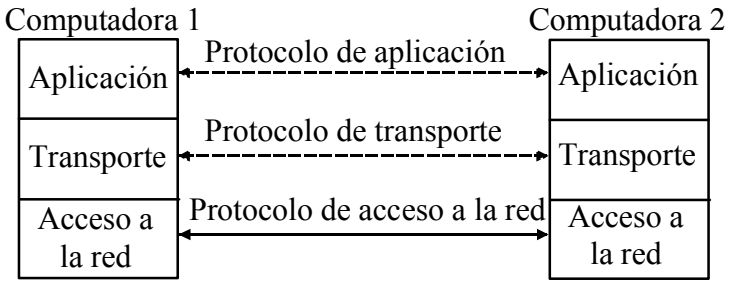

Figura 3.7 Arquitectura de comunicación basada en tres capas [85].

La capa de acceso a la red sería la mas baja en la jerarquía e involucra el intercambio de datos entre la computadora y la red a la cual está conectada. La computadora debe proveer la información necesaria para que la red pueda transferir

los datos a la destinación apropiada. El software específico utilizado en esta capa depende del tipo de red usada. De acuerdo al criterio según el cual las tareas de comunicación se dividen en capas, el software involucrado en las etapas superiores debería funcionar apropiadamente independientemente del tipo de red utilizado. Del mismo modo, el formato de los datos que se pretenden intercambiar es independiente de la naturaleza de las aplicaciones que los generan.

La capa de transporte es la encargada de organizar los datos generados por las aplicaciones en un determinado formato que incluya información de la naturaleza de la aplicación que generó los datos y de la destinación de dichos datos, y otro tipo de información que colabore con el requerimiento de que los datos intercambiados sean confiables.

Finalmente la capa de aplicación contiene la lógica necesaria para soportar las distintas aplicaciones y producir los datos a ser transmitidos de acuerdo a los requerimientos del usuario de la red.

En nuestro caso, la capa de aplicación soporta distintas aplicaciones que producen diferentes bloques de datos, comandos o mensajes de error, dependiendo de la naturaleza de la aplicación y de los requerimientos del usuario. La capa de transporte organiza tales bloques, comandos o mensajes en paquetes, agregando la información contenida en el encabezamiento del paquete, descripta en la sección anterior. Por último, la capa de acceso a la red usa la información contenida en el paquete para determinar su destinación y presenta el paquete a la red, que en nuestro caso es TDM, con un requerimiento de transmisión en el *slot* correspondiente al procesador hacia el cual va dirigido el paquete. En el procesador destino, la capa de acceso a la red recibe el paquete transmitido y controla que la destinación sea la adecuada. Luego lo pasa a la capa de transporte que interpreta el encabezamiento y

transfiere la información contenida en el cuerpo del paquete, o de los paquetes en caso que la información recibida involucre más de un paquete, a la aplicación apropiada que forma parte de la capa de aplicación.

3.4 Reprogramabilidad y Reconfigurabilidad

El sistema de adquisición, tratamiento, control y transmisión de datos del detector RICH-1 cuenta con 200 Procesadores de Señales Digitales (DSP) y 192 dispositivos de lógica programable (FPGA). La red de DSP permite la programación de todos los DSP, o de alguno en particular, y la configuración de todas las FPGA, o de alguna en particular, en cualquier momento que sea requerido, definiendo de este modo al sistema como reprogramable y reconfigurable. Al *power-up* se programan en primer lugar los DSP de la tarjeta Dolina, en segundo lugar los DSP de las tarjetas Bora y en tercer lugar se configuran las FPGA.

3.4.1 Programación de los DSP

La programación de los DSP de las tarjetas Dolina y Bora comienza después de cada *power-up* desde la PC de control del RICH-1.

Los DSP *bootean* desde una memoria EPROM y el proceso es el mismo en Dolina y en Bora. En el momento del *power-up* o al recibir un *reset* por hardware, el DSP cuenta con una característica inherente que carga automáticamente en su memoria interna, desde una memoria EPROM, un programa pequeño de 256 instrucciones. Este programa inicial, llamado *boot loader* [90, 92], tiene como función configurar la memoria interna del DSP y cargar un nuevo programa previamente almacenado en la EPROM.

El programa almacenado en la memoria EPROM es a su vez un cargador de programas, denominado *loader*, preparado para leer y cargar en la memoria interna del DSP un programa proveniente de la PC. El *loader* para el DSP de Dolina carga el programa transmitido directamente desde el "Controlador del RICH", y el *loader* para los DSP de Bora carga el programa proveniente de la PC, pasando por Dolina que ya fue programada, a través de la red de DSP. En los capítulos 4 y 6 se describe el *loader* para los DSP de Dolina y Bora respectivamente. Los *loader*s permiten la reprogramación de todos los DSP del sistema o de alguno en particular en cualquier momento que sea requerido desde el "Controlador del RICH", enviando un comando de *reset* al programa del DSP que lo interpreta iniciando el proceso de *booting* del DSP a través de la generación de un *reset* por software.

Desde la PC de control se inicia entonces la programación de los DSP del sistema, en primer lugar se programan los DSP de la tarjeta Dolina y una vez que los mismos están programados se procede con la programación de los DSP de las tarjetas Bora a través de las redes de DSP. La Figura 3.8 muestra un esquema del flujo de paquetes que se intercambian entre la PC y el DSP, en condiciones normales, durante la programación de un DSP.

Una vez finalizado el proceso de *booting* del DSP, el *loader* espera por un comando de inicio desde la PC. El proceso de programación del DSP se inicia desde el "Controlador del RICH" a través del comando: "PC Inicio", al que el *loader* responde con un comando de requerimiento del programa: "*Loader* Req.". Desde la PC se envía entonces el conjunto de paquetes conteniendo el programa (código y datos) del DSP. El *loader* almacena el programa en la memoria interna del DSP y confirma la completa recepción enviando un comando de reconocimiento a la PC:

"*Loader* Akn.". Luego de enviar el comando de reconocimiento comienza la ejecución del programa del DSP.

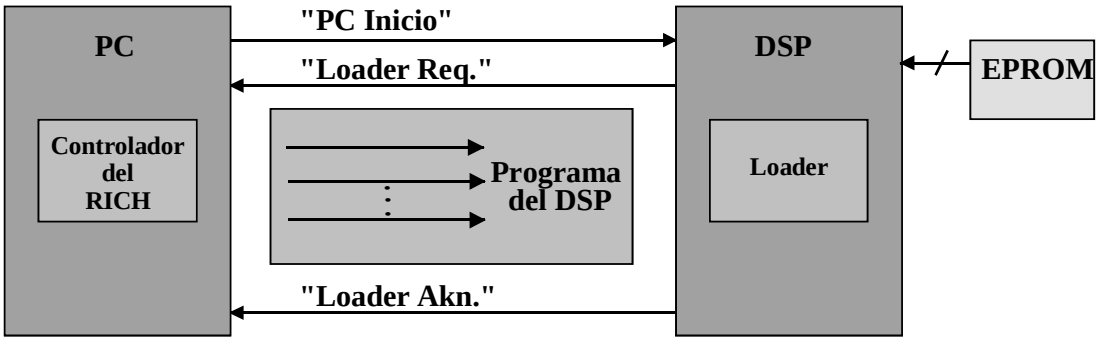

Figura 3.8 Flujo de paquetes durante el proceso de programación de los DSP.

3.4.2 Configuración de las FPGA

Después de cada *power-up*, la configuración de las FPGA del sistema comienza una vez que los DSP de Bora fueron programados.

Se denomina configuración [94] al proceso de cargar la cadena de bits con el diseño de la FPGA en su memoria de configuración interna. Existen diferentes modos de configuración, en nuestro caso se utiliza el modo denominado *Slave Serial* de acuerdo al cual la FPGA actúa como esclavo y es externamente configurada por el DSP de Bora. Una vez finalizado el proceso de *booting* del DSP de Bora, el primer programa que el *loader* carga es también un programa cargador, denominado *FPGA loader,* cuya función es configurar la FPGA. La cadena de bits con el diseño de la FPGA llega a la tarjeta Bora, desde el "Controlador del RICH", a través de la red de DSP.

Durante el proceso de configuración el DSP interactúa con la FPGA por medio de un grupo de señales dedicadas (Sección 7.2). El *FPGA loader* sigue el protocolo establecido para la configuración de la FPGA utilizando tales señales y una vez que la FPGA está lista para ser configurada, manda a la PC un comando requiriendo la

configuración. La Figura 3.9 muestra un esquema del flujo de paquetes que se intercambian entre la PC y el DSP de Bora, en condiciones normales, durante el proceso de configuración de la FPGA.

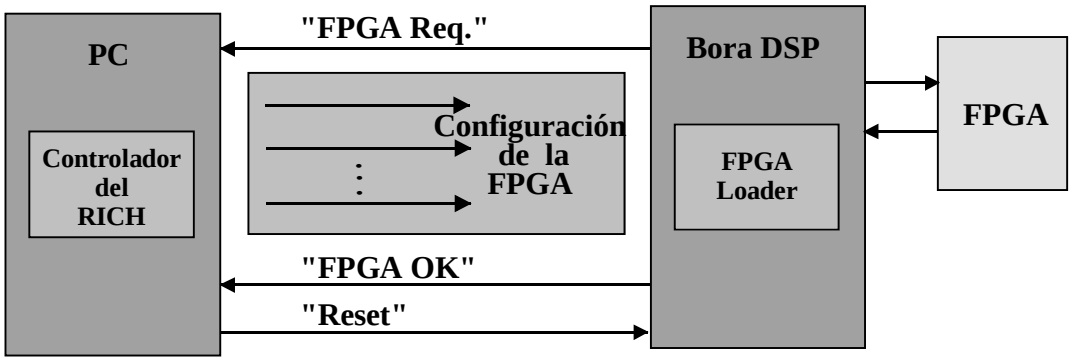

Figura 3.9 Flujo de paquetes durante el proceso de configuración de la FPGA.

El proceso de configuración de la FPGA se inicia por el *FPGA loader*, que transmite el comando: "FPGA Req." requiriendo al "Controlador de RICH" la configuración de la FPGA. Desde la PC se envía entonces el conjunto de paquetes conteniendo la cadena de bits con la configuración. El *FPGA loader* transmite dicha cadena hacia la FPGA y una vez que recibe la señal que determina que la FPGA fue exitosamente configurada, informa a la PC enviando el comando: "FPGA OK". Luego, espera un comando de *reset* por software para iniciar el proceso de *booting* del DSP y, en consecuencia, la programación del DSP que se efectúa con la FPGA ya configurada.

El *FPGA loader* permite la reconfiguración de todas los FPGA del sistema o de alguna en particular en cualquier momento que sea requerido desde el "Controlador del RICH", cargando en el DSP correspondiente el programa *FPGA loader* que inicia el proceso de configuración de la FPGA. En el capítulo 7 se describe el *FPGA loader.*

Capítulo 4

La Tarjeta MultiDSP DOLINA

Dolina es una tarjeta multiprocesador conteniendo ocho DSP que funcionan autónomamente uno de otro. Las tareas principales de Dolina son la administración y control de las ocho redes de DSP que conforman el sistema de adquisición del RICH y la comunicación entre la PC de control, donde Dolina reside, y dichas redes.

En la PC se ejecuta el "Controlador del RICH" por medio del cual se intercambian programas, datos, comandos y mensajes de error entre la PC y Dolina, y entre la PC y las tarjetas Boras, montadas en el detector, pasando por Dolina y las redes de DSP. Las redes de DSP, gestionadas por cada uno de los procesadores de Dolina, permiten la programación y configuración de todos los DSP y FPGA que conforman el sistema de adquisición, en cualquier momento que sea requerido; y el control y monitoreo permanente del sistema desde la PC.

A través de Dolina se distribuyen a todas las Boras las señales de sincronización, comunes al experimento, que delimitan las etapas involucradas en la adquisición de datos: "*inicio de run*", "*fin de run*", "*inicio de spill*" y "*fin de spill*". En el caso de las señales de inicio y fin de *spill*, existen sólo 200 milisegundos de tiempo entre que las mismas son generadas por el TCS y el primer trigger del *spill* y del *interspill* respectivamente, en consecuencia tales señales llegan a Dolina directamente desde el TCS. En cuanto a la señal de inicio de *run*, se cuenta con al menos 1 segundo de tiempo entre que se genera desde la sala "DAQ Control" y el

inicio del primer *spill*, por lo tanto las señales de inicio y fin de *run* llegan a Dolina desde la PC que las recibe por medio de la red de área local del CERN. El hecho que Dolina distribuya tales señales a través de la red de DSP, simplificó el diseño global del sistema puesto que en otro caso deberían conectarse cables a cada una de las 192 tarjetas Bora, específicos para el arribo de las señales de sincronización.

La tarjeta Dolina contiene ocho memorias *dual-port* a través de las cuales los DSP se comunican con la PC. A su vez los DSP de Dolina se comunican con los DSP de las tarjetas Bora por medio de ocho redes TDM de DSP implementadas a través de uno de los puertos seriales del DSP. La tarjeta contiene también ocho memorias EPROM, desde donde los DSP *bootean*, y ocho dispositivos *WatchDog* controlados por los DSP. La Figura 4.1 muestra una fotografía de uno de los lados de la tarjeta Dolina, en la que se puede apreciar un grupo de cuatro DSP (parte superior), cuatro memorias *dual-port* (en el medio), cuatro memorias EPROM (parte inferior), y cuatro dispositivos *WatchDog* (sobre los DSP). El segundo grupo de cuatro DSP, memorias y dispositivos se encuentra en el lado opuesto de la tarjeta.

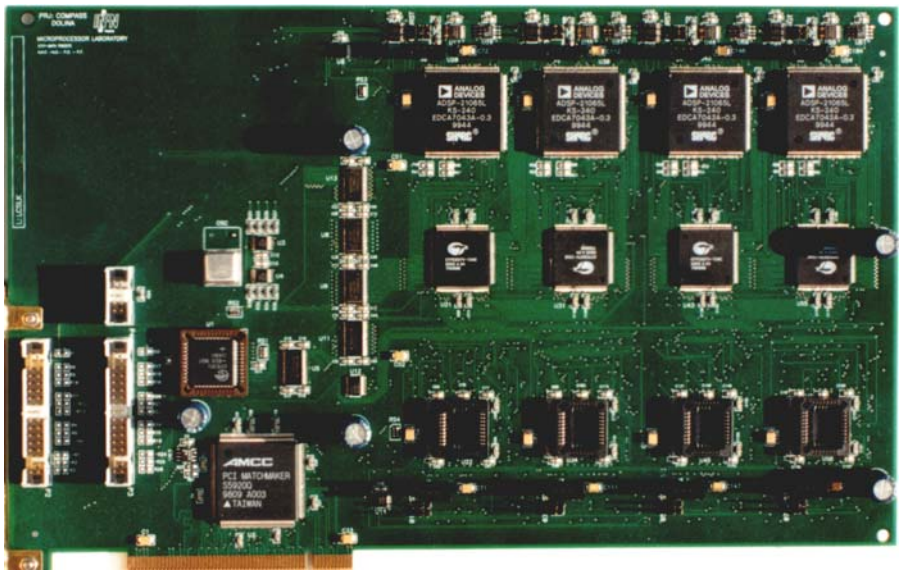

Figura 4.1 La tarjeta Dolina

4.1 Arquitectura de la Tarjeta Dolina

Dolina contiene ocho *Digital Signal Processor Microcomputer* perteneciente a la familia *SHARC* de *Analog Devices* (ADSP-21065L [90, 91) responsables de la funcionalidad de la tarjeta. El DSP se alimenta a 3,3 *volts* y toma un reloj externo de 30 MHz proveniente de un oscilador de cristal. Este reloj se multiplica internamente alimentando el *core* del procesador que trabaja a 60 MHz. Cada DSP tiene asociado una memoria EPROM, un dispositivo *WatchDog* y una memoria *dual-port* (DPM). La Figura 4.2 muestra un diagrama de bloques conteniendo los principales componentes de la tarjeta Dolina y sus interconexiones.

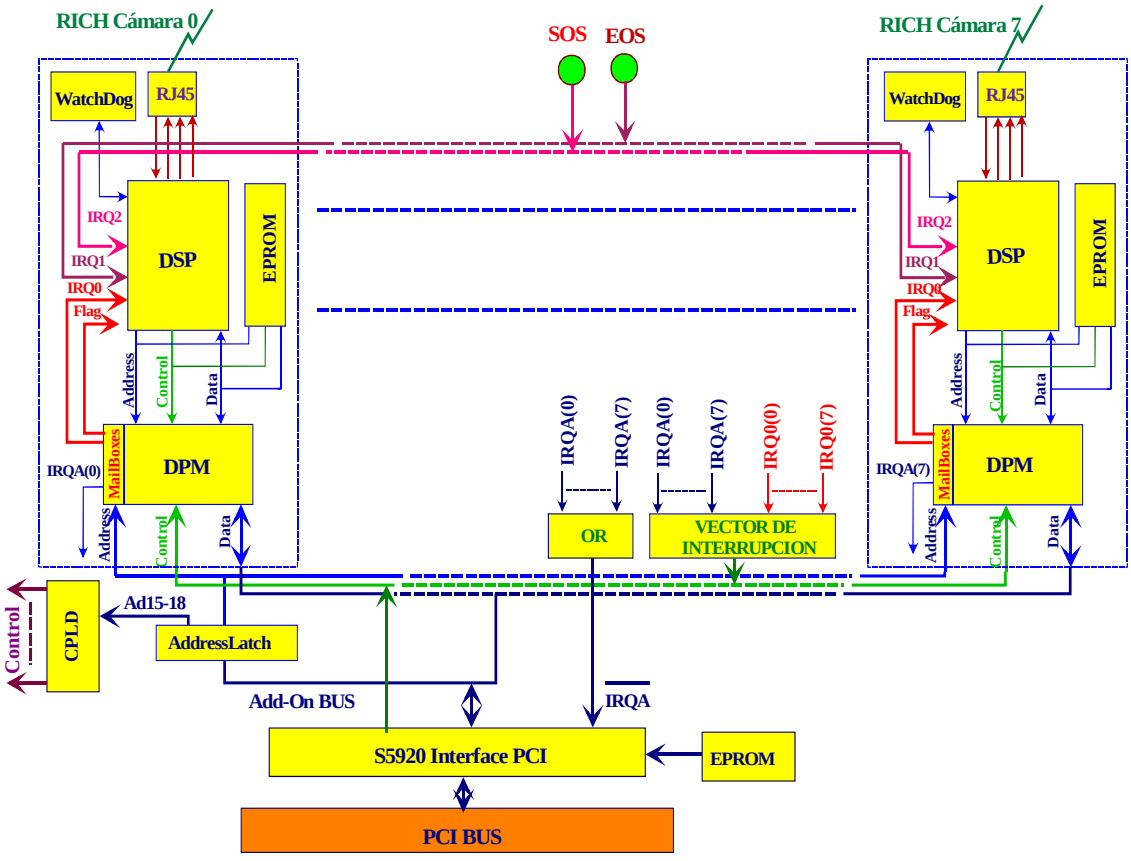

Figura 4.2 Diagrama de bloques de la arquitectura interna de Dolina

La EPROM es la memoria de *booting* del DSP. El *WatchDog* y la DPM son parte de la memoria externa del procesador. La memoria externa del DSP está dividida en cuatro bancos de tamaño fijo (banco 3-0) y cada banco tiene asociado una línea de selección MS 3-0. La interfase entre el DSP y la memoria externa se efectúa principalmente por medio de un *bus* de datos y un *bus* de direcciones externos, una señal de escritura (WR) y una señal de lectura (RD). El acceso por parte del DSP a alguna dirección perteneciente a alguno de estos bancos genera una activación de la línea de selección correspondiente. De este modo es posible usar las líneas de salida MS como *chip selects* para memorias u otros dispositivos externos, eliminando la necesidad de lógica de decodificación externa.

La memoria de *booting* se conecta al *bus* de datos y al *bus* de direcciones externo. Una señal específica generada por el DSP (*boot memory select*) habilita la EPROM y la señal RD activa las salidas de la memoria. Al momento del *booting* se carga automáticamente en el DSP el programa *boot loader* almacenado en la EPROM (Sección 3.4.1).

Para la comunicación entre el DSP y el dispositivo *WatchDog* [41, 95] se utiliza la línea MS1 correspondiente al banco 1 de la memoria externa del procesador. En el caso de producirse una falla en el sistema, originada por algún problema de hardware o software (tal como un *power glitch*), que deje al procesador en un estado en el cual no se puede recuperar por software, el *WatchDog* interviene generando automáticamente un *reset* por hardware. El modo de funcionamiento de tal dispositivo es simple, está configurado para generar un *reset* al DSP, a través del puerto correspondiente, en intervalos fijos de tiempo (1,6 s.). Para evitarlo, el software del DSP envía una señal al *WatchDog* dentro de tal intervalo de tiempo que causa que el dispositivo reinicialice en cero su contador de tiempo. Dicha señal es transmitida

desde el DSP al *WatchDog* a través de la línea MS1, por medio de un acceso (escritura o lectura) a alguna de las direcciones pertenecientes al banco 1 de la memoria externa.

La memoria *dual-port* [96], a través de la cual el DSP intercambia información con la PC, esta mapeada en el banco 0 de la memoria externa del DSP. La DPM es una memoria simétrica que puede ser leída y escrita contemporáneamente por ambos puertos (izquierdo y derecho), y su uso es adecuado como medio de interacción entre sistemas totalmente asíncronos como es el caso del DSP y la PC. Tiene incorporada dos líneas de interrupción (INTL y INTR) para la comunicación entre ambos procesadores por medio de *mail boxes*. Cada puerto tiene asociado una dirección de memoria fija que funciona como *mail box*. Así, cuando por ejemplo el puerto de la derecha (el DSP) quiere enviar un mensaje al puerto de la izquierda (la PC), escribe en el *mail box* correspondiente al puerto de la izquierda y se genera una interrupción (INTL) que le arriba al propietario del *mail box*. El mismo limpia la interrupción cuando lee el mensaje desde su *mail box*. El proceso es el mismo pero inverso si el puerto de la izquierda quiere enviar un mensaje al puerto de la derecha.

El DSP soporta tres interrupciones externas con prioridad establecida por el hardware e individualmente enmascarables por software. Un dispositivo externo puede generar alguna de estas interrupciones, activando la línea de entrada correspondiente (IRQ 2-0). En Dolina tales interrupciones se definen por software como *edge-triggered*, es decir, la interrupción es considerada válida si el procesador la muestrea alta en un ciclo y baja en el siguiente. Las señales de inicio de *spill* (SOS) y de fin de *spill* (EOS) llegan a cada uno de los DSP, desde una tarjeta TCS, a través de las líneas de interrupción externa IRQ2 y IRQ1 respectivamente. La tercer línea de interrupción IRQ0 está conectada con la bandera de interrupción (INTR) de la DPM

correspondiente. A través de la IRQ0 el DSP conoce que existe un mensaje para él, generado desde la PC, en la DPM.

Por otra parte, la línea de interrupción de la DPM correspondiente a la PC (INTL) se conecta también a uno de los *flags* de propósito general del DSP. El DSP cuenta con doce *flags* de propósito general programables por software como puertos de entrada o salida. El DSP utiliza el FLAG11 para establecer si puede o no enviar un mensaje a la PC. Si la PC no limpió la interrupción (INTL) significa que tiene pendiente la lectura de un mensaje y en consecuencia el DSP no puede enviarle uno nuevo porque causaría la sobre escritura, y en consecuencia la pérdida, del mensaje anterior.

La PC cuenta con una sola línea de interrupción externa (IRQA) a través de la cual conoce si alguna de las ocho DPM le generó una interrupción. Por lo tanto cada una de las ocho líneas de interrupción (INTL) de las DPM, en la Figura 4.2 rotuladas como IRQA(0)-IRQA(7), están conectadas a las entradas de un or-lógico de 8 bits cuya salida se conecta al IRQA de la PC. De este modo la PC sabe si alguna (o todas) de las ocho DPM le generó una interrupción, le falta entonces saber cuál (o cuáles) de las DPM se la generó. Para este propósito se utiliza un *buffer tristate* de 16 bits, denominado “vector de interrupción”. Al *buffer* se conectan las ocho líneas de interrupción provenientes de las DPM (IRQA(0)-IRQA(7)) por medio de las cuales la PC determina cuál DPM generó la interrupción.

Al *buffer* también se conectan las líneas de interrupción IRQ0 correspondientes a cada uno de los ocho DSP (IRQ0(0)-IRQ0(7)), la PC las utiliza para determinar si puede o no mandar un mensaje a alguno de los DSP. Si el DSP a quien la PC pretende enviarle un mensaje no limpió la correspondiente línea de interrupción, significa que tiene pendiente la lectura de un mensaje y en consecuencia

la PC no puede enviarle un nuevo mensaje porque implicaría la pérdida del mensaje anterior.

La tarjeta Dolina se conecta al *bus* PCI de la PC. La interacción entre el *bus* y las memorias *dual-port* se realiza por medio de un dispositivo de interface PCI (S5920 [97]) que ofrece un modo simple y flexible de conectar aplicaciones al *bus* PCI. Tal dispositivo convierte las señales complejas inherentes al *bus* PCI en un *bus* de 32 bits simple de usar, denominado *Add-On bus*. El S5920 está preparado para funcionar en distintos modos por lo tanto la configuración inicial es definida por el usuario y se carga en el dispositivo, al *power-up* del sistema, desde una memoria EPROM. En Dolina, el S5920 se configura para transferir datos vía un canal de datos *Pass-Thru* en modo activo, que habilita la lógica interna para adquirir el *Add-On bus* para escritura o lectura de datos en forma independiente.

Para la escritura o lectura de una palabra por parte de la PC, se usa un *address latch* por medio del cual la PC especifica la DPM y la dirección dentro de la DPM. Los bits 0-14 contienen la dirección y los bits 15-17 contienen el número que identifica una de las ocho DPM. Los bits 15-18 son leídos por un dispositivo de lógica programable CPLD (*Complex Programmable Logic Device* [62, 98]), que usa esta información para proveer el *chip select* de la DPM correspondiente en caso que el bit 18 sea 0. Mientras que si el bit 18 es 1, significa que independientemente del contenido del resto de los bits, la PC quiere leer el contenido del vector de interrupción, en consecuencia la CPLD provee el *chip select* del mismo. Las CPLD es un dispositivo no volátil, por lo tanto se programa solo una vez.

La comunicación entre cada DSP de Dolina y los 24 DSP de las tarjetas Bora que conforman una cámara del RICH se efectúa a través de una red TDM (Sección 3.3) implementada por medio de uno de los puertos seriales del DSP (SPORT0). Por

cada red de DSP, existe una tarjeta optoaisladora para aislar eléctricamente las señales que conectan los 24 DSP de una cámara y el DSP de Dolina, evitando de este modo interferencias eléctricas entre la PC y el detector. Los cables que transportan las señales de comunicación entre los DSP de una red, se conectan desde Dolina a la tarjeta optoaisladora y desde la tarjeta optoaisladora a cada una de las 24 Bora.

4.2 Comunicación entre Dolina y la PC de Control

Cada uno de los ocho DSP de Dolina se comunica con la PC a través de una memoria *dual-port* (32K x 32-bit). Tal como se describió en la sección anterior, la DPM tiene incorporado un sistema de comunicación entre ambos procesadores por medio de *mail boxes*. Las dos últimas direcciones de la memoria actúan como *mail boxes*, uno para el DSP y otro para la PC.

La PC y el DSP intercambian datos, programas, comandos y mensajes de error por medio de paquetes de red (Sección 3.3.1). Como la velocidad de comunicación entre la PC y el DSP no es un factor crítico y con el objetivo de simplificar el diseño de hardware y de software, el intercambio de paquetes se realiza utilizando dos regiones de tamaño fijo con espacio para 256 palabras de 32 bits cada una. La PC lee en la región en la que el DSP escribe, y por el contrario el DSP lee en la región en la que la PC escribe. Así, los *mail boxes* se utilizan para comunicar la existencia de un paquete de red en la memoria. Un esquema del direccionamiento y uso de la DPM se muestra en la Figura 4.3. Desde el punto de vista del DSP, la DPM correspondiente está direccionada en el banco 0 de la memoria externa del procesador.

La escritura por parte de la PC en el *mail box* correspondiente al DSP genera la interrupción externa IRQ0 en el DSP, y la escritura por parte del DSP en el *mail box* correspondiente a la PC genera la interrupción IRQA en la PC. En la Figura 4.3,

"IRQA_X" identifica la interrupción generada a la PC desde la "DPM X" con X en el rango 0-7. A su vez, el DSP conoce si la PC tiene alguna interrupción pendiente, en consecuencia no puede enviarle un nuevo mensaje, a través del FLAG11 del DSP que está conectado con la línea de interrupción "IRQA_X". Por el contrario, la PC conoce si el DSP, a quien pretende enviarle un mensaje, tiene una interrupción pendiente por medio del vector de interrupción al cual están conectadas las líneas IRQ0 correspondientes a los ocho DSP de la tarjeta.

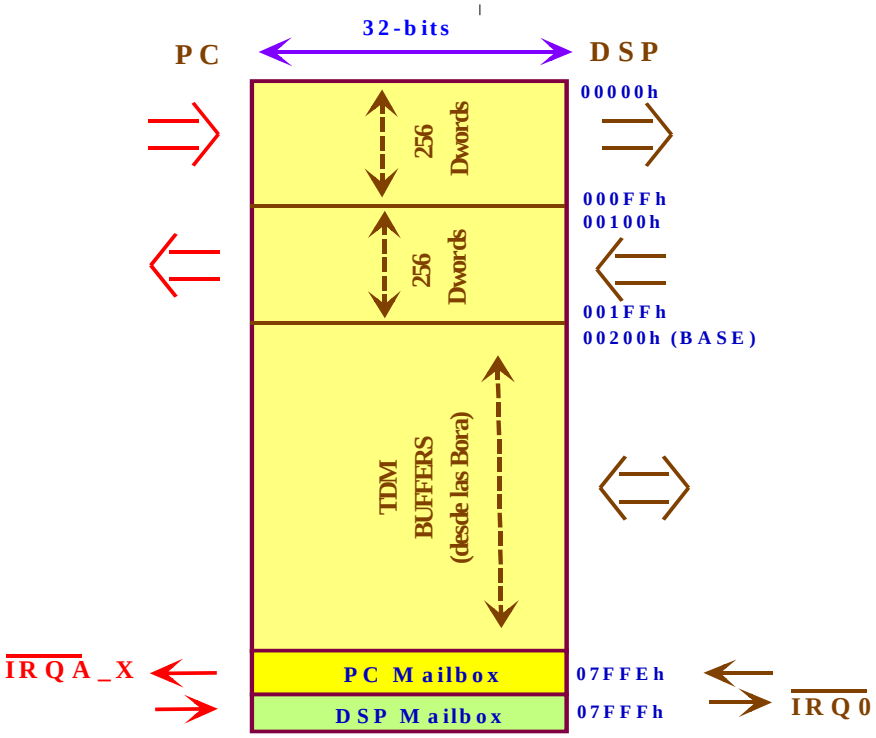

Figura 4.3 Uso y direccionamiento de la memoria *dual-port* correspondiente a un DSP

El protocolo de comunicación entre Dolina y la PC depende de quien sea el emisor del paquete. La Figura 4.4 resume la secuencia de acciones principales ejecutada por el software del DSP de Dolina en los casos en los que Dolina envía un paquete a la PC. En este caso el destinatario del paquete es la PC y el emisor del paquete es alguno de los DSP que conforma la red (DSP de Dolina o DSP de Bora).

1. *El DSP consulta el FLAG11 para determinar si puede enviar el paquete a la PC.*
2. *Si el contenido del flag es 0 significa que la PC tiene una interrupción pendiente por la tanto el DSP espera chequeando continuamente el valor del flag.*
3. *Si el contenido del flag es 1 significa que la PC está disponible para leer el paquete.*
4. *El DSP escribe el paquete en la región de la DPM destinada a la escritura de paquetes por parte del DSP.*
5. *El DSP escribe en el mail box de la PC para generar una interrupción en la PC.*
6. *El DSP escribe en el mail box de la PC para generar una interrupción en la PC.*

Figura 4.4 Pasos principales ejecutados por el DSP de Dolina para la transmisión de paquetes a la PC de control.

La Figura 4.5 resume la secuencia de acciones principales ejecutada por el software del DSP de Dolina en los casos en los que Dolina recibe un paquete desde la PC. En este caso el emisor del paquete es la PC y el receptor del paquete es alguno de los DSP que conforma la red (DSP de Dolina o DSP de Bora).

1. *El DSP recibe la interrupción externa IRQ0 que le indica la existencia de un paquete en la DPM.*
2. *El DSP lee el encabezamiento del paquete desde la región de la DPM destinada a la lectura de paquetes por parte del DSP.*
3. *El DSP controla que el encabezamiento del paquete sea correcto: controla que la primer palabra sea "HOLA" y que la destinación del paquete corresponda al identificador de alguno de los DSP que conforma la red para la cual el master es el DSP que recibe el paquete.*
4. *Si el destinatario es el DSP de Dolina, el DSP controla que el paquete sea corto puesto que la PC envía solo paquetes cortos con destinación el DSP de Dolina.*
5. *Si el encabezamiento es correcto y el destinatario es el DSP de Dolina, el DSP copia la segunda palabra del encabezamiento desde la DPM a la memoria interna.*
6. *El DSP procesa el comando usando como parámetro el encabezamiento del paquete. Desde este punto el protocolo sigue en el paso 10.*
7. *Si el encabezamiento es correcto y el paquete es largo, el DSP controla que la última palabra del paquete sea "CHAU".*
8. *Si ningún error es detectado y el destinatario del paquete es alguno de los DSP de Bora, el DSP copia el paquete desde la DPM a un buffer de la memoria interna del DSP.*

9. *El DSP envía el paquete a la Bora correspondiente a través de la red de DSP.*

10. *Si algún error es detectado, el DSP ignora el paquete proveniente de la PC y prepara un paquete de error conteniendo el código del error correspondiente. Luego envía el paquete de error siguiendo el protocolo especificado en la Figura 4.4.*

11. *El DSP lee el mail box del cual es propietario, limpiando la interrupción generada por la PC. De este modo el DSP avisa a la PC que el paquete fue recibido y procesado.*

Figura 4.5 Pasos principales ejecutados por el DSP de Dolina para la recepción de paquetes provenientes de la PC de control.

La tarjeta Dolina se conecta al *bus* PCI de la PC. Un *device driver*, específicamente desarrollado para la tarjeta, gestiona el intercambio de paquetes de red entre la aplicación de alto nivel (“Controlador del RICH”) y los DSP de Dolina. Por lo tanto, el *driver* es responsable del servicio y atención de las interrupciones generadas desde las DPM.

El “Controlador del RICH” utiliza un conjunto de funciones provistas por el *driver* para intercambiar paquetes con los DSP. Para el envío de paquetes a alguno de los DSP de Dolina, la aplicación realiza la solicitud al *driver* quien gestiona el requerimiento. Para la lectura de los paquetes generados desde los DSP, el *driver* atiende las interrupciones provenientes de las DPM copiando los paquetes desde las DPM a un *buffer* interno administrado como FIFO. La aplicación consulta continuamente el estado de dicho *buffer* y siempre que no esté vacío, procede con la lectura de paquetes directamente desde el mismo. Un semáforo, gestionado por el *driver*, permite la escritura y lectura del *buffer* por parte del *driver* y de la aplicación respectivamente. Desde la PC, las ocho DPM son vistas como un único bloque de memoria. La Figura 4.6 muestra el direccionamiento de las DPM de Dolina.

El rango de direcciones para las ocho DPM comienza en la dirección 0x00000 y termina en la dirección 0x3ffff. La lectura en el rango de direcciones 0x40000 – 0x4ffff (bit 18 = 1) implica la lectura del vector de interrupción que contiene

información de las interrupciones generadas desde las DPM a la PC y del estado de las interrupciones correspondientes a los ocho DSP, generadas por las DPM desde la PC.

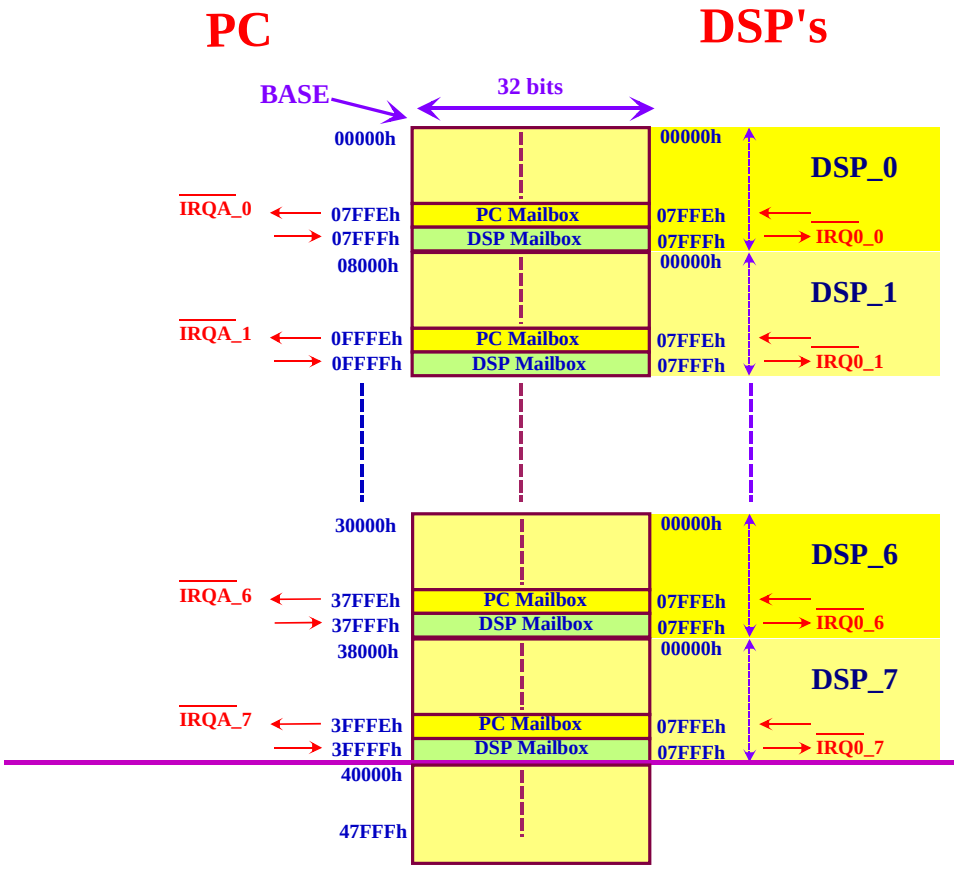

Figura 4.6 Direccionamiento de las ocho DPM de Dolina.

La Figura 4.7 resume los pasos principales seguidos por la PC en los casos en los que la PC envía un paquete a Dolina. En este caso el destinatario del paquete es alguno de los DSP que conforma la red (DSP de Dolina o DSP de Bora).

1. *La PC consulta el vector de interrupción para determinar si puede enviar el paquete al DSP master de la red a la cual va dirigido el paquete.*

2. *Si el contenido del vector de interrupción en la posición correspondiente al DSP master es 0 significa que el DSP tiene una interrupción pendiente en consecuencia la PC espera chequeando continuamente el contenido del vector.*

3. *Si el contenido del vector de interrupción, en la posición correspondiente, es 1 significa que el DSP está disponible para leer el paquete.*

4. *La PC escribe el paquete en la región de la DPM, destinada a la escritura por parte de la PC, correspondiente al DSP master.*

5. *La PC escribe en el mail box de la DPM correspondiente al DSP master, para generar una interrupción en el DSP.*

Figura 4.7 Pasos principales seguidos en la PC de control para la transmisión de paquetes a Dolina.

La Figura 4.8 resume los pasos principales seguidos por la PC en los casos en los que la PC recibe un paquete desde Dolina. En este caso el emisor del paquete es alguno de los DSP que conforma la red (DSP de Dolina o DSP de Bora).

1. *La PC recibe la interrupción IRQA que le indica la existencia de un paquete en alguna de las ocho DPM.*

2. *La PC chequea el vector de interrupción para determinar los DSP que generaron la interrupción.*

3. *Por cada DSP que generó la interrupción, la PC copia el paquete desde la DPM correspondiente a un buffer de la memoria interna de la PC. Desde dicho buffer, administrado como FIFO, el "Controlador de RICH" lee y procesa el paquete.*

4. *Si algún error es detectado, la PC informa al usuario del origen y naturaleza del error por medio del "Controlador del RICH".*

5. *La PC lee el mail box del cual es propietario, de cada DPM desde donde leyó un paquete, limpiando la interrupción generada por el DSP. De este modo la PC avisa al DSP correspondiente que el paquete fue recibido y procesado.*

Figura 4.8 Pasos principales seguidos en la PC de control para la recepción de paquetes provenientes desde Dolina.

4.3 Comunicación entre Dolina y las Tarjetas Bora

Cada DSP de Dolina gestiona una red de DSP formada por el mismo y 24 DSP correspondientes a las tarjetas Bora de una cámara del RICH. El detector cuenta con ocho redes de DSP.

Los DSP contienen el hardware para implementar una red TDM, a través del puerto serial SPORT0 del procesador (Sección 3.3). Por medio del SPORT0, el DSP

puede recibir y transmitir datos simultáneamente. El puerto serial tiene registros de control y dos *buffers* de datos (uno para transmisión y otro para recepción) para la configuración y comunicación del SPORT por parte del software del DSP. La longitud de los registros y de los *buffers* es de 32 bits. La Figura 4.9 muestra esquemáticamente la estructura interna del SPORT en el DSP [90].

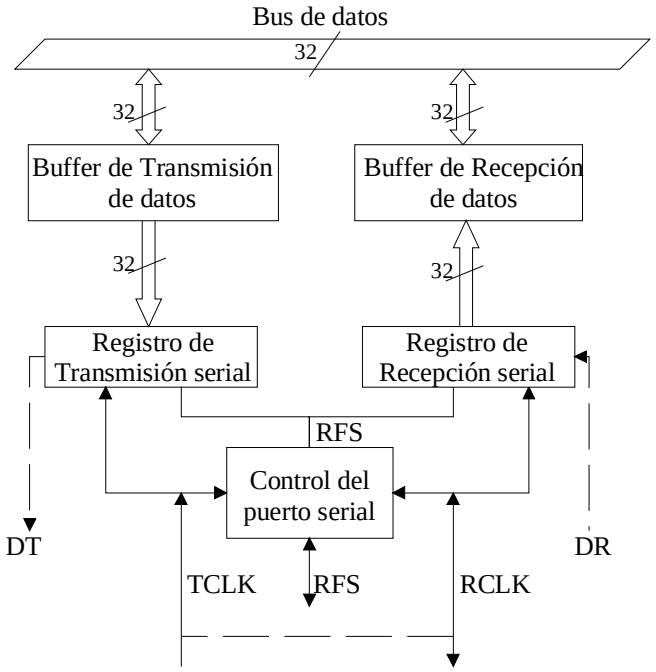

Figura 4.9 Estructura interna del SPORT en el DSP.

Para la transmisión de datos a la red, el DSP escribe palabras de 32 bits en el *buffer* de transmisión de datos, desde donde cada palabra se transfiere automáticamente al registro de transmisión serial. Desde este registro se transfieren los bits a través del puerto DT (*Data Transmit*). Del mismo modo para la recepción de datos, los bits se leen desde el puerto DR (*Data Receive*) al registro de recepción serial. Cuando el registro contiene una palabra completa de 32 bits, la palabra se transfiere automáticamente al *buffer* de recepción de datos desde donde el DSP lee las palabras provenientes de la red. Tanto la recepción como la transmisión se efectúan

sincrónicamente con las señales de inicio de *frame* (RFS) y de reloj de referencia (RCLK).

La gestión del SPORT por parte del software se realiza por medio de dos interrupciones internas, una para la transmisión y otra para la recepción de datos. La transferencia de datos entre el SPORT y la memoria interna se realiza en los dos siguientes modos posibles [90]:

- Usando SPORT DMA (*Direct Memory Acces*s) para transmisión o recepción, el cual provee un mecanismo para recibir o transmitir un bloque entero de datos antes que el SPORT genere una interrupción. El DSP contiene un controlador de DMA que manipula la transferencia de datos, habilitando al procesador a continuar con la ejecución del programa hasta que un bloque entero de datos se transmita o reciba. El software del DSP inicializa el DMA especificando, en los registros correspondientes, la dirección base del *buffer* en memoria interna desde donde o hacia donde se realiza la transferencia de los datos, la cantidad de datos a transferir, y un modificador que determina el incremento del puntero al *buffer* después de cada lectura o escritura. Una vez establecidos los parámetros, el software habilita el DMA y el controlador inicia automáticamente la transferencia de datos desde el *buffer* receptor del SPORT al *buffer* en memoria interna, o desde el *buffer* en memoria interna al *buffer* transmisor del SPORT. Cuando el controlador termina la transferencia, se genera la interrupción correspondiente y se deshabilita automáticamente el DMA.
- Cuando el DMA está deshabilitado para transmisión o recepción, el SPORT genera una interrupción cada vez que recibe o inicia la transmisión de una palabra de dato. La interrupción se produce cuando el *buffer* de recepción del SPORT contiene una palabra completa o cuando el *buffer* de transmisión del SPORT está vacío.

Antes de habilitar las interrupciones asociadas con la transmisión y recepción del SPORT, el software del DSP configura el puerto serial en modo TDM. La configuración incluye principalmente la especificación del número de *slots* que conforma la red TDM y el número de bits por *slot*. En nuestro caso, tal como se explicó en el capítulo 3, el número de *slots* es 25 coincidiendo con el número de procesadores de la red; y por cada *slot* se transmiten o reciben palabras de 32 bits. Como el DSP de Dolina es el *master* de la red, establece también la frecuencia de la señal que determina el inicio de cada *frame* (RFS) y la frecuencia del reloj de referencia de la red (RCLK).

Registros de selección de *slot*, uno para la transmisión y otro para la recepción (ambos de 32 bits), se usan para habilitar o deshabilitar los *slots* durante el uso de la red TDM. Cada bit en estos registros se corresponde con uno de los *slot* de la red, el bit 0 con el *slot* 0, el bit 1 con el *slot* 1, y así siguiendo. El SPORT transmite y recibe únicamente a través de los *slots* habilitados, ignorándolos en otro caso.

El DSP de Dolina puede transmitir y recibir en todos los *slots* correspondientes a los DSP de Bora que conforma la red que él gestiona (Tabla 3.1). Sin embargo, el software del DSP mantiene todos los *slots* de transmisión deshabilitados lo mismo que la interrupción de transmisión hasta el momento en que tenga un paquete para transmitir a la red. La razón es que para la transmisión se usa SPORT DMA. De este modo, cuando el DSP tiene un paquete en un *buffer* de la memoria interna, listo para ser enviado a alguna de las Bora de la red, inicializa los parámetros del DMA y luego habilita en primer lugar el *slot* correspondiente al destinatario del paquete, en segundo lugar el DMA, y por último la interrupción de transmisión del SPORT.

Cuando el controlador de DMA termina de transferir el paquete al *buffer* de transmisión del SPORT, desde donde el SPORT efectúa la transmisión a la red usando el único *slot* habilitado, se genera una interrupción y la rutina que atiende dicha interrupción deshabilita nuevamente los *slots* y la interrupción de transmisión.

La Figura 4.10 resume las acciones principales ejecutadas por el software del DSP de Dolina para enviar un paquete al DSP de la red identificado como el número *x*, donde *x* es un número entre 1 y 24.

1. *Establece los parámetros del DMA con la dirección base del buffer que contiene el paquete, la longitud del paquete (2 si el paquete es corto y 128 si el paquete es largo) y el incremento en 1.*

2. *Habilita el slot número x de la red usando el registro de selección de slots de transmisión (bit x = 1).*

3. *Habilita la interrupción de transmisión del SPORT.*

4. *Habilita el DMA.*

Una vez habilitado el DMA, el controlador de DMA inicia la transferencia del paquete al buffer de transmisión del SPORT desde donde la transmisión se efectúa en el slot número x. Cuando el paquete completo fue transferido, se deshabilita automáticamente el DMA y se genera la interrupción de transmisión del SPORT. La rutina que atiende la interrupción ejecuta los siguientes pasos:

1. *Deshabilita el slot número x de la red usando el registro de selección de slots de transmisión (bit x = 0).*

2. *Deshabilita la interrupción de transmisión del SPORT.*

Figura 4.10 Pasos principales ejecutados por el DSP de Dolina para la transmisión de paquetes a la red de DSP.

Para la recepción de datos desde la red, los *slots* a través de los cuales el DSP de Dolina recibe paquetes permanecen en cambio siempre habilitados lo mismo que la interrupción de recepción del SPORT puesto que el software debe estar preparado en todo momento para recibir paquetes provenientes de las tarjetas Bora. En este caso no es posible el uso del SPORT DMA debido a que las palabras pueden llegar al *buffer* de recepción del SPORT provenientes de cualquiera de los *slots* habilitados,

implicando que la reconstrucción del paquete corresponda al software del DSP de Dolina.

Para este propósito, el software reserva espacio para dos *buffers* de 128 palabras de 32 bits cada uno, que corresponde a la longitud de un paquete largo, por cada uno de los 24 *slots* de los cuales puede recibir datos provenientes de las Bora. Estos 48 *buffers* se almacenan en una región de la DPM que no es usada por la PC (Figura 4.3). Así, cuando el receptor del SPORT genera una interrupción indicando que existe una palabra en el *buffer* de recepción del SPORT proveniente de alguno de los *slots* habilitados, la rutina que atiende la interrupción chequea el *slot* desde donde la palabra proviene y la copia al *buffer* correspondiente en el cual se está realizando la reconstrucción del paquete para ese *slot*.

El DSP provee además una propiedad opcional que compara el dato en entrada, a través de alguno de los *slots* habilitados, con una palabra de 32 bits preestablecida por el software denominada “palabra clave”. Dependiendo del resultado de la comparación, el SPORT acepta o ignora el dato recibido. Un registro del SPORT de 32 bits se usa para habilitar o deshabilitar lo palabra clave en los distintos *slots*. Cada bit en el registro se corresponde con uno de los *slot* de la red, el bit 0 con el *slot* 0, el bit 1 con el *slot* 1, y así siguiendo.

El software del DSP de Dolina utiliza esta propiedad para cada uno de los *slots* de recepción estableciendo que la palabra clave sea “HOLA”, que es la primer palabra de cada uno de los paquetes generados desde las tarjetas Bora. Mientras la palabra clave está habilitada, una interrupción por parte del receptor del SPORT implica que la palabra almacenada en el *buffer* de recepción es la palabra clave indicando entonces el inicio de la transmisión de un paquete por parte de alguno de los *slots*. La rutina de atención a la interrupción chequea el *slot* que transmitió la palabra, deshabilita la

palabra clave para ese *slot* e inicia la reconstrucción del paquete. Cuando el paquete fue totalmente recibido la rutina habilita nuevamente la palabra clave para ese *slot*.

4.3.1 Reconstrucción de los Paquetes Recibidos desde la Red de DSP

Dolina recibe paquetes desde todas las Boras y cada Bora transmite en uno de los 24 *slots* que conforman una red. Por lo tanto, el DSP de Dolina puede recibir palabras provenientes de todos o de un grupo de *slots*, y cada palabra forma parte de un paquete distinto. Por ejemplo, el DSP recibe una palabra del *slot* 1 que pertenece a un paquete transmitido desde el DSP número 1 de la red, e inmediatamente después recibe una palabra del *slot* 2 que pertenece a un paquete transmitido desde el DSP número 2. Para la reconstrucción de los paquetes, el DSP usa un conjunto de estructuras y dos *buffers* de 128 palabras de 32 bits cada uno por cada *slot* de la red. Las Figuras 4.11(a) y 4.11(b) muestran gráficamente tales estructuras.

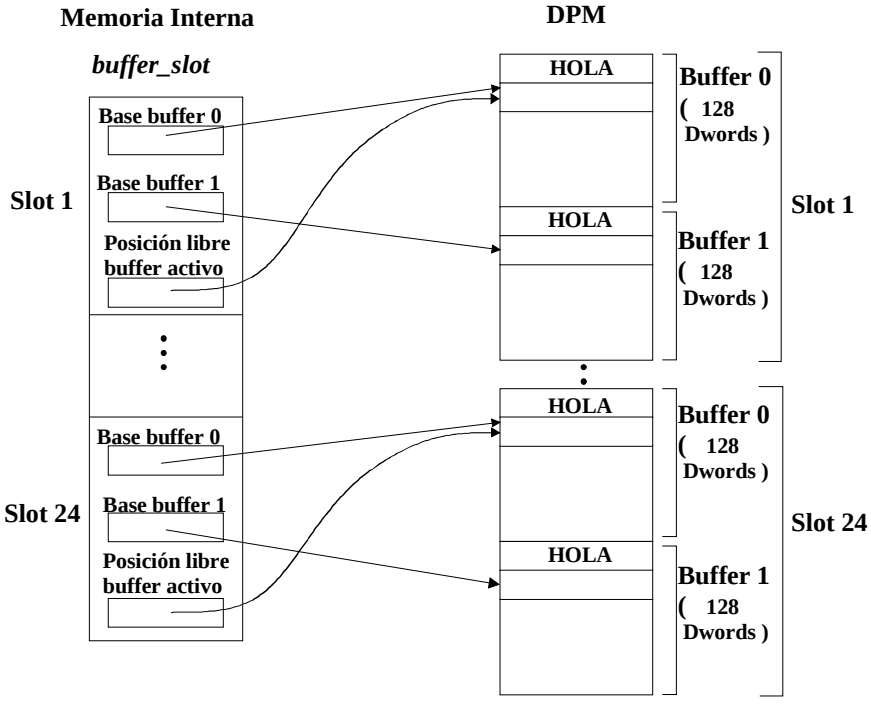

Figura 4.11(a) Estructura para la reconstrucción de paquetes provenientes de la red.

A través de la estructura *buffer_slot* (Figura 4.11(a)) se accede a la posición libre del *buffer* activo para el *slot* corriente. Así, cuando el DSP recibe una interrupción del receptor del SPORT indicando que existe una palabra proveniente de alguno de los *slots*, identifica el *slot* y escribe la palabra en la posición libre del *buffer* activo para ese *slot* (*buffer_slot[slot].posición_libre*). La estructura *buffer_activo* (Figura 4.11(b)) determina el *buffer* activo (*buffer* 0 o *buffer* 1) para el *slot* corriente. Como los *buffers* son solamente dos por cada *slot*, la estructura es un registro de bits en el cual cada posición identifica un *slot*. Si, por ejemplo, *buffer_activo* en la posición 1 contiene un 0, indica que el *buffer* activo para el *slot* 1 es el *buffer* 0, y si contiene un 1 significa que el *buffer* activo para el *slot* 1 es el *buffer* 1. El DSP inicializa la posición libre apuntando al *buffer* activo, indicado por dicha estructura, para cada *slot*.

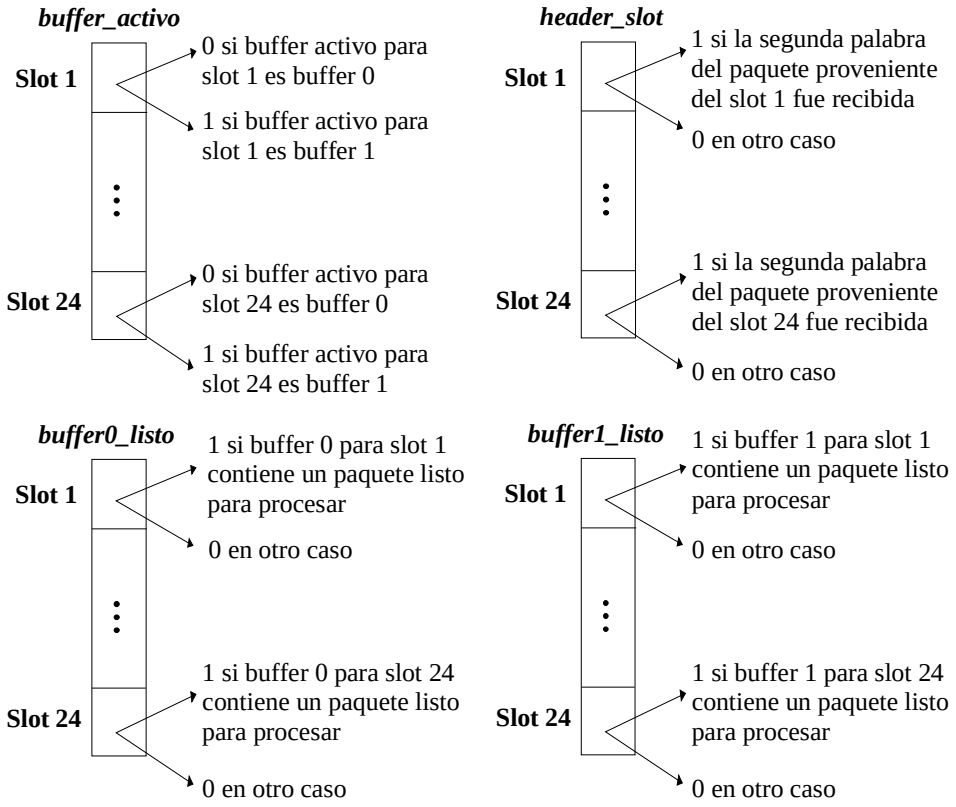

Figura 4.11(b) Estructuras para la reconstrucción de paquetes provenientes de la red.

Las estructuras *buffer0_listo* y *buffer1_listo* (Figuras 4.11(b)) se usan para indicar que un paquete completo fue recibido y está listo para ser procesado en alguno de los *buffer*. Tales estructuras son registros de bits en los cuales cada posición identifica un *slot*. Si, por ejemplo, *buffer0_listo* en la posición 1 contiene un 1 indica que para el *slot* 1 hay un paquete listo para ser procesado en el *buffer* 0 de ese *slot*.

La segunda palabra del paquete determina si el paquete es corto o largo, por lo tanto dicha palabra debe ser identificada puesto que si el paquete es corto implica que un paquete completo fue recibido en el *slot* corriente, en cambio si el paquete es largo continúa la reconstrucción del paquete para ese *slot*. La estructura *header_slot* se usa para indicar si la segunda palabra del paquete para el *slot* corriente fue recibida. Tal estructura es un registro de bits en el cual cada posición identifica un *slot*. Si, por ejemplo, el *slot* corriente es el 1 y *header_slot* en la posición 1 contiene un 0 indica que para ese *slot* la segunda palabra del paquete no fue recibida, en consecuencia se debe chequear el bit correspondiente de la palabra recibida para determinar si el paquete es corto o largo. Sólo si el paquete es largo, se escribe un 1 en la posición 1 de *header_slot* indicando que la segunda palabra de un paquete largo para el *slot* 1 fue recibida.

La Figura 4.12 resume los pasos principales ejecutados por la rutina que atiende la interrupción generada por el receptor del SPORT indicando la llegada de una palabra desde alguno de los *slots* de la red.

1. *Obtiene el slot por medio del cual el SPORT recibió la palabra. Uno de los registros del puerto serial contiene el slot de recepción corrientemente en uso.*
2. *Lee la palabra desde el buffer de recepción del SPORT.*
3. *Si la palabra recibida es "HOLA", indica el inicio de la transmisión de un paquete desde la Bora correspondiente al slot corriente. El DSP deshabilita la palabra clave en el slot corriente y retorna de la rutina de interrupción.*

4. *Si la palabra recibida no es "HOLA", la almacena en la posición libre del buffer activo para el slot corriente (buffer_slot[slot].posición_libre) y actualiza la posición libre del buffer.*

5. *Si la palabra recibida es la segunda palabra del paquete para el slot corriente (header_slot[slot]=0) y corresponde a un paquete corto, o es la ultima palabra de un paquete largo ("CHAU"), indica que un paquete completo (corto o largo) fue recibido. El DSP habilita la palabra clave en el slot corriente, informa que un paquete está listo para ser procesado en uno de los buffers correspondientes al slot corriente (buffer0/1_listo[slot]=1), actualiza el buffer activo para ese slot (buffer_activo[slot]=0/1), inicializa la posición libre apuntando al comienzo del nuevo buffer activo para el slot, y retorna de la rutina de interrupción.*

6. *Si la palabra recibida es la segunda palabra de un paquete largo, indica usando la estructura header_slot[slot] que la segunda palabra de un paquete largo fue recibida y retorna de la rutina de interrupción. La estructura es reinicializada en el slot corriente cuando la última palabra del paquete largo fue recibida.*

Figura 4.12 Pasos principales ejecutados por el DSP de Dolina para la reconstrucción de paquetes provenientes de la red de DSP.

4.4 Señales de Sincronización para la Adquisición de Datos

Los DSP de Dolina distribuyen a todas las Bora las señales que sincronizan el sistema de adquisición del RICH con el resto del experimento durante la adquisición de datos. Para la sincronización con el sistema global de adquisición, todas las Bora deben recibir tales señales dentro de intervalos de tiempo previamente especificados. Por lo tanto, *broadcasting* se implementa para la transmisión de las señales de sincronización a todos los DSP que conforman la red. Un esquema de las etapas involucradas en la adquisición de datos se muestra en la Figura 4.13. En general, la adquisición de datos sólo toma lugar durante las fases de *spill* del acelerador, cada *spill* dura 5,1 segundos. Los *spill* se dividen por periodos de *interspill* cuya duración es de 11,7 segundos. Varios *spills* son agrupados formando un *run*.

La señal de inicio de *run* determina el comienzo del *run* y existe al menos 1 segundo de tiempo entre que la señal se genera desde la sala "DAQ Control" y el inicio del primer *spill* del *run*. Las señales de inicio y fin de *run* llegan a la PC por medio de la red de área local del CERN, desde donde el "Controlador del RICH" las

transmite a Dolina como comandos, usando paquetes cortos con destinación el DSP de Dolina. Los DSP de Dolina reciben los comandos de inicio y fin de *run*, y los envía a todas las Bora de la red.

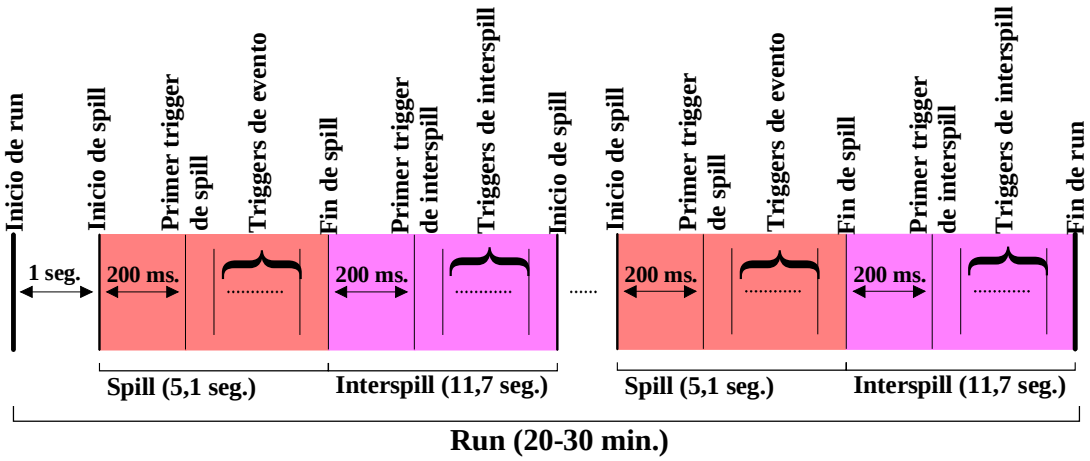

Figura 4.13 Etapas de la adquisición de datos (*run*, *spill*, *interspill*)

Las señales de inicio y fin de *spill* determinan el inicio y el final de cada *spill* del *run* respectivamente. Cada uno de los DSP de Dolina recibe tales señales, desde una tarjeta TCS, a través de dos líneas de interrupción externa del DSP. La señal de inicio de *spill* (SOS) llega al DSP a través de la línea de interrupción IRQ2 y la señal de fin de *spill* (EOS) llega al DSP a través de la línea de interrupción IRQ1. Existen 200 milisegundos de tiempo entre que la señal se genera desde el TCS y el primer trigger del *spill* o del *interspill*. El DSP transmite las señales SOS y EOS a todas las Bora de la red cuando recibe las interrupciones IRQ2 y IRQ1 respectivamente.

Ambas interrupciones permanecen deshabilitadas hasta el momento en que el DSP recibe el comando de inicio de *run* y en ningún momento permanecen habilitadas contemporáneamente. Una vez que el DSP transmite a la red el comando de inicio de *run*, habilita solamente la interrupción asociada con la señal SOS. Luego de recibir y transmitir a la red el comando de inicio de *spill*, deshabilita la interrupción asociada con la señal SOS y habilita la interrupción asociada con la señal EOS. Del mismo

modo, la recepción y transmisión a la red de la señal de fin de *spill* implica deshabilitar la interrupción asociada con la señal EOS y habilitar la interrupción asociada con la señal SOS. Finalmente, ambas interrupciones permanecen deshabilitadas con el arribo del comando de fin de *run* al DSP.

Para la transmisión de las señales de inicio y fin de *spill* a la red se utilizan paquetes especiales conteniendo una única palabra de 32 bits, la palabra “HOLJ” identifica el comando SOS y la palabra “HOLE” identifica el comando EOS. Luego de transmitir los comandos de inicio y fin de *spill* a todas las Bora de la red, el DSP los transmite también a la PC.

El software del DSP envía a todas las Boras de la red los comandos de sincronización para la adquisición de datos utilizando SPORT DMA, pero a diferencia de la transmisión de paquetes a una Bora en particular habilitando solamente el *slot* correspondiente a esa Bora, en este caso la transmisión se efectúa en todos los *slots* de la red. Para este propósito el software del DSP reserva en su memoria interna cuatro *buffers* de datos ya inicializados, uno para cada comando. Los *buffers* para los comandos de inicio de *run* (SOR) y fin de *run* (EOR) contienen 48 palabras de 32 bits cada uno, y los *buffers* para los comandos SOS y EOS contienen 24 palabras de 32 bits cada uno. La Tabla 4.1 muestra el contenido del *buffer* utilizado para transmitir el comando SOR (el *buffer* para la transmisión de EOR es igual variando solamente el código del comando).

El primer grupo de 24 palabras en el *buffer* contiene la réplica de la primer palabra del paquete de comando y el segundo grupo de 24 palabras contiene la réplica de la segunda palabra del paquete de comando en las que varía solamente el campo usado para la destinación del paquete. La primer palabra del *buffer* se transfiere en el *slot* 1 dirigida entonces al DSP número 1 de la red, y así siguiendo hasta la 24-esima

palabra que se transmite en el *slot* 24 dirigida al DSP número 24 de la red. La transmisión de la 24-esima palabra implica que todos los DSP de la red recibieron la primer palabra del paquete. La transmisión sigue en el *slot* 1 con la 25-esima palabra del *buffer* dirigida al DSP número 1 de la red como lo indica el campo de destinación del paquete, y así siguiendo hasta transmitir la 48-esima palabra en el *slot* número 24 dirigida al DSP número 24 de la red.

1	H		O	L	A
2	H		O	L	A
	…				
23	H		O	L	A
24	H		O	L	A
25	1	SOR	0x00	PC Id	Bora DSP Id_1
26	1	SOR	0x00	PC Id	Bora DSP Id_2
	…				
47	1	SOR	0x00	PC Id	Bora DSP Id_23
48	1	SOR	0x00	PC Id	Bora DSP Id_24

Tabla 4.1 *Buffer* de transmisión de la señal SOR (*broadcasting*)

La Tabla 4.2 muestra el contenido del *buffer* utilizado para transmitir el comando SOS (el *buffer* para la transmisión de EOS es igual variando solamente el código del comando). Cada palabra contiene el comando SOS, la primer palabra del *buffer* se transfiere en el *slot* 1 dirigida al DSP número 1 de la red, y así siguiendo hasta la 24-esima palabra que se transmite en el *slot* 24 dirigida al DSP número 24 de la red.

1	H	O	L	J
2	H	O	L	J
	…			
23	H	O	L	J
24	H	O	L	J

Tabla 4.2 *Buffer* de transmisión de la señal SOS (*broadcasting*)

La Figura 4.14 resume los pasos principales ejecutados por el software del DSP de Dolina para la transmisión a la red de los comando de sincronización para la adquisición de datos.

1. Establece los parámetros del DMA con la dirección base del buffer correspondiente al comando, la longitud del buffer en 48 o 24 dependiendo del comando y el incremento en 1.

2. Habilita todos los slots de transmisión correspondientes a los DSP de Bora que conforman la red (slots 1-24), usando el registro de selección de slots de transmisión (bits 1-24 = 1).

3. Habilita la interrupción de transmisión del SPORT.

4. Habilita el DMA.

Una vez habilitado el DMA, el controlador de DMA inicia la transferencia del buffer al buffer de transmisión del SPORT desde donde las palabras se transmiten en todos los slots habilitados. Cuando el buffer completo fue transferido, se deshabilita automáticamente el DMA y se genera la interrupción de transmisión del SPORT. La rutina que atiende la interrupción ejecuta los siguientes pasos:

1. Deshabilita los slots usando el registro de selección de slots de transmisión (bits 1-24 = 0).

2. Deshabilita la interrupción de transmisión del SPORT.

Figura 4.14 Pasos principales ejecutados por el DSP de Dolina para la transmisión a la red de los comandos de sincronización para la adquisición de datos.

4.5 *WatchDog Timer*

Cada uno de los DSP de Dolina controla un dispositivo *WatchDog* usando un *timer* del procesador. El DSP incluye dos *timers* programables que pueden configurarse en modo que generen una interrupción interna en intervalos preestablecidos de tiempo. El *timer* tiene un registro, denominado periodo, en el cual se especifica la duración de dicho intervalo en ciclos de reloj del procesador (16,67 ns); como el registro es de 32 bits, el periodo máximo es de 71,5 s. A partir del momento en que el software habilita el *timer*, el procesador inicializa un contador en 0

y cuando el contador alcanza el número establecido en periodo, se genera la interrupción interna.

Para evitar el *reset* generado por el dispositivo *WatchDog* (Sección 4.1), el software del DSP debe acceder (leer o escribir) a alguna dirección del banco 1 de la memoria externa del procesador en intervalos que no superen los 1,6 segundos. Por lo tanto el periodo del *timer* se inicializa en 1 segundo y la rutina que atiende la interrupción del *timer* efectúa una lectura desde el banco 1 de la memoria externa, causando de este modo la reinicialización en 0 de los contadores de tiempo del dispositivo *WatchDog* y del *time*r.

4.6 Programación del DSP de Dolina

Cada DSP bootea desde una memoria EPROM. En el momento del *power-up* o al recibir un *reset*, el DSP carga automáticamente en su memoria interna, desde la memoria EPROM, un programa pequeño (*boot loader*) cuya función es configurar la memoria interna del DSP y cargar un nuevo programa que fue previamente almacenado en la EPROM. El nuevo programa, que reemplaza al *boot loader* en la memoria del procesador, es a su vez un cargador de programas *(loader)*. La función del *loader* es leer y cargar en la memoria interna del DSP un programa transmitido desde la PC. El *loader* permite la reprogramación del DSP en cualquier momento que sea requerido desde el “Controlador del RICH” (Sección 3.4.1).

La memoria interna del DSP [90] se divide en memoria de datos formada por palabras de 32 bits y memoria de programa formada por palabras de 48 bits. Las instrucciones son de 48 bits. El vector de interrupciones del DSP forma parte de la memoria de programa ocupando direcciones fijas. Por lo tanto, un programa de DSP está conformado por datos (32 bits), instrucciones (48 bits) y el vector de

interrupciones (48 bits). La Tabla 4.3 muestra el formato de los paquetes usados para la transmisión del programa del DSP.

H		O	L	A
0	Programa DSP	Número de Paquete	Fuente	Destino
Dirección de Memoria				
Tipo	Sec.	Longitud		
Datos / Instrucciones				
C		H	A	U

Tabla 4.3 Paquete para la transmisión del programa del DSP

El cuerpo del paquete contiene los siguientes campos de encabezamiento:

a) Dirección de Memoria (32 bits) especifica la dirección de la memoria interna del DSP donde comienza el bloque de datos o instrucciones contenido en el paquete.

b) Tipo (4 *bits*) identifica si el paquete contiene datos o instrucciones. Si se trata de datos pertenecen a la memoria de datos (32 bits), en cambio si son instrucciones pertenecen a la memoria de instrucciones (48 bits).

c) Secuencia (4 *bits*) indica si el paquete es el primero del programa, uno intermedio o el último del programa.

d) Longitud (24 *bits*) especifica el número de palabras de 32 bits si el paquete contiene datos, o el número de instrucciones de 48 bits si el paquete contiene código.

El *loader* almacena el contenido de los paquetes en las direcciones de memoria interna apropiada e inicia la ejecución del programa. Antes de aceptar un paquete, realiza la siguiente serie de chequeos: chequea que la primer palabra del paquete sea la palabra “HOLA”, que la destinación del paquete corresponda al

identificador del DSP, que después del *booting* llegue el comando de inicio desde la PC, que los paquetes sean “largos” luego de recibir el comando de inicio y el tipo de datos de los paquetes corresponda al código “Programa del DSP”, que el número del paquete siga la secuencia correcta, y que la última palabra de los paquetes sea la palabra “CHAU”. Si detecta algún error envía el mensaje a la PC con su correspondiente código de error.

Los paquetes contienen palabras de 32 bits y las instrucciones son de 48 bits, por lo tanto cada instrucción en el paquete abarca una palabra y media, y dos instrucciones se almacenan en tres palabras del paquete. Cuando el paquete contiene instrucciones, el *loader* considera el modo en que las mismas se empaquetan para copiarlas a la memoria de programa. Como la PC se comunica con el *loader* a través de una interrupción del DSP, el vector de interrupciones correspondiente al programa cargado es el último que el *loader* copia en la memoria de programa.

4.6.1 Generación de los Paquetes que Contienen el Programa del DSP

Para generar los paquetes que contienen el programa del DSP, se lleva a cabo la siguiente secuencia de tareas mostrada esquemáticamente en la Figura 4.15:

Figura 4.15 Proceso de conversión de archivos para la generación de los paquetes que contienen el programa del DSP

1. Desde el programa del DSP en formato ejecutable (*.dxe*) se genera un archivo en formato *byte-stacked* (*.stk*) usando un utilitario, denominado *splitter*, provisto por

el *Visual DSP Development Tool* [92]. El *byte-stacked* es un formato de archivo standard usado para la transferencia de datos en microcontroladores.

2. El archivo "*.stk*" se convierte en un archivo en formato de paquete (*.pkt*) usando un utilitario, denominado *convert*, desarrollado específicamente para generar un archivo conteniendo los paquetes para enviar al DSP.

El archivo generado por el utilitario *convert* incluye el cuerpo de todos los paquetes necesarios para almacenar el programa del DSP. Antes de enviar cada paquete al DSP, el "Controlador del RICH" completa el paquete agregándole las palabras de encabezamiento y *trailer*. De este modo, no se necesita repetir la conversión para cada destinación si el programa ejecutable enviado a todos o a un grupo de DSP es el mismo.

4.6.2 *Loader* para el DSP de Dolina

El *loader* carga en la memoria interna del procesador e inicia la ejecución del programa del DSP. Basado en los requerimientos de memoria y considerando que convive con el programa del DSP, el *loader* fue escrito en el lenguaje *assembly* del procesador. La Figura 4.16 muestra un esquema de la memoria interna del DSP de Dolina. En la memoria de programa la zona reservada al vector de interrupciones abarca las direcciones 0x8000 – 0x80ff. El *loader* ocupa las direcciones 0x8100 – 0x82ff que incluyen una zona de almacenamiento temporario que el *loader* utiliza para el vector de interrupciones del programa del DSP. El programa del DSP se carga a partir de la dirección 0x8300. En la memoria de datos el *loader* ocupa las direcciones superiores 0xdfe0 – 0xdfff, quedando el resto de la memoria a disposición del programa.

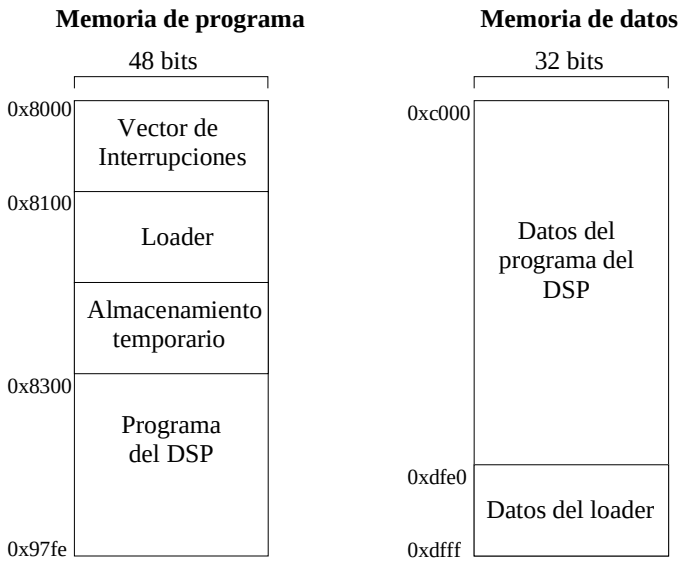

Figura 4.16 Memoria interna en el DSP de Dolina.

El programa principal del *loader* habilita la interrupción IRQ0 para la comunicación con la PC y permanece esperando por el comando de inicio (Figura 3.8). A través de la interrupción el *loader* conoce de la existencia de un paquete en la DPM.

La rutina que atiende la interrupción (ISR) chequea el paquete e informa al programa principal del arribo del paquete a través de un *flag*, indicando además las características del paquete, o en caso de detectar algún error el código del mismo. El programa principal responde al comando de inicio con un comando de requerimiento del programa. Si el paquete recibido es “largo”, el programa principal copia el contenido en las direcciones apropiadas de la memoria interna del DSP. La única excepción es cuando el paquete contiene el vector de interrupciones en cuyo caso almacena el contenido en la zona de almacenamiento temporáneo. Luego de copiar el último paquete de programa, el programa principal transfiere el vector desde la zona de almacenamiento temporáneo a las direcciones reservadas al vector de

interrupciones en la memoria de programa. Finalmente envía a la PC el comando de reconocimiento e inicia la ejecución del programa.

Si algún error es detectado durante la recepción de paquetes, el programa principal informa a la PC enviando el paquete de error correspondiente y genera un *reset* por software ("autoreset") iniciando nuevamente el *booting* del DSP.

4.7 Estructura del Software del DSP e Interrupciones

El software del DSP de Dolina implementa un sistema de tiempo real que satisface restricciones de tiempo duras y blandas. Un sistema es de tiempo real cuando se requiere que complete su trabajo y cumpla con sus servicios respetando ciertas restricciones temporales. Una restricción de tiempo es dura si requiere la validación que el sistema cumpla siempre con tal restricción, el incumplimiento de restricciones de tiempo duras afecta directamente la utilidad de los resultados producidos y en consecuencia la performance total del sistema. Las restricciones de tiempo blandas son, en cambio, generalmente expresadas en términos probabilísticos y su incumplimiento es indeseable pero no afecta drásticamente la performance del sistema [39].

El software responde a las siguientes seis interrupciones (internas e externas), representadas esquemáticamente en la Figura 4.17:

- La interrupción externa IRQ0 comunica la existencia de un paquete en la DPM proveniente de la PC de control.
- Las interrupciones externas IRQ2 y IRQ1 comunican el arribo de las señales de "inicio de *spill*" y "fin de *spill*" respectivamente provenientes del TCS.
- La interrupción interna del *timer* se usa para controlar el dispositivo *WatchDog*.

- La interrupción interna del transmisor del SPORT comunica el final de la transmisión de un bloque de datos, por el controlador DMA, a la red de DSP.
- La interrupción interna del receptor del SPORT comunica el arribo de una palabra de dato proveniente de la red de DSP.

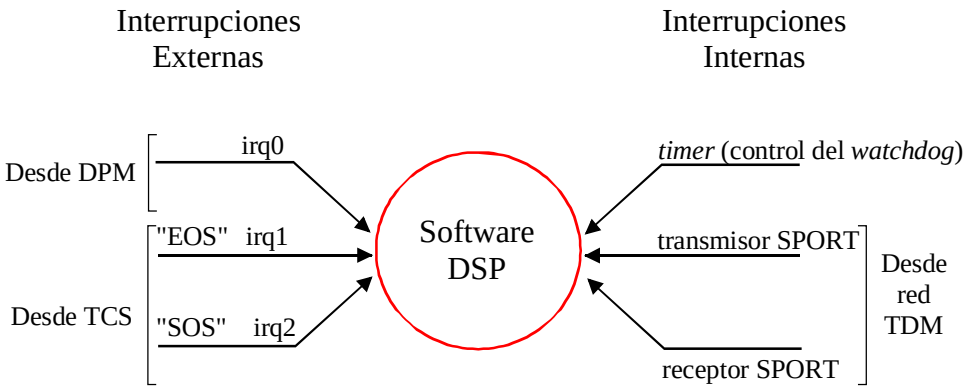

Figura 4.17 Interrupciones externas e internas generadas en el DSP de Dolina.

Las interrupciones en el DSP se habilitan individualmente y globalmente, y sus prioridades están prefijadas en el procesador [90]. Las interrupciones externas tienen la mayor prioridad y entre ellas la IRQ2 tiene mayor prioridad que la IRQ1, y la IRQ1 mayor que la IRQ0. Las interrupciones del SPORT tienen mayor prioridad que la interrupción del *timer* y entre ellas la interrupción del receptor tiene mayor prioridad que la del transmisor.

En Dolina, las interrupciones IRQ2 y IRQ1 sólo se habilitan durante el *run*, y en ningún momento ambas están habilitadas contemporáneamente. La IRQ0 permanece habilitada en todo momento pero en algún modo el software tiene control de dicha interrupción puesto que se genera sólo después que el DSP limpia el *mailbox* correspondiente a la PC de control. La interrupción del transmisor del SPORT se habilita sólo para transmitir un bloque de datos a la red, y es el controlador de DMA quien gestiona la transmisión permitiendo al procesador seguir con la ejecución

normal del programa. La interrupción del receptor del SPORT permanece habilitada en todo momento, teniendo en algún modo la mayor prioridad por la forma en que se gestiona el resto de las interrupciones. La interrupción del *timer* permanece también siempre habilitada pero la atención de la misma implica solamente un par de ciclos de reloj.

Para gestionar las interrupciones no se utilizó la opción de anidamiento, es decir, las rutinas de atención de interrupción (ISR) no se interrumpen quedando pendiente el servicio de una eventual nueva interrupción para el retorno de la ISR. Por lo tanto, el software se desarrolló en modo de usar el menor tiempo posible en las ISR dejando al programa principal (y sus rutinas) la mayor cantidad de actividades.

El *kernel* (núcleo) del software es el programa principal quien ejecuta dos etapas secuenciales. En la primer etapa inicializa los *buffers* y estructuras usadas por el programa, inicializa y habilita individualmente las interrupciones correspondientes y por último habilita las interrupciones globalmente. En la segunda etapa permanece en un ciclo infinito gestionando principalmente la comunicación entre la PC y la red TDM. En esta etapa, el arribo de alguna de las interrupciones habilitadas genera la automática ejecución de la ISR correspondiente. En general, la ISR modifica un *flag* indicando al *kernel* las tareas a ejecutar para completar el servicio de la interrupción.

El *kernel* chequea permanentemente en el ciclo una serie de *flags* y dependiendo de los valores de los mismos, invoca la tarea (una o más rutinas) correspondiente. A su vez, tales *flags* son limpiados una vez ejecutada la tarea correspondiente. De este modo se pueden distinguir los siguientes dos niveles de ejecución: nivel de interrupciones (*foreground*), en el cual las ISR ejecutan las operaciones críticas que involucran restricciones temporales y especifican a través de *flags* internos las tareas a realizar, y nivel de tareas (*background*) en el cual

principalmente se ejecutan las tareas que fueron requeridas a través de alguna de las interrupciones [39, 40]. La Figura 4.18 muestra esquemáticamente la estructura general del software.

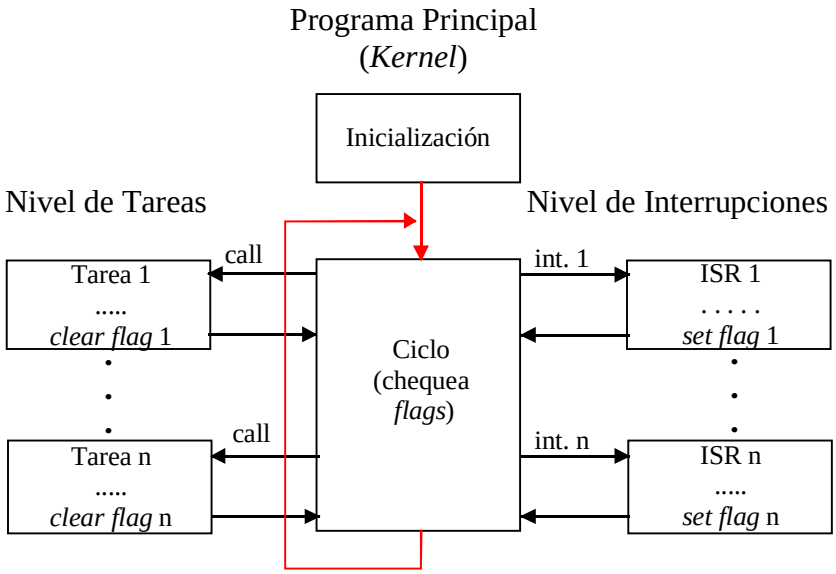

Figura 4.18 Esquema general de la estructura del software del DSP.

Capítulo 5

La Tarjeta Principal de Adquisición de Datos BORA

Las funciones principales de la adquisición, tratamiento y transmisión de las señales generadas por el RICH se realizan en 192 tarjetas Bora montadas en las cámaras de fotones del detector. El RICH consta de ocho cámaras, que en conjunto forman una matriz de 288 x 288 *pixels* en un área de 5,3 m^2. Las 82944 señales analógicas están físicamente disponibles a través de conectores eléctricos específicos. Cada conector conecta 48 canales correspondientes a 48 *pixels* distribuidos rectangularmente en un arreglo de 8 x 6 *pixels* (Figura 3.1). Una tarjeta Bora cada nueve conectores adquiere las señales provenientes de 432 canales del detector. Cada cámara se conforma por 24 tarjetas Bora.

Bora cuenta con chips de *front-end* (Gassiplex [86, 87, 88]) especialmente diseñados para el RICH. Las señales analógicas provenientes del detector se filtran y amplifican en tales chips antes de su digitalización. La tarjeta cuenta también con conversores analógico-digital (ADC [99]) a través de los cuales las señales adquiridas se digitalizan en 10 bits. La funcionalidad de Bora está a cargo de un DSP conjuntamente con una FPGA, que actúa como coprocesador del DSP. La reprogramabilidad del DSP y la reconfigurabilidad de la FPGA determinan la flexibilidad y versatilidad de la tarjeta. También la gran capacidad de cálculo

disponible permite a la Bora proveer diversas funciones más allá de la adquisición y transmisión de datos de evento al *trigger rate* requerido.

En este capítulo se describe la funcionalidad y la arquitectura de la tarjeta Bora. La Figura 5.1 muestra una fotografía de ambos lados de Bora, en la que se indican sus principales componentes los cuales se describen a lo largo del capítulo. En la descripción de la arquitectura de Bora, se dejan de lado mucho de los aspectos eléctricos y físicos a los efectos de no oscurecer los aspectos lógicos y funcionales de la tarjeta.

5.1 Descripción Funcional de Bora

La función principal de Bora es la adquisición, elaboración y transmisión de los eventos generados por el detector RICH, respondiendo a un *trigger rate* que puede alcanzar los 100 kHz. Varias circunstancias hacen que Bora no se agote en esta sola función. La gran cantidad de datos generados por unidad de tiempo hace necesario un filtrado *on-line* de los canales adquiridos a través de la comparación del valor digitalizado de cada canal con un umbral correspondiente. Tales umbrales se almacenan en la misma tarjeta y deben poder ser reajustados en cualquier momento según ciertos criterios experimentales. Generalmente los umbrales se definen a partir de los resultados obtenidos en la caracterización estadística de los canales del detector. Parte del potencial de cálculo del DSP se emplea para caracterizar estadísticamente cada uno de los canales adquiridos por Bora.

Bora provee los servicios necesarios para el estudio del RICH. Diversos parámetros físicos afectan determinantemente la performance del detector y estos deben ser optimizados antes de utilizar el instrumento durante el *run* del experimento. Entre estos parámetros figuran el nivel de ruido total observado en cada canal y el alto

voltaje (~ 2000 *volts*) aplicado a las cámaras de fotones, responsable de la amplitud de la señal generada pero también responsable de potenciales inestabilidades eléctricas.

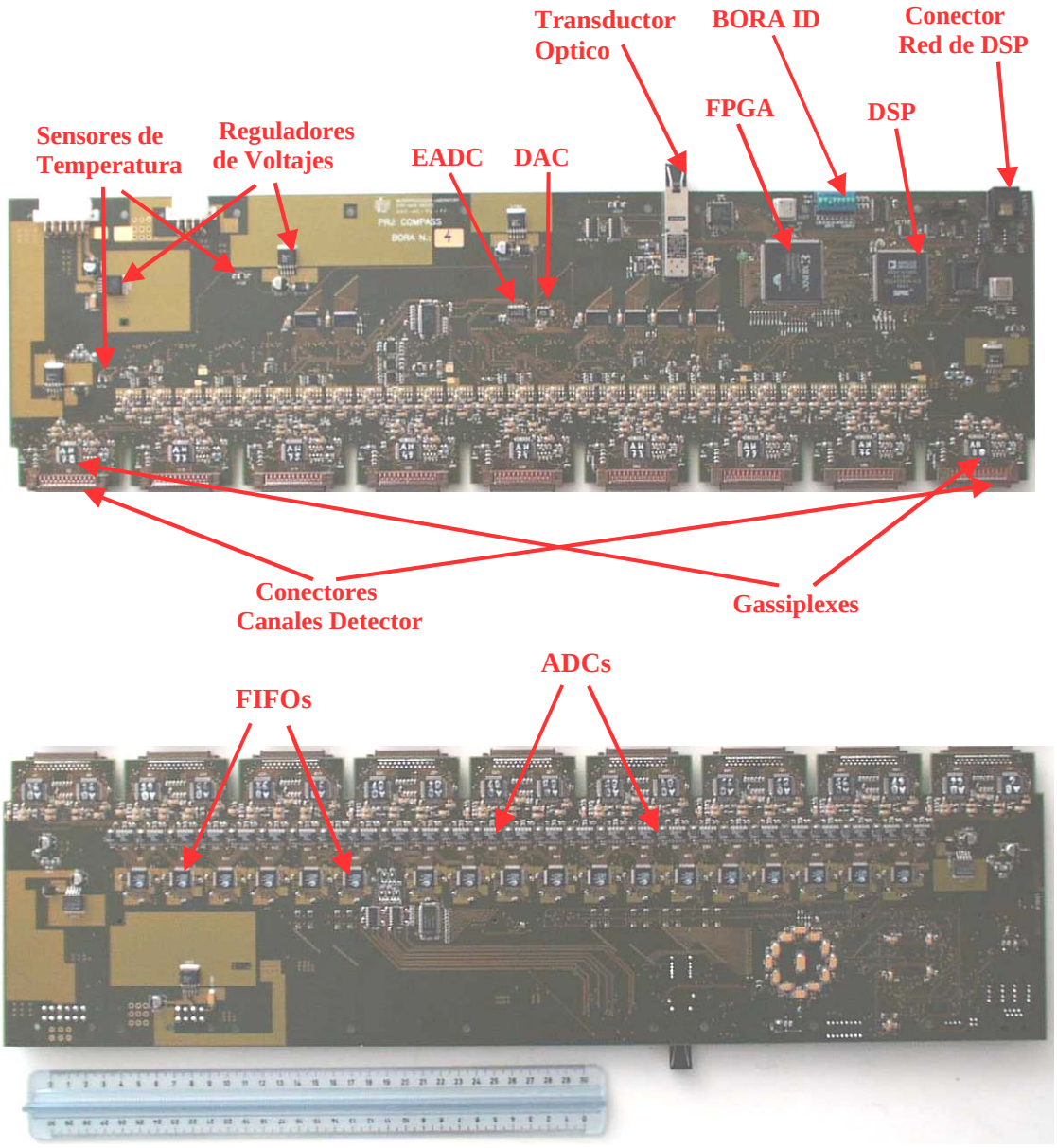

Figura 5.1 La tarjeta Bora

La complejidad intrínseca de Bora impone además que tenga un sistema de auto testeo por medio de la simulación de las señales provenientes del detector. Este sistema permite estudiar ciertos aspectos de los canales como linealidad y *peaking*

time. Bora mide también su temperatura en diversos puntos y distintos voltajes de alimentación, a los efectos de monitorear físicamente las tarjetas en modo remoto.

La Bora se comunica con la cadena de adquisición global de COMPASS a través de dos fibras ópticas que conectan la tarjeta con los dispositivos CATCH, y con la PC de control del RICH por medio de la red de DSP. Durante el *run* del experimento, la Bora recibe los comandos de sincronización (inicio y fin de *run* e inicio y fin de *spill*) a través de la red, y el *trigger* por una de las fibras ópticas. Por la otra fibra, transmite normalmente datos de evento y eventualmente transmite los resultados obtenidos de la caracterización estadística de los canales, los valores de umbrales corrientes almacenados en la tarjeta y las medidas de las temperaturas y los voltajes.

A través de la red de DSP se cargan en la Bora los programas del DSP, la cadena de bits para la configuración de la FPGA, y los valores de umbrales. Por su parte la Bora transmite hacia la PC de control, según sea requerido por medio de comandos específicos, los resultados obtenidos en la caracterización estadística de los canales analógicos y en el proceso de auto testeo (*self testing*), así como también los valores corrientes de umbrales almacenados en la tarjeta. Una vez por *spill* la Bora envía automáticamente hacia la PC, información del *spill*, las medidas de las temperaturas y los voltajes de la tarjeta, y los datos de un evento.

Desde el punto de vista de su funcionalidad, la tarjeta Bora se puede dividir en las siguientes actividades principales:

- adquisición, procesamiento y transmisión de datos,
- caracterización estadística de los canales analógicos,
- *self testing*,
- medición de parámetros físicos de la tarjeta,

- comunicación con la PC de control, y
- comunicación con los dispositivos CATCH.

5.1.1 Adquisición, Procesamiento y Transmisión de Datos

La adquisición de datos comienza con la señal de inicio de *spill* durante el *run* del experimento. Durante el *spill,* la Bora recibe las señales de *trigger* a través de una fibra óptica desde el módulo CATCH. El *trigger* indica a la Bora que los 432 canales analógicos deben ser simultáneamente adquiridos, procesados y transmitidos. Esta actividad se dividide en las siguientes dos fases:

a) Adquisición Propiamente Dicha

Esta fase involucra la amplificación y filtrado de las señales analógicas provenientes de los canales del detector, la digitalización de las mismas en 10 bits, y el almacenamiento temporáneo de los datos resultantes en memorias de tipo FIFO (CYFIFO [100]). Este proceso se activa con el *trigger* y debe respetar rigurosas restricciones temporales, durante el mismo existe una ventana de tiempo en la que la Bora no puede procesar un nuevo *trigger*. Dicha ventana de tiempo determina el tiempo muerto del instrumento.

Una parte de la FPGA está configurada para controlar este proceso autónomamente. El *trigger* inicia una máquina finita de estados que genera, en la secuencia correcta, las señales de control necesarias para los chips de *front-end*, los conversores ADC y la escritura en los CYFIFO. De este modo se puede garantizar un tiempo breve y deterministico de servicio de los *triggers*.

b) Procesamiento y Transmisión

Una vez memorizados los valores digitalizados de los canales en los CYFIFO, estos se leen y confrontan con sus correspondientes umbrales. Los valores de los canales que superan su umbral se empaquetan junto con el identificador del canal, conformando un evento, y se transmiten hacia el sistema global de adquisición y eventualmente hacia la PC de control. Los paquetes de eventos son de longitud variable e incluyen una palabra de *header* y otra de *trailer* que indican el inicio y el fin del paquete. Tales palabras contienen el número de evento y el identificador de la Bora. El procesamiento y transmisión de los eventos no se activa inmediatamente con el *trigger*, es el DSP quien inicia este proceso a través de la FPGA.

Los CYFIFO [1] tienen una capacidad para almacenar hasta 128 eventos, permitiendo de este modo absorber fluctuaciones del *trigger rate* sin perder eventos. Bajo el comando del DSP los datos del CYFIFO son leídos por la FPGA quien compara los valores de cada canal con su umbral correspondiente. El resultado de la substracción del umbral es transmitido solo en el caso en que sea positivo. En el capítulo 6 se describe el formato de los paquetes transmitidos. Estos paquetes son escritos simultáneamente en dos memorias internas de la FPGA administradas como FIFO. Una de estas memorias (*Hotlink FIFO*) es leída automáticamente y los datos leídos se escriben directamente en un dispositivo serializador (*Hotlink Transmitter* [29]), quien a su vez transmite los datos por una fibra óptica hacia el módulo CATCH. Los datos en la otra memoria (*DSP FIFO*) pueden ser eventualmente leídos por el DSP y transmitidos hacia la PC de control.

[1] El proceso de escritura es estocástico y responde a una distribución de *Poisson*. El proceso de servicio es determinístico y tiene un tiempo fijo de servicio por evento, y siendo el servidor único, el problema de colas correspondientes viene clasificado como M/D/1 [105].

5.1.2 Caracterización Estadística de los Canales Analógicos.

En ausencia de señal los canales analógicos tienen un valor medio estacionario propio, y en general diverso para cada canal, llamado *offset*. Esto se debe fundamentalmente a la inevitable falta de uniformidad espacial de los parámetros eléctricos en los chips de *front-end*. Estos canales analógicos exhiben, en ausencia de señal, una variación aleatoria de sus valores en función del tiempo que llamamos *ruido*. Los canales analógicos tienen un ruido propio y en general diverso para cada canal. Asumiendo que este ruido es estacionario, blanco y gaussiano, bastan dos parámetros para caracterizarlo completamente: el valor medio (μ') y la desviación estándar (σ'). Tales parámetros se definen a continuación, donde *x(t)* es el voltaje analógico a la salida del chip de *front-end* en función del tiempo:

$$\mu' = \lim_{T\to\infty} \frac{1}{2T} \int_{-T}^{T} x(t)dt$$

$$\sigma' = \lim_{T\to\infty} \sqrt{\frac{1}{2T} \int_{-T}^{T} (\mu - x(t))^2 dt}$$

Para estimar μ' y σ' se usan los estimadores discretos μ y σ respectivamente, evaluados a partir de una colección discreta de muestras $\{x_i\}$. Tales estimadores se definen a continuación, donde x_i representa el valor digitalizado de *x(t)* evaluado al tiempo de muestreo t_i, y N es el número total de muestras:

$$\mu = \frac{1}{N} \sum_{i=1}^{N} x_i$$

$$\sigma = \sqrt{\frac{1}{N-1} \sum_{i=1}^{N} (x_i - \mu)^2}$$

La tarjeta Bora obtiene y elabora las muestras $\{x_i\}$ de cada canal para calcular los estimadores μ y σ. A tal efecto se procede como en el caso de la adquisición normal de datos pero con las siguientes diferencias:

- El *trigger* no es externo sino que es generado por el DSP.
- No hay substracción de umbral sobre los datos.
- Los valores digitalizados de todos los canales adquiridos por cada *trigger* no son enviados al sistema global de adquisición, sino que son exclusivamente leídos y elaborados por el DSP.
- Resultados parciales de μ y σ para cada canal son transmitidos en paquetes específicos a la PC de control y al sistema global de adquisición para su elaboración final.

5.1.3 *Self Testing*

En la sección anterior se explicó un tipo especial de adquisición en el que no hay estímulo de ningún tipo a la entrada de los chips de *front-end*. Para explorar la funcionalidad de la tarjeta, la misma cuenta con un sistema de *self testing*. Un comando específico enviado desde la PC de control activa este proceso. En este caso el DSP, a través de la FPGA, acciona un subcircuito que estimula los chips de *front-end* a través de la simulación de una señal en las entradas analógicas similar a aquella proveniente del detector. Así el DSP puede generar posteriormente un *trigger* para adquirir los datos generados correspondientes, en modo análogo al caso explicado en la sección anterior.

Este procedimiento permite observar la funcionalidad de prácticamente toda la cadena de adquisición de Bora. Variando el intervalo de tiempo entre el estímulo de test y el *trigger* se puede explorar la respuesta temporal de cada canal. Así mismo

variando la intensidad del estímulo y midiendo la respuesta a través de la adquisición de los datos se puede explorar la linealidad del sistema. El comando para esta operación contiene como parámetros el número de muestras a ser adquiridas, la intensidad del estímulo generado y la demora entre el estímulo y el *trigger*.

5.1.4 Medición de Parámetros Físicos

Durante el *run*, el área experimental se vuelve inaccesible por lo que es importante contar con mecanismos de monitoreo físico de las tarjetas en modo remoto. A tal efecto la Bora cuenta con un conversor analógico-digital específico para medir cinco voltajes de alimentación analógicos y cuatro sensores de temperatura colocados en puntos estratégicos de la tarjeta. A través de tales dispositivos, el DSP obtiene las temperaturas y los voltajes para enviarlos a la PC de control y al sistema global de adquisición de datos donde se monitorean. Una vez por *spill*, el DSP arma y transmite un paquete denominado *Engineering Frame* conteniendo tales datos físicos e información del *spill* (número de *spill* y número de *triggers* por *spill*).

5.1.5 Comunicación con la PC de Control

La Bora se comunica con la PC de control a través del DSP que forma parte de una red TDM conformada por los 24 DSP de las Bora de una cámara del RICH y uno de los DSP de Dolina (Sección 3.3). Por medio de esta red, la Bora intercambia programas, datos, comandos y mensajes con la PC de control. En particular la Bora recibe desde la PC los programas para el DSP, la cadena de bits para la configuración de la FPGA, datos de umbrales, y comandos. A su vez la Bora transmite hacia la PC *engineering frames*, datos de eventos, los valores de umbrales corrientes almacenados

en la tarjeta, los resultados de la caracterización estadística de los canales analógicos y del proceso de *self testing*, y distintos tipos de mensajes.

5.1.6 Comunicación con los Dispositivos CATCH

La Bora se comunica con los dispositivos CATCH a través de dos fibras ópticas. Por una fibra la Bora recibe la señal de *trigger* y por la otra transmite datos hacia la cadena de adquisición global del experimento. La velocidad de transmisión a través de la fibra es de 400 Mbits/s.

La señal de *trigger* es un pulso de aproximadamente 25 ns de duración, y entra directamente en la FPGA durante el *run* del experimento para iniciar automáticamente la primer fase del proceso de adquisición. El DSP también recibe el *trigger*, a través de la FPGA, quien lo usa para contabilizar los eventos de un *spill* y generar el encabezamiento de los paquetes de evento que contienen el número del evento y el identificador de la Bora, y para iniciar en la FPGA la segunda fase del proceso de adquisición.

La FPGA escribe los paquetes de eventos en una memoria interna (*Hotlink FIFO*) desde donde se envían al *Hotlink Transmitter* quien a su vez pilota un transductor para fibra óptica. En ciertas circunstancias es el DSP quien escribe en el *Hotlink FIFO* para enviar datos específicos de la Bora hacia los dispositivos CATCH.

5.2 Arquitectura de la Tarjeta Bora

La arquitectura de la tarjeta Bora está esencialmente basada por un lado en la combinación DSP-FPGA responsables de la funcionalidad de la tarjeta y de la comunicación de la misma con el mundo externo (dispositivos CATCH y PC de

control), y por otro lado en la cadena de adquisición de datos que comprende la adquisición analógica de las señales generadas por el detector, la digitalización de las mismas en 10 bits y el almacenamiento temporáneo de los valores digitalizados. La Figura 5.2 muestra un diagrama de bloques conteniendo los principales componentes de Bora y sus interconexiones.

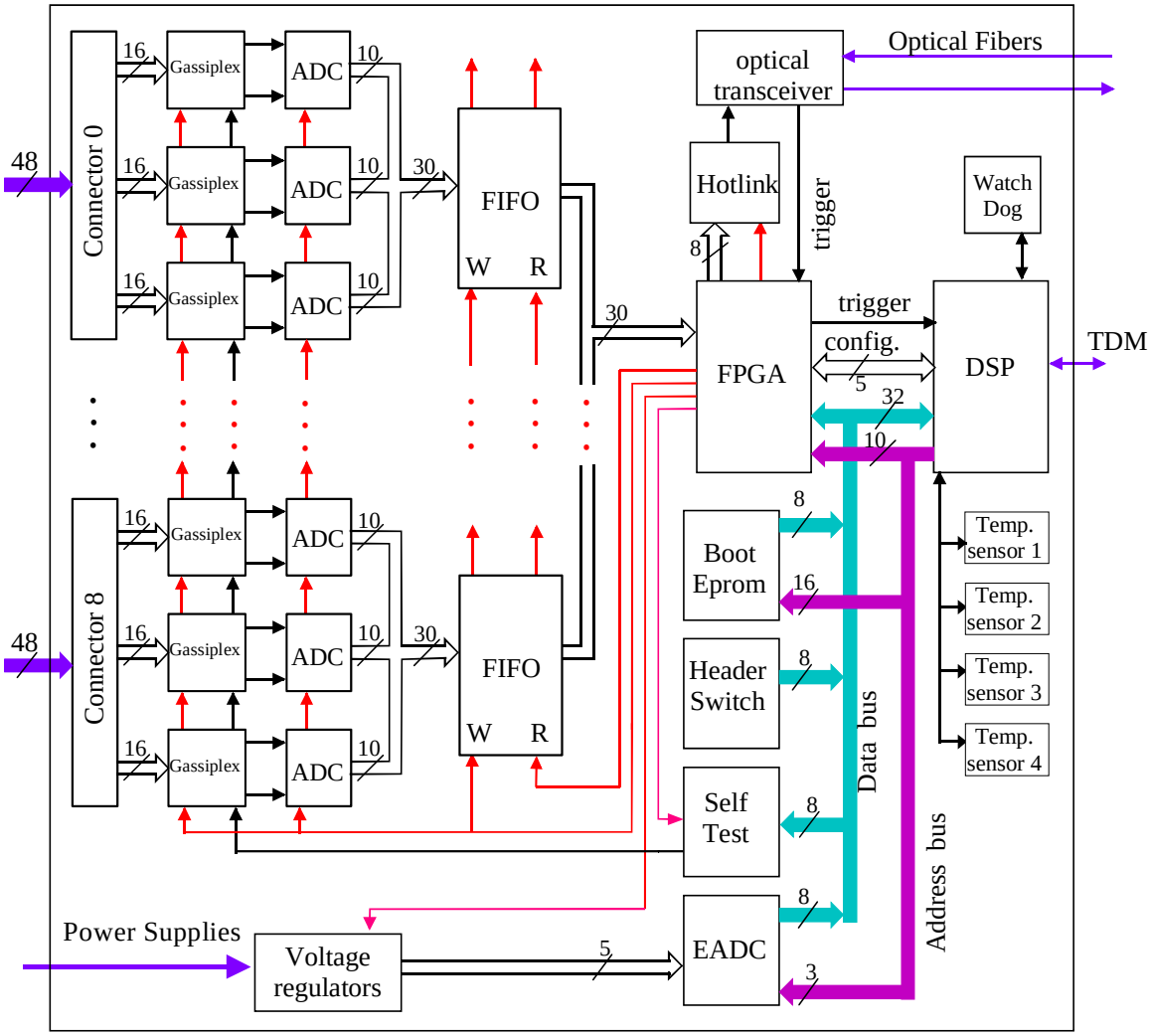

Figura 5.2 Diagrama a bloques de la arquitectura interna de Bora (las líneas rojas identifican señales de control).

La interfase entre el DSP y la FPGA se efectúa principalmente a través del *bus* de datos y el *bus* de direcciones externos del DSP. Además, puesto que es el DSP el

responsable de la configuración de la FPGA, los cinco puertos de configuración de la FPGA están conectados a cinco de los *flags* externos de propósito general del DSP.

Las señales analógicas generadas por el detector se adquieren por Bora a través de 9 conectores. La cadena de adquisición de Bora comienza por 27 chips de *front-end* (Gassiplex) seguidos por 27 conversores analógico-digital (ADC) y 9 memorias FIFO (CYFIFO). La gestión de tales componentes está a cargo de la FPGA quien provee todas las señales de control necesarias para la adquisición, digitalización y almacenamiento temporáneo en los CYFIFO de los valores digitalizados de los 432 canales adquiridos por Bora. A su vez, las salidas de los CYFIFO se conectan a la FPGA a través de un *bus* de lectura específico.

Dos fibras ópticas conectan la Bora con los dispositivos CATCH. Por una de las fibras la Bora envía datos, y por la otra recibe el *trigger*. La FPGA, un chip serializador (*Hotlink*) y un transductor óptico (*optical transceiver*) gestionan la transmisión a través de la fibra óptica, y por medio del transductor la FPGA recibe la señal de *trigger*. A su vez, el *trigger* se retransmite desde la FPGA al DSP quien lo recibe por medio de una de las líneas de interrupción externa del procesador.

La comunicación entre Bora y la PC de control se efectúa por medio de la red TDM de DSP implementada a través de uno de los puertos seriales del DSP (Sección 3.3).

El *booting* del DSP se efectúa, como en Dolina, desde una memoria EPROM conectada al *bus* de datos y de direcciones externos del procesador. La memoria externa del DSP está dividida en cuatro bancos de tamaño fijo (banco 3-0) y cada banco tiene asociado una línea de selección MS 3-0. La interfase entre el DSP y la memoria externa se efectúa principalmente por medio del *bus* de datos y el *bus* de direcciones externos, una señal de escritura (WR) y una señal de lectura (RD). El

acceso por parte del DSP a alguna dirección perteneciente a alguno de estos bancos genera una activación de la línea de selección correspondiente.

El DSP controla un dispositivo *WatchDog*, del mismo tipo que el usado en Dolina, a través de la línea de selección MS3 correspondiente al banco 3. Bora cuenta con un *header switch* mapeado también en el banco 3 por medio del cual el DSP obtiene el identificador de la Bora. Durante el proceso de *self testing*, el DSP genera el estímulo de test aplicado a los chips de *front-end* a través de un conversor digital-analógico (DAC) mapeado en el banco 2. Para la medición de los voltajes de alimentación internos, el DSP utiliza un conversor analógico-digital (EADC) mapeado en el banco 1. La FPGA está mapeada en el banco 0.

Para la medición de las temperaturas en cuatro puntos distintos de la tarjeta, cuatro sensores de temperatura están conectados a uno de los *flags* externos de propósito general del DSP programables como puertos de entrada o salida.

En las siguientes secciones se describen en más detalle los principales bloques de la arquitectura de Bora los cuales interactúan con el DSP y la FPGA. En el capítulo 6 se da una explicación detallada del software del DSP de Bora y su interacción con la FPGA, y en el capítulo 7 se describe la funcionalidad y arquitectura interna de la FPGA.

5.2.1 Cadena de Adquisición de Datos de Bora

La cadena de adquisición de datos de Bora comprende los chips de *front-end* (Gassiplex), conversores ADC *dual-channel* de 10 bits y las memorias FIFO (CYFIFO). La Figura 5.3 muestra la cadena de adquisición de los 48 canales analógicos correspondientes a uno de los nueve conectores de Bora.

El Gassiplex gestiona 16 canales analógicos en ingreso, y cuenta con un circuito de "*track and hold*" para cada canal y dos multiplexores analógicos. Cada canal se filtra en modo de maximizar la relación señal-ruido teniendo en cuenta las características específicas del detector RICH. Al ingreso se espera un pulso corto de corriente que oportunamente integrado alcanza algunos miles de electrones. Como consecuencia del filtrado este pulso de corriente se convierte en un pulso de voltaje semigaussiano que alcanza su pico después de un tiempo característico llamado *peaking time*. Es en el pico de este pulso que se obtiene la relación señal-ruido más alta. Este tiempo es independiente de la intensidad del pulso y es del orden de 1.16 µs.

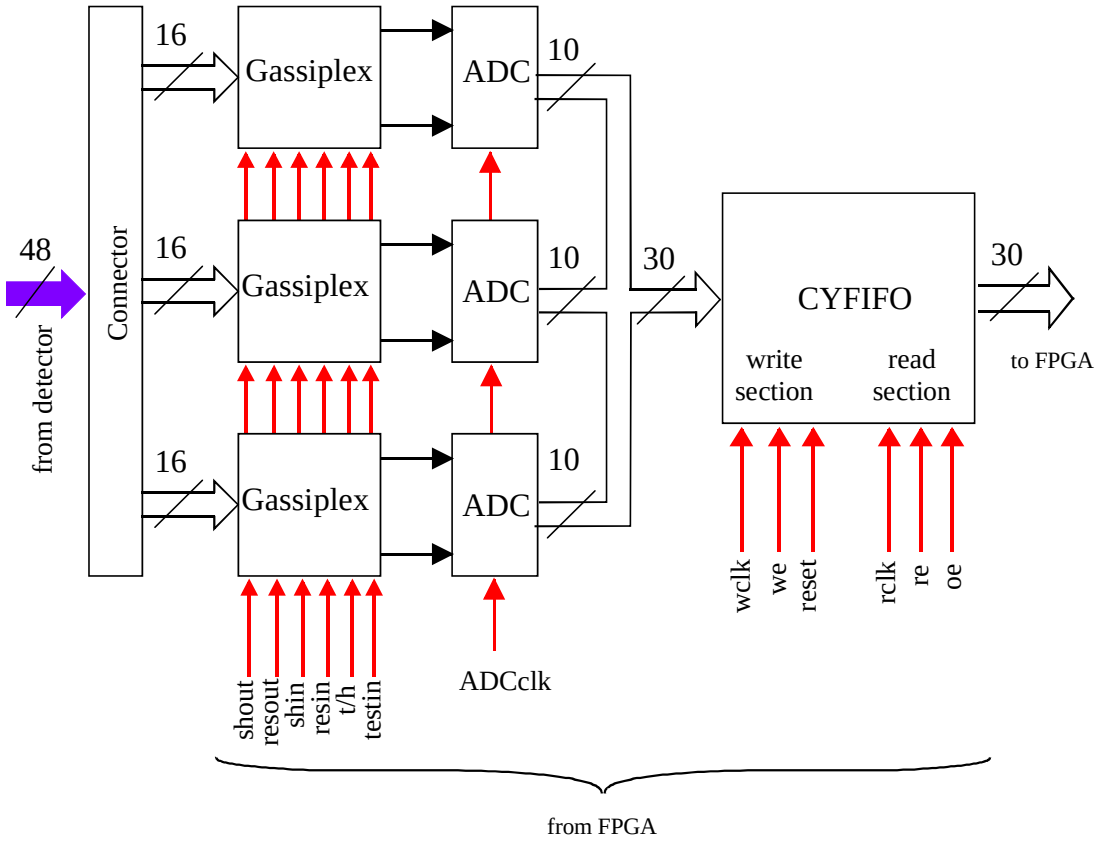

Figura 5.3 Cadena de adquisición de los 48 canales analógicos correspondientes a un conector de Bora.

La señal de *track and hold* (*t/h*) se usa para retener (*hold*) los valores analógicos amplificados y filtrados al instante deseado. Los dos multiplexores analógicos trabajan en paralelo y permiten seleccionar secuencialmente las salidas de un par de canales a la vez. Así los pares de canales 1 y 9, 2 y 10 hasta 8 y 16 se obtienen activando una señal de reloj específica llamada *shout* (*shift out*). Estos dos canales analógicos son muestreados sincrónicamente por el ADC quien los digitaliza a 10 bits según una señal de reloj llamada *ADCclk* a una frecuencia de 20 MHz. La arquitectura *pipeline* del ADC hace que los resultados de la conversión aparezcan a la salida tres ciclos de reloj después del muestreo. Estos dos resultados salen multiplexados en el tiempo, uno en el flanco positivo del *ADCclk* y otro en el flanco negativo. Los resultados de la conversión de tres ADC, correspondientes a un conector, se concatenan formando una palabra de 30 bits que se escribe directamente en un CYFIFO.

Las 432 señales del detector gestionadas por Bora se adquieren por 27 chips Gassiplex (3 por cada conector), y cada Gassiplex tiene su conversor ADC. Las salidas de nueve grupos de tres ADC se escriben simultáneamente en nueve CYFIFO. Las señales de control necesarias para la cadena de adquisición de Bora se generan en la FPGA, siguiendo una secuencia iniciada por un *trigger* y determinada por la tempística de los chips involucrados en este proceso. Los datos almacenados en los CYFIFO son disponibles para su posterior lectura por la FPGA. La secuencia hasta la escritura de los CYFIFO es única y todos los CYFIFO se escriben en paralelo. Por el contrario, la lectura de los CYFIFO se hace secuencialmente, uno a la vez, a través de un *bus* de lectura específico conectado a la FPGA. Teniendo en cuenta que un evento usa 16 palabras de cada FIFO y que los CYFIFO tienen una profundidad de 2048

palabras, los mismos cuentan con una capacidad de almacenamiento temporáneo de 128 eventos.

5.2.2 La FPGA y su Interfase con la Cadena de Adquisición y el Bloque de Comunicación con los CATCH

La FPGA es un sofisticado dispositivo de lógica programable perteneciente a la familia Virtex (VIRTEX 100 [93]) que cuenta con una capacidad equivalente a 100000 compuertas lógicas y 160 puertos genéricos de entrada-salida completamente programables. Esta FPGA tiene, además de los recursos lógicos reconfigurables, 10 bancos de memoria *dual-port* de 512 x 8-bit y cuatro DLL (*Delay Lock Loop*). Se alimenta a 3,3 *volts* para las estructuras de entrada-salida y a 2,5 *volts* para el *core*. A la FPGA se conectan dos relojes externos de 30 MHz y 40 MHz provenientes de dos osciladores de cristal. Tales relojes son usados al interno de la FPGA por las diversas máquinas de estados que la integran.

La FPGA se conecta con la cadena de adquisición de Bora por medio de un *bus* de lectura de 30 bits y un conjunto de señales de control. Tales señales controlan los Gassiplex, los ADC y los CYFIFO. El *bus* de lectura de 30 bits está implementado con tres *buffers tristate* en una estructura de árbol como se muestra en la Figura 5.4.

Actuando sobre los *output enable* (*OE*) de los CYFIFO y de los *buffers tristate*, la FPGA lee en secuencia los CYFIFO conteniendo los valores digitalizados de los canales del detector para elaborarlos internamente. El CYFIFO provee dos *flags* que indican su estado: lleno (*full*) y vacío (*empty*). Tales señales están disponibles en un registro interno de la FPGA, permitiendo al DSP consultar el estado de los CYFIFO.

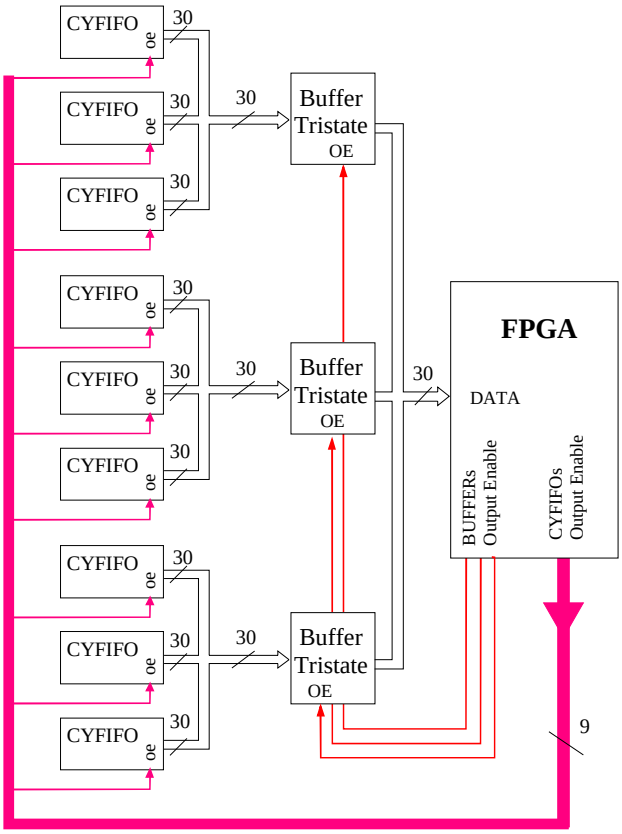

Figura 5.4 *Bus* de lectura a través del cual la FPGA accede a los CYFIFO.

Bora se conecta con los dispositivos CATCH por medio de dos fibras ópticas. La FPGA, un chip serializador (*Hotlink*) y un transductor óptico (*optical transceiver*) gestionan la transmisión a través de la fibra óptica. La Figura 5.5 muestra esquemáticamente la conexión entre la FPGA y el bloque de comunicación con los dispositivos CATCH (*Hotlink y Optical Transceiver*). La FPGA se conecta con el *Hotlink* por medio de un *bus* de escritura de 8 bits, una señal de *chip select* (*ENA*) y un reloj específicamente generado de 40 MHz (*CKW*). El dispositivo transmisor *Hotlink* sirve a una conexión punto a punto sobre una línea serial de alta velocidad (fibra óptica). La conexión alcanza una velocidad de transmisión efectiva de 360 Mbits/s. El *Hotlink* serializa los datos escritos por la FPGA y pilota directamente un transductor óptico que transmite, a través de una de las fibras ópticas, los datos a los

dispositivos CATCH. A su vez, el transductor óptico recibe el *trigger*, por medio de la segunda fibra óptica, el cual se transmite a la FPGA.

A través de la FPGA también se proveen dos señales de control para los *analog switches* utilizados en el proceso de *self testing* (Sección 5.2.4) y tres señales de control para los reguladores de voltajes analógicos (*voltages regulators*) que intervienen en la medición de los voltajes de alimentación internos (Sección 5.2.5). Tales señales se generan en el DSP y se escriben en un registro interno de la FPGA.

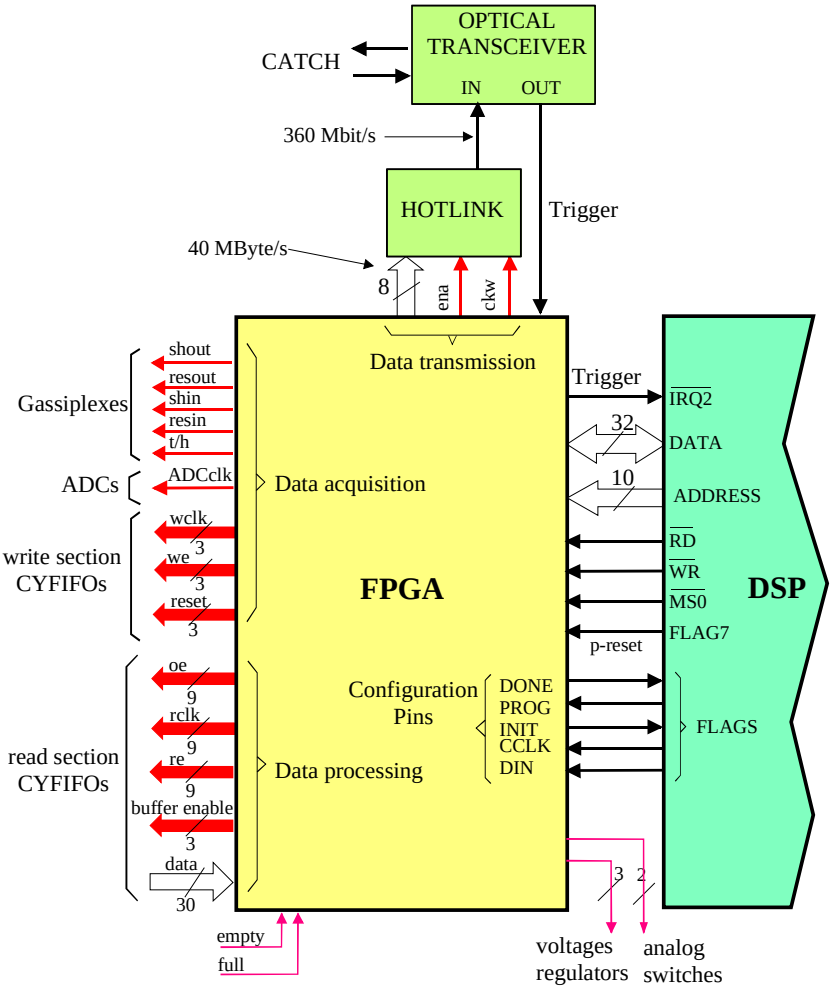

Figura 5.5 Conexión de la FPGA con el DSP, el bloque de comunicación con los dispositivos CATCH y la cadena de adquisición de datos de Bora.

5.2.3 El DSP y su Interfase con la FPGA

El DSP de Bora, como el de Dolina, es un *Digital Signal Processor Microcomputer* perteneciente a la familia *SHARC* de *Analog Devices* (ADSP-21065L [90, 91]), se alimenta a 3,3 *volt*s y toma un reloj externo de 30 MHz proveniente de un oscilador de cristal. Este reloj se multiplica internamente alimentando el *core* del procesador que trabaja a 60 MHz.

El DSP forma parte de una red de DSP TDM (Sección 3.3) implementada a través de uno de los puertos seriales del procesador. La Figura 3.4 muestra la conexión física entre los DSP de una red.

El DSP se comunica con los bancos de memoria externa a través de la correspondiente línea de selección (MS 3-0), el *bus* de datos y el *bus* de direcciones externos, una señal de escritura (WR) y una señal de lectura (RD). La FPGA corresponde al banco 0 de la memoria externa del DSP. Así la FPGA utiliza la señal MS0 para activar las transacciones con el DSP. La Figura 5.5 muestra la interconexión entre el DSP y la FPGA. Puesto que es el DSP el responsable de la configuración de la FPGA, los cinco puertos de configuración de la FPGA están conectados a cinco de los *flags* externos de propósito general del DSP, programables por software como puertos de entrada o salida (Sección 7.2).

A través de uno de tales *flag* (*FLAG7*), el DSP genera una señal de reset en la FPGA (*p-reset*) que causa la inicialización de las estructuras y máquinas de estado de la FPGA. Por su parte la FPGA retransmite al DSP la señal de *trigger* usada en la adquisición de datos durante el *run* del experimento. El DSP recibe el *trigger* a través de una de las tres líneas de interrupción externa del procesador (IRQ2).

El DSP controla un dispositivo *WatchDog* [95] idéntico al usado en Dolina (Sección 4.5), el mismo está configurado para generar un reset al DSP en intervalos

fijos de 1,6 s. Para evitarlo, el DSP envía una señal al *WatchDog* dentro de tal intervalo de tiempo que causa que el dispositivo reinicie en cero su contador de tiempo. Dicha señal es transmitida desde el DSP al *WatchDog* a través de la línea MS3, por medio de un acceso (escritura o lectura) a alguna de las direcciones pertenecientes al banco 3.

En las siguientes secciones se describe la interfase del DSP con la memoria EPROM de *booting*, los sensores de temperatura y los bancos 3-1 de la memoria externa.

5.2.4 *Self Testing*: DSP, DAC y Gassiplex

El chip de *front-end* (Gassiplex) tiene un *input* específico para estimular sus 16 canales analógicos. El modo de simular una señal de test similar a aquella proveniente del detector es cargar un capacitor hasta un voltaje determinado y luego descargarlo a través de una resistencia. La Figura 5.6 muestra esquemáticamente el circuito usado durante el proceso de s*elf testing*. Los *analog switches* (*S1* y *S2*) se controlan con señales digitales provenientes de registros internos de la FPGA. El DSP escribe estos registros en modo de controlar así tales *switches*. El voltaje de carga del capacitor proviene de un conversor digital-analógico [103] (DAC). El DSP gestiona el DAC en modo de cargar el capacitor C_{test} a un valor de tensión determinado. Actuando complementariamente sobre los *switches*, el capacitor de test se descarga a través de la resistencia R_{test} generando en este modo la señal de test (*testin*) en los Gassiplex. El DAC se conecta al DSP a través del byte menos significativo del *bus* de datos externo, la señal de escritura (*WR*) y la línea de selección (*MS2*) correspondiente al banco 2 de la memoria externa del DSP.

Otras dos señales digitales provenientes de registros internos de la FPGA (*shin* y *resin*) actúan sobre el Gassiplex en modo de seleccionar un par de canales analógicos para ser estimulados con la señal de *testin*, en modo análogo a la selección del par de canales para la salida analógica. Así, después de generar un reset con *rsin*, por cada ciclo de *shin* los pares 1 y 9, 2 y 10 hasta 8 y 16 se seleccionan secuencialmente para su estimulación.

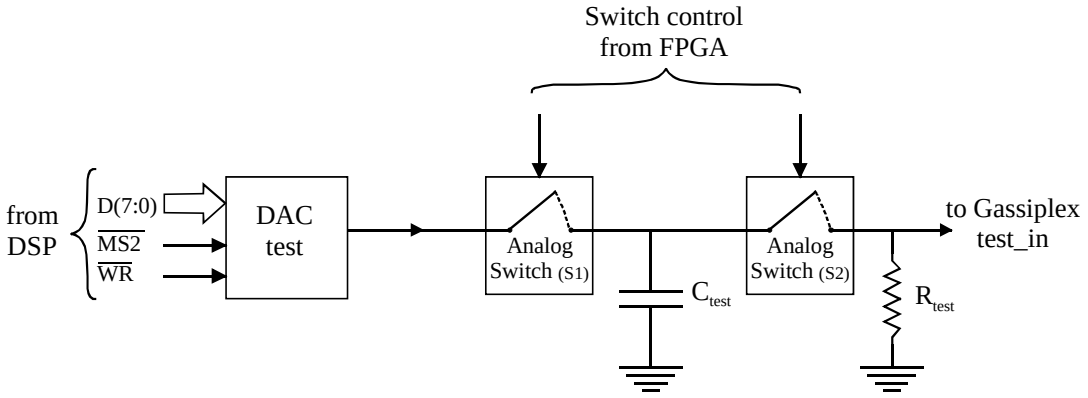

Figura 5.6 Circuito de *Self Testing*

5.2.5 Medición de Voltajes: DSP, EADC y Reguladores de Voltajes

Para medir las tensiones de cinco de los voltajes de alimentación analógicos de Bora, se usa un conversor analógico-digital [102] (EADC) conectado a los correspondientes reguladores de voltajes y al DSP. La Figura 5.7 muestra esquemáticamente dicha conexión. El EADC se conecta al DSP a través del byte menos significativo del *bus* de datos externo, el *bus* de direcciones externo, la señal de lectura (*RD*), la línea de selección (*MS1*) correspondiente al banco 1 de la memoria externa del DSP, y una señal de *acknowledge* (*AKN*). El EADC tiene ocho canales de ingreso analógicos que se seleccionan con tres bits de direccionamiento provenientes del *bus* de direcciones del DSP, y la señal MS1 conectada el *chip select* (*CS*) del conversor. El EADC provee una señal de *acknowledge* para indicar el final de la

conversión. El DSP utiliza dicha señal para completar el ciclo de lectura del EADC. La salida del conversor está conectada al *bus* de datos del DSP.

A través de la FPGA se pueden desactivar los voltajes de alimentación analógicos, mediante específicas señales de control que actúan sobre los respectivos reguladores de voltaje. Tales señales se generan en el DSP y se escriben en un registro interno de la FPGA.

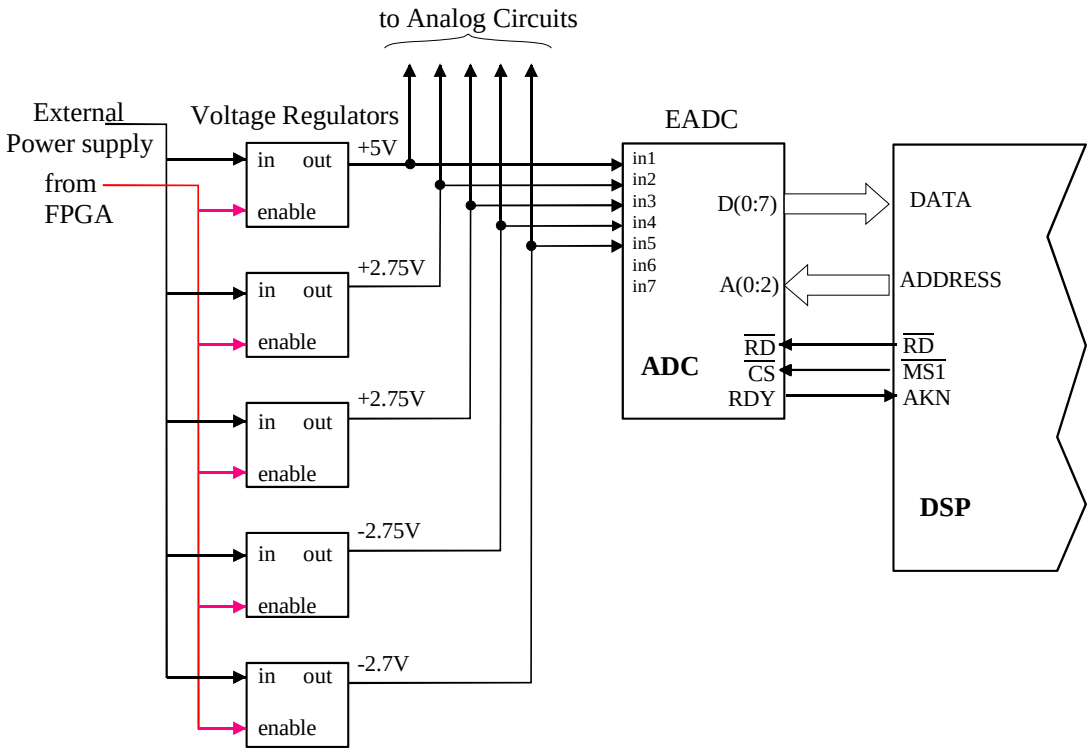

Figura 5.7 Conexión del EADC con los reguladores de voltajes y el DSP.

5.2.6 *Header Switch* para la Identificación de Bora

Un *DIP* (*Dual In line Package*) *Switch* de ocho elementos (“*header switch*”) se utiliza para la identificación de la tarjeta Bora. El estado de este *switch* se ingresa manualmente para identificar unívocamente la Bora al interno del RICH (Sección 6.2). El DSP lee el *switch* a través de un *buffer tristate* como muestra esquemáticamente la Figura 5.8. El *buffer tristate* se conecta al DSP a través del byte menos significativo del *bus* de datos externo, la señal de lectura (*RD*) y la línea de

selección (*MS3*) correspondiente al banco 3 de la memoria externa del DSP. Un acceso de lectura al banco 3 habilita las salidas del *buffer tristate* conectadas al *bus* de datos.

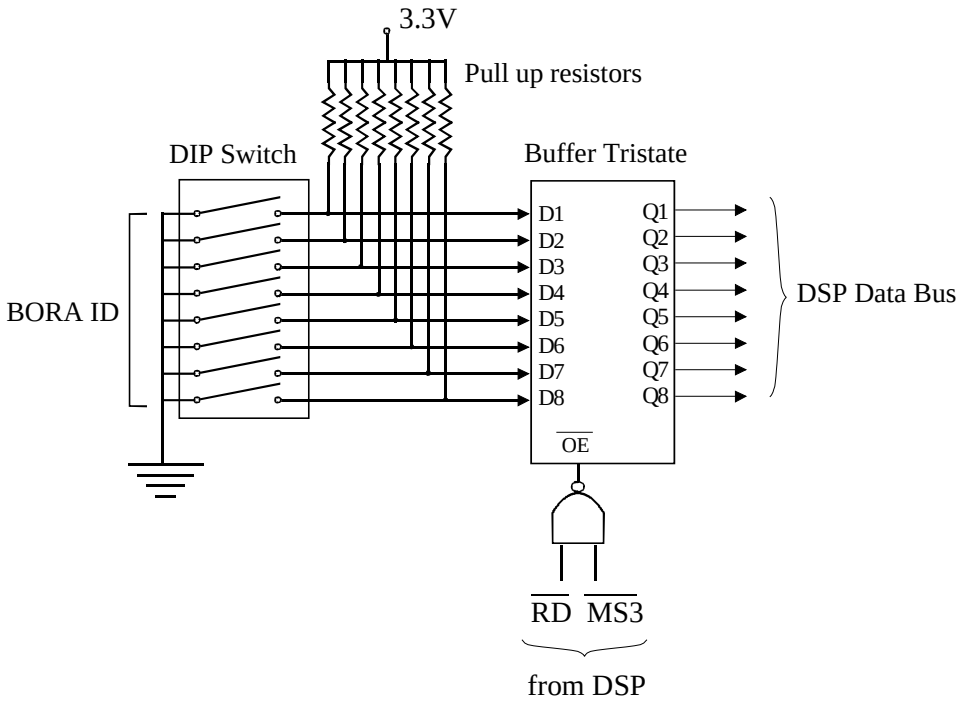

Figura 5.8 *Header DIP Switch.*

5.2.7 Sensores de Temperatura

Bora incluye cuatro sensores de temperatura [101] distribuidos físicamente en distintos puntos de la tarjeta. Tales sensores están predispuestos para su conexión en serie a través de una única línea de control bidireccional. La Figura 5.9 muestra la conexión entre los sensores de temperatura y el DSP. Dos puertos de entrada a los sensores (*ts0* y *ts1*) permiten configurarlos para trabajar en cuatro ventanas temporales distintas de modo de poder gestionarlos por medio de una misma línea sin interferencias entre ellos. Las temperaturas se obtienen midiendo la demora entre el flanco negativo de un pulso generado externamente por el DSP y los flancos positivos de los pulsos subsecuentes generados por los dispositivos. El DSP produce y recibe dichos pulsos usando uno de los *flags* externos (*FLAG*8) de propósito general

programables como puertos de entrada o salida, y mide las distintas demoras utilizando uno de los *timers* (TIMER1).

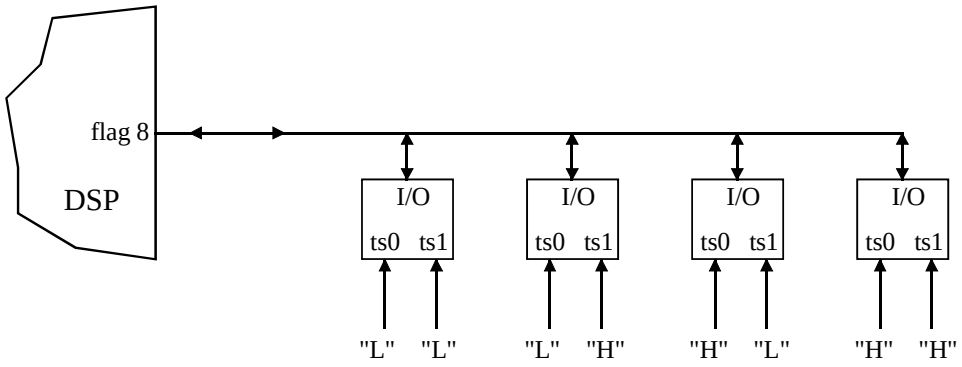

Figura 5.9 Sensores de temperatura.

5.2.8 Memoria de Booting del DSP

La memoria EPROM de *booting* del DSP se conecta al byte menos significativo del *bus* de datos externo junto a los 16 bits menos significativos del *bus* de direcciones externo. La Figura 5.10 muestra la conexión entre la EPROM y el DSP. Al momento del booting, el DSP genera una señal específica (*boot memory select*) a través del puerto *BMS* conectado al *chip select* (*CS*) de la memoria, y por medio de la señal de lectura *RD* conectada al *output enable* (*OE*) de la EPROM activa las salidas de la memoria quien toma de este modo el *bus* de datos. Desde el *bus* de datos el DSP carga automáticamente en su memoria interna el programa *boot-loader* almacenado en la EPROM (Sección 3.4.1).

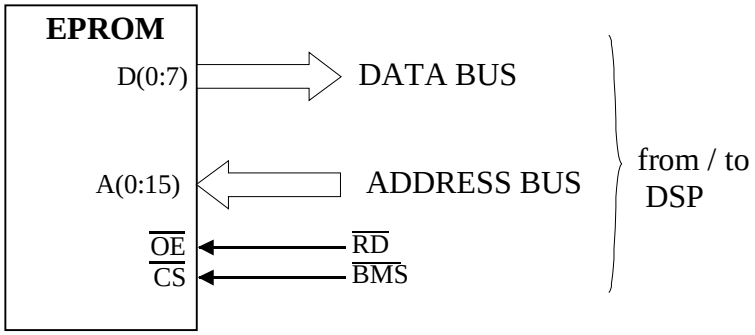

Figura 5.10 Memoria de *booting* del DSP.

Capítulo 6

Software del DSP

El DSP es el cerebro de Bora, su función es la de ejecutar, supervisar y controlar cada una de las actividades efectuadas en Bora actuando conjuntamente con la FPGA que funciona como coprocesador paralelo. La Figura 6.1 muestra esquemáticamente la estructura del software del DSP, en la cual se especifican los servicios principales efectuados por el DSP. Para la ejecución de tales servicios, el DSP interactúa con la FPGA y otros dispositivos externos incorporados en Bora. La FPGA y algunos de dichos dispositivos están mapeados en los bancos de la memoria externa del procesador.

El DSP forma parte de la red que conecta las 24 tarjetas Bora de una cámara con Dolina. La red de DSP permite el intercambio de programas, datos, comandos y mensajes entre la PC de control y las Bora. Los programas del DSP y la cadena de bits para la configuración de la FPGA se cargan por medio de la red, permitiendo la reprogramación del DSP y la reconfiguración de la FPGA en cualquier momento que sea requerido desde la PC de control. La configuración de la FPGA está a cargo del DSP.

El protocolo de comunicación entre el DSP y la FPGA se establece usando "modos de operación". Cada uno de los servicios en los que interviene la FPGA tiene asociado un modo de operación. El DSP determina el modo de operación de la FPGA, y en los distintos modos intercambia datos con la FPGA y recibe información del

estado de la misma. La FPGA controla la cadena de adquisición de datos de Bora, y efectúa las tareas con restricciones temporales duras que requieren un alto grado de procesamiento paralelo.

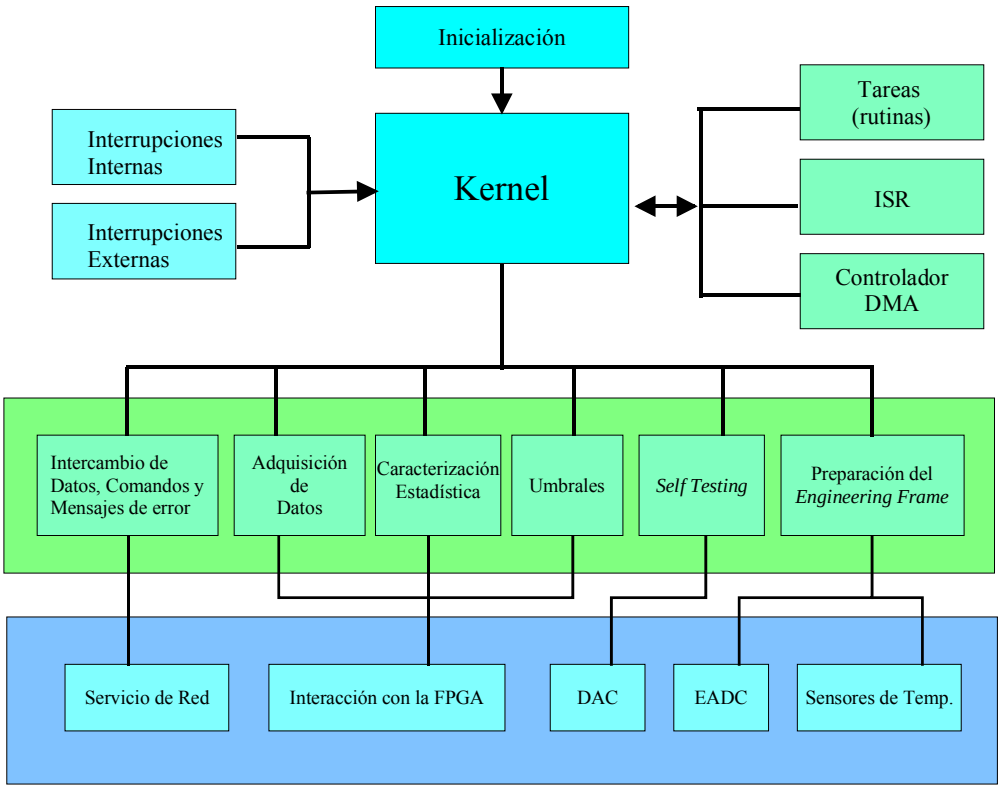

Figura 6.1 Estructura general del software del DSP de Bora.

El DSP recibe en tiempo real los comandos de sincronización para la adquisición de datos a través de la red, y la señal de *trigger* por medio de una de las interrupciones externas del procesador. La atención de los mismos involucra restricciones temporales duras. En particular, el DSP cuenta con menos de 200 ms para el servicio de los comandos de inicio y fin de *spill*, razón por la cual el arribo de tales comandos genera interrupciones internas de software en modo de garantizar la sincronización con el resto del experimento. Para evitar la pérdida de *triggers*, el servicio por parte del DSP no puede superar los 650 ns.

El software del DSP gestiona la comunicación con la red haciendo uso del controlador paralelo de DMA del procesador. El controlador de DMA opera en paralelo con el *core* del DSP, generando una interrupción interna sólo al final de la transmisión o recepción de un paquete completo.

Además de las actividades específicas ejecutadas durante el *run* del experimento, el software del DSP efectúa los siguientes servicios principales: caracterización estadística de los canales del detector, control del proceso de *self testing*, generación de *engineering frames*, cálculo, carga y lectura de los valores de umbrales y transmisión de información específica de la Bora a la cadena de adquisición global y la PC de control. El DSP de Bora, como en Dolina, controla un dispositivo *WatchDog* haciendo uso de uno de los *timers* del procesador.

En este capítulo se describen las funciones efectuadas por el software del DSP de Bora, su interacción con la FPGA y con la red de DSP, y su interacción con el resto de las componentes de la Bora que participan en los servicios provistos por la tarjeta.

6.1 Memoria Externa del DSP de Bora

La memoria externa del DSP está dividida en cuatro bancos de tamaño fijo (banco 3-0) a través de los cuales el procesador se comunica con la FPGA y con otros dispositivos externos que intervienen en los servicios provistos por Bora. Cada banco tiene asociado su propio generador de *wait-state* permitiendo la interacción entre el DSP y dispositivos periféricos más lentos. Dispositivos con diferentes requerimientos de tiempo pueden conectarse a distintos bancos y en cada uno el DSP establece el número requerido de *wait-state*.

El procesador provee una variedad de métodos [90] para simplificar la interfase con memorias y periféricos externos más lentos. El método más simple es

aquel en el cual el software especifica, en ciclos del reloj externo (30 MHz) del procesador, los *wait-state* requeridos para acceder al banco correspondiente. Otro de los métodos, usa también el puerto de entrada AKN del DSP que se conecta al dispositivo externo para extender el tiempo de acceso. De este modo, los ciclos de *wait-state* especifican el tiempo de acceso mínimo y luego el DSP espera por una transición de la señal recibida a través del puerto AKN.

La Figura 6.2 muestra un esquema de la memoria externa del DSP de Bora. La FPGA está mapeada en el banco 0, para cuyo acceso se usa un ciclo de *wait-state*. A través del banco 1, el DSP obtiene los voltajes de alimentación analógicos de la tarjeta provenientes del conversor ADC (EADC). Para acceder al banco 1 se utiliza el máximo número de *wait states* (seis ciclos de reloj externo) y el puerto AKN. El conversor DAC usado en el proceso de *self testing*, está mapeado en el banco 2. El *header switch*, a través del cual el DSP obtiene el identificador de la Bora, y el dispositivo *WatchDog* están mapeados en el banco 3. No se requieren *wait-state* para acceder a los bancos 2 y 3.

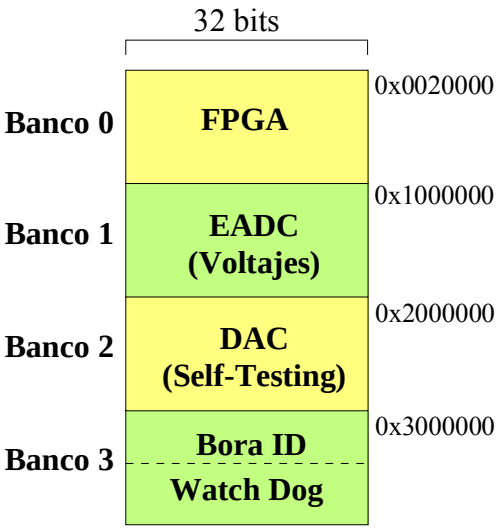

Figura 6.2 Memoria externa del DSP de Bora

6.2 Identificador de la Bora

La Bora contiene un "*header switch*" de ocho elementos a través del cual se ingresa manualmente el identificador de la tarjeta (*Bora Id*). El *Bora Id* identifica unívocamente la Bora al interno del RICH. La Figura 6.3 muestra el formato del *Bora Id* compuesto por 8 bits (xxxyyyyy), donde los 3 bits más significativos (xxx: bits 7-5) identifican la cámara del RICH y los 5 bits menos significativos (yyyyy: bits 4-0) identifican la Bora dentro de la cámara correspondiente. El número de cámaras del RICH es 8 por lo tanto el identificador de la cámara toma un valor en el rango 0-7, y el número de tarjetas Bora que conforman una cámara es 24 por lo que el identificador de la Bora en la cámara toma un valor en el rango 0-23.

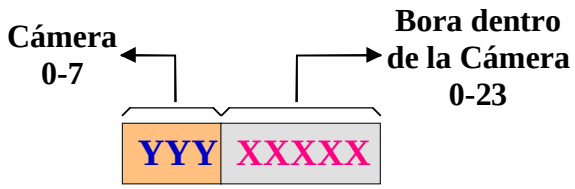

Figura 6.3 Identificador de la tarjeta Bora (*Bora Id*).

El software del DSP obtiene el identificador de la Bora desde el banco 3 de la memoria externa. El *Bora Id* forma parte de los paquetes de red y de evento transmitidos a la PC de control y los dispositivos CATCH respectivamente.

6.3 Interacción entre el DSP y la FPGA

La FPGA actúa como coprocesador paralelo del DSP. La configuración de la FPGA, es decir, el proceso en el cual se carga en la memoria de configuración interna la cadena de bits con el diseño de la FPGA, está a cargo del DSP. Durante el proceso

de configuración, el DSP interactúa con la FPGA a través de señales de configuración específicas (Sección 7.2).

Luego de la configuración, el protocolo de comunicación entre el DSP y la FPGA se establece usando "modos de operación". La interacción se efectúa principalmente por medio de un conjunto de registros y bloques de memoria direccionados en el banco 0 de la memoria externa del procesador. A través de tales estructuras, el DSP determina el modo de operación de la FPGA, intercambia datos con la FPGA, y recibe información del estado de la misma en los distintos modos de operación. Bajo el comando del DSP, la FPGA opera en los siguientes modos:

a) Modo "Evento"

Durante el *spill,* la adquisición de datos se divide en dos fases: "adquisición propiamente dicha" y "procesamiento y transmisión". La adquisición propiamente dicha se inicia autónomamente con el arribo del *trigger*. Es el DSP, en cambio, quien lanza el procesamiento y transmisión estableciendo e iniciando la FPGA en el modo de operación "Evento". Operando en modo "Evento", la FPGA lee las memorias CYFIFO, compara el valor de cada canal con su correspondiente umbral previamente almacenado en la FPGA, empaqueta los valores de los canales que superan el umbral con el correspondiente identificador de canal y transmite los paquetes de evento a la cadena de adquisición global del experimento. La FPGA escribe los paquetes de evento en dos memorias internas administradas como FIFO, desde una de tales memorias (*Hotlink FIFO*) los paquetes se transmiten a los dispositivos CATCH, y desde la otra memoria (*DSP FIFO*) el DSP lee eventualmente el paquete.

b) Modo "Test de Canal"

El DSP establece e inicia la FPGA en el modo de operación "Test de Canal" para efectuar la caracterización estadística de los canales analógicos y durante el proceso de *Self Testing*. El DSP genera internamente los *triggers* en la FPGA (simulando los *triggers* de evento), produciendo de este modo la adquisición, digitalización y almacenamiento en las memorias CYFIFO de los valores de los canales. Operando en modo "Test de Canal", la FPGA, sin efectuar la comparación con los umbrales, escribe en el *DSP FIFO* los valores de los 432 canales adquiridos con su correspondiente identificador de canal; desde donde son leídos y procesados por el DSP.

c) Modo "FPGA hacia DSP"

El modo de operación "FPGA hacia DSP" se usa en combinación con los modos "Evento" y "Test de Canal". En ambos, la FPGA escribe los paquetes de evento o el total de los valores de los canales adquiridos en el *DSP FIFO*. Para la lectura del DSP FIFO, el DSP establece la FPGA en el modo de operación "FPGA hacia DSP".

d) Modo "Escritura Umbrales" y "Lectura Umbrales"

Durante el *spill*, en modo de operación "Evento", la FPGA compara el valor de cada canal con su correspondiente umbral. La FPGA cuenta con una memoria interna destinada al almacenamiento de los valores de umbrales. Los umbrales se definen desde el "Controlador del RICH" y se transmiten a la Bora a través de la red de DSP, o se calculan por el DSP. El DSP establece la FPGA en el modo de operación "Escritura Umbrales" para almacenar los umbrales en la

memoria interna de la FPGA. Por otra parte, el DSP establece la FPGA en el modo de operación "Lectura Umbrales" para la lectura de los umbrales desde la memoria con el propósito de transmitirlos a la PC de control o al sistema global de adquisición de datos (DAQ).

e) Modo "DSP hacia DAQ"

El DSP transmite información específica de la Bora al sistema global de adquisición de datos (DAQ), escribiendo directamente en el "*Hotlink FIFO*". Para este propósito establece la FPGA en el modo de operación "DSP hacia DAQ".

La Figura 6.4 muestra un esquema del banco 0 de la memoria externa del DSP, en el cual se especifican las estructuras a través de las cuales el DSP interactúa con la FPGA. El DSP establece e inicia los modos de operación de la FPGA a través de un registro interno de la FPGA denominado *Command Word,* y recibe información del estado de la FPGA en los distintos modos de operación por medio de otro registro interno denominado *Status Word*. El DSP utiliza también la *Command Word* para bloquear los *trigger* externos provenientes del TCS, para generar *triggers* internos (modo "Test de Canal"), y para bloquear la adquisición de datos cuando es el DSP quien responde al *trigger* externo enviando información específica de la Bora a la cadena de adquisición global (modo "DSP hacia DAQ"**)**. Los registros *Command Word* y *Status Word* se describen en el capítulo 7.

El DSP accede al registro *Command Word* usando la dirección 0x20000 y lee el registro *Status Word* accediendo a la dirección 0x20001. El DSP lee y escribe la memoria de umbrales de la FPGA en los modos de operación "Lectura Umbrales" y

"Escritura Umbrales" respectivamente. La memoria de umbrales está mapeada en el rango de direcciones 0x20200 – 0x202ff. En el modo de operación "FPGA hacia DSP", el DSP lee el *DSP FIFO* de la FPGA a través de la dirección 0x20002. Usando la misma dirección pero en escritura el DSP accede al *Hotlink FIFO* de la FPGA en el modo de operación "DSP hacia DAQ". Para acceder al FIFO, basta una única dirección de memoria que corresponde a la primer palabra del FIFO en lectura y a la primer posición libre del FIFO en escritura.

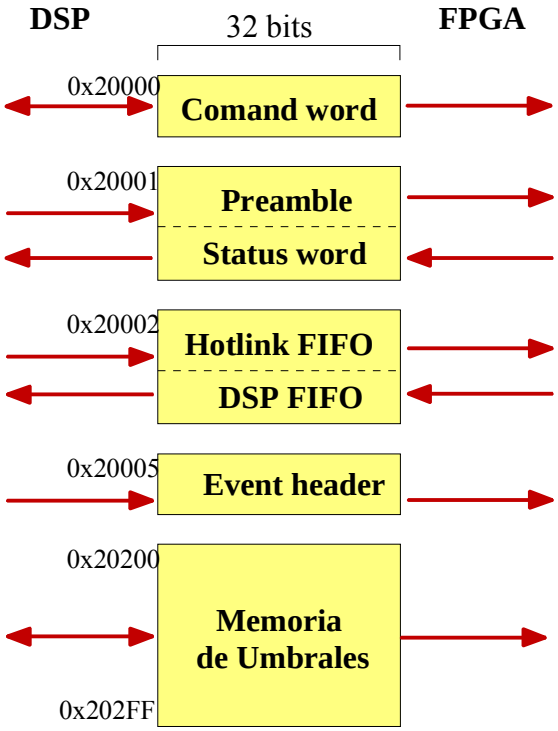

Figura 6.4 Banco 0 de la memoria externa del DSP donde la FPGA está mapeada.

Durante el *spill*, la FPGA recibe los *triggers* de evento provenientes del TCS y los trasmite al DSP. El DSP recibe el *trigger* a través de la línea de interrupción externa IRQ2 del procesador. Por cada *trigger*, el DSP genera el *header* del paquete de evento e inicia la FPGA en el modo de operación "Evento". El DSP escribe el *header* en un registro interno de la FPGA denominado *Event Header*. El *header* de los paquetes de eventos es una palabra de 32 bits que incluye el identificador de la Bora

(*Bora Id*) y el número de evento, usado por los CATCH para la reconstrucción local de eventos. El cuerpo de los paquetes de eventos, generado por la FPGA, contiene el valor y el identificador de los canales que superan el umbral. A su vez, el identificador del canal incluye el *Bora Id* (Figura 7.4). El DSP escribe el *Bora Id* en un registro interno de la FPGA denominado *Preamble,* desde donde la FPGA lo lee para incluirlo en el identificador del canal.

El DSP escribe en el registro *Preamble* y en el registro *Event Header* a través de las direcciones 0x20001 y 0x20005 respectivamente, la dirección del registro *Preamble* coincide con la dirección del registro *Status Word* pero en un caso el acceso por parte del DSP es en escritura y en el otro en lectura.

6.4 Valores de Umbrales

Los valores de umbrales usados en el modo de operación "Evento" se almacenan en un bloque de memoria interno (256 x 32 bits) de la FPGA (memoria de umbrales). Cada Bora adquiere 432 canales del detector, y al valor digitalizado en 10 bits de cada canal se le sustrae su correspondiente umbral también de 10 bits. Como la memoria de umbrales es de 32 bits, se almacenan tres valores por cada palabra por lo que se utilizan 144 palabras de la memoria para almacenar los 432 umbrales. El DSP escribe y lee la memoria de umbrales.

Normalmente, los valores de umbrales se establecen desde el "Controlador del RICH" de acuerdo a ciertos criterios experimentales, en general se definen a partir de los valores de ruido estimados para cada canal (Sección 3.1.1). Una vez definidos, se transmiten a la Bora por medio de la red de DSP. Desde el "Controlador del RICH" pueden también requerirse los valores actuales de umbrales enviando un comando específico al DSP.

Para la transmisión de umbrales se utilizan dos paquetes de red cuyo formato se muestra en la Tabla 6.1. Cada palabra del cuerpo del paquete contiene los valores de tres umbrales ocupando los 30 bits menos significativos (10 bits por umbral: 0-9, 10-19 y 20-29) de la palabra. El primer paquete contiene los umbrales para 375 canales y el segundo paquete contiene los umbrales correspondientes a los 57 canales restantes.

H		O	L	A
0	Umbrales	0x00	Fuente	Destino
-	Umbral_2	Umbral_1		Umbral_0
-	Umbral_5	Umbral_4		Umbral_3
-	Umbral_8	Umbral_7		Umbral_6
…				
-	Umbral_371	Umbral_370		Umbral_369
-	Umbral_374	Umbral_373		Umbral_372
C		H	A	U

H		O	L	A
0	Umbrales	0x01	Fuente	Destino
-	Umbral_377	Umbral_376		Umbral_375
-	Umbral_380	Umbral_379		Umbral_378
-	Umbral_383	Umbral_382		Umbral_381
…				
-	Umbral_428	Umbral_427		Umbral_426
-	Umbral_431	Umbral_430		Umbral_429
C		H	A	U

Tabla 6.1 Paquetes de red para la transmisión de los valores de umbrales.

El DSP reserva en su memoria interna un *buffer* exclusivo para almacenar los dos paquetes de umbrales (*buffer-umbrales*). Cuando recibe ambos paquetes y luego de controlar que la secuencia sea justa, establece la FPGA en el modo de operación "Escritura Umbrales" y copia en secuencia el contenido de ambos paquetes desde la memoria interna hacia la memoria de umbrales de la FPGA. Una vez que los 432 umbrales fueron almacenados en la FPGA, el DSP cambia el modo de operación indicando en este modo el final de la escritura. Si durante la recepción de los paquetes de umbrales el DSP detecta algún error, informa a la PC de control enviando el mensaje correspondiente e ignora la operación.

Por su parte, el "Controlador del RICH" puede solicitar a la Bora los valores de umbrales corrientes, enviando un comando específico (*get thresholds*). Cuando el DSP recibe este comando, configura la FPGA en el modo de operación "Lectura

Umbrales" y copia en secuencia el contenido de la memoria de umbrales de la FPGA hacia *buffer-umbrales*. Una vez que todos los valores fueron almacenados en las direcciones del *buffer* correspondiente al cuerpo de ambos paquetes, el DSP cambia el modo de operación de la FPGA indicando en este modo el final de la operación de lectura, luego actualiza los campos fuente y destino en los encabezamientos de los paquetes y configura el SPORT DMA para transmitir ambos paquetes a Dolina.

6.5 Adquisición de Datos desde el DSP

El DSP puede lanzar la adquisición de datos, cuando el experimento no está en *run,* generando en la FPGA *triggers* internos que simulan los *triggers* externos de evento. El DSP genera el *trigger,* a través de la *Command Word,* produciendo de este modo la adquisición, digitalización y almacenamiento en las memorias CYFIFO de los valores de los canales. La FPGA informa al DSP el final de la adquisición correspondiente a un *trigger* por medio de la *Status Word*. Luego, el DSP establece e inicia la FPGA en el modo de operación "Test de Canal" en el cual, sin efectuar comparación con los valores de umbrales, la FPGA escribe en el *DSP FIFO* el valor y el identificador de cada uno de los 432 canales adquiridos. Una vez más, a través de la *Status Word*, la FPGA informa al DSP el final de la escritura de los valores de los canales correspondientes a un *trigger*. Para la lectura del FIFO, el DSP establece la FPGA en el modo de operación "FPGA hacia DSP".

6.5.1 Estimación del Ruido del Canal

Los canales analógicos del detector presentan, en ausencia de señal, una variación aleatoria de sus valores en función del tiempo que denominamos *ruido*. Para

caracterizar el ruido de cada canal, se calculan los estimadores discretos: μ y σ (Sección 5.1.2) a partir de una colección discreta de muestras. Con el objetivo de optimizar el cálculo del σ, se utiliza la siguiente fórmula equivalente donde x_i es el valor digitalizado del canal obtenido en cada muestra y N es el número total de muestras:

$$\sigma^2 = \frac{1}{N-1}\left[\sum_{i=0}^{N-1} x_i^2 - \frac{1}{N}\left(\sum_{i=0}^{N-1} x_i\right)^2\right]$$

Esta fórmula permite obtener los valores μ y σ a partir de los siguientes tres parámetros: el número de muestras, la suma de los valores obtenidos para el canal en cada muestra y la suma del cuadrado de los valores obtenidos para el canal en cada muestra.

Desde el "Controlador del RICH" se solicita al DSP la caracterización estadística de los canales adquiridos, enviando un comando específico (*channel test*) que incluye como parámetro el número de muestras. El DSP responde al comando generando un *trigger* interno por cada muestra, esperando por la adquisición de los canales y estableciendo la FPGA en el modo de operación "Test de Canal". Una vez que la FPGA escribe los valores adquiridos en el *DSP FIFO*, el DSP establece la FPGA en el modo de operación "FPGA hacia DSP" y procede con la lectura del FIFO. Con los valores adquiridos en cada muestra, el DSP calcula parcialmente el μ y el σ para cada canal obteniendo la suma de los valores del canal y la suma del cuadrado de los valores del canal. Por último envía tales resultados parciales al "Controlador del RICH", quien se encarga de la elaboración final obteniendo el valor medio y la desviación estándar de cada canal.

El DSP reserva en su memoria interna dos *buffer* exclusivos para almacenar la suma (*buffer-suma*) y la suma del cuadrado (*buffer-suma-cuadrados*) de los valores obtenidos de cada canal. Por cada muestra el DSP lee el "DSP FIFO", toma el valor de cada canal y calcula el cuadrado de dicho valor, luego suma los nuevos valores a los valores acumulados para el canal en el *buffer* correspondiente. Una vez finalizada la adquisición y elaboración del número total de muestras requeridas, el DSP arma los paquetes con los resultados obtenidos para enviarlos al "Controlador del RICH" a través de la red de DSP. Por cada uno de los 432 canales adquiridos, el DSP incluye en el paquete la siguiente información:

-	Canal Id dentro de Bora	Suma de los valores del canal
-	Suma del cuadrado de los valores del canal	

a) **Canal Id dentro de Bora (10 bits)** contiene una parte del identificador del canal (*Canal Id*). El identificador del canal (18 bits) está formado por la concatenación entre el *Bora Id* (8 bits) y el *Canal Id* dentro de la Bora (10 bits)**.** Puesto que el *Bora Id* forma parte del campo fuente del paquete, el cuerpo del paquete incluye sólo la segunda parte del identificador del canal.

b) **Suma de los valores del canal (21 bits)** contiene la suma de los valores obtenidos para el canal: $\sum_{i=0}^{N-1} x_i$ donde N es el número requerido de muestras.

c) **Suma del cuadrado de los valores del canal (31 bits)** contiene la suma de los cuadrados de los valores obtenidos para el canal: $\sum_{i=0}^{N-1} x_i^2$ donde N es el número requerido de muestras.

Puesto que la longitud de los paquetes de red es de 128, se requieren siete paquetes, de tipo *channel frame*, para enviar la información adquirida por cada uno de los 432 canales. El primer paquete de la secuencia incluye el número de muestras (N). Luego de armar los paquetes, el DSP configura el SPORT DMA para transmitirlos a Dolina.

6.5.2 Proceso de *Self Testing*

La Bora cuenta con un mecanismo de auto testeo que permite evaluar principalmente la cadena de adquisición de datos local (Sección 5.1.3). Para este propósito, el DSP a través de la FPGA y del conversor DAC (mapeado en el banco 2 de la memoria externa del procesador), acciona un circuito que estimula los chips de *front-end* generando una señal en las entradas analógicas que simula la señal proveniente del detector durante la adquisición de datos normal.

El mismo comando utilizado para solicitar la caracterización estadística de los canales analógicos (*channel test*) se envía desde el "Controlador del RICH" al DSP para iniciar el proceso de *Self Testin*g. El DSP responde al comando siguiendo un procedimiento similar al explicado en la sección anterior pero en este caso, además del número de muestras, el comando incluye también como parámetros la intensidad del estímulo que el DSP aplica al DAC, y el intervalo de tiempo entre la aplicación del estímulo al DAC y la generación del *trigger* por parte del DSP. De este modo, variando el intervalo de tiempo entre el estímulo de test y el *trigger* se puede explorar la respuesta temporal de cada canal, y variando la intensidad del estímulo se puede explorar la linealidad del sistema.

Cuando el DSP recibe el comando *channel test* requiriendo el proceso de *Self Testing*, toma los parámetros del paquete y por cada una de las muestras efectúa la secuencia de operaciones resumida en la Figura 6.5.

1. *Aplica al DAC el estímulo recibido como parámetro, escribiendo dicho valor en el banco 2 de la memoria externa, y genera, a través de la Command Word, la señal de test en las entradas analógicas de los chips de front-end (Sección 5.2.4).*
2. *Espera el número de ciclos especificado como parámetro, el cual establece el intervalo de tiempo entre el estímulo y el trigger.*
3. *Genera un trigger interno en la FPGA, a través de la Command Word, iniciando de este modo la adquisición, digitalización y almacenamiento en los CYFIFOs de los valores de cada canal.*
4. *Espera, consultando la Status Word, por la confirmación por parte de la FPGA del final de la "adquisición propiamente dicha" para el trigger generado.*
5. *Establece e inicia la FPGA en el modo de operación "Test de Canal", en el cual la FPGA sin efectuar la comparación con el umbral, escribe en el DSP FIFO el valor y el identificador de cada canal.*
6. *Espera, consultando la Status Word, por la confirmación por parte de la FPGA del final de la escritura de los valores adquiridos en el DSP FIFO.*
7. *Establece la FPGA en el modo de operación "FPGA hacia DSP", en el cual lee el DSP FIFO tomando el identificador (18 bits) y el valor (10 bits) del canal.*
8. *Si se trata de la primer muestra, copia parte del identificador del canal (los 10 bits menos significativos de los 18 bits que lo conforman) en la posición del paquete correspondiente que contiene la información obtenida para el canal.*
9. *Suma el valor del canal a la suma acumulada de los valores del canal en buffer-suma.*
10. *Calcula el cuadrado del valor del canal y suma el resultado obtenido a la suma acumulada de los cuadrados de los valores del canal en buffer-suma-cuadrados.*

Figura 6.5 Secuencia ejecutada por el DSP durante el proceso de *Self Testing*.

Por cada muestra, se repiten los pasos 7-10 para cada uno de los 432 canales adquiridos. Para el caso de la caracterización estadística de los canales analógicos, explicada en la sección anterior, el proceso es exactamente el mismo comenzando del paso 3 hasta el 10. También como en el caso anterior, una vez elaboradas las muestras, se arman y transmiten a Dolina los siete paquetes *channel frame*

conteniendo la información de cada canal. El primer paquete de la secuencia incluye los parámetros usados en la elaboración.

6.5.3 Comando *Channel Test*

La Tabla 6.2 muestra el formato del paquete *Channel Test* transmitido desde la PC de control al DSP de Bora para iniciar la caracterización estadística de los canales analógicos y el proceso de *Self Testing*.

H		O	L	A
0	Channel Test	0x00	Fuente	Destino
Tipo de Operación				
Número de Muestras (N)				
Estímulo				
Demora del *trigger*				
C		H	A	U

Tabla 6.2 Paquete de red para la transmisión del comando *Channel Test*.

El cuerpo del paquete contiene los parámetros del comando *Channel Test*:

a) **Tipo de Operación** especifica la operación solicitada: caracterización estadística de los canales analógicos o *Self Testing*.

b) **Número de Muestras (N)** establece para ambas operaciones el número de muestras a tomar para calcular el valor medio y la desviación estándar.

c) **Estímulo** especifica la intensidad del estímulo aplicado al DAC durante la operación de *Self Testing* para generar las señales analógicas que simulan las señales provenientes del detector.

d) **Demora del *Trigger*** establece el intervalo de tiempo en número de ciclos del DSP entre la aplicación del estímulo al DAC y la generación del *trigger*

interno que inicia el proceso de adquisición en la FPGA durante la operación de *Self Testing*.

6.6 *Engineering Frame*

Durante el *run* el área experimental se vuelve inaccesible y por lo tanto es importante contar con mecanismos para monitorear remotamente el funcionamiento de las tarjetas Bora. A tal efecto, el DSP mide los voltajes de cinco *power supplies* analógicos y las temperaturas provenientes de cuatro sensores.

El DSP obtiene y empaqueta los voltajes y las temperaturas durante los intervalos de *interspill* para enviarlos al "Controlador del RICH" al inicio de cada *spill* y al sistema DAQ del experimento respondiendo al primer *trigger* del *spill*. Además de los voltajes y las temperaturas, el DSP envía información del *run* tal como el número de *spill* corriente del *run* y el número de *triggers* recibidos durante el *spill* anterior al corriente. Dichos valores permiten verificar, desde el "Controlador del RICH", si las 192 Boras del sistema adquirieron el mismo número de *spill*s y de *triggers*, como así también permiten corroborar, desde el sistema DAQ, la sincronización del sistema de adquisición de datos del RICH con el resto de la cadena de adquisición del experimento.

El DSP reserva en su memoria interna un *buffer* específico, denominado *engineering frame*, para almacenar los voltajes, las temperaturas y los datos del *run* por cada *spill*. Al comienzo del *run*, se inician en cero las variables que contienen el número de *spill* corriente (*spill-counter*) y los *triggers* recibidos por *spill* (*event-counter*). Durante el *spill* se incrementa *event-counter* por cada uno de los *triggers* recibidos, y en el *interspill*, se obtienen y se almacenan los voltajes y las temperaturas.

Los valores de *spill-counter* y *event-counter* se almacenan al comienzo de cada *spill*, previamente al incremento de *spill-counter* y a la reinicialización de *event-counter*. La Tabla 6.3 muestra el formato del paquete de red usado para transmitir el *engineering frame* a Dolina.

H	O	L	A
0 *Engineering Frame*	0x00	Fuente	Destino
Voltaje 0			
Voltaje 1			
Voltaje 2			
Voltaje 3			
Voltaje 4			
Temperatura 0			
Temperatura 1			
Temperatura 2			
Temperatura 3			
Número de *spill* en el *run*			
Número de *triggers* en *spill*			
C	H	A	U

Tabla 6.3 Paquete de red para la transmisión del *Engineering Frame*.

El cuerpo del paquete incluye los siguientes campos:

a) **Voltaje 0-4** contiene los valores de los voltajes de cinco *power suplies* de Bora, obtenidos por el DSP desde el conversor EADC.

b) **Temperatura 0-3** contiene las temperaturas en grados *Celsius,* obtenidas por el DSP desde cuatro sensores posicionados en distintos puntos de Bora.

c) **Número de *spill* en el *run*** contiene el número de *spill* corriente en el *run*.

d) **Número de *triggers* en *spill*** contiene el número de *triggers* recibidos durante el *spill* anterior al corriente.

Al comienzo de cada *spill*, una vez que fueron actualizados los valores del *engineering frame*, el DSP configura el SPORT DMA para enviar el paquete a Dolina.

6.6.1 Medición de Voltajes

El DSP obtiene las tensiones de cinco voltajes de alimentación analógicos de Bora, a través de un conversor ADC (EADC) mapeado en el banco 1 de la memoria externa del procesador. El EADC tiene ocho canales de ingreso analógico que se seleccionan con tres bits de direccionamiento correspondientes al *bus* de direcciones externo, y se digitalizan en 8 bits (Sección 5.2.5). El conversor provee una señal (*ready*) para indicar el final de la conversión del canal seleccionado, conectada al puerto de entrada AKN del procesador. El DSP utiliza el método "*wait-state mas acknowledge*" para acceder al banco 1 de la memoria externa. De este modo, el DSP obtiene cada voltaje leyendo desde una dirección particular del banco 1 que contiene los tres bits de selección del canal correspondiente, y el tiempo de acceso se determina automáticamente por el máximo número permitido de *wait-state* sumado al tiempo de transición de la señal *ready* del EADC.

Durante el *interspill*, luego de recibir el comando de fin de *spill*, el DSP obtiene los voltajes y los almacena en el *engineering frame*.

6.6.2 Medición de Temperaturas

El DSP obtiene las temperaturas desde los cuatro sensores contenidos en Bora, usando una única línea de control (Sección 5.2.7). Las temperaturas se obtienen midiendo la demora entre el flanco negativo de un pulso generado por el DSP y los flancos negativos de los pulsos subsecuentes generados por los dispositivos. El DSP

produce y recibe dichos pulsos usando el FLAG8, y mide las distintas demoras utilizando el TIMER1. La Figura 6.6 muestra la secuencia de pulsos del *flag* durante la lectura de las temperaturas.

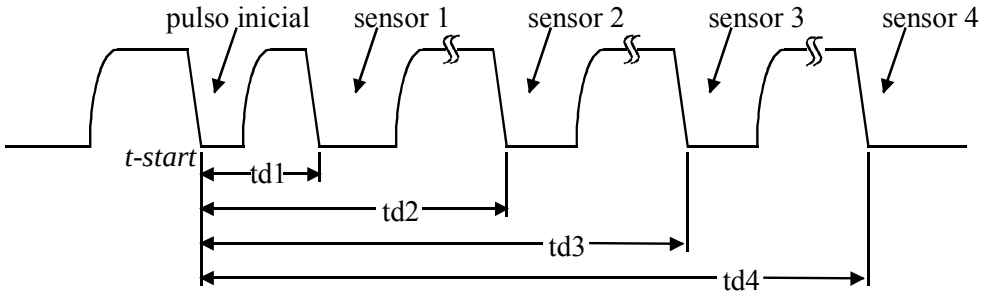

Figura 6.6 Diagrama de tiempo de la línea de control a través de la cual el DSP mide las temperaturas provenientes de los cuatro sensores contenidos en Bora.

El DSP genera un pulso a través del FLAG8 programado inicialmente como puerto de salida, cuando el *flag* se vuelve bajo (*t-start*) el control pasa a los sensores y el *flag* se programa como puerto de entrada. La temperatura de cada sensor se obtiene midiendo las demoras entre *t-start* y los flancos negativos de los pulsos subsecuentes generados por cada uno de los sensores (*td1*, *td2*, *td3* y *td4*)**.** Para medir las demoras, el DSP utiliza el TIMER1 del procesador. Cuando el *timer* se habilita, se inicializa un registro del *timer*, denominado TCOUNT1, el cual se incrementa automáticamente por cada ciclo de reloj del procesador. La figura 6.7 resume los pasos principales ejecutados por el DSP para la medición de las temperaturas

1. *Configura el FLAG8 como puerto de salida y genera un pulso, es decir, una secuencia de transiciones de 0 a 1 y de 1 a 0, a través del flag.*

2. *Configura el FLAG8 como puerto de entrada y habilita el TIMER1 para inicializar el registro TCOUNT1 en 1.*

3. *Lee el flag esperando por un pulso proveniente del sensor correspondiente, es decir, espera en primer lugar por una transición de 0 a 1 y en segundo lugar por una transición de 1 a 0. Luego lee y registra el valor de TCOUNT1.*

4. *Repite el paso 3 para cada uno de los sensores hasta obtener las demoras: td1, td2, td3 y td4.*

5. *Deshabilita el TIMER1.*

Figura 6.7 Secuencia ejecutada por el DSP para la medición de las temperaturas.

Cada sensor tiene asociado un multiplicador cuyo valor depende de la posición del sensor en la secuencia. Los multiplicadores y las demoras obtenidas se utilizan en el cálculo de las temperaturas en grados *Kelvin*. El DSP aplica la siguiente fórmula para transformarlas en grados *Celsius,* donde *tdx* es la demora (expresada en microsegundos) y *multiplicador-x* es el multiplicador, ambos asociados al censor *x*:

$$T(^oC) = \left[\frac{tdx(\mu s)}{multplicador\_x(\mu s/^oK)} \right] - 273{,}15\,^oK$$

Durante el *interspill*, luego de recibir el comando de fin de *spill*, el DSP obtiene las temperaturas en grados *Celsius* y las almacena en el *engineering frame*.

6.7 Comunicación entre el DSP y los CATCH

El primer *trigger* de cada *spill* se reserva para enviar información específica de las tarjetas de adquisición a los dispositivos CATCH. En este caso, es el DSP quien transmite directamente dicha información. Del mismo modo, durante los intervalos de *interspill*, es el DSP quien responde a eventuales *triggers* generados desde el TCS. En tales situaciones, el DSP bloquea, por medio de la *Command Word*, la normal adquisición de los canales del detector. El DSP también se comunica directamente con los CATCH para enviar una serie de datos requeridos por tales dispositivos, para la sincronización con las tarjetas de adquisición, antes del inicio del *run*. Para transmitir

datos a los CATCH, el DSP establece la FPGA en el modo de operación “DSP hacia DAQ” y escribe directamente en el “Hotlink FIFO”.

Normalmente, los paquetes transmitidos a los CATCH son de longitud variable y contienen un *header* y *trailer* que indican el inicio y el fin del paquete. Las palabras que conforman el paquete son de 32 bits. El formato de tales paquetes, mostrado en la Tabla 6.4, es común a un grupo de detectores que conforman la estructura instrumental de Compass [21, 28].

0	0	0	0	BORA ID (8 bits)	Número de Evento (20 bits)
1	Datos de Evento (31 bits)				
1	Datos de Evento (31 bits)				
…					
1	Datos de Evento (31 bits)				
1	Datos de Evento (31 bits)				
0	1	0	0	BORA ID (8-bit)	Número de Evento (20-bit)

Tabla 6.4 Formato del paquete para la transmisión de datos a los CATCH.

El bit más significativo de cada palabra distingue el *header* y el *trailer* de los datos que conforman el cuerpo del paquete. El bit 31 en el header y en el trailer es 0 mientras que para los datos es 1. A su vez el *header* se distingue del *trailer* por el bit 30, el cual es 0 en el primer caso y 1 en el segundo. A excepción del bit 30, el *header* es igual al trailer y contienen los siguientes campos:

a) **Número de Evento (20 bits)** contiene el número de evento que se corresponde con el número de *trigger* generado por el TCS al cual la Bora responde enviando el paquete a los CATCH.

b) **BORA ID (8 bits)** contiene el identificador de la Bora transmisora del paquete.

6.7.1 Primer *Trigger* de *Spill*

Durante el *spill*, es el DSP quien responde al primer *trigger* enviando información específica de la Bora. Para este propósito, al inicio del *run* el DSP configura la FPGA para recibir *triggers* externos bloqueando la normal adquisición de los canales del detector. El DSP responde al primer *trigger* del primer *spill* del *run* enviando los datos del *engineering frame*, los umbrales actuales almacenados en la FPGA y los resultados parciales del valor medio y la desviación estándar obtenidos tomando 2048 muestras por cada canal en ausencia de señal. Para el resto de los *spill*, el DSP responde al primer *trigger* de *spill* enviando sólo los datos del *engineering frame*.

Durante el intervalo de tiempo entre el inicio del *run* y el inicio del *spill*, el DSP obtiene las temperaturas, los voltajes y los valores actuales de umbrales almacenados en la FPGA, calcula los resultados parciales del valor medio y desviación estándar, y prepara el paquete que será enviado a la cadena de adquisición global. Para este propósito, reserva un *buffer* en la memoria interna (*buffer-first-trigger*) desde donde envía el paquete respondiendo al primer *trigger* del *run*. La longitud del paquete es de 1021 palabras conteniendo la siguiente secuencia: *header*, *engineering frame* (11 palabras), umbrales (144 palabras), suma y suma de los cuadrados de los valores obtenidos por cada canal (864 palabras), y *trailer*. Para la transmisión del primer paquete del *run*, el tiempo entre el primer *trigger* y el segundo *trigger* se estableció en 220 microsegundos.

Para el resto de los *spill*, el DSP responde al primer *trigger* enviando los datos del *engineering frame* más el *header* y el *trailer*. El DSP arma el paquete durante el *interspill*, luego de recibir el comando de fin de *spill*, dejándolo listo para su

transmisión con el arribo del primer *trigger* del próximo *spill*. En cada caso, el número de evento contenido en el *header* y el *trailer* del paquete es 1.

Una vez que el DSP recibe el primer *trigger* del *spill*, configura la FPGA para la adquisición normal de los canales del detector.

6.7.2 *Triggers* de *Interspill*

Durante el *interspill*, el TCS puede generar eventuales *triggers* generalmente denominados *triggers* de calibración o monitoreo. La Bora responde a dichos *triggers* enviando los así llamados paquetes de evento "vacíos", mostrados en la Tabla 6.5, conformados solamente por el *header* y el *trailer*.

0	0	0	0	BORA ID	Número de Evento
0	1	0	0	BORA ID	Número de Evento

Tabla 6.5 Formato del paquete de evento "vacío" transmitido a los CATCH durante el *interspill*.

Durante el *interspill*, es el DSP quien responde a tales *triggers* eventuales. Para este propósito, al final del *spill*, el DSP configura la FPGA para recibir *triggers* externos bloqueando la normal adquisición de los canales del detector. El número de evento contenido en el paquete contiene el número correspondiente en la secuencia de *triggers* recibidos desde el inicio del *spill*.

6.7.3 Sincronización con los CATCH

Los dispositivos CATCH reciben datos provenientes de los distintos detectores que conforman la estructura experimental de Compass. Algunos de tales detectores envían paquetes conteniendo palabras de 24 bits y otros, como en el caso

de RICH, envían paquetes cuyas palabras son de 32 bits. Para sincronizarse con las tarjetas de adquisición de los distintos detectores, los CATCH solicitan el envío repetitivo de una palabra (Tabla 6.6) conteniendo el identificador del detector, el ancho de las palabras de datos enviadas desde el detector, y el identificador y la posición geográfica de la tarjeta de adquisición transmisora [28].

Posición Geográfica (16 bits)	Longitud Dato (1 bit)	Detector (4 bits)	0	Tarjeta ID (10 bits)

Tabla 6.6 Palabra para la sincronización entre el RICH y los CATCH.

Las tarjetas de adquisición del RICH son las Bora, cuyo identificador especifica la posición geográfica de la tarjeta en el detector. Por lo tanto, el campo *Posición geográfica* y *Tarjeta ID* contienen el *Bora ID*. El campo *Longitud Dato* contiene el código que establece que los datos enviados desde las Bora son de 32 bits. El campo *Detector* contiene el código que identifica al detector RICH.

Al final del proceso de inicialización del software y cuando se requiere desde el "Controlador del RICH" por medio de un comando específico, el DSP transmite repetitivamente la palabra de sincronización a los CATCH.

6.8 Adquisición de Datos Durante el *Run*

Señales de sincronización comunes a todo el experimento determinan el inicio y el fin de *run*, así como también el inicio y el fin de cada *spill* en el *run*. Tales señales se transmiten desde Dolina a cada una de las tarjetas Bora como comandos a través de la red de DSP (Sección 4.4).

Los comandos de inicio y fin de *run* son tratados como cualquier otro comando recibido desde la red, puesto que para efectuar la atención de los mismos el

DSP cuenta con el tiempo suficiente considerando que el intervalo entre la generación de los comandos de inicio de *run* y de inicio de *spill* es de 1 s. El tiempo, en cambio, entre la generación del comando de inicio de *spill* y la generación del primer *trigger* del *spill*, es de 200 ms. Por lo tanto, para asegurar la atención del comando antes del arribo del *trigger*, el DSP produce una interrupción interna de software con el arribo del comando de inicio de *spill*. El DSP cuenta con tres interrupciones internas que pueden ser definidas y generadas por el software. Del mismo modo, una segunda interrupción software se usa para el comando de fin de *spill*.

El DSP recibe los *triggers* a través de la línea de interrupción IRQ2 del procesador, que es la interrupción externa con mayor prioridad. Para evitar la pérdida de *triggers*, el servicio de dicha interrupción no puede absolutamente superar los 650 ns. El DSP genera el *header* del paquete de evento transmitido a los CATCH. En la rutina de atención a la interrupción (ISR) generada por el *trigger*, el DSP obtiene el número de evento y lo almacena en un *buffer* circular soportado por el hardware del procesador. El DSP cuenta con generadores de direcciones de datos (DAG) que proveen soporte en hardware de algunas funciones comúnmente usadas en algoritmos de procesamiento digital de señales [90]. Los DAG soportan *buffers* de datos circulares que sólo requieren del software para avanzar los punteros al *buffer*, usados para escritura o lectura. Las operaciones de módulo y control de los límites del *buffer* se efectúan automáticamente por el DAG.

EL DSP usa el *bit* 31 de la palabra de 32 bits usada para almacenar el número del evento en el *buffer circular*, denominado *event-buffer*, para especificar si el *trigger* recibido se generó en el *spill* o en el *interspill (bit 31=0 si trigger de spill y bit31=1 si trigger de interspill)*. Otra de las actividades realizadas por el DSP en la ISR del *trigger*, es la configuración de la FPGA para que responda a los *triggers* de

spill efectuando la adquisición normal de los canales del detector. Puesto que es el DSP quien responde al primer *trigger* de cada *spill* enviando a los CATCH información específica de la Bora, la configuración de la FPGA para la adquisición se realiza en la ISR (a través de la *Command Word*) luego del arribo del primer *trigger*. La Figura 6.8 muestra esquemáticamente las funciones realizadas por el DSP en la ISR generada por el *trigger*.

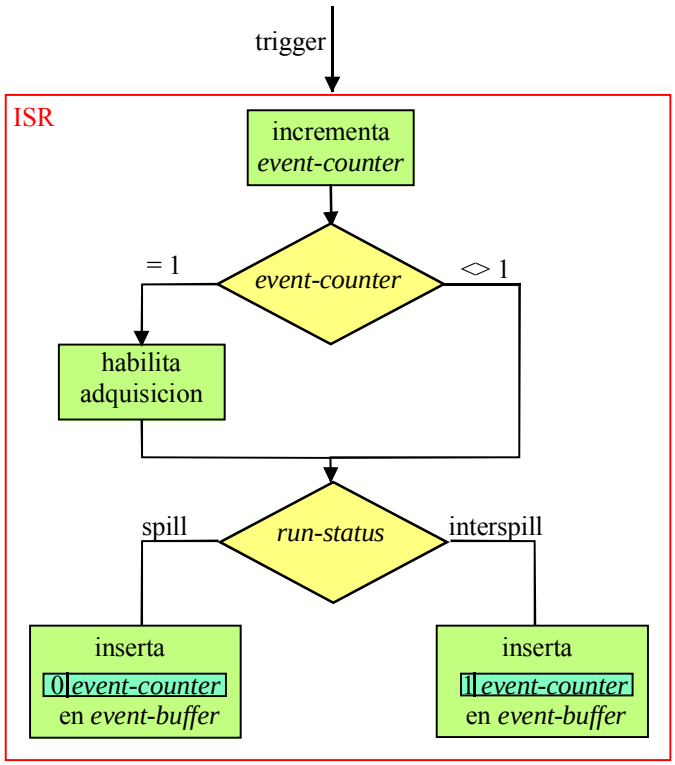

Figura 6.8 Funciones realizadas por el DSP en la ISR generada por el *trigger* durante el *run*.

Al retorno de la ISR del *trigger* se cuenta entonces con un *buffer* circular conteniendo, por cada *trigger*, el número de evento junto con la especificación del tipo del *trigger* (*spill* o *interspill*). De acuerdo a las características del *trigger*, es el DSP o la FPGA quien transmite a los CATCH el paquete correspondiente, y en todo caso es el DSP quien genera el *header* del paquete. Resumiendo, los *triggers* recibidos durante el *run* se dividen en las siguientes categorías:

a) **Primer *trigger* del *run***, para el cual es el DSP quien genera el paquete conteniendo el *engineering frame*, los umbrales y los resultados parciales del valor medio y la desviación estándar.

b) **Primer *trigger* de *spill***, para el cual es el DSP quien genera el paquete conteniendo los datos del *engineering frame.*

c) ***Trigger* de *spill***, para el cual es la FPGA quien genera el paquete de evento conteniendo el valor e identificador de los canales del detector que superaron su correspondiente umbral.

d) ***Trigger* de *interspill***, para el cual es el DSP quien genera el paquete de evento "vacío".

Durante el *run*, el DSP chequea continuamente el *buffer* circular *event-buffer* para determinar si existen *triggers*. Obviamente, el *event-buffer* se administra como una memoria FIFO (los *trigger* se atienden de acuerdo al orden de arribo). Mientras que *event-buffer* contenga elementos, el DSP lee del *buffer* el número de evento y determina la categoría a la que el *trigger* pertenece.

Si se trata del primer *trigger* del *run*, es decir el primer *trigger* del primer *spill* del *run*, o del primer *trigger* de *spill*, es decir el primer *trigger* de los *spill*s sucesivos al primero, transmite el paquete correspondiente previamente almacenado en la memoria interna del DSP (Sección 6.7.1). Si se trata de un *trigger* de *interspill*, prepara y transmite el paquete de evento "vacío" (Sección 6.7.2). En cada caso, para la transmisión del paquete, establece la FPGA en el modo de operación "DSP hacia DAQ".

Si se trata de un *trigger* de *spill*, antes de lanzar el procesamiento y transmisión de los canales adquiridos estableciendo e iniciando la FPGA en el modo de operación "Evento", el DSP realiza los siguientes chequeos:

a) Chequea, a través de la *Status Word*, que la FPGA no esté ocupada en el procesamiento y transmisión del paquete de evento correspondiente al *trigger* anterior al corriente.

b) Chequea, a través de la *Status Word*, que los CYFIFOs no estén vacíos. Si no existen eventos pendientes para ser elaborados y transmitidos, y puesto que el *trigger* arriba contemporáneamente al DSP y a la FPGA, podría ocurrir que el DSP lance el procesamiento y transmisión antes que los valores de los canales sean escritos en el CYFIFO.

Una vez corroborados los chequeos mencionados anteriormente, el DSP arma el *header* del paquete (incluyendo el número de evento) y lo escribe en el registro *Event Header* de la FPGA. Luego establece e inicia la FPGA en el modo de operación "Evento", lanzando de este modo el procesamiento y transmisión del evento correspondiente al nuevo *trigger*. La FPGA modifica el *header* transformándolo en el *trailer* (solamente un *bit* diferencia el *header* del *trailer* (Tabla 6.4)) para incluirlo al final del paquete de evento transmitido a los CATCH.

Operando en modo "Evento", la FPGA escribe el paquete de evento en dos FIFOs internos: *Hotlink FIFO*, desde donde el paquete se transmite a los CATCH, y *DSP FIFO*, desde donde el DSP tiene la opción de leer el paquete. Como la lectura por parte del DSP es opcional, la FPGA genera siempre un *reset* en el FIFO antes de escribir un nuevo paquete. Efectivamente, sólo una vez por *spill*, el DSP lee el paquete de evento desde el *DSP FIFO* para transmitirlo a Dolina. Para la lectura del

FIFO, el DSP establece la FPGA en el modo de operación “FPGA hacia DSP”. El DSP lee del *DSP FIFO* el paquete de evento correspondiente a un número de *trigger* previamente especificado y lo almacena en un *buffer* de la memoria interna, para enviarlo a Dolina luego del final del *spill*. De este modo el DSP no demora, durante el *spill*, el procesamiento y transmisión de los eventos correspondientes a *triggers* pendientes.

El paquete de evento generado por la FPGA contiene el valor y el identificador de los canales que superaron su correspondiente umbral, como muestra la Tabla 6.7.

0	0	0	0	BORA ID	Número de Evento
1	0	0	0	Identificador de canal (18 bits)	Valor de canal (10 bits)
1	0	0	0	Identificador de canal (18 bits)	Valor de canal (10 bits)
…					
1	0	0	0	Identificador de canal (18 bits)	Valor de canal (10 bits)
1	0	0	0	Identificador de canal (18 bits)	Valor de canal (10 bits)
0	1	0	0	BORA ID	Número de Evento

Tabla 6.7 Paquete de evento generado por la FPGA en el modo de operación “Evento”.

Luego del arribo del comando de fin de *spill*, el DSP arma los paquetes de red conteniendo el evento tal como fue generado desde la FPGA. La longitud del evento es variable en el rango entre 2 (*header* y *trailer*) y 432 (número máximo de canales adquiridos por Bora). Por lo tanto, puesto que la longitud de los paquetes de red es de 128, se requieren de uno a cuatro paquetes (de tipo “Evento”) para enviar el evento. Luego de armar los paquetes, el DSP configura el SPORT DMA para transmitirlos a Dolina.

6.8.1 Inicio de *Run*

Cuando el DSP recibe el comando de inicio de *run* (SOR), obtiene las temperaturas, los voltajes y los valores actuales de umbrales almacenados en la

FPGA, calcula la suma y la suma de los cuadrados de los valores de los canales tomando 2048 muestras, y prepara y almacena en *buffer-first-trigger* el paquete que será enviado a la cadena de adquisición global con el primer *trigger* del *run*. Luego inicia en 0 los contadores de *spill* y de *triggers*, y configura la FPGA, a través de la *Command Word*, para recibir *triggers* externos dejando bloqueada la adquisición de los canales del detector.

Un comando específico (*autothreshold*) puede ser enviado desde el "Controlador del RICH" para solicitar al DSP el cálculo de los valores de umbrales. En este caso el DSP también calcula el μ y el σ, a partir de las 2048 muestras, y obtiene los valores de umbrales (T_i) para cada canal, donde *i* identifica al canal y el factor α es un parámetro del comando (Sección 3.1.1):

$$T_i = \mu_i + \alpha\sigma_i$$

6.8.2 Inicio de *Spill*

Con el arribo del comando de inicio de *spill* (SOS), el DSP genera una interrupción interna de software (*soft-0*). El *engineering frame* se transmite a Dolina al inicio de cada *spill,* y a los CATCH con el primer *trigger* del *spill*. La rutina que atiende la interrupción (ISR) generada por el comando SOS, completa el *engineering frame*, almacenando en el mismo los valores de los contadores de *spill* y de *triggers*, e inicia el SPORT DMA para transmitirlo a Dolina. Luego incrementa el contador de *spill* e inicia en 0 el contador de *triggers*, dejando por último establecido en un *flag* interno (*run-status*) el inicio del *spill*. El DSP utiliza dicho *flag* en la ISR del *trigger* para determinar si el *trigger* recibido fue generado durante el *spill*.

6.8.3 Fin de *Spill*

Con el arribo del comando de fin de *spill* (EOS), el DSP genera una interrupción interna de software (*soft-1*). La rutina que atiende la interrupción (ISR) generada por el comando EOS, bloquea, a través de la *Command Word*, la adquisición de los canales puesto que es el DSP quien responde a los *triggers* de *interspill* enviando paquetes de evento "vacíos". Luego, especifica en el *flag run-status* el fin del *spill* de modo que el DSP pueda determinar en la ISR del *trigger* que el *trigger* recibido fue generado durante el *interspill*. Por último, deja establecido en un par de *flags* internos las dos tareas que el DSP debe efectuar luego del arribo del comando EOS, las cuales son la preparación del *engineering frame* para enviar a Dolina y los CATCH en el próximo *spill*, y la preparación y transmisión a Dolina de uno de los eventos generados en el *spill*. El DSP limpia los *flags* correspondientes una vez efectuadas, durante el *interspill*, cada una de estas tareas.

6.8.4 Fin de R*un*

Cuando el DSP recibe el comando de fin de *run* (EOR), bloquea, a través de la *Command Word*, la recepción de *triggers* externos.

6.9 Comunicación entre Bora y Dolina

Bora se comunica con Dolina por medio de una red TDM de DSP gestionada por el DSP de Dolina que actúa como *master* (Sección 4.3). El RICH cuenta con ocho redes conformadas por cada uno de los ocho DSP de Dolina y 24 DSP de las Bora que forman parte de cada una de las ocho cámaras del detector. El DSP de Bora actúa como *slave*, recibiendo las señales de inicio de *frame* (RFS) y reloj de referencia

(RCLK) generadas por el DSP de Dolina (Figura 3.5). La red se configura en modo TDM estableciendo el número de *slots* en 25, que coincide con la cantidad de integrantes de la red, y la longitud del *slot* en 32 *bit*s.

A cada DSP de Bora le corresponde un *slot*, y sólo durante tal *slot* intercambia paquetes con Dolina (Tabla 3.1). El *slot* correspondiente a cada DSP de Bora coincide con el identificador del DSP en la red (Figura 3.6), el cual se obtiene del *Bora Id*. El DSP incluye el identificador en el campo *fuente* de los paquetes de red, y lo utiliza también para habilitar el *slot* de comunicación y chequear que la destinación de los paquetes recibidos sea correcta.

Como la comunicación se efectúa en un único *slot*, el DSP usa SPORT DMA tanto para la transmisión como para la recepción de paquetes. El controlador de DMA manipula la transferencia de datos entre la memoria interna del procesador y los *buffers* de transmisión y recepción del SPORT, habilitando al DSP a continuar con la ejecución del programa hasta el final de la transferencia de un bloque entero de datos. Esta propiedad permite al DSP de Bora recibir y transmitir paquetes desde y hacia la red durante la adquisición de datos en la que el DSP, a causa de las severas restricciones temporales, debe dedicarse casi exclusivamente al servicio de los *triggers* y a la interacción con la FPGA.

Para la transmisión de paquetes, el DSP inicializa en primer lugar los parámetros del DMA (dirección base del *buffer* que contiene el/los paquetes y longitud del *buffer*), luego habilita el transmisor DMA, y por último habilita la interrupción de transmisión del SPORT. Una vez que el *buffer* completo fue enviado por el controlador de DMA, el SPORT genera la interrupción y el DMA se deshabilita automáticamente. Por lo tanto, la única tarea que realiza la rutina que atiende la interrupción de transmisión del SPORT es deshabilitar la interrupción.

Para la recepción de paquetes, el DSP reserva en su memoria interna un *buffer* (*rc-buffer*) con capacidad para un paquete "largo" (128 palabras de 32 bits). Sin embargo, Dolina trasmite hacia la Bora los siguientes tipos de paquetes:

a) Paquetes conteniendo una sola palabra en el caso de las señales de sincronización *"Inicio de spill"* y *"Fin de spill"* (Sección 4.4).
b) Paquetes de comando conteniendo dos palabras.
c) Paquetes de datos conteniendo 128 palabras.

Para la recepción de los comandos de inicio y fin de *spill* se enmascara la palabra clave, la cual usualmente coincide con la primer palabra de los paquetes recibidos por el receptor del SPORT. El DSP provee la posibilidad de enmascarar la palabra clave en modo tal de poder seleccionar solamente un grupo de los 32 bits que la conforman. Por este motivo, la palabra que identifica el comando de inicio de *spill* es "HOLJ" y la palabra que identifica el comando de fin de *spill* es "HOLE", cuando para el resto de los paquetes la primer palabra es "HOLA". Esto permite establecer en el DSP de Bora la palabra clave como "HOL" enmascarando el byte menos significativo. Así, cuando el receptor del SPORT genera una interrupción indicando el arribo de una palabra que coincide con la palabra clave, la rutina que atiende la interrupción ejecuta los siguientes pasos:

a) Si la palabra coincide con uno de los comandos de inicio o fin de *spill*, genera la interrupción de software correspondiente al comando (Secciones 6.8.2 y 6.8.3).
b) Si la palabra es "HOLA", deshabilita la palabra clave, configura y habilita el DMA para recibir una única palabra, dejando indicado en un *flag* (*rc-status*) que la próxima interrupción indicará el arribo de la segunda palabra del paquete.

Cuando se genera la interrupción por la llegada de la segunda palabra del paquete, la rutina que atiende la interrupción ejecuta los siguientes pasos:

a) Chequea la destinación del paquete y si detecta error, informa del mismo con su correspondiente código al *kernel* que se encargará de transmitir el mensaje de error a Dolina. Luego habilita la palabra clave, dejando especificado en *rc-status* que la próxima interrupción indicará la llegada de la primer palabra de un paquete.

b) Si la destinación es correcta, determina si la palabra corresponde a un paquete de comando o es la segunda palabra de un paquete de datos.

c) Si se trata de un paquete de comando, informa al *kernel* por medio de un *flag* (*new-packet*) que existe un paquete de comando en *rc-buffer* listo para ser procesado, y habilita la palabra clave, dejando especificado en *rc-status* que la próxima interrupción indicará el arribo de la primer palabra de un paquete.

d) Si se trata de un paquete de datos, deshabilita la palabra clave, y configura y habilita el DMA para recibir el resto del paquete (126 palabras), dejando especificado en *rc-status* que la próxima interrupción indicará el arribo de las palabras restantes de un paquete de datos.

Cuando se genera una interrupción indicado la llegada de las últimas 126 palabras de un paquete de datos, la rutina que atiende la interrupción ejecuta los siguientes pasos:

a) Chequea que la última palabra del paquete sea “CHAU” y si detecta error, informa del mismo con su correspondiente código al *kernel* que se encargará de transmitir el mensaje de error a Dolina. Luego habilita la palabra clave, dejando especificado en *rc-status* que la próxima interrupción indicará el arribo de la primer palabra de un paquete.

b) Si la última palabra es correcta, informa al *kernel* (por medio de *new-packet*) que existe un paquete de datos en *rc-buffer* listo para ser procesado, y habilita la palabra clave, dejando especificado en *rc-status* que la próxima interrupción indicará el arribo de la primer palabra de un paquete.

6.10 Programación del DSP de Bora

El *booting*, el *loader* y la reprogramación del DSP es igual en Dolina y en Bora (Sección 4.6). La única diferencia es que el *loader* de Bora carga los paquetes con el programa del DSP por medio de la red TDM, mientras el *loader* de Dolina carga los paquetes desde la DPM.

El *loader* carga en la memoria interna del procesador e inicia la ejecución del programa del DSP. La Figura 6.9 muestra un esquema de la memoria interna del DSP de Bora. En la memoria de programa, las direcciones 0x8000 – 0x80ff contienen el vector de interrupciones y el *loader* ocupa las direcciones 0x8100 – 0x834f. El *loader* utiliza una zona de almacenamiento temporario para el vector de interrupciones del programa. El programa del DSP se carga a partir de la dirección 0x8350. En la memoria de datos el *loader* ocupa las direcciones superiores 0xdee0 – 0xdfff, quedando el resto de la memoria a disposición del programa.

El *loader* utiliza SPORT DMA tanto para la transmisión como para la recepción de paquetes, por lo tanto reserva dos *buffers*, uno para almacenar los paquetes de comandos que serán enviados a la PC por el transmisor DMA, y otro para almacenar los paquetes de comando o de programa recibidos desde la PC por el receptor DMA. Cuando el controlador DMA finaliza la transmisión o recepción de un paquete completo, el SPORT genera la interrupción correspondiente. Para la recepción de paquetes, se estable la palabra clave “HOLA” en el slot activo y se

configura el DMA para recibir en primer lugar las dos palabras de encabezamiento del paquete. Si el encabezamiento no contiene errores y corresponde a un paquete "largo", se deshabilita la palabra clave y se configura nuevamente el DMA para recibir el resto de las palabras del paquete. En consecuencia, al final de la recepción de un paquete de programa se vuelve a habilitar la palabra clave.

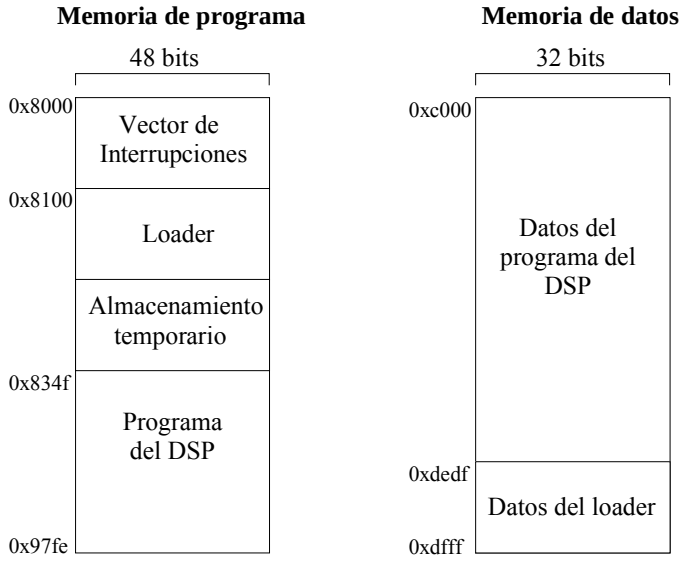

Figura 6.9 Memoria interna en el DSP de Bora.

El programa principal del *loader* configura el SPORT en modo TDM y habilita el *slot* correspondiente al identificador del DSP en transmisión y recepción, habilita también la palabra clave, el receptor DMA y la interrupción de recepción del SPORT. Luego permanece esperando por el comando de inicio (Figura 3.8). El transmisor DMA y la interrupción de transmisión del SPORT sólo se habilitan para enviar a la PC los comandos de requerimiento y reconocimiento, o en caso de error el mensaje correspondiente.

La rutina que atiende la interrupción de recepción del SPORT chequea el paquete e informa al programa principal del arribo del paquete a través de un *flag*, indicando además las características del paquete, o en caso de detectar algún error el código del

mismo. El programa principal responde al comando de inicio con un comando de requerimiento del programa. Si el paquete recibido es “largo”, el programa principal copia el contenido en las direcciones apropiadas de la memoria interna del DSP. La única excepción es cuando el paquete contiene el vector de interrupciones en cuyo caso almacena el contenido en la zona de almacenamiento temporario.

Luego de copiar el último paquete de programa, el programa principal transfiere el vector desde la zona de almacenamiento temporario a las direcciones reservadas al vector de interrupciones en la memoria de programa. Finalmente envía a la PC el comando de reconocimiento e inicia la ejecución del programa. Si algún error es detectado durante la recepción de paquetes, el programa principal informa a la PC enviando el paquete de error correspondiente y genera un *reset* por software iniciando nuevamente el *booting* del DSP.

6.11 Estructura General del Software e Interrupciones

El software del DSP de Bora implementa un sistema de tiempo real que satisface restricciones de tiempo duras y blandas. En particular, las restricciones temporales duras que deben ser satisfechas durante el *spill*, exigieron la escritura del software en lenguaje *assembly* del procesador en modo de minimizar el tiempo de servicio de las interrupciones y hacer un uso eficiente del paralelismo provisto por la arquitectura del DSP. Si bien el DSP puede ser programado en lenguaje C, generalmente el compilador no hace un uso eficiente de la arquitectura subyacente del DSP e incrementa el tiempo de servicio de las interrupciones. El tiempo que el compilador utiliza para salvar en el *stack* de ejecución la información de contexto al inicio del servicio de una interrupción está en el orden de los microsegundos, lo mismo que el tiempo requerido para recuperar dicha información al final del servicio.

Para atender las interrupciones durante el *spill*, manteniendo el tiempo de servicio en el orden de los nanosegundos, se utilizan los 16 registros de propósito general del DSP para minimizar la cantidad de datos a salvar y recuperar. Cada una de las rutinas de servicio de interrupción usa, en manera exclusiva, un grupo de tales registros evitando de este modo que los mismos sean salvados al inicio del servicio de la interrupción, y en consecuencia recuperados al final. Por otra parte, la arquitectura *Super Harvard* del DSP [52, 53, 55, 57], caracterizada por contar con *buses* separados para la memoria de datos y la memoria de programa, permite el uso de un grupo de instrucciones que se ejecutan en paralelo, es decir en un único ciclo del reloj del procesador.

El software responde a las siguientes seis interrupciones (internas e externas), representadas esquemáticamente en la Figura 6.10:

- La interrupción externa IRQ2 comunica la llegada del *trigger*.
- Las interrupciones internas de software *soft-0* y *soft-1* comunican el arribo de los comandos de inicio y fin de *spill* respectivamente.
- La interrupción interna del *timer* se usa para controlar el dispositivo *WatchDog*.
- La interrupción interna del transmisor del SPORT comunica el final de la transmisión de un bloque de datos, por el controlador DMA, a la red de DSP.
- La interrupción interna del receptor del SPORT comunica el final de la recepción de un paquete, por el controlador DMA, proveniente de la red de DSP.

Las interrupciones en el DSP se habilitan individualmente y globalmente, y sus prioridades están prefijadas en el procesador [90]. Las interrupciones externas tienen la mayor prioridad. Las interrupciones del SPORT tienen mayor prioridad que la interrupción del *timer* y entre ellas la interrupción del receptor tiene mayor prioridad que la del transmisor. Las interrupciones de software son las de menor

prioridad. Como en Dolina, para gestionar las interrupciones no se utilizó la opción de anidamiento, es decir, las rutinas de atención de interrupción (ISR) no se interrumpen quedando pendiente el servicio de una eventual nueva interrupción para el retorno de la ISR. Por lo tanto, el software se desarrolló en modo de usar el menor tiempo posible en las ISR dejando al programa principal (y sus rutinas) la mayor cantidad de actividades.

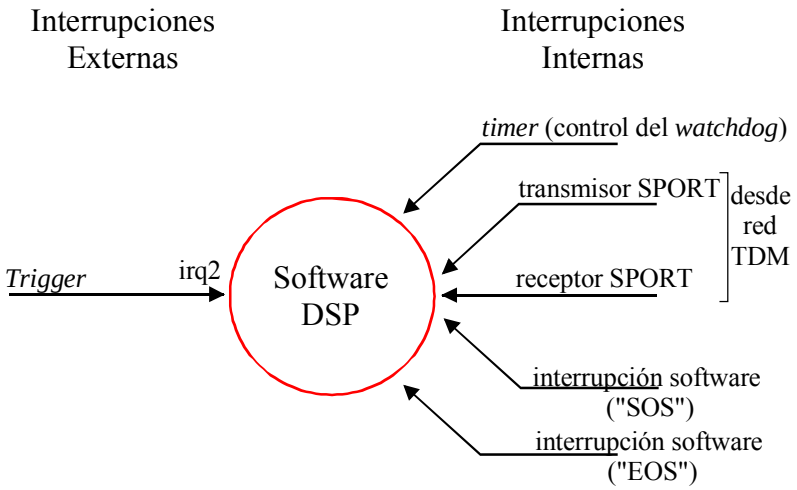

Figura 6.10 Interrupciones externas e internas generadas en el DSP de Bora.

La estructura global del software del DSP de Bora es la misma que la del DSP de Dolina (Figura 4.18). El *kernel* (núcleo) del software es el programa principal quien ejecuta una primer etapa de inicialización seguida de una segunda etapa en la cual las interrupciones están habilitadas y pueden generarse en cualquier momento. En esta segunda etapa, la llegada de alguna de las interrupciones habilitadas genera la automática ejecución de la ISR correspondiente. Generalmente la ISR modifica un *flag* indicando al *kernel* las tareas a ejecutar para completar el servicio de la interrupción. El *kernel* permanece en un ciclo infinito en el cual chequea permanentemente una serie de *flags* y dependiendo de los valores de los mismos,

invoca la tarea (una o más rutinas) correspondiente. A su vez, tales *flags* son limpiados una vez completado el servicio de la interrupción.

Capítulo 7

La FPGA como Coprocesador del DSP

La FPGA, en Bora, opera como coprocesador paralelo del DSP. Dada su naturaleza versátil, la FPGA se configura para ejecutar varias tareas en paralelo y descargar así al DSP de ciertas operaciones intensivas de cálculo y de control del hardware de Bora. Entre las tareas más importantes que la FPGA ejecuta se encuentra el control de la cadena de adquisición de datos de Bora, y la elaboración y transmisión de los paquetes de evento durante el *run* del experimento.

La configuración de la FPGA está a cargo del DSP. Una vez configurada, la FPGA opera diversamente según distintos modos de operación establecidos por el DSP. En este capítulo se describe la FPGA de Bora como coprocesador del DSP. Se describe su arquitectura global, y el procedimiento de configuración efectuado por el DSP.

7.1 Arquitectura Global de la FPGA

El diseño de la FPGA de Bora está basado en una arquitectura global que utiliza distintos bloques arquitecturales específicos disponibles en la familia de FPGAs Virtex [93], como memorias RAM, DLLs, y estructuras de *input-output*. El diseño fue descripto en el lenguaje de descripción de hardware VHDL [76, 78, 79, 80].

Para comprender la arquitectura global se resume a continuación las funciones principales efectuadas por la FPGA:

a) Interacción con el DSP

La FPGA permite acceder al DSP a los registros y bloques de memoria a través de los cuales el DSP interactúa con la FPGA. El DSP accede a tales estructuras por medio del *bus* de datos y el *bus* de direcciones externos, junto a las señales RD, WR y MS0.

b) Control de la Cadena de Adquisición de Datos de Bora

La FPGA provee todas las señales necesarias para el control de los Gassiplex, los ADC y los CYFIFO a fin de adquirir, digitalizar y almacenar los datos generados por cada canal.

c) Sustracción de los Umbrales a los Datos Adquiridos

La FPGA provee todas las señales necesarias para la lectura de los CYFIFO, efectúa la sustracción de los umbrales correspondientes a cada canal, y prepara los paquetes de evento adjuntando el identificador de cada canal.

d) Control del *Hotlink* para la Transmisión de Datos

La FPGA provee las señales necesarias para el *Hotlink* a fin de transmitir los paquetes de evento a los dispositivos CATCH.

La FPGA también cumple otras funciones tales como conectar ciertas señales externas con bits de registros internos de la FPGA a los cuales accede el DSP. Con este mecanismo el DSP puede consultar el estado de los CYFIFO, y puede controlar los *switches* analógicos del circuito de *self testing* (Sección 5.2.4).

La Figura 7.1 muestra esquemáticamente la arquitectura global implementada en la FPGA de Bora en la que se distinguen los principales bloques funcionales, estructuras y señales conectadas a la FPGA. En general, cada bloque funcional tiene asociado una máquina finita de estados. Las mismas cuentan con una señal de reset (*p-reset*) que proviene de un *flag* del DSP.

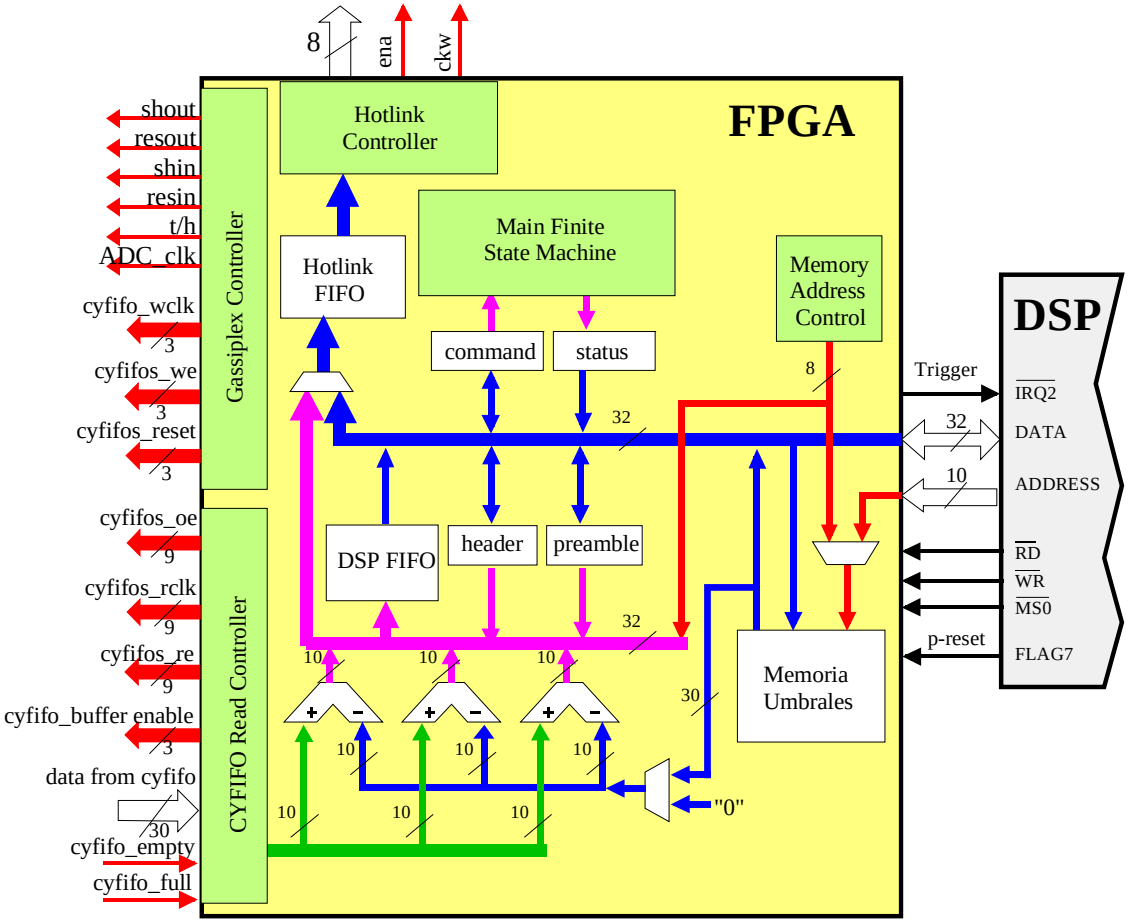

Figura 7.1 Arquitectura Global de la FPGA de Bora.

Los *buses* externos del DSP se conectan directamente con *buses* internos de la FPGA permitiendo de este modo el acceso a las estructuras a través de las cuales el DSP interactúa con la FPGA.

La generación de señales para la cadena de adquisición de datos está a cargo de un bloque funcional llamado *Gassiplex Controller*.

Tres sustractores de diez bits que operan en paralelo efectúan las sustracciones de umbrales tomando los datos desde los CYFIFO y los umbrales desde la memoria de umbrales. El bloque funcional *Memory Address Control* genera las direcciones de memoria para la lectura secuencial interna de los umbrales, y el bloque funcional *CYFIFO Read Controller* gestiona de la lectura de los CYFIFO. Los resultados de las sustracciones se conectan a un *bus* interno, el cual a su vez se conecta a los puertos de escritura de las memorias FIFO internas: *Hotlink FIFO y DSP* FIFO.

Una máquina principal de estados llamada MFSM (*Main Finite State Machine*) controla el procesamiento de los datos adquiridos y el armado de los paquetes de evento.

El *Hotlink FIFO* es leído por un bloque funcional llamado *Hotlink Controller* que escribe el contenido del FIFO en el *HotLink*.

A continuación se describe cada una de las funciones de la FPGA y sus diversas interrelaciones, y se especifica también la tempística de las funciones críticas determinantes de la performance del sistema de adquisición.

7.1.1 Interacción con el DSP

La interacción con el DSP se realiza principalmente por medio del *bus* de datos y de direcciones externos a través de los cuales el DSP accede a diversos registros y bloques de memoria de la FPGA (Figura 6.4). La FPGA está mapeada en el banco 0 de la memoria externa del DSP, así la activación de la señal MS0 es una condición necesaria para cualquier acceso del DSP a la FPGA. La FPGA tiene un bloque de memoria RAM dedicado al almacenamiento de los umbrales (Memoria de

Umbrales), a la cual el DSP tiene acceso de lectura y escritura. La FPGA tiene también dos memorias FIFO: *Hotlink FIFO* y *DSP FIFO*, a las cuales el DSP tiene acceso de escritura y lectura respectivamente. El DSP accede a cuatro registros internos de la FPGA: *Command Word* (lectura-escritura), *Status Word* (lectura), *Event Header* (escritura) y *Preamble* (escritura).

Las diversas funcionalidades de la FPGA se activan siguiendo seis modos de operación mutuamente excluyentes según sea requerido por el DSP. La Sección 6.3 describe tales modos de operación. La máquina principal de estados (MFSM), controla periódicamente tres bits del registro *Command Word* correspondientes al modo de operación, para decidir ciertos cambios de estado en función del nuevo modo de operación requerido. El DSP es quien decide los modos de operación escribiendo estos tres bits. Ciertas funciones requieren el paso de la FPGA por distintos modos, en estos casos el software del DSP establece la FPGA en los distintos modos de operación en la secuencia correcta.

El registro *Command Word* (32 bits) se utiliza por el DSP para establecer e iniciar la FPGA en los distintos modos de operación, para bloquear los *triggers* externos, para generar *triggers* internos, para bloquear la adquisición de datos respondiendo a los *triggers* externos, para pasar parámetros y para actuar sobre señales específicas. La Tabla 7.1 describe la funcionalidad de los bits que conforman la *Command Word*.

El registro *Status Word* (32 bits) contiene información sobre el estado de la FPGA en los distintos modos de operación. Específicamente describe el estado de las distintas máquinas de estado de la FPGA, y el estado de los CYFIFO. La Tabla 7.2 describe la funcionalidad de los bits que conforman la *Status Word*.

Bit	Nombre	Descripción
0	*Start*	Inicio en modo “evento” y “test de canal”.
1:3	*Mode*	Modo de operación de la FPGA.
4:7	-	-
8:12	*Ritardo*	Retardo entre la llegada del *trigger* y el *hold* de las salidas de los canales analógicos de los Gassiplex, expresado en ciclos de reloj de 12,5 ns.
13	*Auto*	Activación de la adquisición respondiendo a *triggers* externos.
14	*block-trigger*	Bloqueo del *trigger* externo.
15	*DSP-trigger*	Generación del *trigger* interno.
16:17	*p-voltage*	Activación de los reguladores de voltajes analógicos.
19	*Gass-rst*	Gassiplex Reset.
20	*p-charge-cap*	Control del *switch* analógico que establece el contacto entre la salida del DAC y el capacitor de test.
21	*p-discharge-cap*	Control del *switch* analógico que establece el contacto entre el capacitor de test y la resistencia de test.
22:26	-	-
27	*cy-full-sticky-reset*	Reset del bit *cy-full-sticky (Status Word*).
28	*p-shift-test-in*	Señal de reloj del *shift register* de *testin* de los Gassiplex. Selecciona secuencialmente los canales para la estimulación vía la señal *testin*.
29	*p-rst-test-in*	Reset del *shift register* de *testin* de los Gassiplex.
30	*cy-rst*	CYFIFO Reset.
31	-	-

Tabla 7.1 Registro *Command Word*.

Bit	Nombre	Descripción
0	*Hotlink-busy*	Estado del *Hotlink Controller*.
1	*DSPfifo-empty*	Estado del *DSP FIFO*.
2	*idle-flag*	Estado de la *MFSM*.
3	*DAQ-busy*	Estado del bloque de adquisición de datos.
4	*New-trigger*	Posibilidad de servir un nuevo *trigger*.
5	*p-cy-empty*	CYFIFO vacío.
6	*p-cy-full*	CYFIFO completo.
7	*cy-full-sticky*	CYFIFO completado (requiere reset explícito).
8:31	-	-

Tabla 7.2 Registro *Status Word*.

7.1.2 Control de la Cadena de Adquisición de Datos de Bora

La actividad de la FPGA en la adquisición de datos comienza con el arribo de un *trigger*, sea este interno o externo. El *trigger* inicia una máquina finita de estados perteneciente al bloque funcional *Gassiplex Controller*. Esta máquina de estados progresa por una secuencia determinada de estados generando de este modo las señales necesarias a los Gassiplex, los ADC y los CYFIFO. Como resultado de esta

actividad se almacenan en los CYIFO los valores de los 432 canales adquiridos por Bora.

Al arribo del *trigger* se activa un contador (*timer*) para activar la señal de *track/hold* y retener (*hold*) los valores analógicos de los canales en el instante en que los canales con señal alcanzan su valor máximo. El tiempo que transcurre entre el estímulo y el arribo del *trigger* (prácticamente instantáneo) se denomina *trigger delay time*, y es un tiempo constante y característico de la estructura experimental que depende esencialmente de la lógica de generación del *trigger* y de la demora en la transmisión de las señales involucradas. Al *peaking time* (Sección 5.2.1) se le sustrae el *trigger delay time*, y el tiempo resultante es el tiempo que debe esperar la FPGA para lanzar la secuencia de adquisición de datos. El parámetro *ritardo*, almacenado en la *Command Word*, expresa este tiempo en unidades de ciclos de reloj de 80 MHz.

Una vez activada la señal de t*rack/hold* (t/h), los 16 canales de cada Gassiplex salen multiplexados dos a dos por medio de la señal *shift out*. Estas salidas analógicas son muestreadas simultáneamente por el conversor *dual-channel* ADC en el flanco positivo del reloj del ADC, quien después de tres ciclos de reloj entrega el resultado de las conversiones multiplexadas en el tiempo en ambos flancos del reloj. Las salidas de los ADC se escriben directamente en los CYFIFO. Así por cada *trigger* se escriben 16 palabras en cada CYFIFO. La Figura 7.2 muestra las formas de onda de las señales principales involucradas en la cadena de adquisición de datos.

En la figura, *ADC data out* representa la salida digital del conversor, y los números representan el canal correspondiente al dato. Debido a la arquitectura *pipeline* del ADC, los datos aparecen tres ciclos de reloj después del muestreo. En cada flanco positivo del *ADC clock*, el ADC muestrea dos señales analógicas

simultáneamente y sus datos correspondientes aparecen multiplexados en el tiempo sobre ambos flancos del reloj.

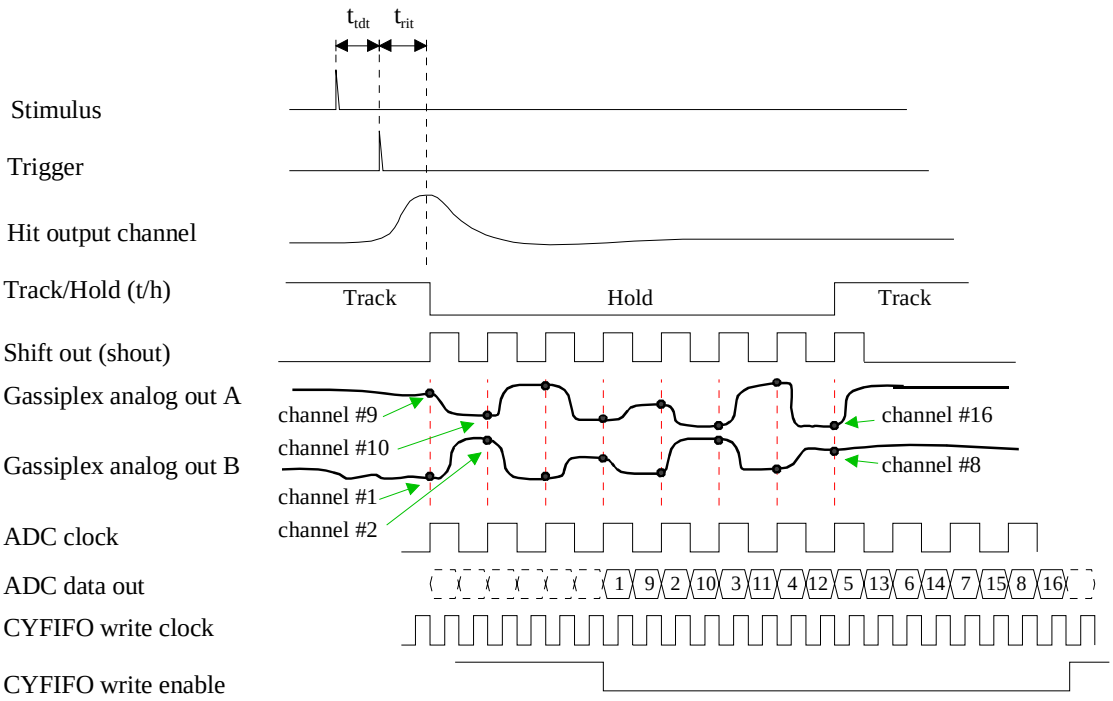

Figura 7.2 Secuencia de las señales principales involucradas en la cadena de adquisición de datos de Bora. Los tiempos t_{tdt} y t_{rit} corresponden a *trigger delay time* y *ritardo* respectivamente. El *peaking time* es igual a la suma de los tiempos t_{tdt} y t_{rit}.

El tiempo efectivo empleado por *trigger* para almacenar un evento en los CYFIFO es de 1,2 µs, es decir que la FPGA puede soportar una frecuencia "instantánea" de *triggers* de hasta 0,8 MHz dependiendo del espacio disponible en los CYFIFO. Más aún, *el Gassiplex Controller* puede aceptar, pero no procesar, un solo *trigger* después de 500 ns del *trigger* precedente si es que este *trigger* precedente encontró al *Gassiplex Controller* libre. Un *trigger* así aceptado no comenzará a procesarse hasta después de 1,2 µs del arribo del *trigger* precedente. La figura 7.3 muestra una posible sucesión de *triggers* aceptados y rechazados, y sus correspondientes tiempos de servicios.

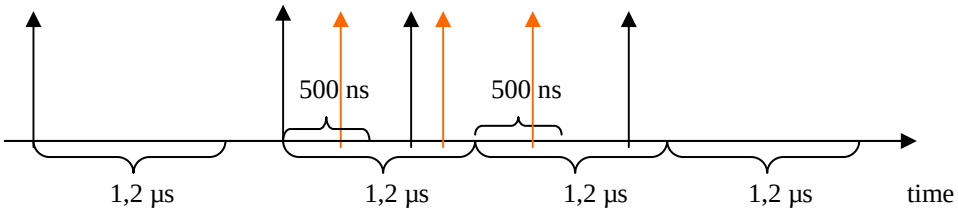

Figura 7.3 Una posible sucesión de *triggers* y sus tiempos de servicio. Las flecha rojas indican los *triggers* rechazados y las negras los aceptados. A cada *trigger* corresponde un tiempo de servicio de 1,2 µs y un tiempo muerto de 500 ns.

7.1.3 Sustracción de los Umbrales a los Datos Adquiridos.

El DSP configura la FPGA para efectuar la lectura y procesamiento de los datos adquiridos, una vez almacenados en los CYFIFO. La lectura de los datos se realiza secuencialmente leyendo una palabra de 30 bits a la vez de cada CYFIFO. Los 30 bits corresponden a tres canales, uno por cada uno de los tres Gassiplex de cada conector.

En el modo de operación “Evento”, la FPGA lee una palabra del CYFIFO y simultáneamente lee una palabra de la memoria interna de umbrales, conteniendo los tres umbrales correspondientes a los tres canales. El valor de cada canal y su respectivo umbral se transfieren a las entradas de los tres sustractores de la FPGA. La FPGA realiza así las tres sustracción es de umbral en paralelo. Sólo los resultados mayores o iguales a cero son almacenados en una palabra de 32 bits, mostrada esquemáticamente en la Figura 7.4, conteniendo el resultado de la sustracción y el identificador del canal (*Channel ID*). Dichas palabras de 32 bits conforman el cuerpo del paquete de evento y se escriben en ambos FIFOS internos (*DSP FIFO y Hotlink FIFIO*).

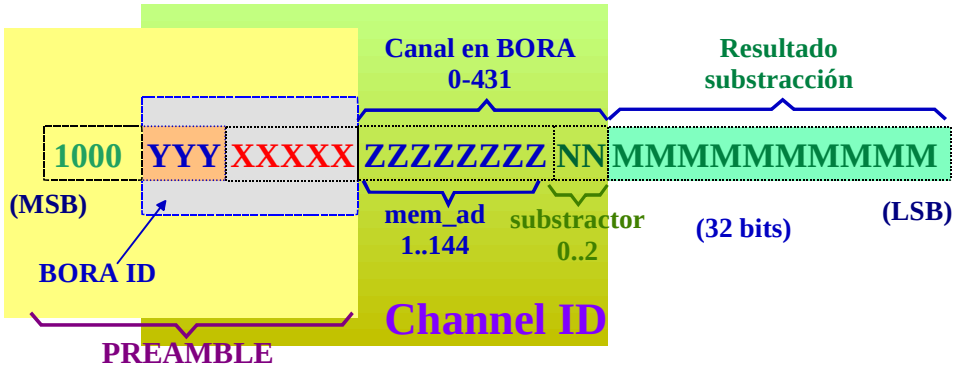

Figura 7.4 Formato de dato de un canal de evento.

El dato de un canal de evento se conforma por los siguientes campos [89] (comenzando del MSB):

1. Los cuatro bits más significativos sirven para la sincronización con los dispositivos CATCH, son utilizados por los mismos para distinguir el dato del *header* y del *trailer* del evento.
2. Los siguientes ocho bits YYYXXXXX contienen el *Bora Id*. El DSP escribe los bits de sincronización y el *Bora Id* en el registro *Preamble* de la FPGA.
3. Los siguientes ocho bits ZZZZZZZZ corresponden a la dirección de la memoria de umbrales que contiene el valor de umbral que la FPGA usó para la sustracción.
4. Los siguientes dos bits NN identifican el sustractor (0, 1 o 2) que fue utilizado para determinar que el valor del canal supera su correspondiente valor de umbral.
5. Los últimos 10 bits MMMMMMMMMM contienen el valor de la sustracción entre el valor del canal digitalizado y su correspondiente umbral.

En el modo de operación “Test de Canal”, la FPGA procede de la misma manera que en el modo “Evento” con la excepción que se sustituye el valor de umbral por *cero* en las entradas de los sustractores, en modo tal de adquirir el valor absoluto de cada canal. En este caso el dato de evento se escribe solamente en el DSP FIFO.

Por cada palabra leída de un CYFIFO (tres canales) la FPGA emplea 6 ciclos de reloj de 80 MHz para su procesamiento, es decir 75 ns. Así la estricta lectura y procesamiento de los 432 canales comporta 10,8 µs. A este tiempo se le debe agregar el tiempo de escritura del *header* y del *trailer* del paquete de evento en los FIFOS internos de la FPGA. El tiempo total de procesamiento y transmisión de un evento es de 12,7 µs.

7.1.4 Control del HotLink para la Transmisión de Datos

Los datos escritos en el *Hotlink FIFO* son automáticamente leídos por una máquina finita de estados que se activa cada vez que el FIFO tiene algún dato. Cada palabra leída del *Hotlink FIFO* se escribe, un byte a la vez, en el transmisor *Hotlink*. Para que el *Hotlink* acepte el byte para la transmisión, se provee un reloj específico de 40 MHz (CKW) y una señal específica de habilitación (ENA). CKW y ENA son provistos por el bloque *Hotlink Controller* de la FPGA. La velocidad efectiva de transmisión de datos es de 40 MBytes/s. Después de la serializacion y protocolización efectuada por el *Hotlink*, la velocidad de transmisión a través de la fibra óptica es de 400 Mbits/s, de los cuales 360 Mbits/s es la transmisión efectiva.

7.1.5 Control de Señales Específicas a través de la *Status Word* y de la *Command Word*.

Las señales *cyfifo-empty* y *cyfifo-full* provenientes de los CYFIFO entran en la FPGA y son disponibles en el registro *Status Word*. El DSP puede leer este registro y así consultar el estado de los CYFIFO. Un tercer bit *cyfifo-full-sticky*, retiene el último valor alto de *cyfifo-full* hasta que no sea explícitamente reestablecido por el DSP. De

esta manera es posible saber si el CYFIFO estuvo *full* en algún momento anterior a la consulta.

Desde el registro *Command Word* es posible comandar otras señales externas que controlan diversos dispositivos de la tarjeta Bora. Estas señales están directamente conectadas a ciertos bits de la *Command Word*, de modo tal que el DSP puede establecer el estado de tales señales escribiendo en los bits correspondientes. Este mecanismo permite al DSP controlar algunas señales externas a través del *bus* de datos y de direcciones externos sin tener que recurrir al uso directo de los *flags* programables del procesador, especialmente cuando la velocidad de control requerida no es crítica.

7.2 Configuración de la FPGA

La configuración de la FPGA (Sección 3.4.2) se efectúa en modo *Slave Serial* [94]. El DSP carga la cadena de bits conteniendo el diseño en la memoria de configuración interna de la FPGA. La FPGA cuenta con ocho puertos de entrada/salida, descriptos en la Tabla 7.3, reservados para la configuración.

Nombre	Dirección	Descripción
CCLK	I	Reloj de configuración
Program	I	Reset asíncrono
Done	O	Estado de configuración
M2, M1, M0	I	Modo de configuración
DIN	I	Entrada de datos serial
Init	O	Demora y error de configuración

Tabla 7.3 Puertos de configuración de la FPGA (“I” significa entrada y “O” significa salida)

El proceso de configuración de la FPGA se inicia al *power-up* o luego de un reset generado a través del puerto de entrada *Program*. Por medio de los puertos de

entrada *M2-0* se selecciona el modo de configuración. Durante la configuración se efectúan al interno de la FPGA las siguientes actividades:

1. Al *power-up* o luego de un reset generado a través de *Program*, el cual es activo bajo, la FPGA limpia la memoria de configuración. Cuando *Program* se vuelve alto, la FPGA mantiene *Init* bajo hasta que la memoria esté vacía. Para generar el reset, el tiempo mínimo en el cual *Program* se debe mantener bajo es de 300 ns.
2. Una vez que *Init* se vuelve alto, puede comenzar la carga de la cadena de bits con la configuración la FPGA.
3. En modo *Slave Serial* (*M2*=1, *M1*=1, *M0*=1), la FPGA se configura cargando un *bit*, a través del puerto de entrada *DIN*, por cada ciclo de reloj generado externamente, por medio del puerto de entrada *CCLK*. La carga comienza por el bit más significativo de cada byte de la cadena.
4. Durante la carga, se efectúan controles de seguridad comparando un valor de CRC (*Cyclic Redundancy Code*) embebido en la cadena con otro calculado internamente. En caso de error, se aborta el proceso de configuración e *Init* se vuelve bajo indicando la presencia del error de CRC.
5. Si durante la carga no se detectan errores, el puerto de salida *Done* se vuelve alto indicando el final del proceso de configuración y la FPGA se vuelve funcional con la configuración cargada.

El DSP cuenta con doce *flags* de propósito general programables por software como puertos de entrada o salida. Durante el proceso de configuración, el DSP interactúa con la FPGA a través de cinco de tales *flags*. La Figura 7.5 muestra la interfase entre los *flags* programables del DSP y los puertos de configuración de la FPGA.

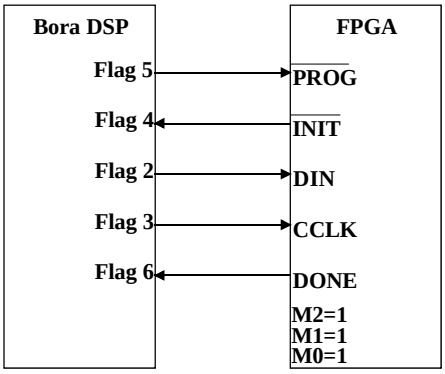

Figura 7.5 Interacción entre el DSP y la FPGA durante el proceso de configuración de la FPGA.

7.2.1 FPGA *Loader*

El programa *FPGA loader* se carga en el DSP de Bora para configurar la FPGA. El *FPGA loader* permite la reconfiguración de la FPGA en cualquier momento que sea requerido desde el "Controlador de RICH".

La primer actividad que el *FPGA loader* realiza es generar un reset en la FPGA a través del *Flag* 5 conectado al puerto de configuración *Program* (Paso 1 en el proceso de configuración). Luego espera que la señal *Init* se vuelva alta, chequeando continuamente el *Flag* 4 conectado al puerto de configuración *Init* (Paso 2 en el proceso de configuración). Cuando *Init* indica que la FPGA está lista para recibir la cadena de bits con la configuración, comienza la interacción entre el *FPGA loader* y la PC (Figura 3.9).

El *FPGA loader* solicita la configuración a través del comando *"FPGA Req."*, al que el "Controlador de RICH" responde enviando el conjunto de paquetes conteniendo la cadena de bits con la configuración. El *FPGA loader* carga dicha cadena en la FPGA usando los *flags* 2 y 3 conectados a los puertos de configuración *DIN* y *CCLK* respectivamente (Paso 3 en el proceso de configuración). Cuando la cadena de bits del último paquete fue cargada en la FPGA, chequea el *Flag* 6 conectado al puerto de configuración *Done* para determinar si la FPGA fue

configurada (Paso 5 en el proceso de configuración). Si la configuración fue exitosa informa a la PC enviando el comando “FPGA OK”, y se queda esperando un comando de reset por software para iniciar el proceso de *booting* del DSP y en consecuencia la programación del DSP que se efectúa con la FPGA ya configurada.

El *FPGA loader* se carga en el DSP sólo a los efectos de configurar la FPGA. La Tabla 7.4 muestra el formato de los paquetes usados para la transmisión de la cadena bits con la configuración de la FPGA. El cuerpo del primer paquete de la secuencia contiene el número total de bits de la cadena de configuración.

<table>
<tr><td colspan="2">H</td><td>O</td><td>L</td><td>A</td></tr>
<tr><td>0</td><td>Configuración FPGA</td><td>Número de Paquete</td><td>Fuente</td><td>Destino</td></tr>
<tr><td colspan="5">Número de bits</td></tr>
<tr><td colspan="5">Cadena de bits conteniendo la configuración de la FPGA</td></tr>
<tr><td colspan="2">C</td><td>H</td><td>A</td><td>U</td></tr>
</table>

Tabla 7.4 Paquete de red para la transmisión de la configuración de la FPGA.

El *FPGA loader* se comunica con la red TDM exactamente del mismo modo que el *loader* del DSP de Bora (Sección 6.10). SPORT DMA es usado tanto para la recepción como para la transmisión de paquetes. Las rutinas de atención a las interrupciones de recepción y transmisión del SPORT son las mismas que las utilizadas por el *loader*.

El *FPGA loader* toma del primer paquete el número de bits y a medida que recibe cada paquete, carga en la FPGA (bit a bit) la cadena de configuración generando también el reloj que acompaña a cada bit. Si durante la configuración el *FPGA loader* detecta algún error, en el paquete o través de la señal *Init* (Paso 4 en el proceso de configuración), envía el mensaje a la PC con su correspondiente código de error y aborta la configuración.

En ambas situaciones, es decir, luego de enviar el comando “FPGA OK” si la configuración fue exitosa o luego de enviar el mensaje de error si se detectó algún error durante la configuración, el *FPGA loader* permanece esperando el comando de reset por software.

7.2.2 Generación de los Paquetes con la Configuración de la FPGA

Para generar los paquetes con la configuración de la FPGA, se lleva a cabo la siguiente secuencia de tareas mostrada esquemáticamente en la Figura 7.6:

Figura 7.6 Proceso de conversión de archivos para la generación de los paquetes con la configuración de la FPGA

1. Desde el programa fuente de la FPGA (.vhd) escrito en VHDL se genera un archivo (.rbt) usando el paquete de diseño *Xilinx Foundation 2.1i* [104]. Tal archivo contiene la cadena de bits con la configuración de la FPGA precedida de un encabezamiento que especifica el nombre del programa que generó el archivo, el tipo de FPGA para la cual se generó la cadena de bits y el número de bits de la cadena.
2. El archivo “.rbt” se convierte en un archivo en formato de paquete (.pkt) usando el utilitario *convert* (Sección 4.6.1).

El archivo generado por el utilitario *convert* incluye el cuerpo de todos los paquetes necesarios para almacenar la configuración de la FPGA. Antes de enviar cada paquete al DSP, el “Controlador del RICH” completa el paquete agregándole las palabras de encabezamiento y trailer. De este modo, no se necesita repetir la

conversión para cada destinación la configuración de la FPGA es la misma para todas o un grupo de FPGA.

7.3 Programación en VHDL

El diseño de la FPGA de Bora fue descripto en el lenguaje de descripción de hardware VHDL siguiendo el estilo sugerido para el sintetizador lógico *Synopsis* [77, 78] que forma parte del paquete de diseño *Xilinx Foundation 2.1i* [104].

El lenguaje VHDL permite describir y simular sistemas digitales complejos en forma natural. En particular permite describir el comportamiento simultáneo de subsistemas digitales y sus interacciones en función del tiempo. Esta tarea es en general difícil de llevar a cabo con los lenguajes secuenciales tradicionales. Una ventaja importante del VHDL es que hoy se cuenta con sintetizadores lógicos que toman una descripción en VHDL y generan una *netlist*[1] de componentes estándar funcionalmente equivalente a la descripción original. Estos componentes estándar pueden ser *macros*[2] o *primitivas*[3] de librerías predefinidas.

En VHDL se comienza por definir una entidad (*Entity*), que es el objeto a describir. Esta definición se hace a través de sus puertos en modo que el objeto pueda verse como una caja negra. Los puertos pueden ser de distintos tipos: *input, output, input-output y buffer*. A cada entidad le corresponde al menos una arquitectura (*architecture*) en la que se define el comportamiento (funcionalidad) del objeto a describir. Básicamente hay dos estilos de descripción: comportamental y estructural

[1] Por *netlist* se entiende una descripción estructural en términos de elementos interconectados. Se especifican los modelos de los componentes y sus puertos, y se especifican las señales usadas para las interconexiones de los componentes.
[2] Un *macro* es un componente obtenido de una configuración específica de ciertos recursos lógicos de la FPGA.
[3] Una *primitiva* es un bloque hardware disponible en la FPGA como los bloques de memoria RAM, DLL, o estructuras de *input-output*.

(*behavioral y structural*), aunque estos estilos no son mutuamente excluyentes y pueden presentarse ambos en una misma descripción. La descripción comportamental recurre a operaciones y estructuras lógicas abstractas como AND, OR, If-then-else, etc. Así se puede modelar objetos complejos describiendo la relación entre *inputs*, *outputs*, señales internas y estados internos. Supongamos por ejemplo que queremos describir el objeto CONJ de tres puertos, dos de entrada (A y B) y uno de salida (Y), que realiza la operación lógica AND, el código correspondiente en VHDL en estilo comportamental podría ser el siguiente:

```
Entity CONJ is Port (A,B: Input, Y: Ouput)

Architecture BEHAVE of CONJ is
Y <= A And B;
End BEHAVE;
```

En la primer línea se define la entidad de nombre CONJ, y se explicitan los nombres y el tipo de sus puertos. Luego se define para la entidad CONJ una posible arquitectura, llamada BEHAVE, que describe el comportamiento de dicha entidad.

El estilo estructural de descripción se basa en los conceptos de interconexión y jerarquía. Se trata de describir entidades complejas en términos de interconexión de componentes más simples. Estos componentes pueden a su vez ser descriptos en función de componentes aun más sencillos, y así siguiendo, formando una estructura jerárquica. Un conjunto de bloques elementales se interconectan para formar macro bloques y estos a su vez se interconectan para formar macro-macro bloques, y así siguiendo. Las características más evidentes de este estilo son las siguientes:

1) Estructuras complejas son representadas por estructuras más simples.
2) Bloques básicos son definidos una sola vez y pueden ser usados un número indefinido de veces.

3) Posibilidad de construir librerías de componentes.

En esta estructura jerárquica los últimos componentes básicos deben tener su descripción comportamental.

7.4 Síntesis e Implementación del Diseño de la FPGA

Para la implementación del diseño de la FPGA de Bora, se procedió según la metodología estándar de diseño *top-down*: especificación, descripción en VHDL, simulación, síntesis, implementación física, y verificación.

Luego de compilar y simular la descripción en VHDL, se utilizó el ambiente de diseño *Foundation 2.1i* de Xilinx [104] para sintetizar e implementar el diseño en la FPGA. Si bien todo el proceso de implementación puede ser completamente automático, difícilmente se pueda alcanzar una alta performance sin una atenta supervisión y control del proceso en sus diversas etapas. En nuestro caso se actuó sobre diversas opciones de síntesis entre las cuales figuran: optimización por velocidad, frecuencia nominal de operación, tipo de codificación para máquinas de estados, preservación de la jerarquía estructural, etc.

La implementación física implica elegir los bloques lógicos configurados y luego interconectarlos; estos dos procesos se denominan usualmente *placing* y *routing* respectivamente. Para el proceso de *placing* se hizo un *floor planning*[4] a través de ciertas restricciones espaciales, como la asignación y prohibición de áreas, para el posicionamiento de distintos bloques estructurales. Para el *routing* se establecieron diversas restricciones temporales sobre ciertas señales críticas como los *clocks*, *enables* y otras señales tipo *read* o *write*. Después de cada implementación se

[4] *Floor planning* es un procedimiento por el cual se optimiza el uso del área disponible para la implementación de un diseño.

estudiaron atentamente las señales más lentas y los caminos críticos, para luego eventualmente hacer un *re-routing* de las señales identificadas como críticas. El *Foundation 2.1i* produce diversos informes temporales (*timing reports*) al final de cada implementación. Si los resultados reportados no eran satisfactorios se procedía a alterar las opciones de síntesis, restricciones temporales y restricciones espaciales en modo de ayudar al software a converger a diseños más optimizados.

Cada diseño se probó directamente sobre la Bora en modo de evaluar directamente su funcionalidad y performance.

Capítulo 8

Discusión y Generalización de los Resultados Obtenidos

En esta tesis hemos descripto un sistema reconfigurable masivamente paralelo de adquisición y procesamiento de datos de tiempo real duro, basado en Procesadores de Señales Digitales (DSP) conectados en red y combinados con dispositivos de lógica programable de tipo FPGA.

Los DSP son procesadores optimizados para aplicaciones de tiempo real y cálculo aritmético intensivo. Una programación adecuada del DSP permite explotar la potencialidad de tales procesadores, conectarlos en red y combinarlos con dispositivos FPGA.

El uso de una FPGA como coprocesador permite al DSP de asignar a la FPGA las tareas con restricciones temporales duras que requieren un alto grado de procesamiento paralelo. La combinación DSP-FPGA es adecuada para implementar sistemas reconfigurables de tiempo real duros, conjugando el paralelismo del hardware con la flexibilidad del software.

En un sistema masivamente paralelo de adquisición de datos, una red de DSP es adecuada para la sincronización y el control de los subsistemas adquiriendo y procesando en paralelo un número del total de canales del sistema. La conexión entre la red de DSP y una PC, permite la programación, configuración, control y monitoreo permanente del sistema desde una aplicación de software de alto nivel.

De los resultados obtenidos por el sistema de adquisición y procesamiento de datos desarrollado y descripto en esta tesis, podemos generalizar cuatro conclusiones importantes que describimos a continuación.

a) La potencialidad de un DSP puede ser expandida usando una FPGA como coprocesador.

Un DSP es un procesador optimizado para el cálculo aritmético intensivo frecuentemente requerido para el procesamiento continuo de señales digitales [53, 55, 57, 58, 74]. Entre las aplicaciones más frecuentes se encuentran multiplicaciones matriciales, filtros digitales, y transformaciones rápidas de Fourier. Siendo que estas aplicaciones requieren en general que sean efectuadas en tiempo real es necesario ser capaz de procesar los datos a un ritmo mayor o igual al ritmo de producción de estos datos, y así un DSP típico tendrá intrínsecamente un cierto paralelismo interno.

Las aplicaciones mencionadas incluyen normalmente un gran número de multiplicaciones, adiciones, y accesos a memoria (interna o externa) por unidad de tiempo. Los DSP modernos cuentan con al menos una unidad de multiplicación y acumulación (MAC) de único ciclo, y con otras unidades independientes implementadas en hardware que ejecutan en paralelo (ALU y *shifter*, buffer circular, controlador DMA, *zero-overhead-looping*, etc.).

Una operación MAC, por ejemplo, requeriría varias instrucciones en un procesador tradicional y en consecuencia varios ciclos de reloj. No debería sorprender entonces que ciertos DSP provean un número mayor de instrucciones equivalentes por segundo que un procesador tradicional que trabaja con un reloj más rápido. Para lograr estas características los DSP se basan en una arquitectura interna denominada Harvard que, en contraste con la clásica arquitectura Von Neumann

típica de los procesadores tradicionales de propósito general, cuenta con zonas de memoria separadas para almacenar datos e instrucciones y cada zona posee su propio bus de datos y de direcciones.

Si bien los DSP y los procesadores tradicionales de última generación cuentan con arquitecturas complejas y sofisticadas siguen siendo esencialmente diferentes. Los procesadores tradicionales son optimizados para implementar sistemas operativos multitarea y multiusuario, y los DSP son optimizados para implementaciones de tiempo real y cálculo aritmético intensivo. Todavía los DSP comparten con los procesadores tradicionales una característica fundamental: el hardware es fijo y optimizado para la ejecución esencialmente secuencial de instrucciones, independientemente de la aplicación específica del software. El procesador lee cada instrucción desde la memoria, decodifica su significado, y luego la ejecuta; resultando en una alta sobrecarga de ejecución para cada operación individual. Un costo adicional se incorpora en el caso de implementar alguna instrucción que no sea parte del conjunto de instrucciones propias del procesador.

En contraste a dicha característica se disponen en la actualidad de dispositivos de lógica programable sofisticados como las FPGA que permiten implementar soluciones hardware optimizadas para cada aplicación específica [61, 63, 68, 73, 74,]. Estos dispositivos consisten en un gran numero de bloques lógicos reconfigurables e interconectables. Una memoria especial de configuración almacena la información necesaria para la configuración e interconexión de los bloques lógicos. Así basta ingresar esta información en la FPGA para transformarla en un objeto hardware capaz de realizar tareas específicas. Cambiando la configuración el objeto se transforma en otro capaz de realizar otras tareas y comportarse diversamente. Si bien las FPGA existen en el mercado desde hace ya muchos años, es solo recientemente que estos

dispositivos han crecido en complejidad y sofisticación dando un salto cualitativo en cuanto a sus potencialidades y posibles aplicaciones.

Disponiendo de un hardware así flexible y versátil es fácil intuir sus ventajas. Por ejemplo se puede explotar el paralelismo intrínseco del hardware, en oposición a la secuencialidad del software, para obtener respuestas rápidas en tiempo real y una alta performance en la ejecución de ciertos algoritmos. También, la reconfigurabilidad de estos dispositivos permite la adaptación a cambios en las condiciones o exigencias de trabajo, sin necesidad de sustituir el hardware.

Otra característica importante de la nueva generación de FPGA es la presencia de recursos de hardware específicos como DLL, memorias *dual-port* RAM y bloques aritméticos optimizados. Los DLL son bloques que aparte de su función esencial de ajustar una fase de reloj para compensar un retardo intrínseco, proveen diversas señales de reloj obtenidas a partir de un reloj inicial (multiplicación y división de frecuencia). Estos DLL se pueden combinar entre si para obtener relojes internos mas rápidos y sofisticados. Los bloques de memoria son reconfigurables en distintas relaciones de ancho y profundidad, y pueden usarse para implementar memorias de tipo FIFO. Estas memorias embebidas en la FPGA no recurren a los bloques lógicos genéricos como sucedía con las FPGA precedentes, las cuales emulaban una memoria RAM usando los flip-flop de los bloques lógicos reconfigurables disminuyendo la cantidad total de recursos lógicos disponibles para implementar el resto del diseño. Los bloques lógicos reconfigurables están en general optimizados para la generación y propagación del bit de acarreo que se usa en la mayoría de las funciones aritméticas binarias y que frecuentemente determina la máxima frecuencia de operación.

Actualmente se disponen de FPGA con millones de compuertas lógicas equivalentes y varios Mbits de memoria RAM. Así las potencialidades de estos dispositivos son enormes. La limitación en explotar estas potencialidades proviene en general de la capacidad humana de diseño y de la sofisticación de las herramientas de software de diseño CAD (*Computer Assisted Design*).

Para hacer una comparación más específica entre DSP y FPGA, supongamos que tenemos que hacer un gran número de multiplicaciones por segundo. Un DSP generalmente dispone de un multiplicador optimizado que puede realizar un gran número de multiplicaciones al segundo, pero las realiza secuencialmente y así su limitación en performance depende de su frecuencia de reloj. En una FPGA se puede implementar un multiplicador configurando e interconectando algunos de sus bloques lógicos reconfigurables. En general este multiplicador no alcanzará la performance de aquel del DSP, pero si la FPGA es suficientemente grande podría replicarse este multiplicador muchas veces, digamos 100, y así la FPGA podría ejecutar 100 multiplicaciones por ciclo de reloj confrontándose seriamente con el DSP.

Todavía resta evaluar la facilidad de implementación, ya que para los DSP se disponen de compiladores que permiten la programación en un lenguaje de alto nivel, mientras que en una FPGA se requiere de la descripción del hardware, la que en general implica más tiempo y su implementación final no es trivial. En ambos casos, cuando se requieran performances extremas cercanas a los límites teóricos de los dispositivos, hay que aumentar el esfuerzo de programación, ya sea programando en *assembly* el DSP o haciendo descripciones de hardware más detalladas para una FPGA. Si bien ha habido un gran progreso en las herramientas CAD para las FPGA todavía no se ha logrado la situación óptima en la que se aprieta una tecla y una descripción hardware a alto nivel viene implementada automáticamente con

resultados óptimos. Normalmente se espera que el diseñador supervise toda la cadena de diseño, desde la síntesis lógica hasta la implementación física final, proveyendo diversas restricciones y parámetros para ayudar al CAD a converger hacia diseños optimizados que puedan alcanzar las especificaciones requeridas.

La combinación DSP-FPGA permite explotar las características complementarias del DSP y de la FPGA obteniendo una mayor performance que el software y manteniendo una mayor flexibilidad que el hardware. Esta es la característica clave de la denominada Computación Reconfigurable [70, 74, 75]. En este contexto, el término reconfigurabilidad indica que la funcionalidad lógica e interconexión de un sistema de computación puede adecuarse a aplicaciones específicas a través de una programación definida por el usuario. En investigaciones corrientes, el término computación reconfigurable se refiere a sistemas incorporando alguna forma de programabilidad del hardware [73].

Las exigencias de alta performance, respuesta en tiempo real y flexibilidad del sistema de adquisición y procesamiento de datos descripto en esta tesis, fueron cumplidas usando DSPs combinados con FPGAs. El protocolo y la arquitectura de comunicación entre el DSP y la FPGA fueron explicados en los capítulos 5, 6 y 7. El DSP es el cerebro del sistema, su función es la de ejecutar, supervisar y controlar cada una de las actividades efectuadas actuando conjuntamente con la FPGA que funciona como coprocesador. La FPGA principalmente desarrolla las tareas con restricciones temporales duras que requieren un alto grado de procesamiento paralelo. En particular, durante el *spill*, se exigía una rápida velocidad de respuesta al *trigger* en modo de minimizar el tiempo muerto (del orden del microsegundo), y al mismo tiempo se exigía un tiempo de servicio por evento en el orden de los diez microsegundos. Considerando que el servicio de cada evento consiste en la lectura de

memoria externa de 432 canales (digitalizados en 10 bits), la sustracción de umbral por canal, la incorporación del identificador del canal, y la elaboración y transmisión del paquete de evento; tales restricciones temporales solo podían ser satisfechas por la FPGA.

b) Una red determinística de DSP permite la sincronización y el control de un sistema masivamente paralelo de adquisición y procesamiento de datos.

Cuando un gran número de canales deben ser adquiridos y procesados en paralelo, la única opción es distribuir el sistema de adquisición en varios subsistemas más pequeños. Surge así la necesidad de sincronizar y controlar tales subsistemas para asegurar su funcionamiento en paralelo. A tal efecto, es necesaria algún tipo de red que permita la distribución y el intercambio de información. Un tipo elemental de información podría ser, por ejemplo, comenzar o terminar una secuencia de adquisición, y en este caso sería necesario comunicar esta información tempestivamente dentro de ciertos límites temporales estrictos en modo de no perder la sincronización entre subsistemas.

Los DSP modernos están predispuestos para su conexión en redes seriales soportando distintos protocolos (TDM, I^2S, RS232, SPI, etc.). Una parte de su hardware está destinado a la gestión de esta red en paralelo con el *core* del procesador. El *core* configura sus puertos seriales para soportar un cierto protocolo y la comunicación se establece a través de interrupciones internas. En general se dispone también de controladores DMA los cuales transfieren bloques de datos entre el puerto serial y la memoria interna, liberando de este modo al *core* de las tareas de *input-output* correspondientes.

La red *Time Division Multiplexed* (TDM) de DSP, explicada principalmente en el capítulo 3, tiene las siguientes ventajas respecto a otras redes: es determinística, permite la conexión de un número considerable de DSP, y es relativamente fácil de implementar y gestionar. Por red determinística entendemos una red en la que es posible prever con exactitud los tiempos de transmisión de cualquier paquete de datos independientemente de las condiciones de tráfico, característica que las hace adecuada para su uso en sistemas de tiempo real.

El DSP usado en el sistema descripto en esta tesis permite la conexión de hasta 32 DSP[1] en modo TDM *full duplex*. La facilidad de implementación se debe al hardware que permite configurar la red, a través de registros de control específicos, estableciendo, la velocidad de transmisión, el número total de *slots*, la longitud en bits de cada *slot*, y por cada procesador el modo de operación de cada *slot* (transmisión, recepción y transmisión-recepción). La facilidad de gestión se debe a ciertas propiedades que automatizan por hardware muchas de las tareas que de otra manera empeñarían el *core* disminuyendo su potencial efectivo de cálculo para otras tareas. Entre tales propiedades se encuentra la posibilidad de habilitar y deshabilitar dinámicamente sean los *slots* de transmisión o recepción como la palabra clave en los *slots* de recepción, interrupciones internas específicas que comunican el fin de una transmisión o recepción, y controladores DMA que habilitan al procesador a continuar con la ejecución del programa durante la transferencia de un bloque entero de datos. Estas propiedades facilitan la implementación de la capa de acceso en nuestro modelo de red basado en tres capas, el cual fue descripto en el capítulo 3.

[1] Al momento del desarrollo del sistema, el DSP usado era el más potente de la familia de DSP de punto flotante de 32 bits. Actualmente existen DSP diez veces más veloces y que permiten la conexión en modo TDM de hasta 128 DSP. Esto da una idea de la velocidad en la que evolucionan tales procesadores.

El sistema de adquisición y procesamiento de datos descripto en esta tesis está conformado por 192 subsistemas paralelos, cada uno adquiriendo y procesando 432 canales. Las señales de sincronización para la adquisición de datos, comunes a todo el experimento, se distribuyen a los subsistemas a través de ocho redes TDM de DSP. A tal efecto, como fue explicado en el capítulo 4, se implementa *broadcasting* cumpliendo de este modo con las restricciones temporales duras asociadas con la transmisión de las señales de sincronización. Las redes de DSP permiten también controlar el funcionamiento de cada uno de los subsistemas y si los mismos están sincronizados.

Para el intercambio de información a través de la red de DSP, se implementó en software el protocolo y la arquitectura de comunicación de la red. Se definieron paquetes de red específicos para la transmisión de comandos, mensajes, datos y programas. La arquitectura de comunicación se basa en tres capas jerárquicas (capa de acceso, capa de transporte y capa de aplicación) según las cuales las capas superiores son independientes del tipo de red usado. Del mismo modo, el formato de los datos que se intercambian es independiente de las aplicaciones que los generan. El protocolo y la arquitectura de comunicación de la red de DSP fueron explicados en los capítulos 3, 4 y 6.

c) Una programación adecuada permite explotar la potencialidad del DSP, y al mismo tiempo gestionar eficientemente la red TDM y un coprocesador implementado en una FPGA.

Aunque un DSP pueda contar con una FPGA como coprocesador específico y gestionar fácilmente su conexión a una red TDM, todavía resta el problema que el programa que se ejecuta en el DSP debe garantizar la realización global de todas las

tareas asignadas respetando severas restricciones temporales. Un sistema de tiempo real de alta performance basado en el uso intensivo del DSP es en cierta medida un sistema complejo y por lo tanto es esencial optimizar la programación. Si bien existen compiladores evolucionados que permiten la programación de los DSP en un lenguaje de alto nivel (típicamente C), en general el código ejecutable obtenido difícilmente explotará eficientemente todos los recursos del procesador. Para obtener la máxima performance es importante conocer en detalle la arquitectura interna del DSP y su funcionamiento.

El conjunto de instrucciones de los DSP ha sido diseñado con el objetivo de explotar al máximo el hardware subyacente, incrementando así su eficiencia. Para alcanzar este objetivo, generalmente se permite especificar al programador varias operaciones paralelas en una única instrucción, típicamente incluyendo uno o dos accesos a memoria (con actualización de punteros) en paralelo con la principal operación aritmética. Así, hacer un uso eficiente del conjunto de instrucciones del DSP, requiere un esfuerzo particular de programación en el lenguaje *assembly* del procesador. La arquitectura típica de los DSP, con sus múltiples espacios de memoria, múltiples buses, conjunto de instrucciones irregulares, y unidades de hardware altamente especializadas, es difícil de usar eficientemente por los compiladores [53]. Otra limitación de los compiladores es el tiempo de servicio de las interrupciones. Si bien en los DSP el tiempo mínimo de respuesta a una interrupción se encuentra en el orden de los nanosegundos, usando compiladores este tiempo se incrementa al orden de los microsegundos. Este incremento se debe al tiempo usado por el compilador para salvar y recuperar la información de contexto.

Por lo tanto, para explotar eficientemente la potencialidad de un DSP en aplicaciones con exigencias severas de performance y respuesta en tiempo real,

optimizar la programación se vuelve un requisito esencial. Programando en *assembly* el programador pone bajo su total control todos los recursos disponibles del hardware y conoce los tiempos exactos de ejecución del programa en sus distintas partes.

En nuestro caso, tal como se explicó principalmente en el capítulo 6, el DSP realiza una gran variedad de servicios respondiendo en tiempo real a seis interrupciones y gestionando la comunicación con la red TDM, la FPGA, y otros dispositivos tales como ADC, DAC y sensores de temperatura. El software del DSP fue escrito en lenguaje *assembly*. La estructura general del software se basa en dos niveles de ejecución: nivel de interrupciones (*foreground*), en el cual las ISR ejecutan las operaciones críticas con restricciones temporales y especifican a través de *flags* internos las tareas a realizar, y nivel de tareas (*background*) en el cual se ejecutan las tareas que fueron requeridas a través de alguna de las interrupciones.

Para garantizar un tiempo mínimo de servicio de las interrupciones se utilizaron los registros de propósito general del DSP con el fin de minimizar el tiempo requerido para salvar la información de contexto en el *stack* de ejecución. Cada una de las ISR usa, en manera exclusiva, un grupo de tales registros evitando de este modo que los mismos sean salvados al inicio del servicio de la interrupción, y en consecuencia recuperados al final del servicio. Para el servicio de los comandos de sincronización para la adquisición de datos, los cuales son recibidos a través de la red e involucran restricciones temporales duras, se generaron interrupciones internas de software. Para evitar la pérdida de *triggers* se utilizó un buffer circular implementado en hardware. Siempre que fue posible se utilizaron los controladores DMA para la comunicación con la red, y se aplicaron las operaciones aritméticas paralelas durante la caracterización estadística de los canales del detector y el cálculo de los umbrales. De este modo, conociendo la arquitectura subyacente del DSP y los tiempos exactos

de ejecución en las distintas partes del programa, se explotó eficientemente la potencialidad del DSP alcanzando la alta performance requerida por el sistema de adquisición y procesamiento de datos descripto en esta tesis.

d) Una tarjeta PCI MultiDSP provee la gestión eficiente de un número de redes de DSP y la comunicación de tales redes con una PC.

La conexión entre una red de DSP y una PC permite la programación, configuración, control y monitoreo de un sistema masivamente paralelo de adquisición y procesamiento de datos, como el descripto en esta tesis, desde una aplicación de software de alto nivel.

Un estándar difuso en las PC es el bus PCI [97] que permite expandir la PC con dispositivos periféricos. La tarjeta MultiDSP (Dolina), descripta en el capítulo 4, provee la comunicación entre ocho redes de DSP y una PC a través del bus PCI. Un dispositivo *driver*, específicamente escrito para Dolina, gestiona la comunicación entre la tarjeta y una aplicación de software de alto nivel.

Dolina contiene 8 DSP cada uno de los cuales gestiona una red de 24 DSP idénticos. Las principales funciones de esta tarjeta son la administración de las redes y la comunicación entre la PC y los 192 subsistemas de adquisición y procesamiento de datos (Bora) que conforman el sistema descripto en esta tesis. Dolina también recibe dos señales críticas de sincronización directamente del TCS, las cuales retransmite inmediatamente a todas las Bora sin mediación del sistema operativo del PC que en general no es un sistema operativo de tiempo real.

El intercambio de información entre la PC y las redes de DSP se efectúa usando paquetes de red específicos para la transmisión de comandos, mensajes, datos y programas. La información contenida en los paquetes de red incluye la fuente y el

destino del paquete, y el tipo de información transmitida. El protocolo y la arquitectura de comunicación entre la tarjeta MultiDSP Dolina y la PC fueron explicados en los capítulos 3 y 4.

La conexión entre Dolina y la PC permite el control y monitoreo de los 192 subsistemas de adquisición y procesamiento de datos desde una aplicación de software de alto nivel, como así también permite la reprogramación de los DSP y la reconfiguración de las FPGA en cualquier momento que sea requerido por un operador. Esta propiedad garantiza la flexibilidad del sistema en cuanto a que el mismo puede ser reprogramado o reconfigurado desde una aplicación de software de alto nivel adaptándose a nuevos requerimientos o condiciones experimentales.

Vale la pena notar que esta tarjeta en sí misma, o junto a los 192 DSP que conforman las redes, constituye un sistema capaz de expandir, en línea de principio, la capacidad de cálculo de la PC. Ciertas tareas de cálculo extensivo podrían ser ejecutadas en los DSP explotando el paralelismo masivo provisto por los 200 procesadores. En cierto modo, esto es lo que ocurre cuando se efectúa la caracterización estadística de los 82944 canales del sistema descripto en esta tesis. Claramente en nuestro caso no se pretendió prioritariamente la expansión de la capacidad de cálculo sino resolver un problema específico, sin embargo gran parte del hardware y software desarrollado demuestran que esta posibilidad es viable y podría ser estudiada ulteriormente.

Conclusiones

En esta tesis hemos demostrado, en modo experimental, que métodos de programación de tiempo real aplicados a una arquitectura basada en Procesadores de Señales Digitales (DSP) conectados en red y combinados con dispositivos de lógica programable FPGA como coprocesadores, constituyen una estrategia válida y eficiente para implementar sistemas de adquisición y procesamiento masivo de datos con exigencias severas de performance, flexibilidad y respuesta en tiempo real.

Tales eran las exigencias del sistema de adquisición del detector RICH del experimento COMPASS que se desarrolla al CERN. COMPASS es el experimento adquiriendo, desde el año 2002, la mayor cantidad de eventos por unidad de tiempo comparado con otros experimentos de física de las altas energías, y su funcionamiento está previsto hasta el año 2010. La complejidad del RICH de COMPASS, el elevado número y distribución física de los canales del detector y el alto *trigger rate* con mínimo tiempo muerto, definieron al sistema de adquisición y elaboración de datos del detector como un problema único para el cual fue necesario el desarrollo de una solución nueva y original.

El RICH cuenta con 82944 canales, distribuidos en una superficie de 5,3 m^2, los cuales son adquiridos y digitalizados en 10 bits a un *trigger rate* que puede alcanzar los 100 kHz con un tiempo muerto de aproximadamente 1 µs. Considerando que solo un porcentaje de los canales adquiridos por cada *trigger* contienen

información significativa, se aplica sobre los mismos una elaboración en tiempo real en modo de seleccionar los canales significativos para ser enviados al sistema de adquisición global (DAQ) del experimento. A tal efecto, se compara cada uno de los valores digitalizados con un umbral programable por software y sólo los valores que superan el umbral con su correspondiente identificador de canal son empaquetados y enviados al sistema DAQ. Los datos de evento son transmitidos al sistema DAQ a través de 192 fibras ópticas.

Los requerimientos de paralelismo masivo, alta performance, respuesta en tiempo real y flexibilidad del sistema de adquisición y elaboración de datos del RICH de COMPASS, fueron cumplidos por un sistema reconfigurable de adquisición y elaboración de datos de tiempo real duro, basado en redes de DSP con FPGA como coprocesadores. La siguiente tabla resume la performance obtenida por el sistema.

Número de canales:	82944 canales
Canal adquirido:	1,25 bytes
Canal transmitido:	4 bytes (header + trailer: 16 bytes)
Media de canales transmitidos:	20 %
Tiempo muerto de adquisición:	1,2 µs
Tiempo de servicio de evento:	12,7 µs
Trigger rate (medio):	78 kHz (1/12,7 µs)
Frecuencia media de datos adquiridos:	~ 8 GBytes/s
Frecuencia media de datos transmitidos:	~ 5 GBytes/s
Trigger rate (pico):	833 kHz (1/1,2 µs)
Frecuencia pico de datos adquiridos:	~ 86 GBytes/s
Frecuencia pico de datos transmitidos:	~ 7,7 GBytes/s

El tiempo muerto de adquisición es el tiempo requerido para adquirir, digitalizar y almacenar, en memorias FIFO, los 82944 canales del detector. El tiempo de servicio de evento es el tiempo empleado para elaborar, empaquetar y transmitir la

información de los canales (valor e identificador) que superan su correspondiente umbral. El sistema puede alcanzar frecuencias pico de *triggers* de hasta 833 kHz, dependiendo del estado de las memorias FIFO que cuentan con la capacidad para almacenar 128 eventos. La frecuencia pico de datos transmitidos es determinada por la velocidad efectiva de transmisión de las fibras ópticas (40 MBytes/s).

La alta performance del sistema es lograda a través de 192 subsistemas de adquisición y elaboración de datos operando en paralelo. El cerebro de cada subsistema es un DSP que cuenta con una FPGA como coprocesador.

Los DSP son procesadores optimizados para aplicaciones de tiempo real y cálculo aritmético intensivo. Una programación adecuada del DSP permite explotar la potencialidad de tales procesadores, conectarlos en red y combinarlos con dispositivos FPGA. El uso de una FPGA como coprocesador consiente al DSP de asignar a la FPGA las tareas con restricciones temporales duras que requieren un alto grado de procesamiento paralelo.

La combinación DSP-FPGA permite explotar los aspectos positivos de cada dispositivo compensando recíprocamente sus aspectos negativos. El objetivo de la combinación es obtener una mayor performance que el software y una mayor flexibilidad que el hardware, disminuyendo de este modo la brecha existente entre el hardware y el software. Esta es la característica clave de la denominada *Computación Reconfigurable*, la cual está adquiriendo un rol cada vez más importante en el área de investigación de arquitecturas de computadora y sistemas de software. Asignado al hardware reconfigurable las porciones de una aplicación que requieren computación o paralelismo intensivo, la aplicación puede ser acelerada notablemente. Así, se cuenta con el potencial para aumentar la performance explotando el paralelismo del hardware sin perder la flexibilidad del software. Los sistemas de computación reconfigurables

han mostrado la habilidad para acelerar notablemente la ejecución de programas, proveyendo una alternativa de alta performance a las implementaciones de software.

Esta estrategia satisface también la flexibilidad necesaria en un sistema que debe funcionar durante mucho tiempo (más de 10 años). La reprogramabilidad del DSP y la reconfigurabilidad de la FPGA permiten la adaptación a nuevas condiciones experimentales que puedan surgir durante el tiempo de vida útil del sistema, sin necesidad de sustituir el hardware. Además de la adquisición, elaboración y transmisión de eventos, la gran capacidad distribuida de cálculo y la versatilidad alcanzadas con esta estrategia permite implementar otras funciones complejas como la medición y monitoreo de parámetros físicos del sistema, y la caracterización estadística, el cálculo y actualización de los valores de umbrales y el autotesteo y calibración de cada uno de los canales del detector.

En un sistema masivamente paralelo de adquisición de datos, una red de DSP es adecuada para la sincronización y el control de los subsistemas adquiriendo y procesando en paralelo un número del total de canales del sistema.

Los DSP modernos están capacitados físicamente para formar parte de una red *Time Division Multiplexed* (TDM) y cuentan con la memoria interna para implementar en software el protocolo y la arquitectura de comunicación de la red. La red TDM tiene la ventaja de ser determinística, permite la conexión de un número considerable de DSP, y es relativamente fácil de implementar y gestionar. Por red determinística entendemos una red en la que es posible prever con exactitud los tiempos de transmisión de cualquier paquete de datos independientemente de las condiciones de tráfico, característica que las hace adecuada para su uso en sistemas de tiempo real.

La facilidad de implementación y de gestión de la red, se debe a una parte del hardware del DSP destinada a la administración de la red en paralelo con el *core* del procesador. De este modo, la comunicación entre la red y el *core* se establece a través de interrupciones internas, y es posible hacer uso de controladores de DMA paralelos que transfieren bloques de datos entre la red y la memoria interna del procesador. Así, se libera al *core* de las actividades de *input-output* correspondientes dejando su potencial efectivo de cálculo para otras tareas.

Se implementó en software el protocolo y la arquitectura de comunicación de la red. Se definieron paquetes de red específicos para la transmisión de comandos, mensajes, datos y programas. La arquitectura de comunicación se basa en tres capas jerárquicas (capa de acceso, capa de transporte y capa de aplicación) según las cuales las capas superiores son independientes del tipo de red usado. Del mismo modo, el formato de los datos que se intercambian es independiente de las aplicaciones que los generan. Para la transmisión de las señales de sincronización se implementó *broadcasting* cumpliendo así con las restricciones temporales duras asociadas con tales señales.

En un sistema de tiempo real de alta performance, explotar la potencialidad del DSP y al mismo tiempo gestionar eficientemente la red TDM y un coprocesador implementado en una FPGA, implica optimizar la programación del DSP. Para obtener la máxima performance es importante conocer en detalle la arquitectura interna del DSP y su funcionamiento. En general, los compiladores de lenguajes de alto nivel (típicamente C) no hacen un uso eficiente de la arquitectura subyacente del DSP incrementando además el tiempo de servicio de las interrupciones. Programando en el lenguaje *assembly* del procesador, el programador pone bajo su total control

todos los recursos disponibles del hardware y conoce los tiempos exactos de ejecución del programa en sus distintas partes.

El software de los DSP fue escrito en lenguaje *assembly*. La estructura general del software se basa en dos niveles de ejecución: nivel de interrupciones (*foreground*), en el cual las ISR ejecutan las operaciones críticas con restricciones temporales y especifican a través de *flags* internos las tareas a realizar, y nivel de tareas (*background*) en el cual se ejecutan las tareas que fueron requeridas a través de alguna de las interrupciones.

La conexión entre la red de DSP y una PC, permite la programación, configuración, control y monitoreo de los 192 subsistemas paralelos de adquisición y elaboración de datos desde una aplicación de software de alto nivel; como así también la reprogramación de los DSP y la reconfiguración de las FPGA en cualquier momento que sea requerido. Esta propiedad garantiza la flexibilidad del sistema en cuanto a que el mismo puede ser reprogramado o reconfigurado adaptándose de este modo a nuevos requerimientos o condiciones experimentales.

Una tarjeta PCI MultiDSP (Dolina) provee la comunicación entre ocho redes de DSP y una PC. Dolina contiene 8 DSP cada uno de los cuales gestiona una red de 24 DSP idénticos. Las principales funciones de Dolina son la administración de las redes y la comunicación entre la PC y los 192 subsistemas de adquisición y elaboración de datos (Bora). Dolina también recibe directamente las señales críticas de sincronización, las cuales retransmite inmediatamente a todas las Bora sin mediación del sistema operativo del PC que en general no es un sistema operativo de tiempo real.

La tarjeta MultiDSP Dolina en sí misma, o junto a los 192 DSP que conforman las redes, constituye un sistema capaz de expandir, en línea de principio, la

capacidad de cálculo de la PC. Ciertas tareas de cálculo extensivo podrían ser ejecutadas en los DSP explotando el paralelismo masivo provisto por los 200 procesadores. En cierto modo, esto es lo que ocurre cuando se efectúa la caracterización estadística de los 82944 canales del detector RICH. Si bien en nuestro caso no se pretendió prioritariamente la expansión de la capacidad de cálculo, gran parte del hardware y software desarrollado demuestran que esta posibilidad es viable y podría ser estudiada ulteriormente.

La solución implementada para el sistema de adquisición y elaboración de datos del detector RICH de COMPASS fue posible gracias al grado de sofisticación alcanzado por los DSP y las FPGA al momento de la concepción y diseño del entero sistema. El DSP elegido es una microcomputadora que cuenta con una arquitectura interna compleja que permite la ejecución en paralelo de diversas tareas sin intervención del *core*. En el caso de las FPGA, si bien existen en el mercado desde hace ya muchos años, es solo recientemente que estos dispositivos han crecido en complejidad dando un salto cualitativo en cuanto a sus potencialidades y posibles aplicaciones. Con respecto a la FPGA utilizada, además de disponer de una cantidad considerable de bloques lógicos reconfigurables, cuenta con recursos de hardware específicos como DLL, memorias *dual-port* y bloques aritméticos optimizados.

Así como ocurrió con el sistema de adquisición y elaboración de datos del RICH de COMPASS al CERN, la nueva generación de experimentos de física de las altas energías desafían constantemente el “estado del arte” en ciencias de la computación, requiriendo soluciones tecnológicas originales e innovativas. En muchos casos, estas soluciones y sus consecuentes aplicaciones serán después incorporadas en otros campos de la ciencia y de la industria. A su vez, la complejidad exponencialmente creciente de los dispositivos reprogramables y reconfigurables

requiere el desarrollo cada vez más sofisticado de software que permita explotar dicha complejidad, y en consecuencia la intervención de expertos en ciencias de la computación.

Por último, queremos destacar que con esta tesis hemos resuelto un problema único cuyas características y exigencias requirieron el desarrollo de una solución nueva y original que provea paralelismo masivo, alta performance, respuesta en tiempo real y flexibilidad. En este sentido, considerando que el sistema fue implementado, probado y esta siendo usado desde el año 2002, se espera que las aproximaciones adoptadas puedan servir como modelo para resolver problemáticas similares en el área de ciencias de la computación y en otras ramas de la ciencia.

Glosario

ADC	**A**nalog to **D**igital **C**onverter
ATM	**A**synchronous **T**ransfer **M**ode
CAD	**C**omputer **A**ssisted **D**esign
CASTOR	**C**ERN **A**dvanced **Stor**age Manager
CATCH	**C**ompass **A**ccumulate **T**ransfer and **C**ontrol **H**ardware
CCF	**C**ompass **C**omputing **F**arm
CDR	**C**entral **D**ata **R**ecord
CERN	**E**uropean Organization for **N**uclear **R**esearch
CLB	**C**onfigurable **L**ogic **B**lock
CMC	**C**ommon **M**ezzanine **C**ard
COMPASS	**Co**mmon **M**uon and **P**roton **A**pparatus for **S**tructure and **S**pectroscopy
CORAL	**CO**MPASS **R**econstruction and **A**na**l**ysis Software
CPLD	**C**omplex **P**rogrammable **L**ogic **D**evice
CPU	**C**entral **P**rocessing **U**nit
DAC	**D**igital to **A**nalog **C**onverter
DAQ:	**D**ata **Ac**q**u**isition
DATE	**D**ata **A**cquisition **T**est **E**nvironment
DLL	**D**elay **L**ock **L**oop
DMA	**D**irect **M**emory **A**ccess
DPM	**D**ual **P**ort **M**emory
DR	**D**ata **R**eceive
DSP	**D**igital **S**ignal **P**rocessor
DT	**D**ata **T**ransmit
EOR	**E**nd **o**f **R**un

EOS	**E**nd **o**f **S**pill
EPROM	**E**rasable **P**rogrammable **R**ead-**O**nly **M**emory
FIFO	**F**irst **I**n **F**irst **O**ut memory
FLT	**F**irst **L**evel **T**rigger
FPGA	**F**ield **P**rogrammable **G**ate **A**rray
ISR	**I**nterrupt **S**ervice **R**outine
LSB	**L**east **S**ignificant **B**it
MSB	**M**ost **S**ignificant **B**it
PC	**P**ersonal **C**omputer
PCB	**P**rinted **C**ircuit **B**oard
PCI	**P**eripheral **C**omponent **I**nterconnect
PS	**P**roton **S**ynchrotron
RAM	**R**andom **A**ccess **M**emory
RCLK	**R**eference **Cl**oc**k**
RFS	**R**eference **F**rame **S**ync
RICH	**R**ing **I**maging **Ch**erenkov
ROB	**R**ead**o**ut **B**uffer
SHARC	**S**uper **H**arvard **Arc**hitecture
SOR	**S**tart **o**f **R**un
SOS	**S**tart **o**f **S**pill
SPI	**S**erial **P**ort **I**nterface
SPORT	**S**erial **Port**
SPS	**S**uper **P**roton **S**ynchrotron
TCP/IP	**T**ransmission **C**ontrol **P**rotocol/**I**nternet **P**rotocol
TCS	**T**rigger **C**ontrol **S**ystem
TDC	**T**ime to **D**igital **C**onverter
TDM	**T**ime **D**ivision **M**ultiplexed
VME	**V**ersa **M**odule **E**urocard

Bibliografía

[1] COMPASS Collaboration (G. Baum et al.). *COMPASS: A Proposal for a Common Muon-Proton Apparatus for Structure and Spectroscopy*. CERN-SPSLC-96-14, Mar 1996.

[2] Bradamante, F. *The COMPASS Experiment at CERN*. Nuclear Physics A, Volume: 622, Issue: 1-2, August 25, 1997, pp. 50c-65c.

[3] Schmitt, L. *Physics with Hadron Beams in the COMPASS Experiment*. Progress in Particle and Nuclear Physics, Volume: 44, Issue: 1, 2000, pp. 365-367.

[4] Bradamante, F. *The Gluon Contribution to the Nucleon Spin and the COMPASS Experiment at CERN*. Progress in Particle and Nuclear Physics, Volume: 44, Issue: 1, 2000, pp. 339-359.

[5] Pagano, P. *Preliminary Measurement of Transversity at COMPASS experiment*. PhD Thesis, Trieste (Italy), 2003.

[6] Mallot, G. *The COMPASS spectrometer at CERN*. Nuclear Instruments and Methods in Physics Research, Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, Volume: 518, Issue: 1-2, February 1, 2004, pp. 121-124.

[7] Konrad Kleinknecht. *Detectors for particle radiation*. Second Edition, New York: Cambridge University Press, 1998.

[8] Dan Green. *The Physics of Particle Detectors*. New York: Cambridge University Press, 2000.

[9] Schmidt, T. *A Common Readout for the COMPASS Experiment*. PhD Thesis, Freeburg, 2002.

[10] Baum et al. *The COMPASS RICH 1 Detector*. Nuclear Physics, Section B 78, 1999, pp. 354.

[11] Baum et al. *The COMPASS RICH Project*. Nuclear Instruments & Methods in Physics Research (NIM-A), Section A 433, 1999, pp. 207.

[12] Albrecht E. et al. *COMPASS RICH-1*. Nuclear Instruments & Methods in Physics Research (NIM-A), Section A 478, 2002, pp. 340-343.

[13] Albrecht E. et al. *COMPASS RICH-1*. Nuclear Instruments & Methods in Physics Research (NIM-A), Section A 502, 2003, pp. 112-116.

[14] Albrecht E. et al. *The radiator gas and the gas system of COMPASS RICH-1*. Nuclear Instruments & Methods in Physics Research (NIM-A), Section A 502, 2003, pp. 266.

[15] Albrecht E. et al. *The mirror system of COMPASS RICH-1*. Nuclear Instruments & Methods in Physics Research (NIM-A), Section A 502, 2003, pp. 236.

[16] Baum, G. et al. *RICHONE: a software package for the analysis of COMPASS RICH-1 data***.** Nuclear Instruments & Methods in Physics Research (NIM-A), Section A 502, 2003, pp. 315-317.

[17] Albrecht E. et al. *First performances of COMPASS RICH-1*. Nuclear Instruments & Methods in Physics Research (NIM-A), Section A 518, 2004, pp. 586-589.

[18] Fisher, H. et al. *The COMPASS Data Acquisition System*. IEEE Transactions on Nuclear Science, Volume: 49, 2002, pp. 443-447.

[19] Schmitt L. et al. *The* DAQ *of the COMPASS Experiment*. IEEE Transactions on Nuclear Science, Volume: 49, 2003, pp. 443-447.

[20] *http://wwwcompass.cern.ch/compass/detector/daq/welcome.html*

[21] Fischer, H. et al. *The COMPASS Online Data Format*. COMPASS Note 2002-8, July 2003.

[22] McLaren, R. et al. *S-LINK, a data link interface specification for the LHC era*. IEEE Transactions on Nuclear Science, Volume: 44 Issue: 3 Part: 1, Anaheim, CA, USA, June 97, pp. 398-402.

[23] Konorov I. et al. *COMPASS TCS documentation*. COMPASS Note 2001-9, Munich, 2001.

[24] Konorov I. et al. *The Trigger Control System for the COMPASS Experiment*. IEEE Nuclear Science Symposium, San Diego 2001, Conference Record 2002.

[25] K. Siu and R. Jain. *A Brief Overview of ATM: Protocol Layers, LAN Emulation, and Traffic Management*. Computer Communications Review (ACM), Volume: 25, No. 2, April 1995, pp. 6-28.

[26] B. Taylor. *TTC distribution for LHC detectors*. IEEE Transactions on Nuclear Science, Volume: 45, No. 3, 1998, pp. 821-828.

[27] Bus Architecture Standards Committee of the IEEE Computer Society, *Standard for a Common Mezzanine Card Family: CMC.* P1386/Draft 2.2, April 22, 2000.

[28] Braun G. et al. *Designing Frontend Boards for use with the CATCH-HOTLink Interface*. COMPASS Note 1999-7, Freiburg, May 1999.

[29] Cypress Semiconductor Corp. *HOTlink Transmitter/Receiver Data Sheet (CY7B923/CY7B933)*. April 5, 1999.

[30] *http://hsi.web.cern.ch/HSI/s-link/devices/slink-pci*

[31] Renshall H. et al. *A Comparison of GSIFTP and RFIO on a WAN*. Proceedings of the International Conference on Computing in High-Energy Physics and Nuclear Physics (CHEP 2001), September 3-7, 2001, Beijing, China.

[32] CERN ALICE DAQ group, DATE 3.7, ALICE DATE User's Guide, January 2001 ALICE Internal Note/DAQ ALICE-INT-2000-31 v.2.

[33] A. Martin. *The COMPASS off-line system.* Computer Physics Communications 140, 2001, pp. 82–91.

[34] M. Lamanna. *The COMPASS Computing Farm Project*. CHEP 2000 Proceedings, February 7-11, 2000, Padova, Italy. AIP Conference Proceedings Volume: 583(1), August 20, 2001, pp. 238-240.

[35] Toeda, T., Lamanna, M., Duic, V. and Manara, A. *Conditions database system of the COMPASS experiment*. Computer Physics Communications, Volume: 152, Issue: 2, May 1, 2003, pp. 135-143.

[36] Duic, V. and Lamanna, M. *The COMPASS Event Store in 2002.* Computing in High Energy and Nuclear Physics, 24-28 March 2003, La Jolla, California.

[37] J. P. Baud, O. Barring and J. D. Durand. *CASTOR project status*. Proceedings of the CHEP 2000, 7-11 February 2000, Padova, Italy.

[38] *http://coral.cern.ch.*

[39] Jane W. S. Liu. *Real-Time Systems*. Prentice-Hall, 2000.

[40] Jean Labrosse. *The Real-Time Kernel*. MicroC/OS-II. R&D Books, 1999.

[41] Arnold S. Berger. *Embedded Systems Design. An Introduction to Processes, Tools, & Techniques*. CMP Books, 2002.

[42] David B. Stewart. *Twenty-Five Most Common Mistakes with Real-Time Software Development*. Embedded Systems Conference, San Francisco, March 2002.

[43] Michael E. Anderson. *Selecting the Right RTOS. A Comparative Study.* Embedded Systems Conference, San Francisco, March 2002.

[44] Stankovic, John and Ramamritham, Krithi. *Advances in real-time systems*. Los Alamitos, CA, IEEE Computer Society Press, 1993.

[45] Steven W. Smith. *The Scientist and Engineer's Guide to Digital Signal Processing*. Second Edition, California Technical Publishing, 1997.

[46] Ronald L. Fante. *Signal Analysis and Estimation. An Introduction*. John Wiley & Sons, 1988.

[47] A. Bruce Carlson. *Sistemas de Comunicaciones. Introducción a las señales y al ruido en comunicaciones eléctricas*. Editorial Pueblo y Educación, 1984.

[48] R. W. Hamming. *Digital Filters*. Third Edition, Prentice-Hall International Editions, 1989.

[49] Lynn, P. and Fuerst, W. Introductory *Digital Signal Processing with Computer Application*, Revised Edition, John Wiley, 1994.

[50] Richard A. Haddad and Thomas W. Parsons. *Digital Signal Processing: Theory, Applications and Hardware*. New York: Computer Science Press, 1991.

[51] Richard G. Lyons. *Understanding Digital Signal Processing*. First Edition, Prentice Hall PTR, September, 1996.

[52] Yu Hen Hu. *Programmable Digital Signal Processors: Architecture, Programming, and Applications*. Marcel Dekker Inc, 2001.

[53] J. Eyre and J. Bier. *The Evolution of DSP Processors from Early Architectures to the Latest Developments*. IEEE Signal Processing Magazine, Volume: 17, No. 2, March 2000, pp. 43-51.

[54] J. Du, G. Warner, E. Vallow and T. Hollebach. *High-Performance DSPs*. IEEE Signal Processing Magazine, Volume: 17, No. 2, March 2000, pp. 16-26.

[55] J. Eyre and J. Bier. *DSP Processors Hit the Mainstream*. Berkeley Design Technology, Inc., August 1998.

[56] Robert Oshana. *Embedded Systems Programming Using Digital Signal Processors*. Texas Instruments, DSP Engineering, 2001.

[57] Jordi Salazar. *Procesadores digitales de señal (DSP). Arquitecturas y criterios de selección.* Mundo Electrónico, Número 314, Noviembre 2000, pp. 46-57.

[58] Jennifer Eyre. *The Digital Signal Processor Derby*. IEEE Spectrum, June 2001, pp. 62-68.

[59] E. Lee. *Programmable DSP Architectures, Part I / II*. IEEE Signal Processing Magazine, Volume: 5 / 6, No. 4 / 1, October 1988 / January 1989, pp. 4-19 / 4-14.

[60] Phil Lapsley, Jeff Bier, Amit Shoham and Edward. Lee. *DSP Processor Fundamentals: Architectures and Features*. John Wiley and Sons Inc, IEEE Press Series on Signal Processing, January 1997.

[61] Trenz Electronic. Spartan-II, Development System. *Tutorial: Introduction to FPGA Technology*. November 2001.

[62] Stephen Brown and Jonathan Rose. *Architecture of FPGAs and CPLDs: A Tutorial*. IEEE Design & Test of Computers, Volume: 13, No. 2, 1996, pp. 42-57.

[63] Stephen Trimberger. *Field-Programmable Gate Array Technology*. Kluwer Academic Publishers, January 1994.

[64] J. Rose, A. El Gamal and A. Sangiovanni-Vincentelli. *Architecture of Field-Programmable Gate Arrays*. IEEE, Volume: 81, No. 7, July 1993, pp. 1013-1029.

[65] John V. Oldfield and Richard C. Dorf. *Field Programmable Gate Arrays: Reconfigurable Logic for Rapid Prototyping and Implementation of Digital Systems*. John Wiley & Sons Inc, New York, January 1995.

[66] R. J. Petersen and B. Hutchings. *An Assessment of the Suitability of FPGA-Based Systems for use in Digital Signal Processing*. Fifth International Workshop on Field Programmable Logic and Applications, Oxford, England, August 1995, pp. 293-302.

[67] Gregory Ray Goslin. *A Guide to Using Field Programmable Gate Arrays (FPGAs) for Application-Specific Digital Signal Processing Performance*. Digital Signal Processing Program Manager, Xilinx Inc., 1995.

[68] Scott Hauck. *The Roles of FPGAs in Reprogrammable Systems*. IEEE Volume: 86, No. 4, April 1998, pp. 615-638.

[69] Kiran Bondalapati and Viktor Prasanna. *Reconfigurable Computing Systems*. Proceedings of the IEEE, Volume: 90, No. 7, July 2002, pp. 1201- 1217.

[70] Kiran Bondalapati and Viktor Prasanna. *Reconfigurable Computing: Architectures, Models and Algorithms.* Current Science, Volume: 78, No. 7, April 2000, pp. 828-837.

[71] J. Villasenor and W. Mangione-Smith. *Configurable Computing*. Scientific American, Volume: 276, No. 6, June 1997, pp. 66-71.

[72] J. Villasenor and B. Hutchings. *The Flexibility of Configurable Computing*. IEEE Signal Processing Magazine, September 1998, pp. 67-84.

[73] Katherine Compton and Scott Hauck. *Reconfigurable Computing: A Survey of Systems and Software*. ACM Computing Surveys, Volume: 34, No. 2, June 2002, pp. 171-210.

[74] Russel Tessier and Wayne Burleson. *Reconfigurable Computing for Digital Signal Processing: A Survey.* Journal of VLSI Signal Processing Systems, Volume: 28, No. 1-2, May-June 2001, pp. 7–27.

[75] Hernández A. et al. *Arquitecturas y aplicaciones de la computación configurable*. Revista Española de Electrónica, No. 572/573, Julio 2002, pp. 50-56.

[76] Peter J. Ashenden. *The VHDL Cookbook*. First Edition, Dept. Computer Science, University of Adelaide, South Australia, July 1990.

[77] Synopsys, *FPGA Compiler II / FPGA Express, VHDL Reference Manual*. Version 1999.05, May 1999.

[78] Skahill K., Legenhausen J., Wade R., Wilner C. and Wilson, B. *VHDL for programmable logic*. Reading, MA, Addison-Wesley, June 1996.

[79] Charles H. Roth. *Digital systems design using VHDL*. Boston, MA, PWS Publishing Co. January 1998.

[80] Jayaram Bhasker. *A VHDL primer*. Englewood Cliffs, Prentice Hall, 1992.

[81] E. Sternheim, R. Singh and Y. Trivedi. *Digital Design with Verilog HDL*. Automata Publishing Company, 1990.

[82] Baum, G. et al. *BORA: a Front End Board, with local intelligence, for the RICH Detector of the COMPASS Collaboration*. Nuclear Instruments & Methods in Physics Research (NIM-A), Section A 433, 1999, pp. 426-431.

[83] Baum, G. et al. *The COMPASS RICH-1 read-out system*. Nuclear Instruments & Methods in Physics Research (NIM-A), Section A 502, 2003, pp. 246-250.

[84] William Stallings. *Data and Computer Communications*. Fourth Edition, New York, Macmillan Publishing Co., 1994.

[85] Andrew S. Tanenbaum. *Computer Networks*. Second Edition, Prentice-Hall, 1989.

[86] Santiard C. et al. *Gassiplex: a low noise analog signal processor for readout of gaseous detectors*, presented at the 6th Pisa Meeting on Advanced Detector, La Biodola, Isola d'Elba, Italy, May 1994.

[87] Bennedetti F. et al. *First Test of the COMPASS Gassiplex Chip*. COMPASS Note 1998-3, Trieste, 1998.

[88] Christaudo P. et al. Further Tests of the COMPASS Gassiplex Chip with a dedicated evaluation board. COMPASS Note 1999-19, Trieste, 1999.

[89] Colavita A. *All you wanted to know about decoding the RICH-1 but were afraid to ask*. COMPASS Note 2001-5, Trieste, 2001.

[90] Analog Devices, Inc. *ADSP-21065L SHARC User's Manual and Technical Reference.* September, 1998.

[91] Analog Devices, Inc. *DSP Microcomputer ADSP-21065L Data Sheet*. Rev B, 2000.

[92] Analog Devices, Inc. *VisualDSP User's Guide & Reference*. Model Number: VDSP-2106x-UGR. First Edition, July 1998.

[93] Xilinx, Inc. *Virtex Field Programmable Gate Arrays (FPGA) Data Sheet (XCV100)*, v 2.5. April 2, 2001.

[94] Xilinx, Inc. *Virtex FPGA Series Configuration and ReadBack*. Application Note (XAPP 138), v 2.5. November 5, 2001.

[95] Maxim Integrated Products. *Watchdog Timer (MAX706P/R/S/T)*. Rev 2, September 1995.

[96] Cypress Semiconductor Corp. *Asynchronous Dual-Port Static RAM Memory (CY7C057V) Data Sheet*. September 2001.

[97] AMCC Semiconductor Corp. *PCI Matchmaker S5920 User and Technical Reference Manual*. Rev 1.3, April 1998.

[98] Cypress Semiconductor Corp. *UltraLogic 32-Macrocell Flash In-System Reprogrammable Complex Programmable Logic Device (CPLD) CY7C371i Data Sheet*. April 1998.

[99] Analog Devices, Inc. *Dual Channel, 20 MHz 10-bit Resolution CMOS ADC (AD9201) Data Sheet*. Rev D, 1999.

[100] Cypress Semiconductor Corp. *Sychronous FIFO (CY7C4235V) Data Sheet*. Rev B, August 1997.

[101] Maxim Integrated Products. *SOT Temperature Sensor with Multidrop Single-Wire Digital Interface* (*MAX6575L/H) Data Sheet*. Rev 0, April 1999.

[102] Maxim Integrated Products. *8-channel 8-bit ADC (MAX117) Data Sheet*. Rev 1, August 1996.

[103] Analog Devices, Inc. *Parallel Input, Voltage Output 8-Bit DAC (AD7801)*. Rev 0, 1997.

[104] Xilinx Foundation, Series Software, Version 2.1i. July 1999.

[105] Donald Gross and Carl Harris. *Fundamentals of Queueing Theory*. Wiley Series in Probability and Mathematical Statistics. Second Edition, 1985.