

---

# CAPÍTULO

# 3

---

## DESARROLLO DEL MÓDULO DE TRANSICIÓN TM4PLUS1

---

<b>1. INTRODUCCIÓN</b>	<b>1</b>
<b>2. MÓDULO DE TRANSICIÓN (TM4PLUS1)</b>	<b>1</b>
2.1 Descripción del Módulo de Transición	1
2.2 Interface S-LINK	4
<b>3. DISPOSITIVOS LÓGICOS PROGRAMABLES</b>	<b>7</b>
3.1 Arquitectura APEX 20K Altera	9
3.2 Metodología Diseño FPGAs	10
3.3 Reformatting Altera FPGA	11
3.3.1 Bloque FIFO	11
3.3.2 Bloque connect FIFO	12
3.3.3 Bloque control S-LINK	15
3.3.4 Bloque data multiplexing, control & filtering unit	15
3.3.5 Bloque auxiliary FPGA	18
3.3.6 Bloque reformatting FPGA control & test	19
3.4 Auxiliary Altera FPGA	19
3.4.1 Bloque S-LINK	19
3.4.2 Bloque reformatting FPGA	20
3.4.3 Bloque J2B connection	20
3.4.4 Bloque data control unit	20
3.4.5 Bloque auxiliary FPGA clock, control & test	20
3.4.6 Bloque ODIN LSC (G-LINK)	20
<b>4. FORMATO DE DATOS DEL ROD</b>	<b>22</b>
4.1 Estructura de los Datos de Salida	22
<b>5. BIBLIOGRAFÍA</b>	<b>25</b>



## 1. INTRODUCCIÓN

Después de encuadrar el ROD en la estructura de adquisición de datos de ATLAS y de ver las características de dos de los elementos fundamentales del mismo (Motherboard y PUs), en este capítulo se va a estudiar en profundidad el Módulo de Transición Activo (Tm4Plus1) ya comentado en puntos anteriores.

Veremos una descripción detallada de sus características principales, los elementos que lo componen, un estudio de los dispositivos programables incluidos en él, los protocolos de comunicación implementados en las FPGAs y los desarrollos realizados para tratar y dar formato a los datos que recibe del primer nivel de trigger.

Es en este módulo donde se ha centrado este Trabajo de Investigación, tanto en su diseño como en su test, su programación y su puesta en marcha en el Test Beam del CERN. Trabajando así mismo en colaboración directa con el desarrollo del RODDemo y del próximo sistema ROD.

## 2. MÓDULO DE TRANSICIÓN (TM4PLUS1)

El Módulo de Transición [1] es un interfaz entre el front-end de TileCal, la motherboard del ROD y los links de salida de este sistema hacia el ROB. Es un paso intermedio en la adecuación de los datos que se reciben y los algoritmos de filtrado que se implementan en las PUs de la Motherboard. Así mismo, está encargado de rutar los datos al siguiente nivel de trigger ya con el formato del ROD.

Es un módulo que se inserta en la parte trasera del ROD, conectado a él por VME y que tiene la capacidad de incorporar cuatro tarjetas S-LINK (veremos más adelante en que consisten estas tarjetas) para la entrada de datos y una tarjeta ODIN integrada para la salida de datos. Mediante las dos FPGAs incluidas en el módulo, podemos realizar operaciones de filtrado, multiplexado y comprobación de los datos que entran al ROD así como la implementación de protocolos de comunicación entre los diferentes módulos presentes en él.

### 2.1 DESCRIPCIÓN DEL MÓDULO DE TRANSICIÓN

El módulo activo Tm4Plus1 [2] es un sistema VME64x insertado en una crate 9U que presenta slots de conexión para cuatro tarjetas S-LINK y que tiene integrada una tarjeta ODIN. En los slots S-LINK se puede conectar cualquier modelo de tarjeta LDC (link destination card). El módulo se ajusta en la parte trasera de las nuevas extensiones de las crates VME64 y puede usar cualquier entrada/salida de las conexiones con la motherboard para la transferencia de datos.

Es capaz de leer 8 x 32 bits de datos a 40 MHz, cada tarjeta LDC conectada al módulo permite la conexión de dos fibras ópticas provenientes del detector. Así pues, nos permite tratar 4 canales con redundancia en los datos. Sus características principales se pueden detallar mediante la siguiente tabla.

Ítem	
Tipo de Crate	VME64x standard, con adaptador P3 back-plane
Tipo de tarjeta	9U
Slots entrada datos	4 (a 4 FIFOs de 3 Kwords)
Número de fibras ópticas entrada	8 (32 bits @ 40 MHz)
Número de LDCs conectadas	4 (conector doble)
Número de fibras ópticas salida	2 (32 bits @ 40 MHz)

Número de FPGAs (APEX 20K)	2 (BGA 252 pines)
Tipo de Alimentación	5 V – 3.3 V (tomados VME)

TABLA 3.1 Características generales del Módulo de Transición.

Por tanto, el Módulo de Transición presenta cuatro conectores para las tarjetas LDC, cuatro FIFOs para la lectura y almacenamiento de datos, dos FPGAs para el tratamiento de los mismos, una tarjeta ODIN integrada para la salida de datos, dos convertidores electrónico-ópticos con salida a dos fibras ópticas, un convertor para obtener los 3.3 V, dos conectores JTAG y dos EPROMs para la programación de las FPGAs, dos osciladores para generar los relojes de 40 MHz de la placa, un buffer para su distribución y dos conectores (J2 y J3) mediante los cuales se conecta a la crate.

En estas figuras podemos ver el diagrama de bloques del Módulo de Transición con el flujo que siguen los datos y una fotografía del mismo.

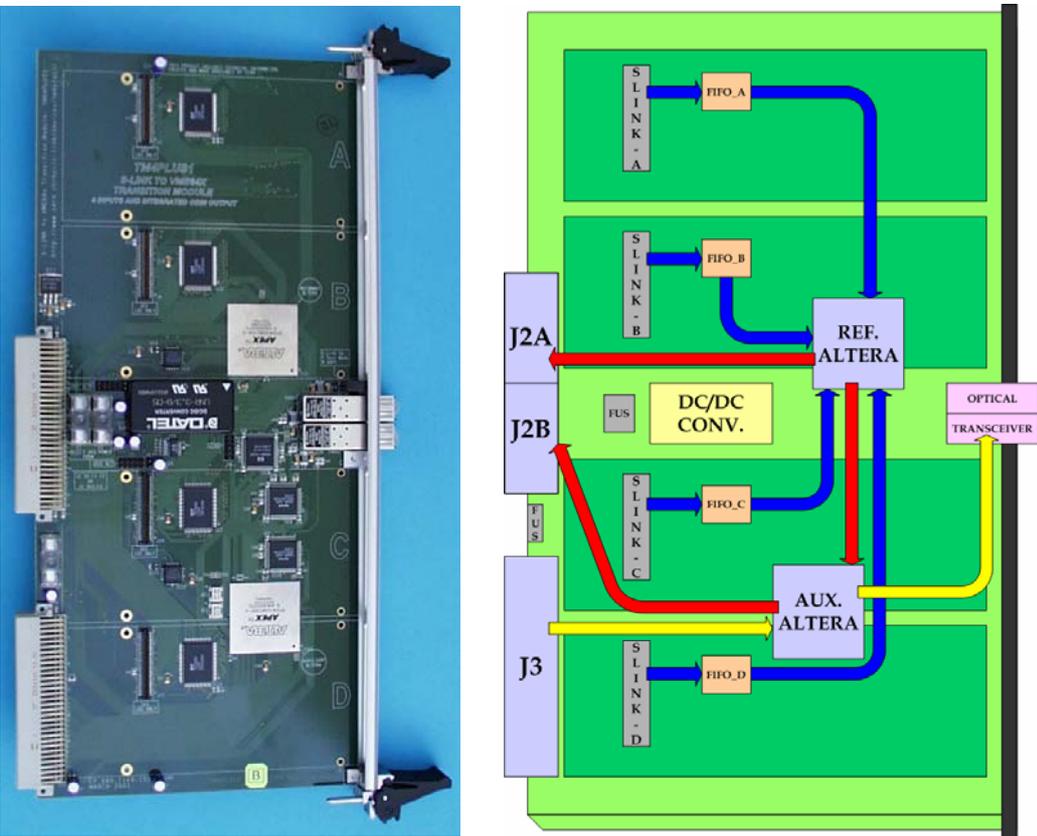


FIGURA 3.1 Fotografía y diagrama de bloques del Módulo de Transición.

Como se puede observar en el diagrama de bloques anterior, el flujo que siguen los datos (se explicará con mayor detalle en apartados posteriores) comienza por la recepción de los datos en las cuatro tarjetas LDC conectadas a los slots (SLINK A B C D) su posterior envío a las FIFOs (A B C D) y su rutado a la Reformatting Altera FPGA.

En dicha FPGA se reciben los 8 canales de datos de 32 bits a 40 MHz, se sincronizan los datos al reloj del módulo, se comprueba si el CRC es correcto, se

realiza un primer filtrado (este caso dependerá del funcionamiento que se le esté dando al ROD) y se multiplexan los datos en dos bloques de 16 bits para ser enviados a las PUs de la motherboard. Este envío se realiza por los conectores J2A y J2B. Los datos de los slots A y B se envían directamente desde la Reformatting Altera FPGA y los datos de los slots C y D pasan primero por la Auxiliary Altera FPGA para ser rutados hacia J2B (por una cuestión de número de pines).

Una vez realizado el procesado en la motherboard, los datos son enviados al Módulo de Transición por J3 y de aquí a la Auxiliary Altera FPGA. Dichos datos ya viene con el formato del ROD en forma de palabras de 32 bits a 40 MHz, conteniendo la información de la energía el tiempo y la forma del pulso. La Auxiliary FPGA multiplexa estos datos en dos bloques de 16 bits a 40 MHz para enviarlos a los transmisores de la tarjeta ODIN integrada en el módulo que serán los encargados de enviarlos por fibra óptica al siguiente nivel de trigger (ROB).

En este otro diagrama de bloques se puede ver con mayor detalle todo el ciclo que siguen los datos desde su salida por fibra óptica del detector hasta su envío al ROB.

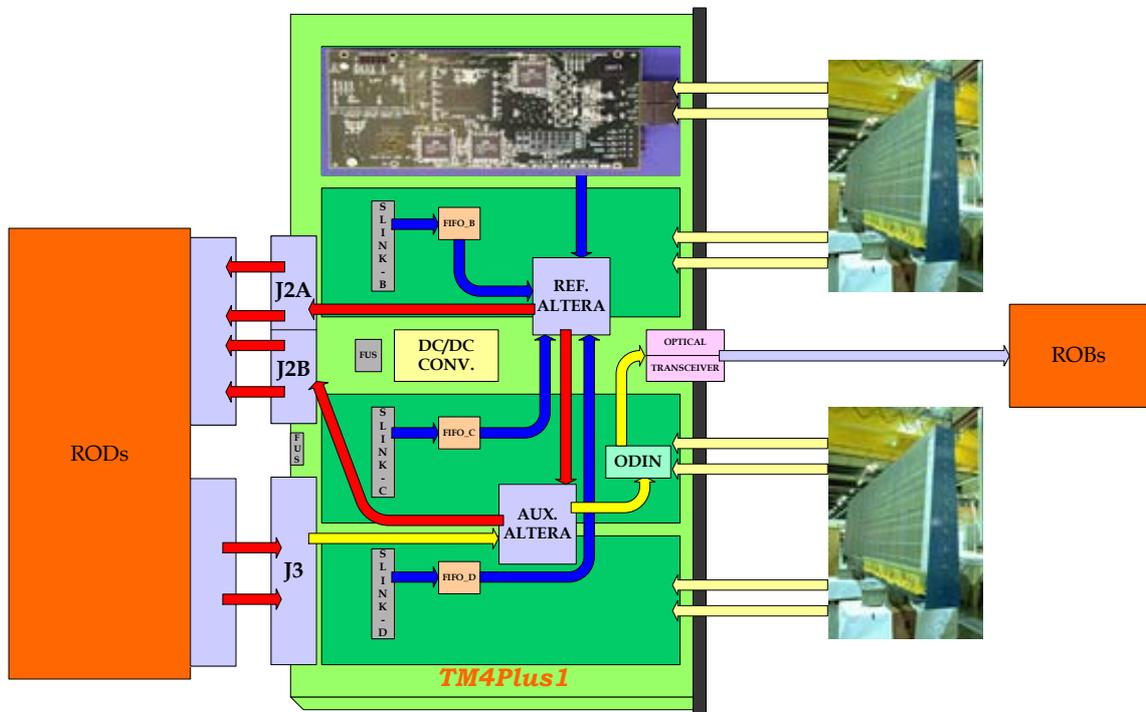


FIGURA 3.2 Flujo de los datos de TileCal.

Una vez tenemos estructurado y definido el sistema de adquisición de datos de TileCal sobre el cual hemos trabajado, vamos a ver en los siguientes apartados los protocolos usados para el envío y recepción de los datos, las implementaciones y características de las FPGAs y el formato de los datos que estudiamos.

## 2.2 INTERFACE S-LINK

El desarrollo de sistemas de comunicación donde trabajan un gran número de tarjetas requiere de protocolos de comunicación simples a la vez que muy fiables. Las conexiones de las fibras ópticas deben funcionar perfectamente, con el mínimo retardo posible y la mayor eficiencia. Para este fin, y después de estudiar distintos sistemas de comunicación, nació el protocolo S-LINK [3] (Simple LINK interface). Se usa para conectar cualquier capa de la electrónica del Front-End con la siguiente capa de la electrónica del Read-Out [4].

Nos ofrece unas especificaciones para el envío de los datos de un punto a otro, ofreciendo además:

- Una detección de errores (línea LDERR).
- Un control de los datos (línea UCTRL).
- La posibilidad de trabajar con diferentes tamaños de palabra (8, 16 y 32 bits).
- Un canal de retorno para controlar el flujo de los datos (línea UXOFF).
- Una función de autotest (línea UTEST) para comprobar si el enlace es correcto.
- Una función de reset para sincronizar correctamente las dos tarjetas y que el envío sea correcto (línea URESET).

En nuestro caso, se usa doblemente. Existe un primer enlace entre el Front-End y el Módulo de Transición y un segundo enlace entre Módulo de Transición y ROB. Para poder usar este interface se han diseñado en el CERN dos tipos de tarjetas, las Link Source Cards (LSC) y las Link Destination Cards (LDC) [5]. La primera de ellas encargada del envío de los datos y la segunda de la recepción de los mismos. Una imagen de este enlace es la siguiente

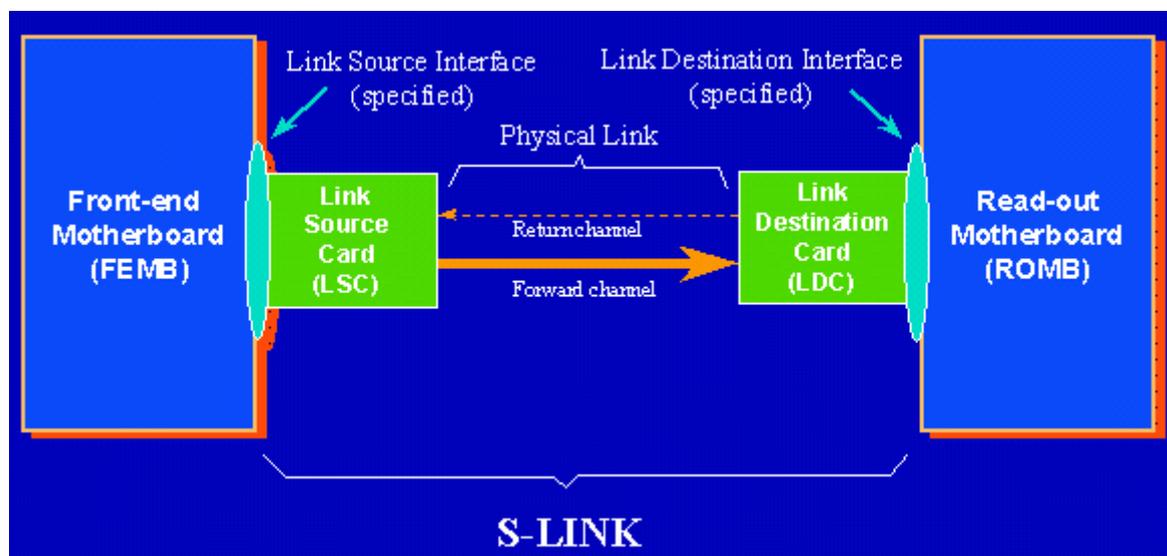


FIGURA 3.3 Protocolo de comunicación S-LINK.

Como acabamos de comentar, S-LINK necesita de dos componentes, las tarjetas LSC y las tarjetas LDC. Su comunicación se basa en las siguientes líneas, las cuales realizan las funciones definidas antes, durante y después de la recepción o envío de los datos.

Pin Symbol	Pin Name	I/O	Description
UD[31..0]	User Data input lines	Input to S-LINK	Data on these lines is transferred to the LSC on a low-to-high transition of UCLK when UWEN is low. UD[3..0] are ignored if UCTRL# is low. Synchronous with UCLK.
LFF#	Link Full Flag	Output from S-LINK	Data <i>shall</i> only be written to the S-LINK when this line is high. After it goes low, up to two more words <i>may</i> be written. Functional when S-LINK is in test mode. Undefined when URESET# is low. Synchronous with UCLK.
URESET#	User Reset line	Input to S-LINK	When low initiates a reset cycle. Asynchronous.
UTEST#	User Test line	Input to S-LINK	When low switches the LSC to test mode. Causes LDOWN# to go low. Asynchronous.
UDW[1..0]	User Data Width lines	Input to S-LINK	Define the data width the S-LINK is to be operated in. Data width codes at LSC and LDC must be the same. Sampled after a reset cycle.
UCTRL#	User Control line	Input to S-LINK	When low indicates that the data to be transmitted is a control word. Causes UD[3..0] to be ignored. Synchronous with UCLK.
UWEN#	User Write Enable	Input to S-LINK	When low enables data to be transferred to the S-LINK on the low-to-high transition of UCLK. Synchronous with UCLK.
UCLK	User Clock	Input to S-LINK	Data is transferred to the S-LINK on the low-to-high transitions of UCLK when UWEN# is low. This is a free running clock.
LRL[3..0]	Link Return Lines	Output from S-LINK	These lines reflect the state of URL[3..0] at the LDC. In the simplex version these lines are not functional. A simplex FEMB <i>shall</i> leave these lines unconnected. A simplex LSC <i>shall</i> connect these lines to GND. Asynchronous, even relative to each other.
LDOWN#	Link Down	Output from S-LINK	When low indicates that the S-LINK is not operational. Asynchronous. Can go low due to: <ul style="list-style-type: none"> <li>- S-LINK failure; then LDOWN# is latched low until cleared by a reset cycle.</li> <li>- S-LINK undergoing reset cycle, then LDOWN# goes high when reset cycle is complete.</li> <li>- S-LINK in test mode, then LDOWN# goes high when test mode is ended.</li> </ul> The FEMB <i>shall</i> pull LDOWN# low if the LSC is not present or not powered up.

**TABLA 3.2 Descripción de las señales de una tarjeta LSC [3].**

Para una tarjeta LDC las líneas que utiliza para su reconocimiento y funcionalidad son las siguientes.

Pin Symbol	Pin Name	I/O	Description
LD[31..0]	Link Data output lines	Output from S-LINK	Data present on these lines may be latched on the low-to-high transition of LCLK when LWEN# is low. LD[3..0] have special meanings when LCTRL# is low. Synchronous with LCLK.
UXOFF#	User Transmit Off	Input to S-LINK	When low, signals the LSC to stop transmitting data. Functions in test mode if UTDO# is low but does not stop transmission of test pattern if UTDO# is high. Asynchronous. In the simplex version this pin is not functional. A simplex ROMB <i>shall</i> pull-up this line to Vcc. A simplex LDC <i>shall</i> leave this line unconnected.
URESET#	User Reset line	Input to S-LINK	When low initiates a reset cycle. Asynchronous.
UTDO#	User Test Data Out	Input to S-LINK	When low, the received test pattern is transferred from the LDC to the ROMB during test mode. Sampled after a reset cycle.
UDW[1..0]	User Data Width lines	Input to S-LINK	Define the data width the S-LINK is to be operated in. Data width codes at LSC and LDC must be the same. Sampled after a reset cycle.
LCTRL#	Link Control line	Output from S-LINK	When low indicates that a control word is being transferred. Causes LD[3..0] to have a special meaning. Synchronous with LCLK.
LWEN#	Link Write Enable line	Output from S-LINK	When low indicates that valid data will be transferred to the ROMB on the low-to-high transition of LCLK. Synchronous to LCLK.
LCLK	Link Clock	Output from S-LINK	Data is transferred to the ROMB on each low-to-high transition of LCLK when LWEN# is low. This is a free-running clock if LDOWN# is high. If LDOWN# is low, this signal is undefined.
LDERR#	Link Data Error	Output from S-LINK	When low indicates that a data transmission error has occurred or that a pattern error has occurred during test mode. With word-by-word error reporting, LDERR# goes low with the word in error and with the control word for that block. With block basis error reporting, LDERR# goes low only with the control word for the block containing the word with error.
URL[3..0]	User Return Lines	Input to S-LINK	The state of these lines is sampled, transmitted back to the LSC and presented on LRL[3..0]. Asynchronous, even relative to each other. In the simplex version these pins are not functional. A simplex ROMB <i>shall</i> connect these lines to GND. A simplex LDC <i>shall</i> leave these lines unconnected.
LDOWN#	Link Down	Output from S-LINK	When low indicates that the S-LINK is not operational. Asynchronous. Can go low due to: - S-LINK failure; then LDOWN# is latched low until cleared by a reset cycle. - S-LINK undergoing reset cycle, then LDOWN# goes high when reset cycle is complete. - S-LINK in test mode, then LDOWN# goes high when test mode is ended. The ROMB <i>shall</i> pull LDOWN# low if the LDC is not present or not powered up.

TABLA 3.3 Descripción de las señales de una tarjeta LDC [3].

Una vez definido el enlace entre el detector y el Módulo de Transición, los datos usando este interface se almacenan en las FIFOs presentes en el módulo. El siguiente paso es la lectura de esas FIFOs mediante la Reformatting Altera FPGA y su adecuación a nuestras especificaciones.

### 3. DISPOSITIVOS LÓGICOS PROGRAMABLES

Dentro del Módulo de Transición podemos decir que los elementos principales del mismo son dos FPGAs. Gracias a ellas hemos podido diseñar un sistema compacto para la recepción, tratamiento, distribución y salida de los datos.

Las FPGAs (Field Programmable Gate Arrays) son una clase de Dispositivo Lógico Programable [6], es decir, dispositivos digitales que contiene estructuras lógicas compuestas de puertas lógicas, recursos de interconexión programables, estructuras de entrada/salida flexibles y estructuras de memoria interna.

Permiten al usuario configurar la implementación hardware de una determinada máquina algorítmica definida en un lenguaje de descripción de alto nivel, que va a ser sintetizado en función de la tecnología y el dispositivo en cuestión. Se ha usado para nuestro sistema dos FPGAs de Altera, el software QUARTUS II [7] y el lenguaje VHDL. La siguiente figura muestra la estructura interna de una FPGA genérica.

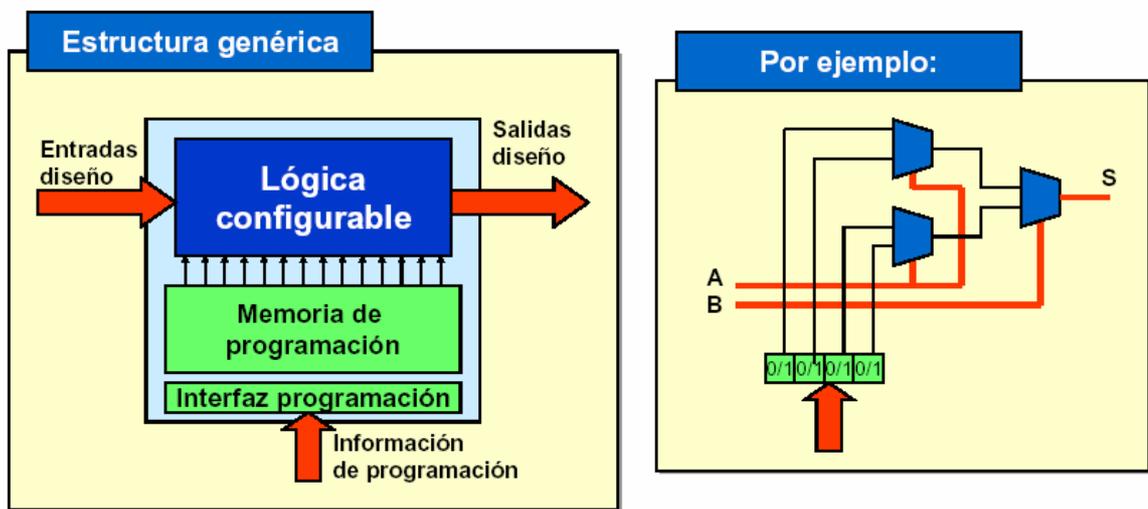


FIGURA 3.4 Estructura genérica de las FPGAs.

Las FPGAs pueden considerarse como una evolución de los Dispositivos Lógicos Programables (PLDs), aunque el incremento de complejidad ha supuesto cambios estructurales. Una característica importante de estos sistemas, que ayudó a su inserción en el Módulo de Transición, es la autonomía del diseñador respecto al sistema en sí. Permite una simulación completa del dispositivo sin necesidad de trabajar directamente en él.

Respecto a la evolución tecnológica de los últimos años, los dispositivos reconfigurables han ido progresivamente comiendo terreno a soluciones tipo ASIC (Application Specific Integrated Circuit), predominantes en la década anterior. Ello ha sido debido a su mejor adaptación a la constante evolución tecnológica.

Recordar, en cualquier caso, que la diferencia fundamental entre las soluciones de implementación ASIC y FPGA reside en el nivel de personalización de la aplicación. Mientras que al nivel de ASIC el diseñador puede interaccionar con diferentes etapas del proceso tecnológico al nivel de FPGA únicamente se configura el sistema, es decir, se asignan los valores a las funciones lógicas, las interconexiones activas entre dispositivos y la funcionalidad de las celdas de entrada/salida del circuito integrado.

Como consecuencia, todo el proceso de concepción, desde la descripción hasta la implementación hardware puede ser realizado en “casa” del diseñador (“Field Programmable”) evitándose tiempos de espera debidos a procesos de fabricación. Esto tiene una segunda lectura: cualquier rediseño requerido por nuevas especificaciones o por detección de errores no deseados se puede realizar de una forma muy rápida al evitar la interacción con el fabricante. La clave está en saber escoger un dispositivo que se adecue a nuestras especificaciones tanto actuales como futuras.

Un ejemplo del interior de una FPGA puede observarse en el siguiente diagrama de bloques.

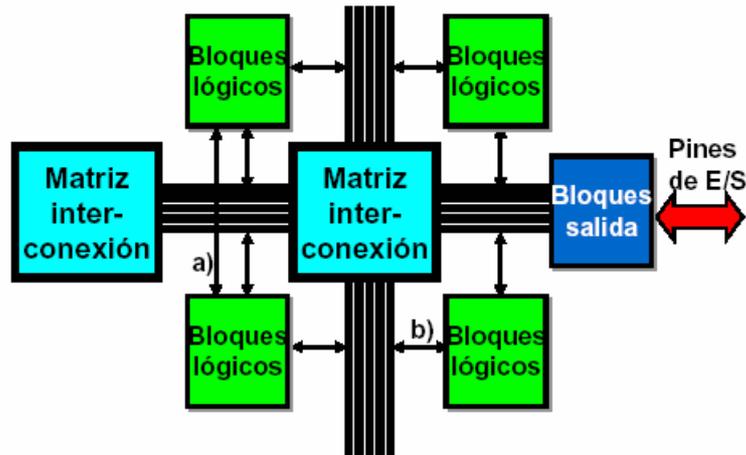


FIGURA 3.5 Estructura interna de las FPGAs.

En contrapartida, las prestaciones equivalentes en superficie de silicio y tecnología utilizada de los dispositivos reconfigurables son menores que las de los dispositivos ASIC. Esta relación suele cifrarse en un factor 5 respecto a densidad de integración (medido como número de puertas por unidad de superficie) y en un factor 3 en términos de velocidad (medido en velocidad máxima de reloj).

Uno de los argumentos del auge de las FPGAs es la sencillez de las herramientas, aunque los procesos involucrados en el diseño de las mismas son muy similares a los correspondientes al diseño con ASICs. Para la programación de las FPGAs del Módulo de Transición se usó el Software QUARTUS II de Altera, tal y como habíamos comentado anteriormente, mediante el cual se realizan Síntesis de Alto Nivel (comportamental y lógico) como Síntesis de Nivel Físico (floorplan, placement y routing).

Las FPGAs que contiene el módulo son de la familia APEX 20K de Altera, a continuación veremos sus características fundamentales.

### 3.1 ARQUITECTURA APEX 20K ALTERA

Se trata del primer dispositivo lógico programable que incorpora la integración de un SOPC (System On a Programmable Chip) [7]. Presenta una arquitectura MultiCore integrada por lógica Look Up Table (LUT), lógica producto-término y memoria embebida. Su nombre viene de Advance Programmable Element matrix (APEX) usando tecnología SRAM (2.5 V y 0.25/0.2  $\mu$ ).

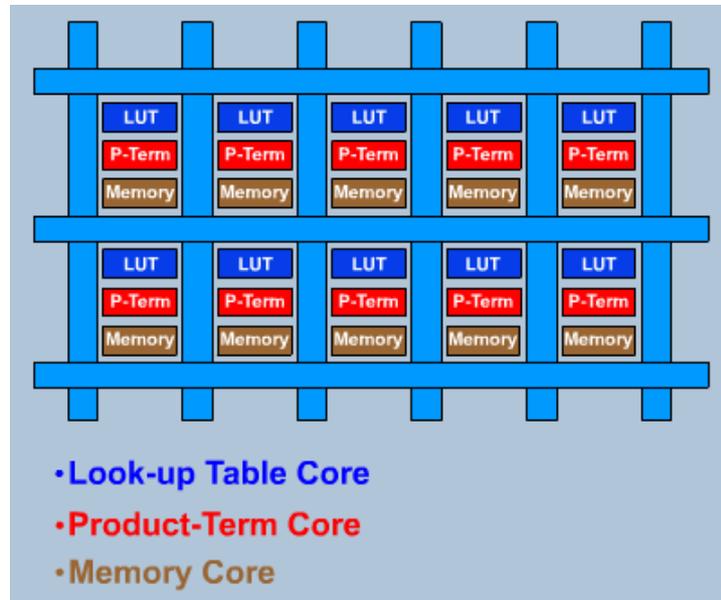


FIGURA 3.6 Arquitectura principal de la APEX 20K.

El bloque computacional básico de las FPGAs esta formado por Elementos Lógicos (LE), Logic Array Blocks (LAB) que tienen un total de 10 LEs, Embedded System Block (ESB) que son memorias o bloques lógicos, MegaLABs formadas por n-LEs + ESB, FastTrack o interconexiones locales, Entradas/Salidas y Relojes específicos.

Las MegaLABs se conectan con interconexiones Fila o Columna y los elementos de E/S se sitúan al final de las filas o las columnas. La siguiente figura muestra esta disposición.

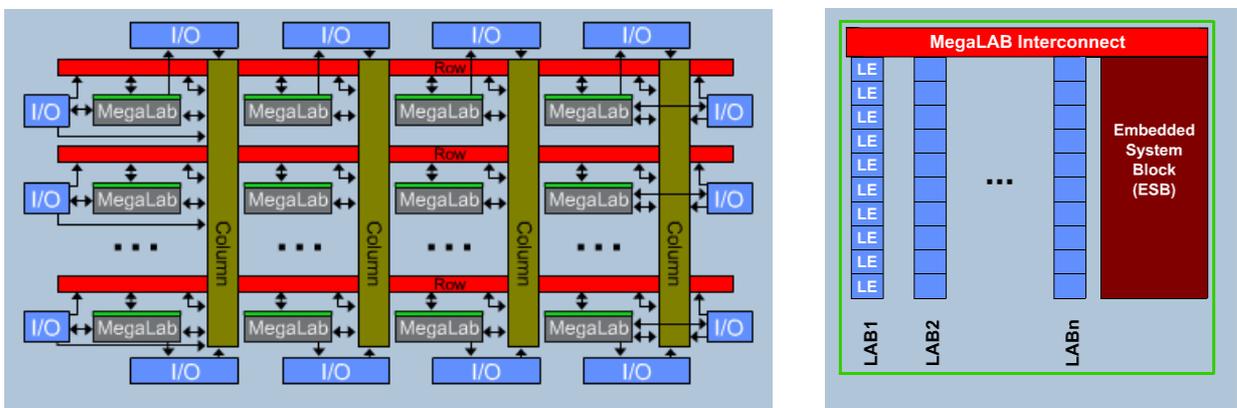


FIGURA 3.7 Interior de una FPGA.

### 3.2 METODOLOGÍA DISEÑO FPGAS

La metodología de diseño top-down usada en este Trabajo de Investigación se puede resumir en estos pasos:

- Diagrama de bloques. Primera idea mental de los bloques que se desean implementar.
- Captura del diseño. Realización del diagrama de bloques en la aplicación del fabricante, introducción de la programación en VHDL.
- Compilación. Comprobación de la no existencia de errores funcionales.
- Análisis de tiempos. Confirmación de que los requerimientos de tiempo se cumplen sin presentar retardos.
- Simulación. Creación de los vectores de test y verificación de tener una salida correcta.
- Programación. Implementación de nuestro diseño en la FPGA.

Para la realización de todos estos pasos se ha usado la herramienta QUARTUS II que nos ofrece el mismo fabricante de las FPGAs. Se trata de un software de desarrollo que nos ofrece un entorno de diseño completo para los SOPC. Tiene una total integración con múltiples herramientas EDA (Electronic Design Automation).

En los siguientes diagramas de bloques se puede observar el camino que se debe seguir para una implementación completa del diseño.

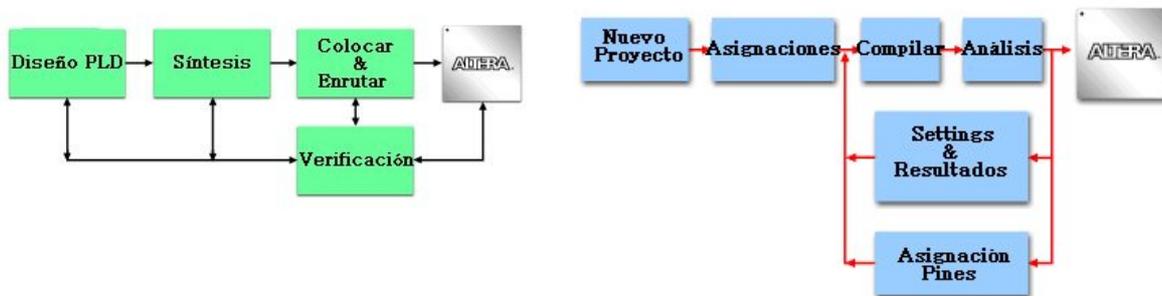


FIGURA 3.8 Metodología de desarrollo.

### 3.3 REFORMATTING ALTERA FPGA

Siguiendo la metodología de diseño expuesta en el apartado anterior, la implementación de nuestras especificaciones [8] de trabajo para un entorno de funcionamiento hardware en tiempo real se basa en el siguiente diagrama de bloques. Será a partir de él desde donde explicaremos bloque a bloque el funcionamiento del sistema.

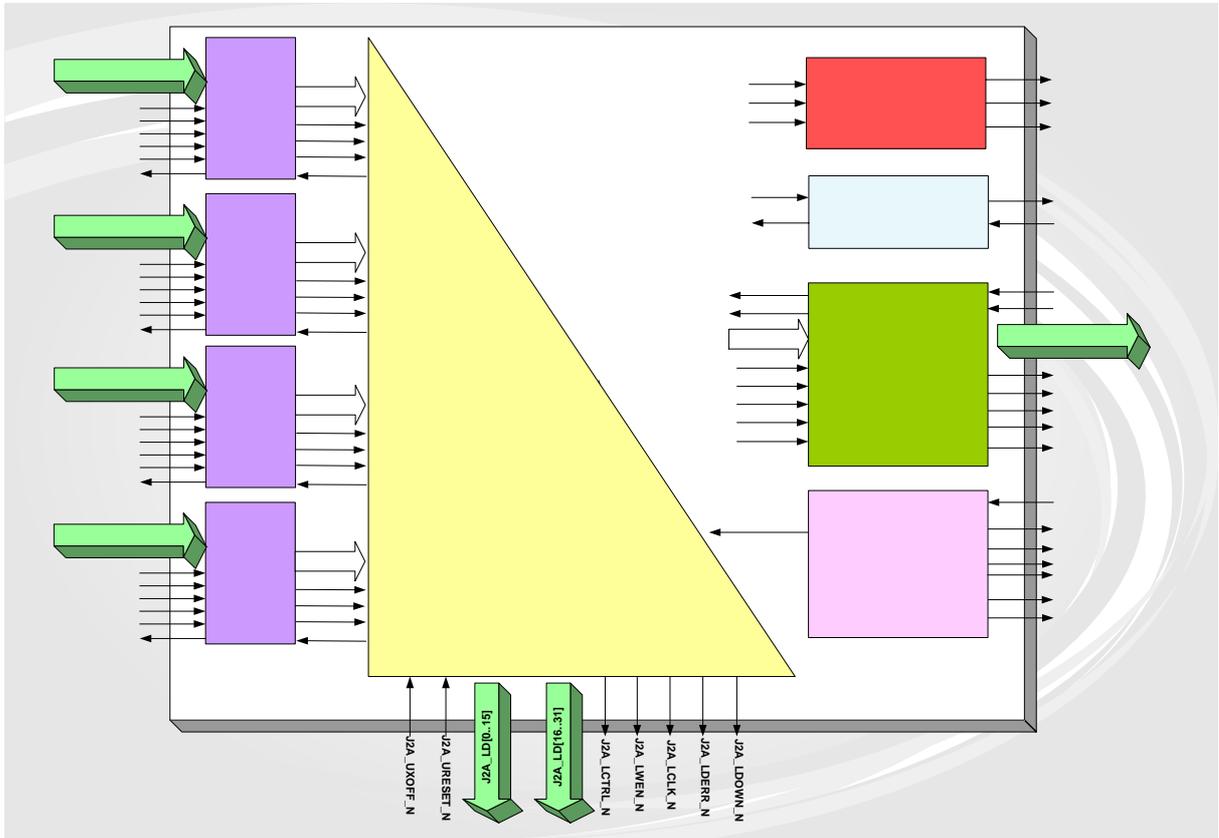
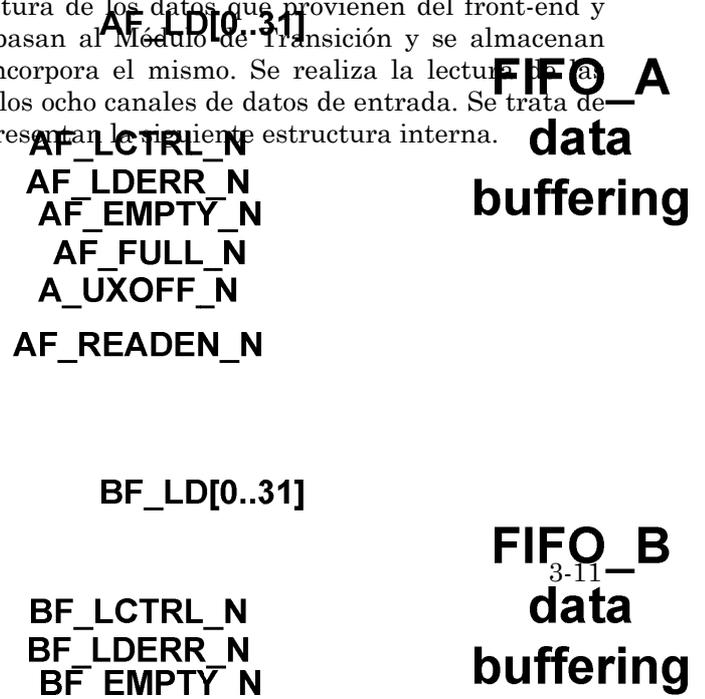


FIGURA 3.9 Diagrama de bloques Reformatting Altera FPGA.

#### 3.3.1 Bloque FIFO

En estos bloques se realiza la lectura de los datos que provienen del front-end y que mediante el protocolo S-LINK pasan al Módulo de Transición y se almacenan temporalmente en las FIFOs que incorpora el mismo. Se realiza la lectura a cuatro tarjetas LDC conectadas y de los ocho canales de datos de entrada. Se trata de cuatro IDT 72V3660 FIFOs [9] que presentan la siguiente estructura interna.



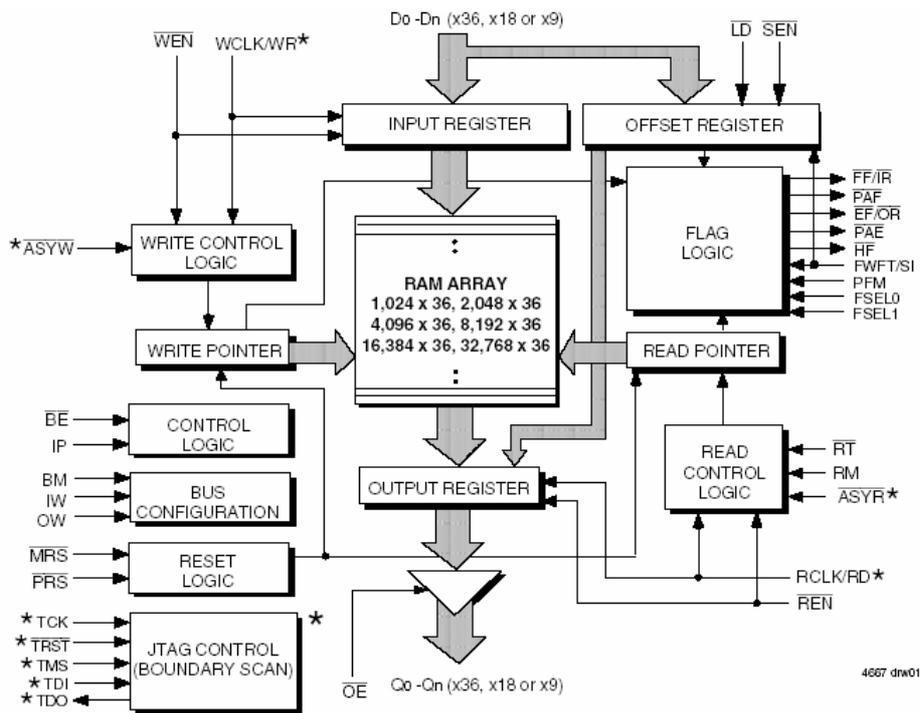


FIGURA 3.10 Diagrama de bloques interior FIFO.

Para la lectura de los datos (LD[31..0]), se envía una señal READEN que permite el inicio de la toma de datos. Como se puede observar en el diagrama de bloques, en este bloque se realiza la lectura de las señales de control de S-LINK (LCTRL, LDERR, UXOFF) y las señales de control de la FIFO (EMPTY, FULL). Los datos están formados por palabras de 32 bits @ 40 MHz que entran en paralelo en la FPGA.

Este bloque se encarga por tanto de sincronizar dichos datos a la frecuencia de reloj del módulo, permitiendo así un flujo de datos continuo y sin retardos. En caso de que la FIFO esté vacía, el proceso de lectura se realiza (envío de la señal READEN) hasta que se produzca un flag de FIFO llena y se pare el proceso. Las señales de S-LINK se envían al bloque central para su posterior tratamiento.

### 3.3.2 Bloque connect FIFO

Encargado de inicializar el modo de operación de las FIFOs, necesario para seleccionar el IDT Standard Mode Timing (modo de operación definido por las FIFOs donde sólo se envía el dato cuando mandamos una señal de lectura) [9]. Mediante la señal MRESET se realiza un master reset que permite inicializar las funciones de la FIFO. La señal FWFT selecciona el modo de operación comentado anteriormente y la señal BUSM la anchura del bus de datos a transmitir.

Las siguientes figuras muestran los timings de reset, lectura y escritura.

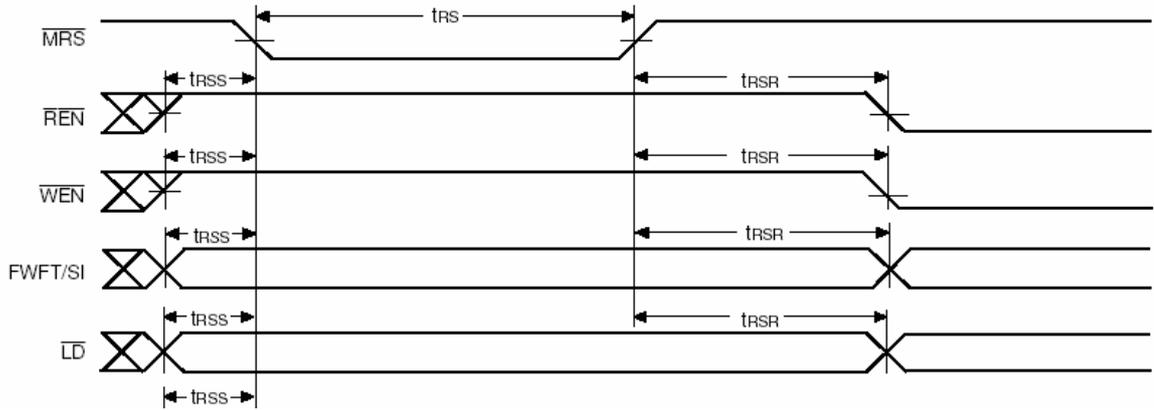


FIGURA 3.11 Timing Master Reset.

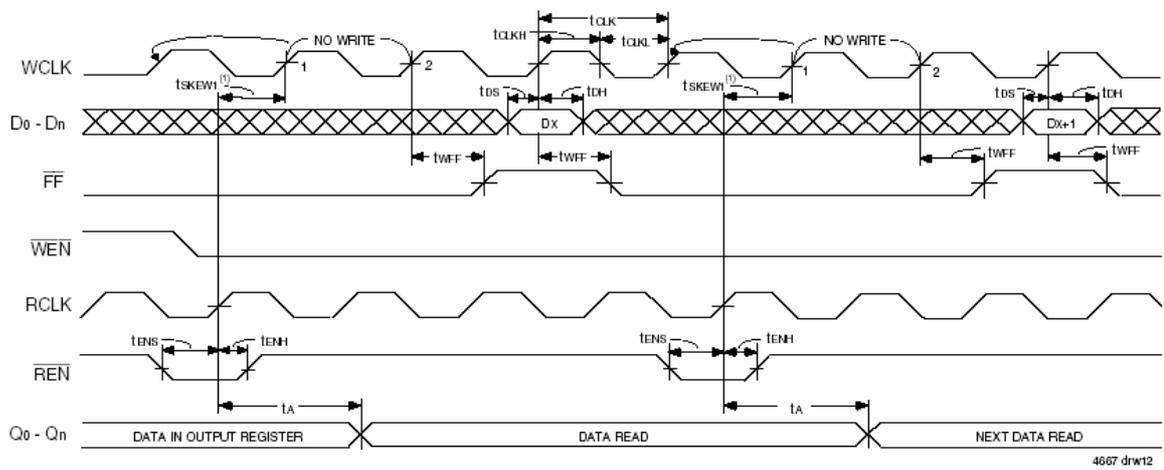


FIGURA 3.12 Write Cycle Timing.

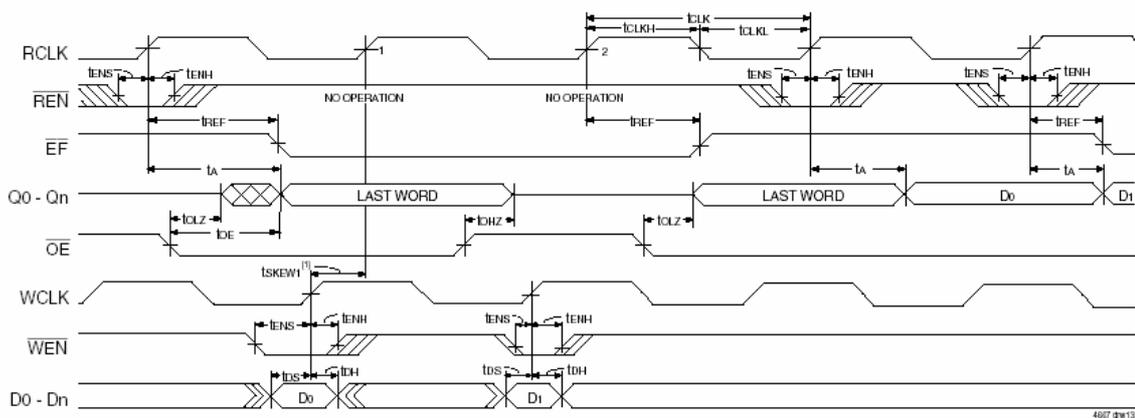


FIGURA 3.13 Read Cycle Timing.

La siguiente tabla muestra todas las líneas implicadas en el proceso de lectura/escritura de la FIFO [2] y su descripción.

Name	Symbol (Schematics)	Symbol (FIFO)	I/O, Pins	Description
Data Inputs	A_LD[0..31]	D	I-34	From the S-LINK (AF_LD, AF_LCTRL#, AF_LDERR#)
Data in (pins 18, 36)		D	I-2	Unused data input pins (total inputs: 36; used: 34)
Master Reset	F_MRESET#	MRS#	I-1	The only reset feature implemented
Partial Reset		PRS#	I-1	Disabled: only master reset is used
Retransmit		RT#	I-1	Useful only if data needs to be read again but not sent again (no use in the design)
First Word Fall Through/ Serial In	F_FWFT#	FWFT/SI	I-1	During master reset, selects First Word Fall Through or IDT Standard mode.
Output width		OW	I-1	Selects the bus width of the read port (18 or 36-bit)
Input Width		IW	I-1	Selects the bus width of the write port (fixed to 36-bit)
Bus-matching	F_BUSM#	BM#	I-1	Selects the bus width of the read/write port
Big Endian/ Little Endian		BE#	I-1	Always set to Little-Endian format
Retransmit timing mode		RM	I-1	Set to Normal Latency Mode
Programmable Flag Mode		PFM	I-1	Set to Synchronous Programmable Flag Timing Mode
Interspersed Parity		IP	I-1	Set to Non- Interspersed Parity Mode
Flag Select Bit0		FSEL0	I-1	During master reset, selects (with FSEL1 and LOAD) the default offset values for PAF (fixed to 1k)
Flag Select Bit1		FSEL1	I-1	During masterreses (with FSEL0 and LOAD) the default offset values for PAF (fixed to 1k)
Write Clock	A_LCLK	WCLK	I-1	Clock from the S-LINK (LCLK); write clock for the FIFO
Write Enable	A_LWEN#	WEN#	I-1	Signal from the S-LINK (LWEN)
Read Clock	BUFFERCLK_A	RCLK	I-1	Clock from the Buffer; read clock for the FIFO
Read Enable	AF_READEN#	REN#	I-1	Signal from the Altera (READEN)
Output Enable		OE#	I-1	Controls the output impedance of Q
Serial Enable		SEN#	I-1	Parallel loading of programmable flag offsets. Feature not available since it works on WCLK, not available in the Altera.
Load	LOAD#	LD#	I-1	During master reset, selects (with FSEL0 and FSEL1) the default offset values for PAF (fixed to 1k)
Full Flag/ Input Ready	AF_FULL#	FF/IR#	O-1	Indicates whether or not the FIFO memory is full
Empty Flag/ Output Ready	AF_EMPTY#	EF/OR#	O-1	Indicates whether or not the FIFO memory is empty
Programmable Almost-Full Flag	A_UXOFF#	PAF#	O-1	Goes HIGH if the number of free locations in the FIFO memory is more than offset m (LOW if less or equal to m); m=1024
Programmable Almost-Empty Flag		PAE#	O-1	LOW if the number of words in the FIFO memory is less than offset n (HIGH if more or equal). Not used
Half-Full Flag		HF#	O-1	Not used
Data Output	AF_LD[0..31]	Q	O-34	To the Altera (AF_LD, AF_LCTRL#, AF_LDERR#)
Data out, pins 18, 36		Q	O-2	Unused data output pins (total inputs: 36; used: 34)

TABLA 3.4 Descripción de las señales de en las FIFOs.

### 3.3.3 Bloque control S-LINK

Mediante la línea URESET conseguimos inicializar la máquina de estados de la tarjeta LDC conectada al módulo. A su vez, y mediante la línea LDOWN, sabemos si el enlace es correcto y si vamos a tener datos de entrada. Estas líneas cumplen el protocolo S-LINK comentado con anterioridad, no pasan por las FIFOs sino que van directas a la tarjeta LDC.

El conjunto de todas las líneas [2] de estos tres bloques se puede observar con claridad en la siguiente figura.

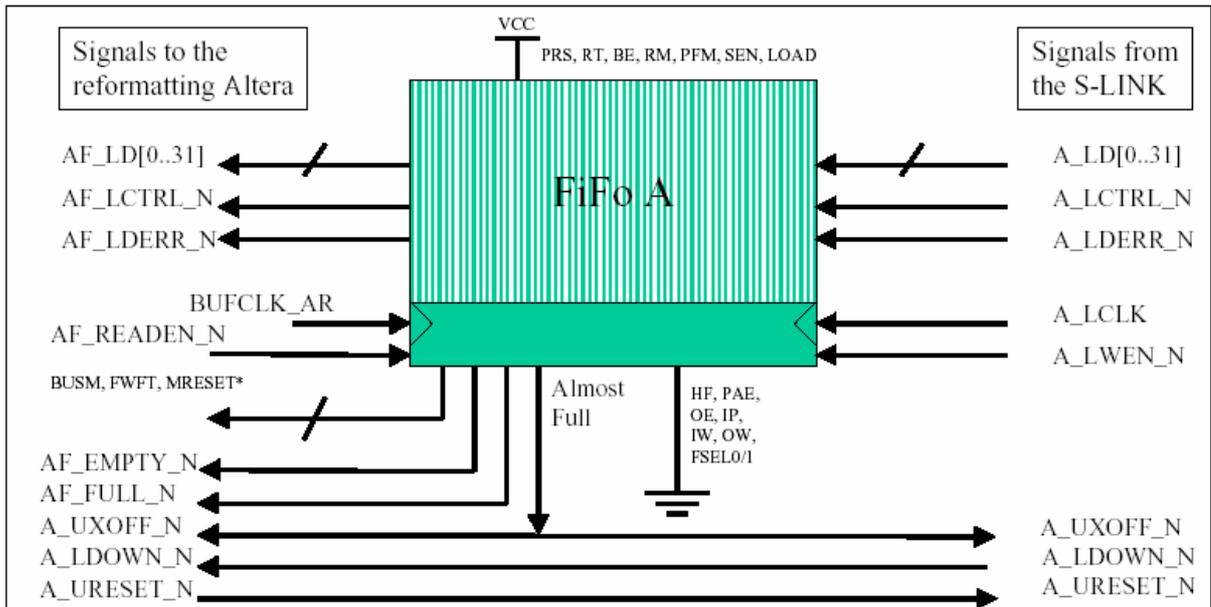


FIGURA 3.14 Señales entrada/salida LDC.

### 3.3.4 Bloque data multiplexing, control & filtering unit

Nos encontramos ante el bloque principal del diseño de esta FPGA; una vez los datos han entrado a la FPGA y han sido sincronizados, es aquí donde se realiza un control del CRC (Cyclic Redundancy Checks), el Flat Filtering y el Multiplexado de los datos para su posterior envío hacia la motherboard del ROD (conector J2A) o hacia la Auxiliary Altera FPGA que los enviará luego a la motherboard (conector J2B).

Lo primero que se produce es la comprobación del CRC, esto significa que del formato de datos de entrada tenemos una palabra de 32 bits que incluye un código representativo de cada TileDMU (Tile Data Management Unit) que debe coincidir con la que se genera internamente en la FPGA. Si se detecta cualquier error, se debe enviar un flag CRC ERROR WORD.

En el Anexo III están desarrollados en Código VHDL todos los algoritmos de programación vistos hasta ahora, así como todo el desarrollo del proyecto.

El formato de datos de TileCal [1] viene determinado por la siguiente figura.

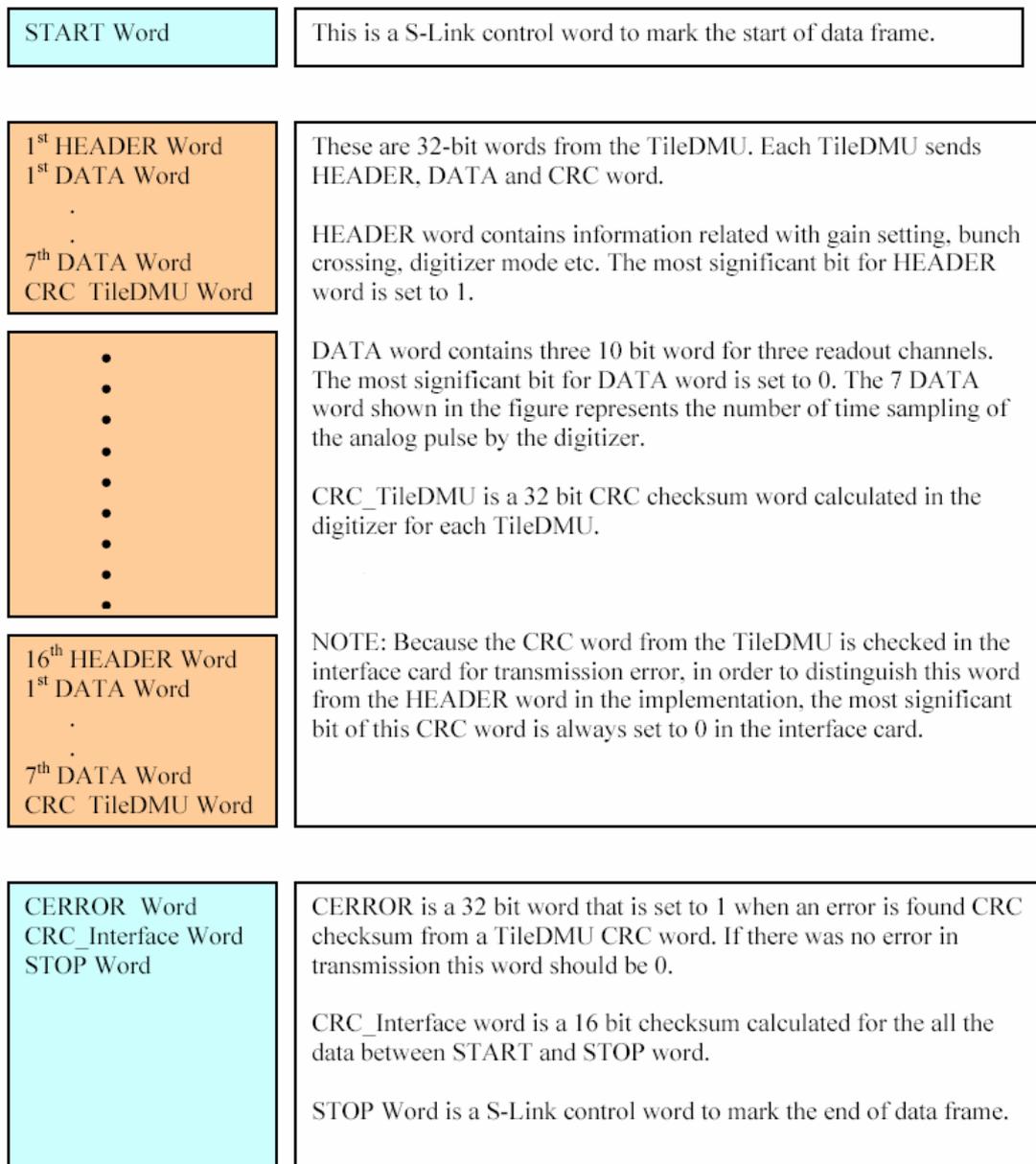


FIGURA 3.15 Formato de datos entrada TileCal.

Una vez comprobado el CRC, los datos son filtrados mediante el algoritmo Flat Filtering, una versión más sencilla del algoritmo Optimal Filtering que se implementa en las PUs del ROD.

Cada palabra de 32 bits de datos contiene tres canales de datos que debemos separar para sumar a las siete palabras. Con esos datos sumados tendremos la energía depositada en cada celda del detector, en total 3 celdas por cada evento recibido.

Las siguientes figuras representan el formato de un evento de TileCal y el formato de cada una de las palabras de datos.

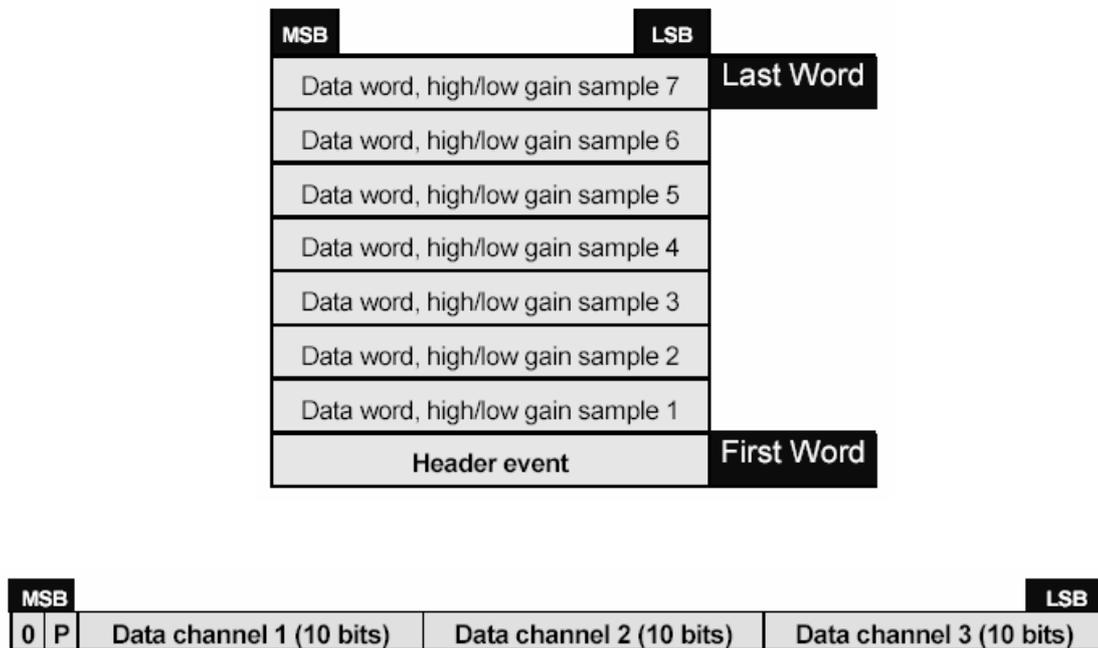


FIGURA 3.16 Formato de un fragmento de un evento y de tres canales de datos [12].

El algoritmo Flat Filtering sólo calcula la energía depositada, no tiene en cuenta el tiempo ni la forma del pulso. Para ello, antes del cálculo debemos normalizar los datos recibidos eliminando el pedestal mediante una sencilla fórmula. Con esos datos y con una constante de calibración ( $K_{FF}$ ) obtendremos el valor de la energía.

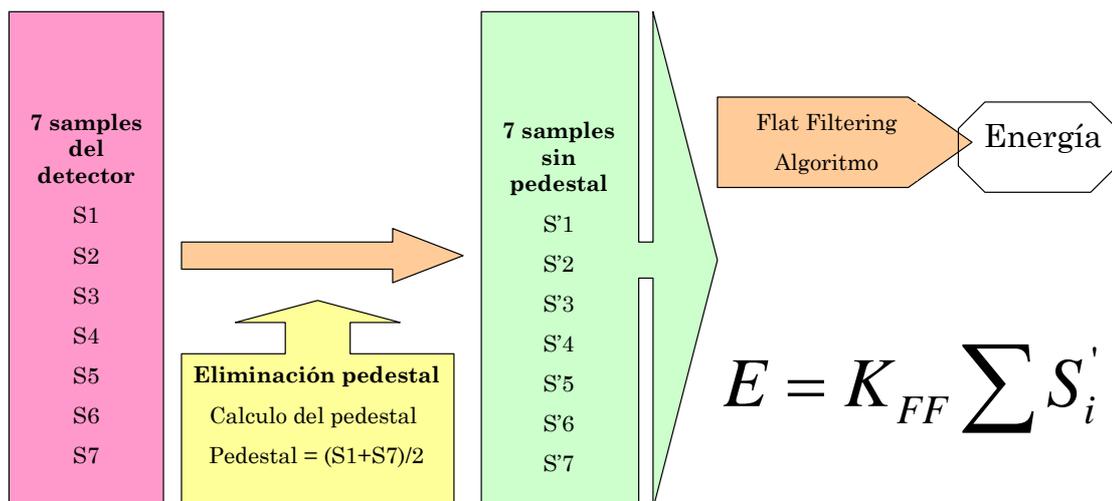


FIGURA 3.17 Algoritmo Flat Filtering.

Como paso final en el tratamiento de la información enviada por el detector debemos multiplexar los datos que recibimos en formato de 32 bits en dos palabras de 16 bits. Esto se debe a las especificaciones de entrada de las PUs del ROD.

El proceso de multiplexado se basa en el siguiente diagrama de bloques.

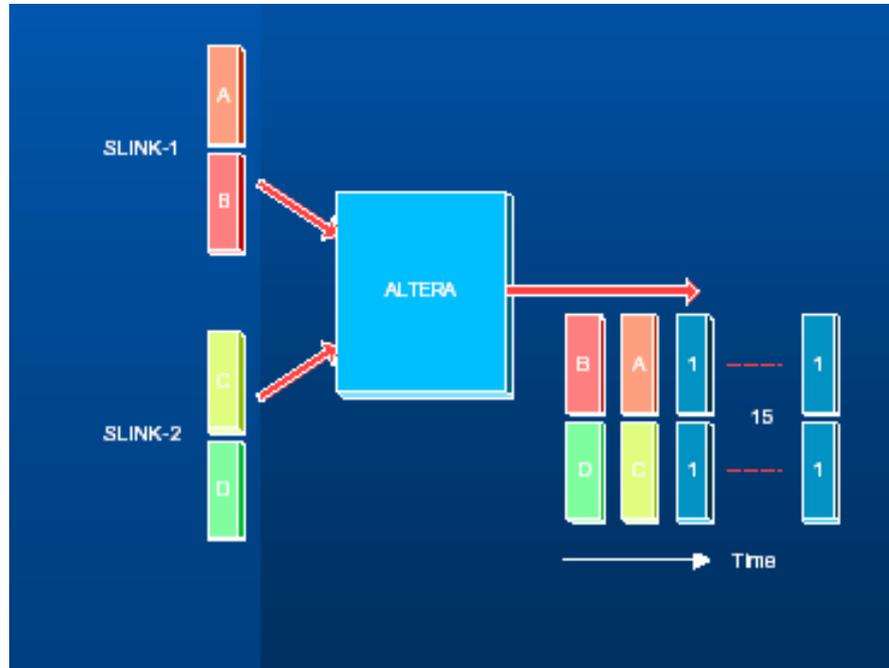


FIGURA 3.18 Multiplexado de datos.

Se puede observar como los 32 bits que se reciben en paralelo en la FPGA de cada uno de los conectores S-LINK, se convierten en bloques de 16 bits que se mandan sincronizados hacia la motherboard del ROD.

Una característica fundamental en el desarrollo del Módulo de Transición es la posibilidad de trabajar de forma independiente a la motherboard. Este modo de funcionamiento permite realizar todo el proceso en las FPGAs y enviar los datos directamente al siguiente nivel de trigger sin pasar por el proceso de la motherboard.

No se trata del modo correcto de funcionamiento, pero si de una forma real de trabajo que en caso de necesidades extremas siempre tendremos en consideración. Este modo ha sido usado dentro del proceso de test que se comentará en el siguiente capítulo en situaciones de fallo de la motherboard.

### 3.3.5 Bloque auxiliary FPGA

Es en este bloque donde se realiza el rutado de los datos recibidos en los conectores C y D del módulo hacia la Auxiliary Altera FPGA. Así mismo, nos puede servir para enviar todos los datos a dicha FPGA en el caso de estar funcionando de forma independiente a la motherboard.

Se trata de una conexión de gran utilidad, tanto para el test del módulo como para su funcionamiento real. No sólo presenta líneas de datos, también tenemos las líneas de control del protocolo S-LINK; pudiendo mantener la escalabilidad y determinación de errores tanto en funcionamiento como en desarrollo del proceso.

**3.3.6 Bloque reformatting FPGA control & test**

Bloque encargado de recibir el reloj del módulo y rutarlo a todos los componentes del sistema. Además presenta puntos de test que están colocados en la superficie del módulo para comprobar líneas que sean de nuestro interés.

**3.4 AUXILIARY ALTERA FPGA**

Al igual que ha ocurrido en el punto anterior, el diseño y funcionamiento de la Auxiliary Altera FPGA se hará siguiendo el diagrama de bloques siguiente.

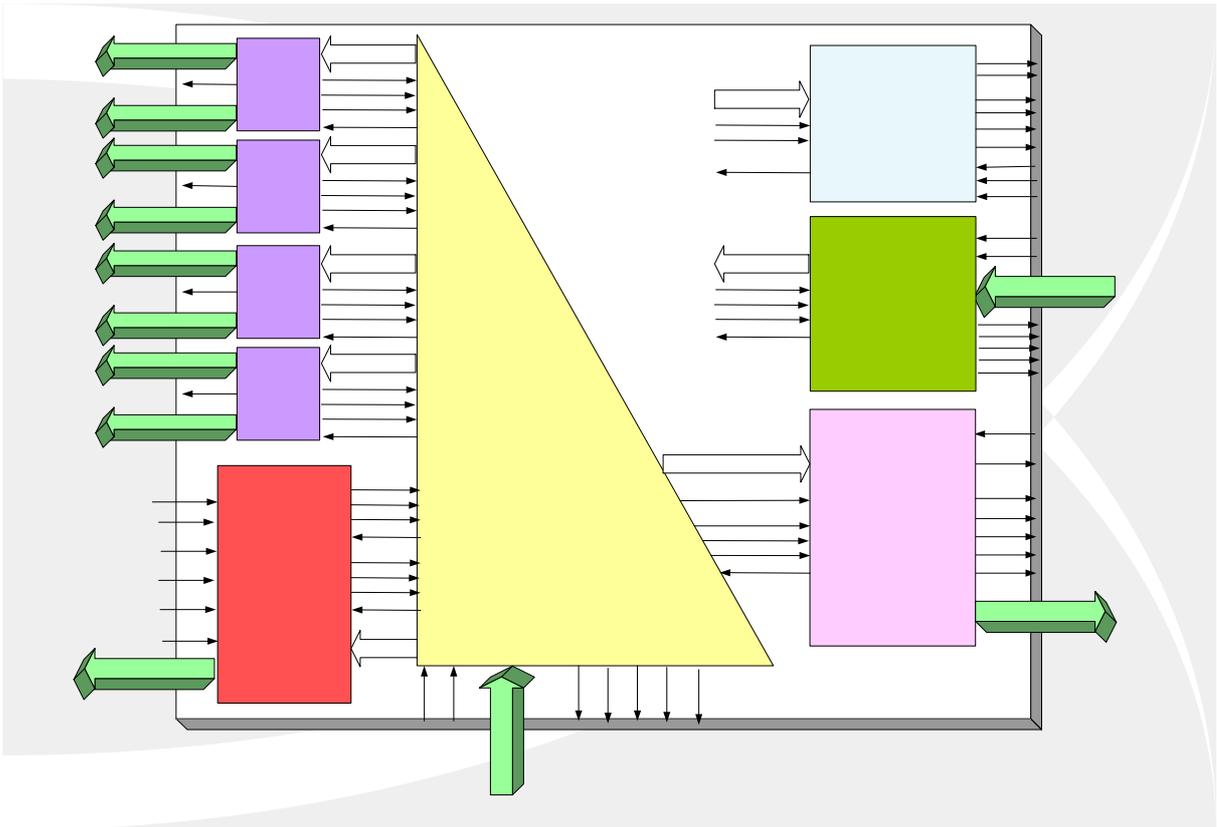


FIGURA 3.19 Diagrama de bloques Auxiliary Altera FPGA.

Las características de cada uno de los bloques que componen este diseño se desarrolla a continuación, explicando en detalle cada una de las funciones internas que se realizan.

**3.4.1 Bloque S-LINK**

Dedicado a controlar por parte del usuario el protocolo S-LINK de recepción de datos. Se pueden definir las características de lectura y de escritura de las FIFOs presentes en las tarjetas LDC que se están utilizando para la recepción de los datos.

A\_URL[0..3] S-LINK  
A\_UTDO\_N  
A\_UDW[0..1]  
B\_URL[0..3]  
B\_UTDO\_N S-LINK

### **3.4.2 Bloque reformatting FPGA**

Bloque encargado de la recepción de los datos que provienen de la Reformatting Altera FPGA. Capaz de generar señales de control y error en caso de fallo en la transmisión. Según el modo de funcionamiento en el que nos encontremos recibirá datos de los cuatro enlaces ópticos o de sólo dos de ellos.

### **3.4.3 Bloque J2B connection**

Conexión directa de esta FPGA con la motherboard del ROD. Dicho enlace se produce por el conector J2B, enviando los datos en dos bloques de 16 bits hacia las PUs. No se realiza ninguna función de multiplexado, al haberse realizado previamente en la Reformatting Altera FPGA.

### **3.4.4 Bloque data control unit**

Este bloque central realiza el rutado y la sincronización de los datos que se reciben en esta FPGA, tanto desde la motherboard como desde la otra FPGA. A su vez, se encarga de actuar sobre el protocolo S-LINK y de controlar todo el sistema.

### **3.4.5 Bloque auxiliary FPGA clock, control & test**

Realiza la función de recepción y rutado del reloj, de control y test de las líneas dedicadas a tal fin y de encender los LEDs que indican buena comunicación con el siguiente nivel de trigger, fallo en la comunicación o modo test.

### **3.4.6 Bloque ODIN LSC (G-LINK)**

Bloque dedicado a implementar las funciones de una tarjeta física ODIN LSC [10] basada en el protocolo S-LINK pero con chipset G-LINK de bajo consumo.

La tarjeta ODIN S-LINK es una tarjeta estándar con doble canal de salida de datos, en nuestro caso, se trata de una tarjeta integrada por completo en el Módulo de Transición y sólo testada por nosotros. Se realizó el diseño de esta manera por la necesidad de tener cuatro conectores para la entrada y no poder colocar tarjeta alguna para la salida.

Usa el chipset G-LINK HDMP-1032/34 y trabaja a una frecuencia de 40 MHz. Presenta una tasa de transferencia máxima de 160 Mbytes/s. Este dispositivo necesita de un oscilador en la placa y de un reloj de referencia para enganchar el flujo de datos que se producen.

En este apartado describiremos como se mapean los 32 bits que se reciben desde la motherboard y como se usan los comandos internos para la comunicación.

El protocolo G-LINK está definido en base al envío de palabras para encontrar la comunicación correcta entre el emisor y el receptor. Los 32 bits de entrada se almacenan en una FIFO de entrada cada flanco positivo del reloj UCLK. La lógica de rutado lee los datos en cada flanco positivo del reloj XCLK para ser comprobados los bits de paridad.

El siguiente paso es multiplexar los datos recibidos en bloques de 16 bits (del mismo modo que se hacía en la Reformatting Altera FPGA para los datos que se recibían y se enviaban hacia la motherboard).

Una vez realizados todos esos pasos, se codifican los datos con un CRC para ser enviados a través de los transmisores al siguiente nivel de procesado de datos.

El protocolo que se ha diseñado en QUARTUS II e implementado en esta FPGA sigue el siguiente diagrama de bloques [11].

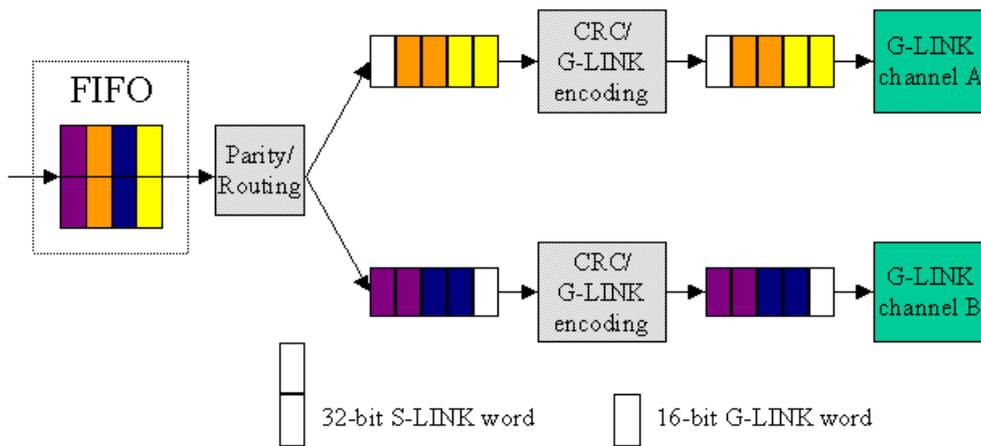


FIGURA 3.20 Diagrama de bloques del protocolo G-LINK.

Como se ha comentado, la comunicación se realiza gracias a un primer protocolo de reset que se realiza entre emisor y receptor, dicho protocolo nos sirve para encontrar errores si los hay entre ambos elementos. La máquina de estados de reset se puede ver en esta figura. Así mismo, podemos observar como se produce el cambio de estados que definen correctamente el protocolo que hemos definido.

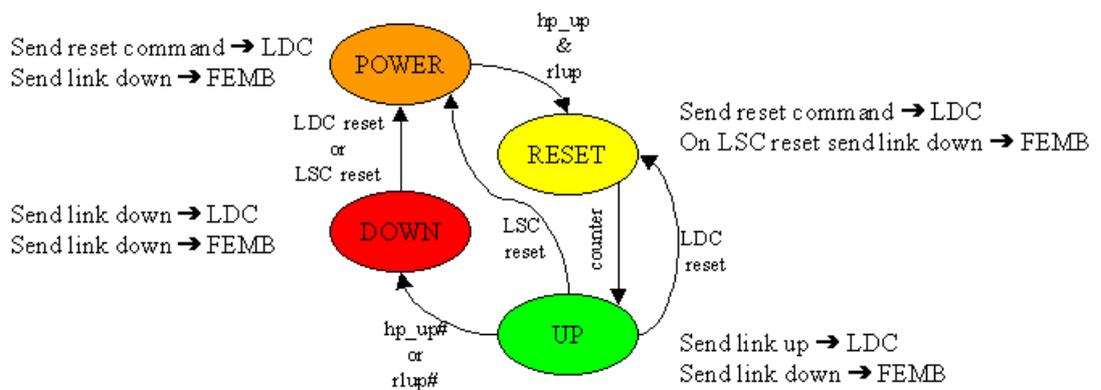


FIGURA 3.21 Máquina de estados del funcionamiento de G-LINK.

Cuando conectamos la alimentación, inmediatamente nos encontramos en el estado POWER. Si las fibras no estuvieran bien conectadas o algún reloj fallara y no se pudieran enganchar el transmisor y el receptor, pasaríamos al estado DOWN.

Si todo es correcto, se envía un comando de reset que comprueba que el enlace se está produciendo satisfactoriamente. En ese momento, pasamos al estado UP en el cual se permanece hasta el final de la comunicación o al aparecer un error.

La secuencia correcta de lo que acabamos de explicar viene definida en esta figura.

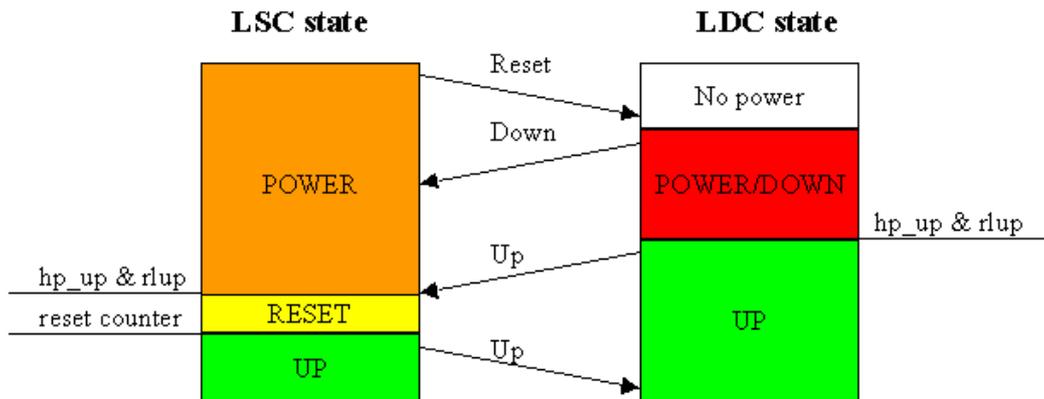


FIGURA 3.22 Secuencia POWER-UP de funcionamiento.

Con este bloque quedan definidas todas las operaciones que se realizan internamente en la Auxiliary Altera FPGA. El siguiente apartado nos definirá y mostrará el formato de los datos que se tratan en la salida del Módulo de Transición.

## 4. FORMATO DE DATOS DEL ROD

El formato de datos que presenta el ROD de TileCal [12] viene determinado por múltiples discusiones con la comunidad del detector. Se han tenido en cuenta consideraciones para decrementar al máximo el ancho de banda de entrada y salida de los enlaces ópticos. Siempre teniendo en mente perder la mínima cantidad de información y la máxima flexibilidad para futuros cambios en la estructura hardware.

Una vez descrita la forma con la cual se reciben y se envían los datos en el ROD, vamos a establecer en este apartado el formato exacto que le daremos internamente a dichos datos y el cual deberá ser interpretado correctamente por el siguiente nivel de adquisición.

### 4.1 ESTRUCTURA DE LOS DATOS DE SALIDA

La motherboard presenta cuatro PUs que procesan un determinado número de canales, dependiendo de la recepción que se realice en el Módulo de Transición. Cada PU tiene información sobre su propio canal, por lo que necesitamos dos formatos para los datos: un formato de dato individual para cada PU y un formato de dato global para la motherboard y todas las PUs.

El Formato de Datos del ROD será la unión entre los datos de las cuatro PUs y el formato del evento del DAQ-1. El Formato de Datos Intermedio será la salida de cada PU.

El Formato de Datos de Salida completo del ROD se puede observar en la siguiente figura.

Header	0	<b>Beginning of fragment (0x0000B0F)</b>			
	1	Start of header marker (0xEEEEEEEE)			
	1	Header size (0x20 in bytes)			
	1	Format version number			
	1	Source Identifier			
	1	Level 1 ID			
	1	Bunch crossing ID <sup>11</sup>			
	1	Level 1 Trigger Type			
	1	Detector Event Type			
	Detector Data	Status word	1	Nb of PU	PU Mask
Processing Unit 1		Nb. #0	Variable Block #0		
		Nb. #1	Variable Block #1		
		Nb. #2	Variable Block #2		
		Nb. #3	Variable Block #3		
		Nb. #4	Variable Block #4		
Processing Unit 2		Nb. #0	Variable Block #0		
		Nb. #1	Variable Block #1		
		Nb. #2	Variable Block #2		
		Nb. #3	Variable Block #3		
		Nb. #4	Variable Block #4		
Processing Unit 3		Nb. #0	Variable Block #0		
		Nb. #1	Variable Block #1		
		Nb. #2	Variable Block #2		
		Nb. #3	Variable Block #3		
		Nb. #4	Variable Block #4		
Processing Unit 4		Nb. #0	Variable Block #0		
		Nb. #1	Variable Block #1		
		Nb. #2	Variable Block #2		
		Nb. #3	Variable Block #3		
		Nb. #4	Variable Block #4		
Status Word		1	Status flag from Output Controller		
Trailer		1	Number of Status Elements		
		1	Number of Data Elements		
	1	Status Block Position			
	0	<b>End of Fragment (0x0E0F)</b>			

FIGURA 3.23 **Formato de los Datos de Salida del ROD.**

Pasamos a continuación a definir uno por uno todos los elementos presentes en el formato de datos expuesto anteriormente:

- Los bloques de las **Processing Units** están activados sólo si la PU correspondiente no está enmascarada por la lectura de los datos.
- El **Format Version Number** es un número entero de 32 bits que se carga vía VME al mismo tiempo que la configuración. Define la versión del ROD data fragment header, no la versión del formato de los datos del detector.
- El **Source Identifier** es una palabra que define el fragmento, queda subdividida de la siguiente forma.

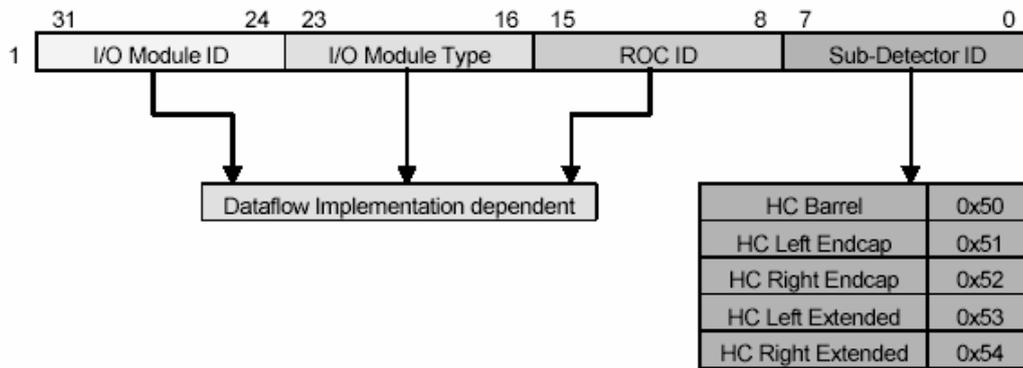


FIGURA 3.24 **Source Identifier Word.**

La implementación exacta de estas palabras dependerá de la organización de lectura/escritura del detector.

- El **I/O Module ID** y el **I/O Module Type** deben definirse para la implementación hardware final, hay suficientes bits para concretar todas las posibilidades que tengamos.
- El **Level 1 ID** es el identificador del evento generado por el sistema de trigger del nivel 1 (24 bits).
- El **Bunch Crossing ID** define el cruce de haz y se genera en el sistema de trigger del nivel 1 (12 bits).
- **Detector Event Type.** Este elemento identifica un evento el cual puede haber sido generado por un subdetector independiente de los otros subdetectores y de los sistemas de trigger de ATLAS.
- **PU Mask.** Se trata de un patrón de 8 bits que define que PU del ROD está leyendo los datos.
- **Detector Format Version Number.** Indica la versión del formato para el bloque de datos del detector.
- **Number of Data Elements.** Nos da la suma de los bloques del detector.
- **Status Block Position.** Se pone a cero para indicar que el Status Block precede a los datos.
- El **Status Flag from Output Controller.** Usado para marcar eventos en los cuales la lectura por medio de las PUs no ha sido correcta.

Con este apartado queda definida toda la estructura de los datos que se usan en el proceso de lectura/escritura en el Módulo de Transición. Así mismo, queda completada la distribución, diseño e implementación del dispositivo aquí estudiado.

## 5. BIBLIOGRAFÍA

- [1] Matricon, P., *The TM Module of the ATLAS Lar Calorimeter ROD System*, Atlas Internal Note, 2002.
- [2] ATLAS Collaboration, *TM4Plus1 Active S-LINK to VME64x Transition Module*, ECP Division. CERN, <http://hsi.web.cern.ch/HSI/s-link/devices/tm4plus1/>, 2000.
- [3] Van der Bij, E., McLaren, R., *The S-LINK Interface Specification*, ECP Division. CERN, 1997.
- [4] ATLAS Collaboration, *S-LINK Overview*, ECP Division. CERN, <http://hsi.web.cern.ch/HSI/s-link/introduc/overview.htm>, 1995.
- [5] ATLAS Collaboration, *ATLAS Tile Calorimeter S-Link Interface Card*, Atlas Internal Note, 2001.
- [6] Torroja, Y., *Diseño Lógico mediante FPGAs*, Universidad Politécnica Madrid, 2001.
- [7] Torres, J., González, V., *QUARTUS II: Una Herramienta para el Diseño Digital Avanzado*, Apuntes Curso Extensión Universitaria, 2002.
- [8] Castelo, J., Torres, J., González, V., *On the Developments of the Read Out Driver for the ATLAS Tile Calorimeter*, CERN/LHCC 2001-34, 2001.
- [9] Integrated Device Technology, *IDT72V3660 Datasheet*, FIFO Applications Guide, <http://www.idt.com/products/pages/FIFOs-72V3660.html>, 2003.
- [10] ATLAS Collaboration, *ODIN LSC*, ECP Division. CERN, <http://hsi.web.cern.ch/HSI/s-link/devices/odin/>, 2001.
- [11] ATLAS Collaboration, *ODIN S-LINK Hardware Specification*, ECP Division. CERN, <http://hsi.web.cern.ch/HSI/s-link/devices/odin/hwspec.html>, 2001.
- [12] Castelo, J., *The I/O Dataformat for the Tilecal Read Out System*, ROD TileCal Group Internal Note, 2003.