CAPÍTULO 3.- PROPUESTA DE ARQUITECTURA ABIERTA PARA LA ASIGNATURA DE ARQUITECTURA DE COMPUTADORAS.

3.1 INTRODUCCIÓN.

El procesador OpenSPARC T1 es el primer chip multiprocesador (CMT) que implementa totalmente el rendimiento de cualquier procesador actualmente. Es un gran procesador integrado en que se aplica la arquitectura SPARC de 64 bits. Este procesador está diseñado para aplicaciones comerciales tales como servidores de aplicación y servidores de bases de datos.

OpenSPARC T1 contiene ocho núcleos de procesador SPARC los cuales tienen soporte de hardware para cuatro subprocesos. Cada núcleo SPARC tiene una instrucción caché, una caché de datos, una instrucción totalmente asociativa y conversión de buffers. Los ocho núcleos de SPARC están conectados a través de una barra transversal a un chip de caché L2.

Los cuatro chips de memoria de acceso aleatorio dinámico (DRAM) controlan directamente la interfaz de datos. Además existe un chip controlador J-Bus que proporciona una interconexión entre las entradas y salidas del subsistema del procesador OpenSPARC T1.

Las características del procesador OpenSPARC T1 incluyen:

- 8 núcleos SPARC V9, con cuatro hilos por núcleo, por un total de 32 hilos.
- 132 Gbytes/seg de barra transversal de interconexión para la comunicación entre el chip.
- 16 Kbytes de caché de instrucción primaria por núcleo.
- 8 Kbytes de memoria caché de datos primarios por núcleo.
- 3 Mbytes de memoria caché secundaria.
- Sistema de interfaz serie (SSI) para el arranque PROM.
- Interfaz J-Bus para entrada y salida.

La figura 3.1-1 muestra un diagrama de bloques del procesador OpenSPARC T1 que ilustra las diversas interfaces y componentes integrados del chip.

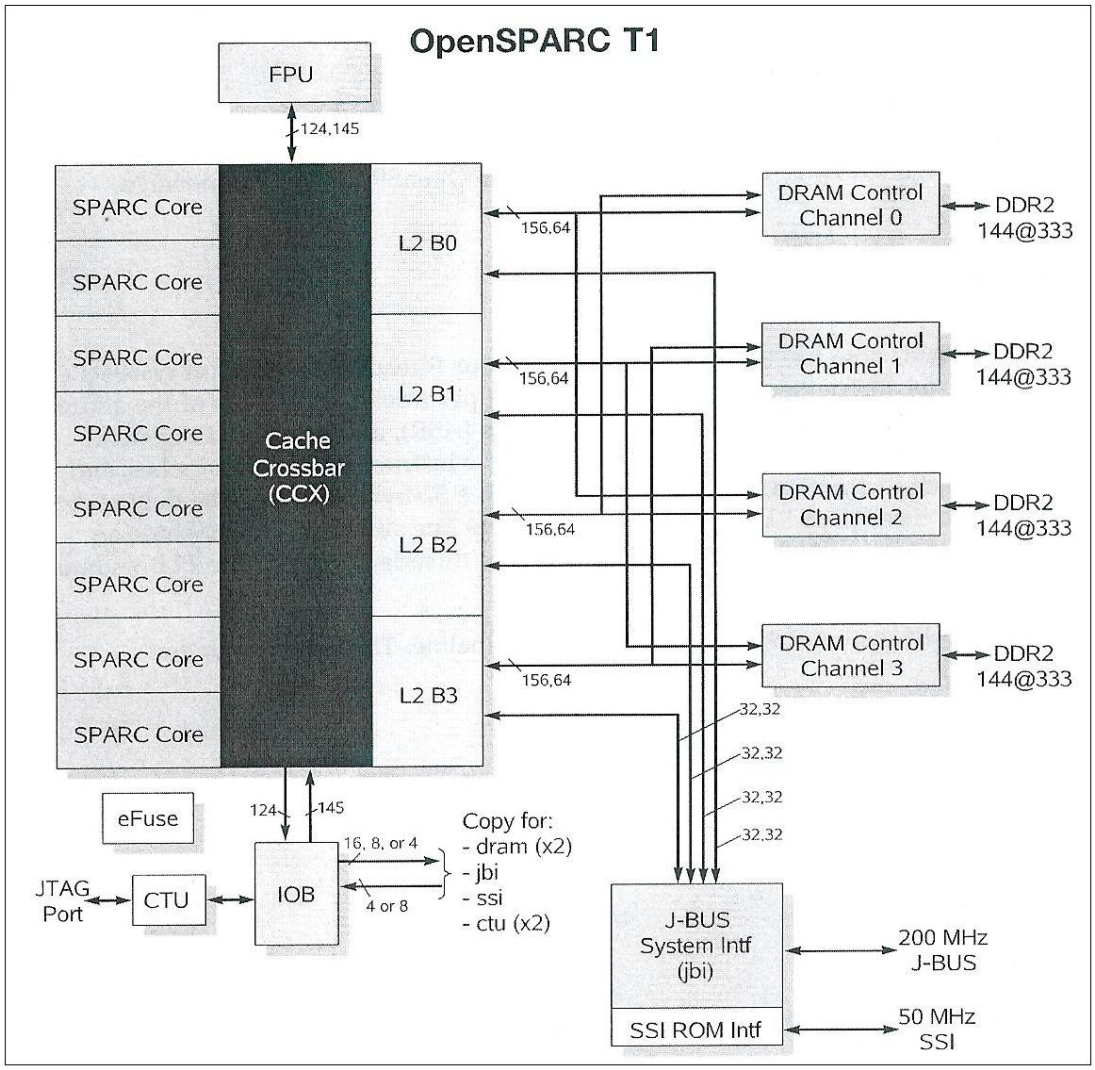

Figura 3.1-1 Diagrama de Bloques del Procesador OpenSPARC T1. Cortesía OpenSPARC.

Núcleo SPARC

Cada núcleo tiene soporte de hardware SPARC para cuatro hilos o subprocesos. Este apoyo consiste en un archivo de registro completo (con ocho ventanas de registros) por hilo, con la mayor parte de la dirección con identificadores de espacio, los registros auxiliares de estado y los registros privilegiados replicados por cada hilo. Los cuatro hilos comparten la instrucción, los caché de datos y los TLB. Cada caché de instrucciones es de 16Kbytes con un tamaño de línea de 32 bytes y sus datos se escriben a través de 8Kbytes.

Cada núcleo SPARC tiene una cuestión concreta, con seis etapas de tubería o pipeline. Estas seis etapas son:

1. Fetch.
2. Thread selection.

3. Decode.
4. Execute.
5. Memory.
6. Write Back.

La siguiente figura muestra la tubería del núcleo SPARC utilizada en el procesador OpenSPARC T1.

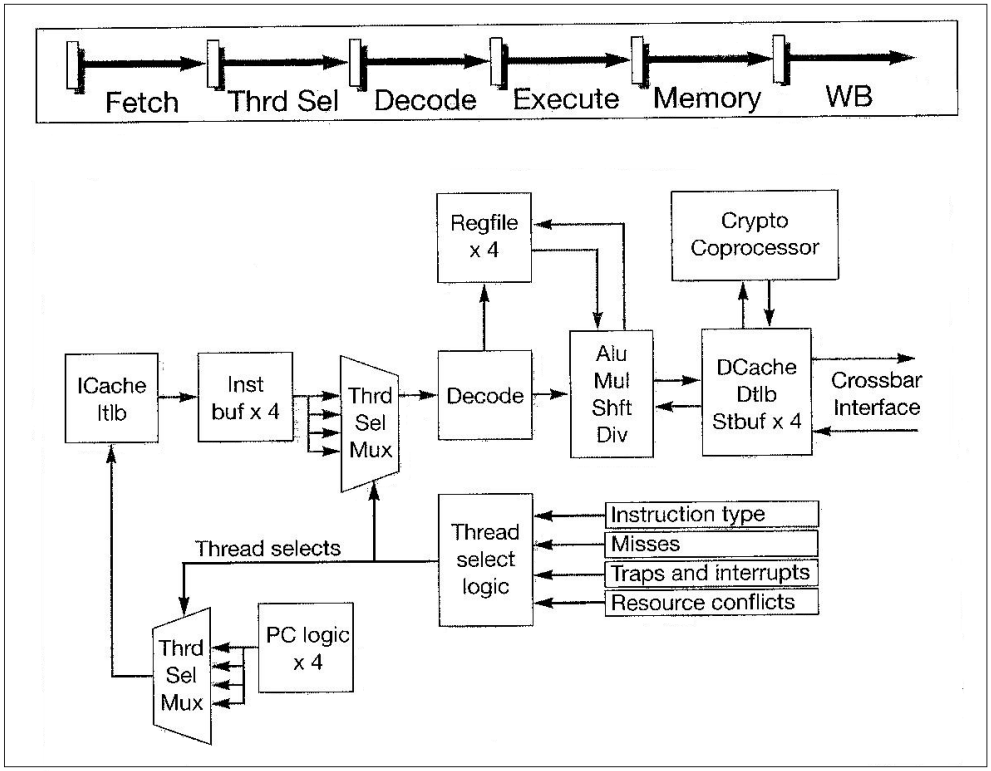

Figura 3.1-2 Tubería del Núcleo SPARC (Pipeline). Cortesía OpenSPARC T1.

Cada núcleo SPARC tiene las siguientes unidades:

1. Unidad de Instrucción de captura (IFU-Instruction Fetch Unit). Que incluye las etapas de tuberías de buscar, seleccionar, decodificar.
2. Unidad de ejecución (EXU-Execute Unit). Incluye la ejecución de la etapa de la tubería.
3. Unidad de carga y almacenamiento (LSU-Load/Store Unit). Incluye memoria de reescritura de etapas y un caché de datos complejos.
4. Unidad de trampa lógica (TLU-Trap Logic Unit). Incluye una trampa lógica y contadores de programa.
5. Unidad de procesamiento de flujo (SPU-Stream Processing Unit). Se utiliza para las funciones de la aritmética modular.
6. Unidad de manejo o gestión de memoria (MMU-Memory Management Unit).

7. Unidad de interfaz de punto flotante (FFU-Floating point Frontend Unit).

Barra Transversal Caché.

Los ocho núcleos de SPARC, los cuatro bancos de memoria caché L2, el puente de entrada y salida, y la FPU, todos ellos conectados a la barra transversal (CrossBar). La siguiente figura 3.1-3 muestra el diagrama de bloques de la barra transversal (CCX).

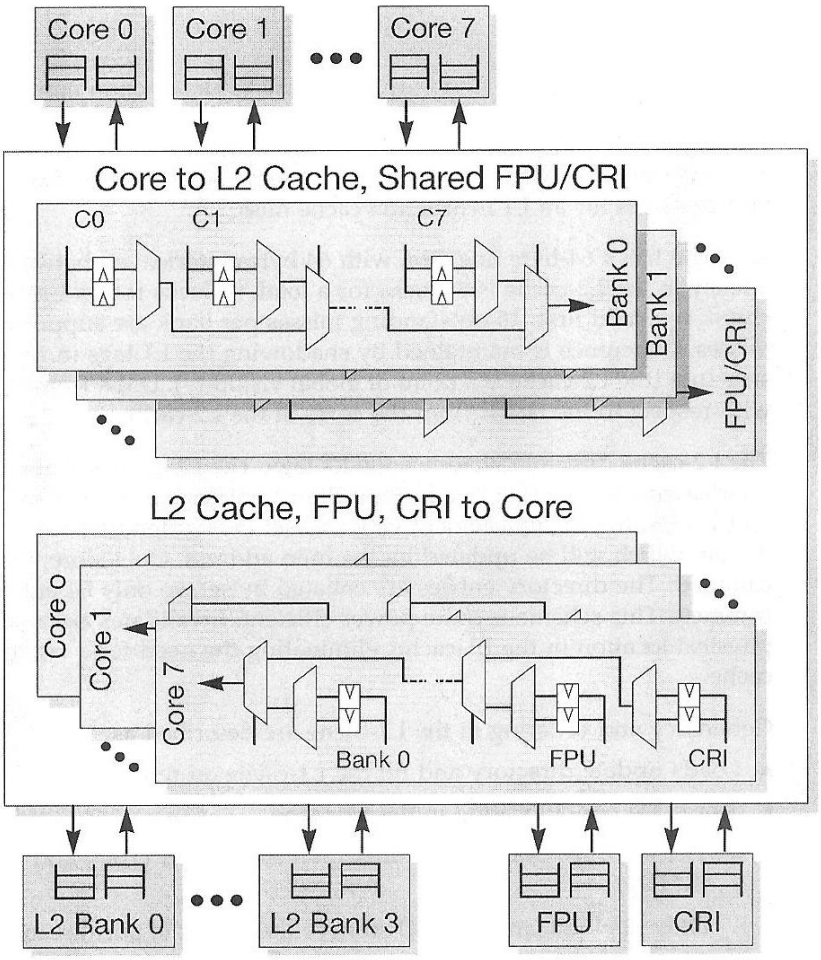

Figura 3.1-3 Diagrama de Bloques de la Barra Transversal CCX. Cortesía OpenSPARC T1.

Interfaz J-Bus.

La interfaz J-Bus (JBI) es la interconexión entre el procesador OpenSPARC T1 y el subsistema de entrada y salida. Tiene una velocidad de 200 MHz, 128 bits de ancho, direcciones multiplexadas, un bus de datos, y es utilizado sobre todo para el tráfico del acceso directo a memoria (DMA), además de una entrada y salida programables (PIO) utilizadas para su control.

La interfaz J-Bus es el bloque funcional de conexión de la recepción, así como responder a las solicitudes de DMA, realiza expediciones a los bancos L2, así como la expedición de las transacciones PIO en nombre de los hilos del procesador y el reenvío de respuestas.

Unidad de Carga y Almacenamiento (Load Store Unit-LSU).

Esta unidad procesa la memoria y hace referencia a los códigos de operación, tales como los diversos tipos de cargas, almacenamiento y descargas. Las interfaces de la LSU junto con su núcleo SPARC y sus unidades funcionales actúan como la puerta de enlace entre el núcleo y el CCX. A través de el CCX los datos de rutas de transferencia pueden ser establecidas con el subsistema de memoria y las entradas y salidas del subsistema (la transferencia de datos es realizada con paquetes).

La arquitectura de hilos de la LSU puede procesar cuatro cargas, cuatro almacenamientos, una búsqueda, operación de flujo, interrupción y un paquete de reenvío.

LSU Pipeline.

Existen cuatro etapas en la tubería LSU. La siguiente figura muestra sus diferentes etapas:

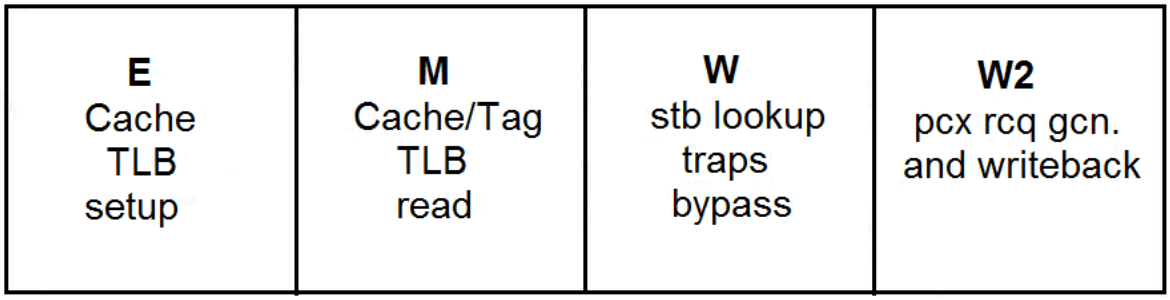

Figura 3.1-4 Tubería de la LSU. Cortesía OpenSPARC T1.

Flujo de Datos.

La LSU incluye un caché de 8 Kbytes la cual es una parte de caché de nivel 1 que comparten cuatro subprocesos. Se tiene un almacenamiento de buffer por hilo (STB).

Las cargas perdidas se mantienen en la LSM (Load miss queue) las cuales se comparten con otros subcódigos tales como los atómicos y los de búsqueda.

Los paquetes entrantes del CCX son ordenados y puestos en cola para la distribución de otras unidades a través del DFQ (Data Fill Queue). La DTLB es totalmente asociativa y responde hacia la traducción de direcciones. Así mismo las operaciones ASI son serializadas por la LSU y también secuenciadas por la cola

ASI hacia las unidades destino del chip. En la siguiente figura se puede visualizar el concepto de flujo de datos de la LSU (Figura 3.1-5).

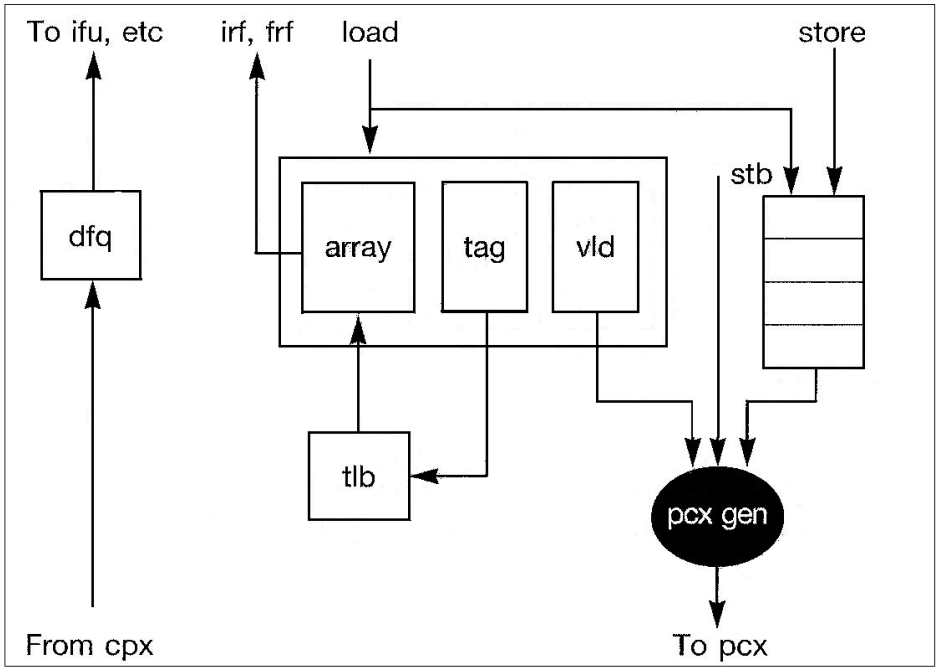

Figura 3.1-5 Concepto de Flujo de Datos de la LSU. Cortesía OpenSPARC T1.

Level 1 Data Cache (D-Cache).

Los 8Kbytes del nivel 1 D-cache tiene cuatro formas de conjunto asociativo y el tamaño de la línea es de 16 bytes. El D-Cache tiene un solo puerto de lectura y escritura para el arreglo de datos y etiquetas. El arreglo V-bit (Valid bit) está constituido por un doble puerto de lectura y un puerto para la escritura (1R/1W). El arreglo de bit válido se encarga de mantener el estado de la línea de cache válida o inválida.

Una carga cacheable carga o pierde la asignación de línea y ejecutará la escritura por almacenamiento. El almacenamiento no hace asignaciones y almacenamientos locales pueden actualizar la memoria caché L1 según lo determinado por la cache L2. Si se considera que no está presente en la cache L1, el almacenamiento local hará que las líneas sean detectadas como invalidas, y puede llegar a soportar simultáneamente hasta cuatro líneas de datos invalidas.

Data Translation Lookaside Buffer (DTLB).

El DTLB se encuentra en la TLB y tiene cuatro hilos (threads) los cuales son compartidos en la DTLB. Tiene un puerto CAM y otro puerto de lectura y escritura (1RW). La DTLB soporta las siguientes operaciones de 32bits de traducción de direcciones:

- VA->PA [virtual address to physical address].

- VA=PA [dirección de derivación para las operaciones de modo de supervisión].
- RA->PA [Real address to physical address].

La etiqueta y datos de TTE (Translation Table Entries) están protegidos y si hay errores serán imposibles de corregir. TTE accesa a los errores de paridad para cargar instrucciones con lo cual causará precisar la etiqueta.

Store Buffer.

La estructura física del Store buffer (STB) consiste en una regulación de almacenamiento CAM y un almacenamiento de un arreglo de datos (STBDATA). Cada hilo está asignado con ocho entradas fijas en una estructura de datos compartida. La SCM tiene un puerto CAM y un puerto RW, y la STBDATA tiene un puerto de lectura (1R) y otro de escritura (1W).

Todos los almacenes residen en el buffer de almacenamiento hasta que sean llevados a un almacenamiento total (TSO). El ciclo de vida de un almacenamiento TSO cumple con las siguientes etapas:

1. Validación.
2. Perpetración (expedido a cache L2).
3. Reconocimiento (cache L2 envía la respuesta).
4. Invalidación o Actualización.

El buffer de almacenamiento implementa parcial y totalmente la revisión de la lectura y después la escritura (RAW). Todos los datos RAW son devueltos a la lista de archivos por la tubería. Los éxitos parciales RAW forzarán la carga para acceder a la cache L2 mientras son enlazados con el almacenamiento y será entregado a CCX.

Data Fill Queue (DFQ).

Un núcleo SPARC se comunica con la memoria de entrada y salida mediante paquetes. Los paquetes destinados a un núcleo SPARC se ponen en cola en el DFQ. El DFQ mantiene una obligación predefinida de ordenar los requerimientos de todos los paquetes entrantes, y su objetivo principal es entregar los paquetes que incluyan la instrucción de búsqueda de unidad (IFU), LSU, la unidad de procesamiento de flujo (SPU).

3.2 UNIDAD DE EJECUCIÓN (EXU).

La unidad de ejecución contiene las siguientes subunidades:

- Unidad Aritmética y Lógica (ALU).
- Shifter (SHFT) o palanca de cambios.
- Integer Multuplier (IMUL) o multiplicador de Entero.

- Integer Divider (IDIV) o divisor de Entero.

A continuación se presenta un diagrama de nivel superior de la unidad de ejecución:

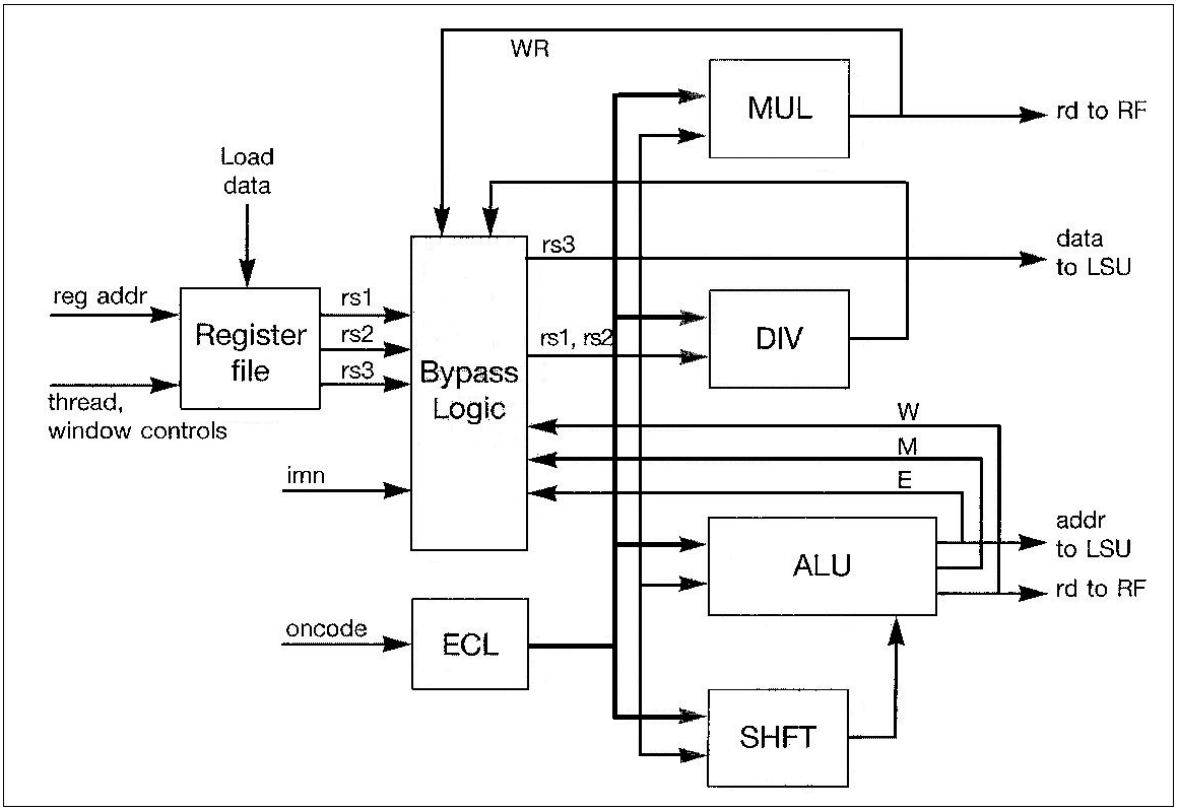

Figura 3.2-1 Diagrama de la Unidad de Ejecución. Cortesía OpenSPARC T1.

La unidad aritmética y lógica (ALU) está formada por una cadena de operaciones lógicas tales como: ADD, SUB, AND, NAND, OR, NOR, XOR, XNOR y NOT. La ALU es también usada en cálculos de direcciones virtuales. La Figura 2-17 ilustra el diagrama de bloques de la ALU.

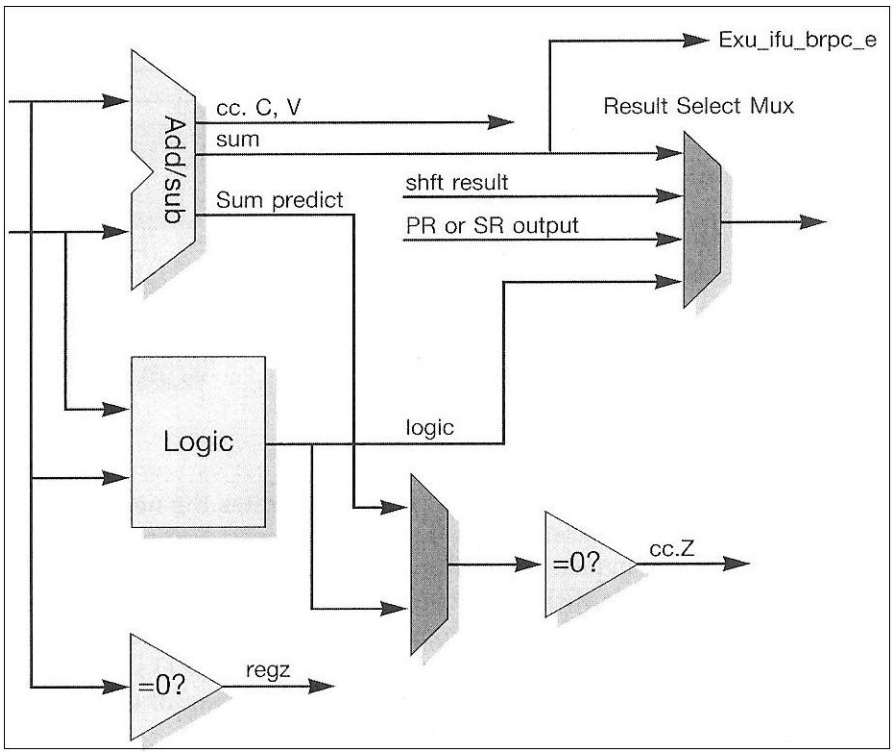

Figura 3.2-2 Diagrama de Bloques de la ALU. Cortesía OpenSPARC T1.

La subunidad Shifter (SHFT) aplica de 0 a 63 bits por cambios o turnos. En la figura siguiente se muestra el diagrama de bloques del SHFT. Figura 3.2-3

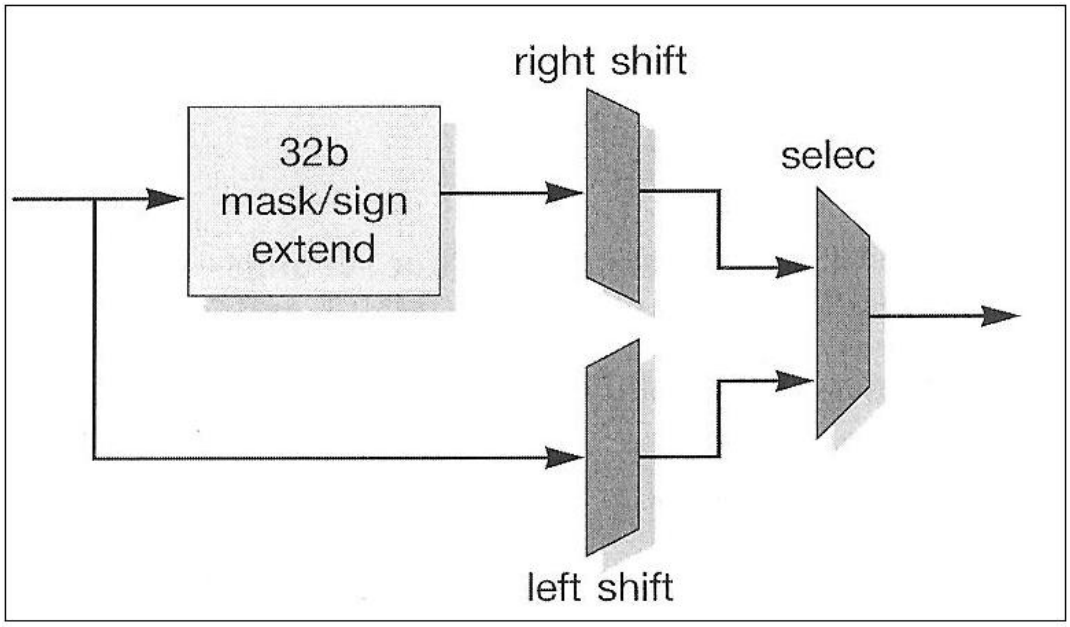

Figura 3.2-3 Diagrama de Bloques de "Shifter". Cortesía OpenSPARC T1.

3.3 UNIDAD DE INTERFAZ PUNTO FLOTANTE (FLOATING POINT FRONTENED UNIT-FFU).

La unidad de interfaz de punto flotante (FFU) es la responsable de la ordenación de las operaciones de punto flotante (OPS FP) hacia la unidad de punto flotante (FPU) a través de la LSU, así como la ejecución de operaciones sencillas e instrucciones. La FFU también mantiene el registro del estado del punto flotante (FSR) y los registros de estado gráficos (GSR). Sólo puede haber una instrucción pendiente en la FFU a la vez.

La FFU está compuesta por cuatro bloques:

- Los archivos de registro de punto flotante (FFU_FRF).
- El bloque de control (FFU_CTL).
- El bloque del Data-Path (FFU_DP).
- El bloque de ejecución VIS (FFU_VIS).

La figura 3.3-1 muestra el diagrama de bloques de la FFU ilustrando sus cuatro subbloques:

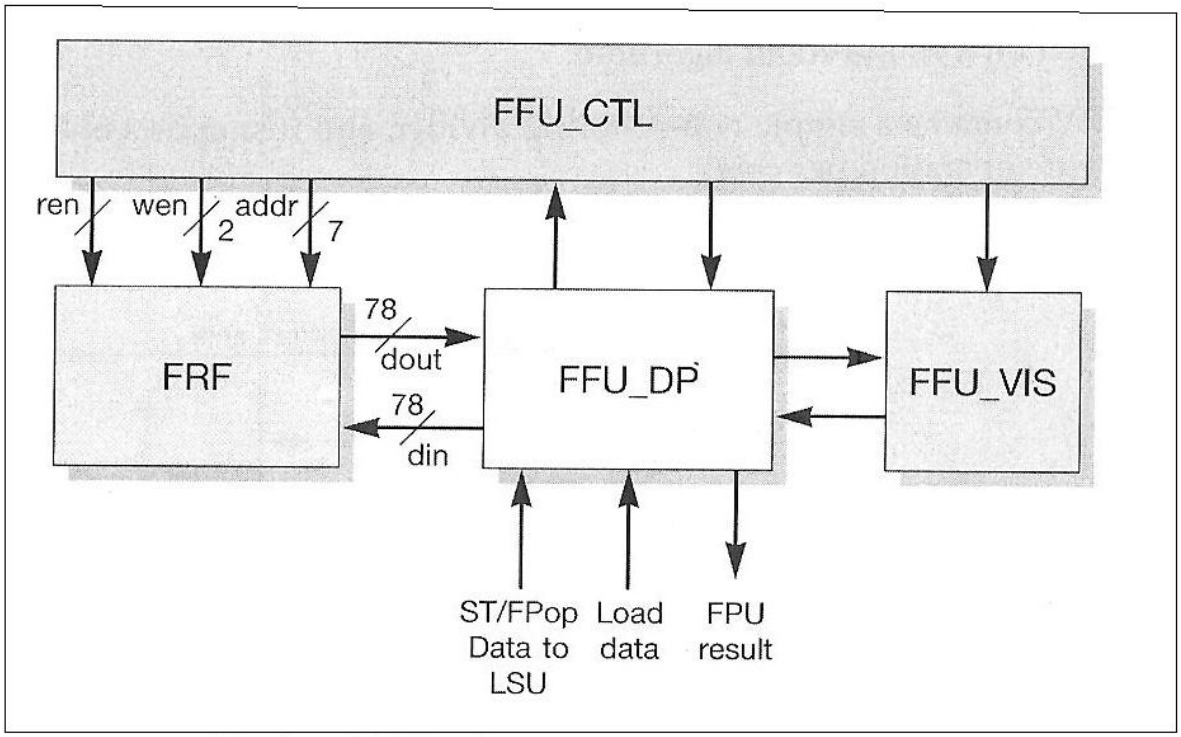

Figura 3.3-1 Diagrama de Bloques de la FFU. Cortesía OpenSPARC T1.

Archivos de registro de Punto Flotante.

El archivo de registro de punto flotante (FRF) dispone de 128 entradas de 64Bits de datos, además de 14Bits de ECC. El puerto de escritura está habilitado para la

mitad de los datos y se manejan Bits para la longitud de palabra, puede ser inferior o superior.

Bloque de Control (FFU_CTL).

El bloque de control FFU implementa el control lógico de la FFU y genera que el multiplexor seleccione señales adecuadas del Data-Path. El control de la FFU también decodifica el fp_opcode y contiene el estado de máquina para la tubería FFU. También genera las trampas de FP y elimina señales, así como la señalización de la LSU cuando los datos están listos para ser procesados.

Bloque del Data-Path (FFU_DP).

Este bloque se compone de multiplexores y datos que han sido leídos o que están a punto de escribirse en el FRF. El FFU_DP opera los datos para el STF, hace implementaciones como FMOV, FABS, etc. Realiza revisiones al ECC para la lectura de datos desde la FRF y genera que en la ECC los datos estén escritos en la FRF.

Bloque de Ejecución VIS (FFU_VIS).

En este bloque se realizan implementaciones a un subconjunto de instrucciones gráficas VIS, incluyendo la adición y sustracción así como operaciones lógicas. Todas las operaciones se ejecutan en un solo ciclo, y las entradas y salidas de datos están conectadas a la FFU_DP.

3.4 UNIDAD MULTIPLICADORA (MUL).

La unidad multiplicadora SPARC (MUL) realiza la multiplicación de dos entradas de 64-Bits. La MUL es compartida entre la EXU y la SPU y tiene un bloque de control y ruta de acceso a datos por categorías. A continuación se ilustra cómo está el multiplicador conectado a otros bloques funcionales:

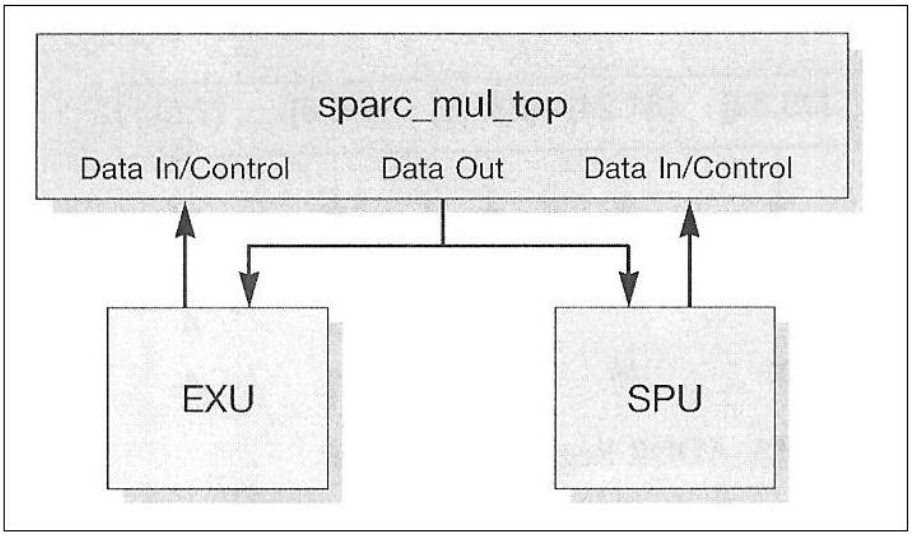

Figura 3.4-1 Diagrama de Bloques de Multiplexor (MUL). Cortesía OpenSPARC T1.

3.5 UNIDAD DE PROCESAMIENTO DE FLUJO (SPU).

Cada núcleo SPARC está equipado con una unidad de procesamiento de flujo (SPU) la cual realiza el soporte de operaciones asimétricas de hasta un tamaño de clave de 2048-Bits.

La SPU comparte el entero multiplicador con la unidad de ejecución (EXU) para las operaciones aritméticas (MA). Mientras la SPU se comparte entre todos los hilos del núcleo SPARC, sólo un hilo puede utilizar la SPU a la vez. La operación SPU es configurada por un almacenamiento de un hilo al control de registro y luego regresa al procesamiento normal, así, la SPU iniciará la carga de flujo o el almacenamiento de operaciones al nivel caché L2 y calcular operaciones para el entero multiplicador. Una vez que la operación se puso en marcha puede operar paralelamente a la ejecución de instrucciones del núcleo SPARC.

Registros ASI de la SPU.

Todos los identificadores alternativos de espacio (ASI) se registran en el SPU y tienen una longitud de 8 bytes. El acceso a todos los registros ASI de la SPU cuentan con supervisión, y sólo se puede acceder en modo supervisión. En la siguiente lista se destacan los registros de la ASI:

1. El registro de la dirección física de aritmética modular (MPA).

Este registro lleva la dirección física utilizada para acceder a la memoria principal.

2. El registro de la dirección de memoria (MA_ADDR).

Este registro lleva la dirección de memoria para la compensación de varios operandos y el tamaño del exponente.

En la siguiente figura se muestra la disposición de los campos de bits.

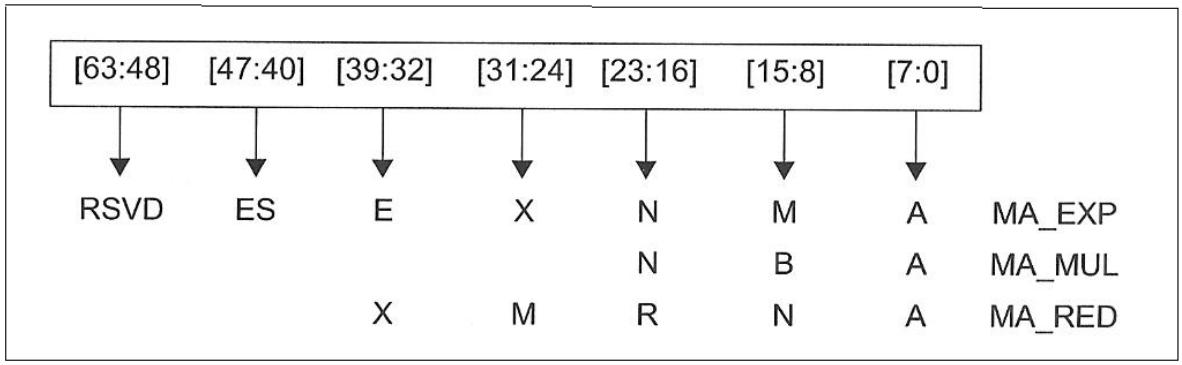

Figura 3.5-1 Registro de Campos de Bits MA_ADDR. Cortesía OpenSPARC T1.

Flujo de Datos de las Operaciones Aritméticas Modulares.

Se ilustran a continuación el flujo de datos de las operaciones aritméticas:

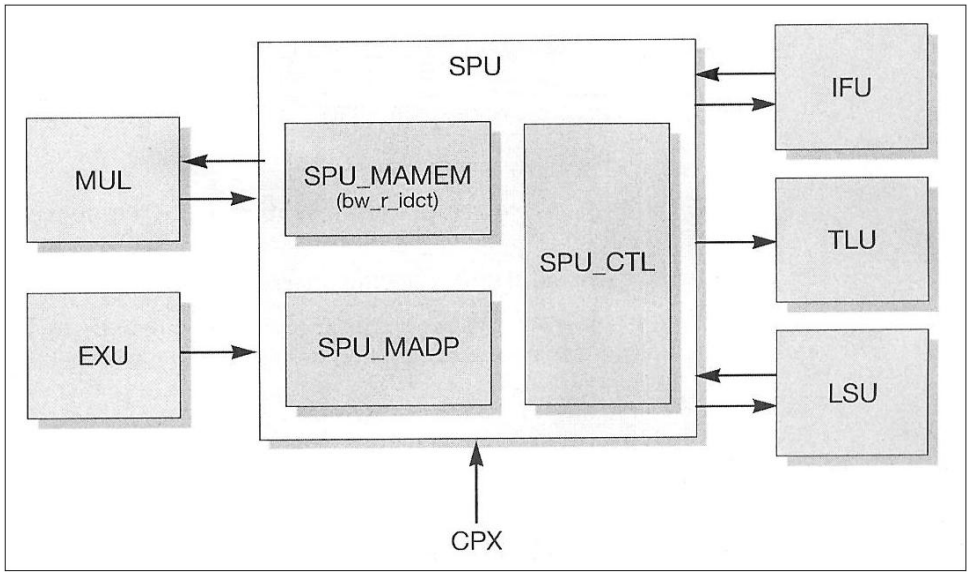

Figura 3.5-2 Diagrama de Bloques del Flujo de Datos de las Operaciones Modulares. Cortesía OpenSPARC T1.

Memoria Modular Aritmética (MA Memory).

Con un total de 1280 bytes de memoria local y un solo puerto de lectura y escritura (1RW) que es utilizado para el suministro de operandos de la Aritmética modular. La memoria MA alberga 5 operandos de 32 palabras cada uno, que admitirá un tamaño máximo de 2048 bits y está protegida con 2 bits por cada palabra de 64 bits.

La memoria MA requiere una inicialización de software antes del comienzo de las operaciones realizadas en ella. Tres operaciones MA_LD están obligadas a iniciar

las 160 palabras de la memoria ya que el campo de longitud MA_CTL permite un máximo de 64 palabras para ser cargadas en la memoria MA.

Los accesos escritos en la memoria MA pueden estar en los límites de 16 bytes o de 8 bytes. Para los accesos de lectura en la memoria MA, se debe estar en los 8 bytes.

Operaciones Modulares Aritméticas (MA ops).

Todos los registros de aritmética modular se deben iniciar antes de lanzar un sistema de operación modular aritmética. Las operaciones de la aritmética modular (MA ops) comienzan con una *"stxa"* hacia el registro de la MA_CTL si el buffer almacenado para ese hilo se encuentra vacío. De lo contrario los hilos esperarán hasta que el buffer esté vacío antes de ser enviado el stx_ack a la LSU. Una operación MA que esté en proceso, puede ser abortada por otro hilo por medio de *"stx"* hacia el registro MA_CTL.

La figura 3.5-3 muestra las operaciones MA usando un diagrama de estado de transición.

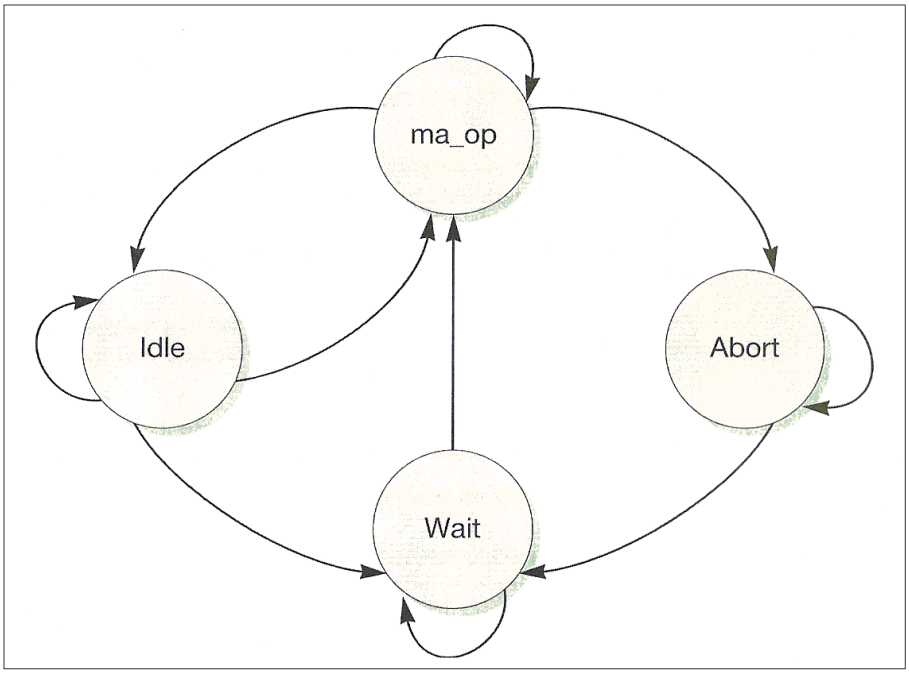

Figura 3.5-3 Diagrama de Estado de Transición de Operaciones MA. Cortesía OpenSPARC T1.

El estado de transición está mostrado por las siguientes ecuaciones:

```
tr2_maop_frm_idle = cur_idle & stxa_2ctlreq & ~wait_4stb_empty & ~wait_4trapack_set;
tr2_abort_frm_maop = cur_maop & stxa_2ctlreg;
```

```
tr2_wait_frm_abort = cur_abort & ma_op_complete;
tr2_maop_frm_wait = cur_wait & ~(stxa_2ctlreg | wait_4stb_empty | wait_4trapack_set);
tr2_idl_frm_maop = cur_maop & ~stxa_2cltreg & ma_op_complete;
tr2_wait_frm_idle = cur_idle & stxa_2ctlreg & (wait_4stb_empty | wait_4trapack_set);
```

La operación MA_ST se inicia con un *"stxa"* hacia el código de registro MA_CTL que es igual a MA_ST y su longitud de campo especifica el número de palabras a enviar al nivel cache L2. La SPU envía al procesador una memoria interfaz cache (PCX) solicitar la LSU, así mismo espera una confirmación de la LSU antes de enviar otra petición. Cuando es necesario guarda los reconocimientos que son devueltos desde la cache L2 a la interfaz del procesador (PCX), y se irá a la LSU con el fin de invalidar el nivel cache L1. Posteriormente el LSU hará llegar al SPU un acuse de recibo, y la SPU realizará un decremento a un contador local y espera a que todos los almacenamientos sean enviados para ser reconocidos y la transición al estado realizado.

3.6 UNIDAD DE MANEJO DE LA MEMORIA (MMU).

La MMU mantiene el contenido de la instrucción del buffer de transacción y del buffer de datos. El buffer de transacción (ITLB) reside en la unidad de búsqueda de instrucción (IFU) y el buffer de transacción de datos (DTLB) reside en la unidad de carga y almacenamiento (LSU). A continuación se ilustra la relación entre la MMU y el TLB.

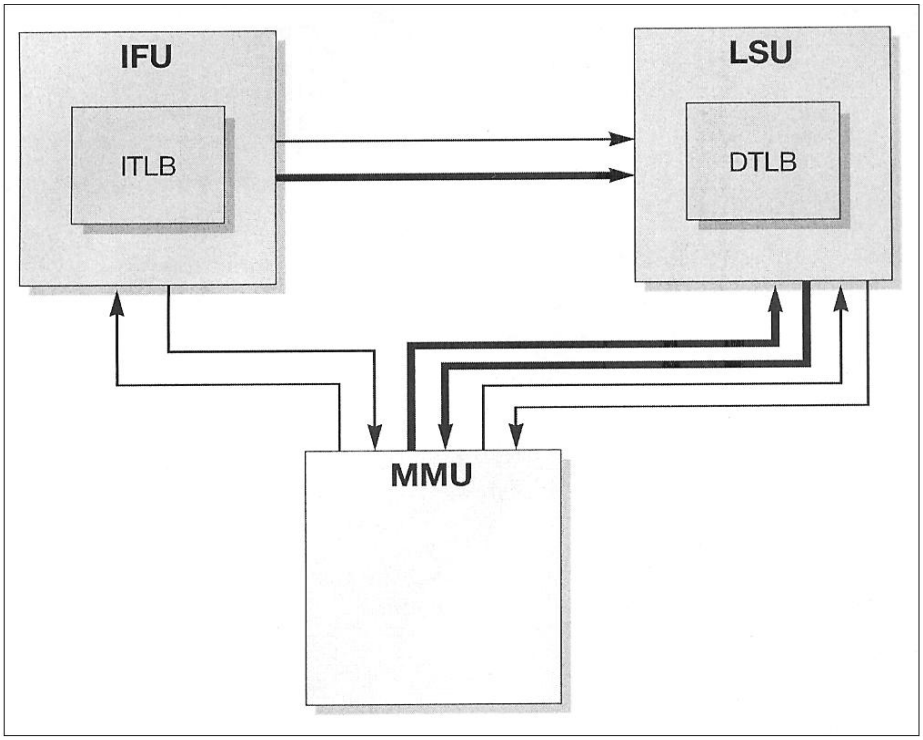

Figura 3.6-1 Relación entre MMU y los TLB´s. Cortesía OpenSPARC T1.

Estructura del Buffer de Traducción.

El buffer de traducción (TLB) consiste en un contenido de memoria direccionable (CAM) y una memoria direccionable aleatoria (RAM). CAM tiene un puerto de comparación y un puerto de lectura y escritura (1C1RW), y la RAM tiene sólo un puerto de lectura y escritura (1RW). El TLB soporta los siguientes eventos:

1. CAM.
2. Lectura.
3. Escritura.
4. Bypass o derivación.
5. Demap.
6. Reset suave.
7. Reset duro.

CAM está compuesto por el siguiente campo de bits, un identificador de partición ID (PID), un identificador de contexto ID (CTXT) y una dirección virtual (VA). La dirección virtual se subdivide en el campo del tamaño de página. El campo CTXT también tiene su propia habilitación para su flexibilidad al estar implementada. La porción de los campos CAM son para efectos de comparación. En cuanto a la RAM, es constituida por los siguientes bits; la dirección física (AF) y atributos. La porción de los campos RAM están únicamente para propósitos de lectura, donde la lectura puede ser causada por un software de lectura o una lectura basada en CAM.

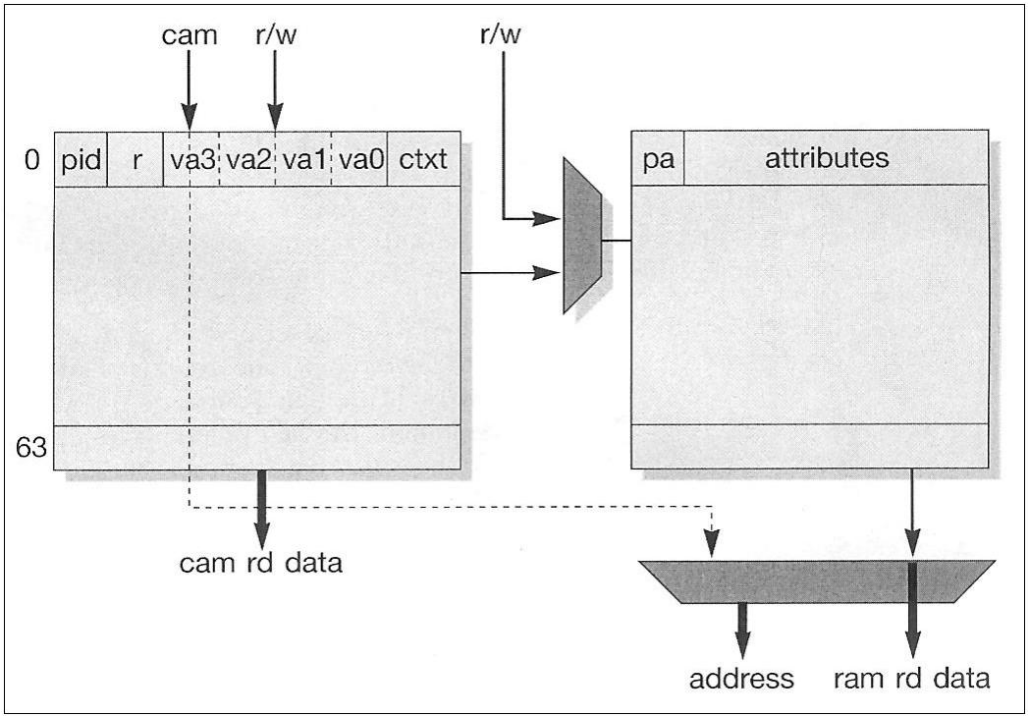

Figura 3.6-2 Estructura del TLB. Cortesía OpenSPARC T1.

Especificaciones de Acceso de Escritura al TLB.

Un stxa hacia la entrada de datos o acceso de datos hace que una operación de escritura sea asíncrona al flujo de la tubería (pipe). Las peticiones de escritura son originadas por la entrada cuatro del FIFO en la LSU. La LSU pasará la petición de escritura hacia la MMU la cual la reenviará a la ITLB (Instruction Translation Lookaside Buffer) o a la DTLB.

Especificaciones de Acceso de Lectura al TLB.

El TLB lee las operaciones siguiendo un protocolo tal y como lo hace en la escritura de operaciones. El acceso de datos ASI hará la lectura en la parte RAM, entonces el TLB leerá datos y serán regresados a través de un direccionamiento en la LSU. Si no existe error de paridad, la LSU reenviará los datos.

3.7 TRAP LOGIC UNIT (TLU).

Contiene seis niveles de trampa, y una trampa puede estar en uno de los siguientes cuatro modos:

- Modo Reset-error-debug (RED).
- Modo hipervisor (HV).
- Modo supervisor (SV).
- Modo usuario.

Las trampas harán en la tubería del núcleo SPARC una limpieza y se producirá un Switch de hilo para que sea producido hasta que el vector de trampa haya sido resuelto.

Interrupciones de software se entregan a cada uno de los núcleos virtuales utilizando el nivel de interrupción a través del registro SOFTINT_REG. Interrupciones de entrada y salida son llevadas a cada núcleo virtual usando el vector de interrupción (interrupt_vector). Se pueden tener en cola hasta 64 interrupciones pendientes por hilo o subproceso, uno por cada vector de interrupción. Los vectores de interrupción tienen prioridad, así cada fuente de interrupción de entrada/salida tiene un número de cableado de interrupción que se utiliza como vector de interrupción por el puente de bloqueo de entrada/salida.

El TLU se encuentra en una posición lógicamente central para recoger todas las trampas y las interrupciones para después ser reenviadas. La siguiente figura ilustra el papel de la TLU respecto a los demás bloques en un núcleo SPARC.

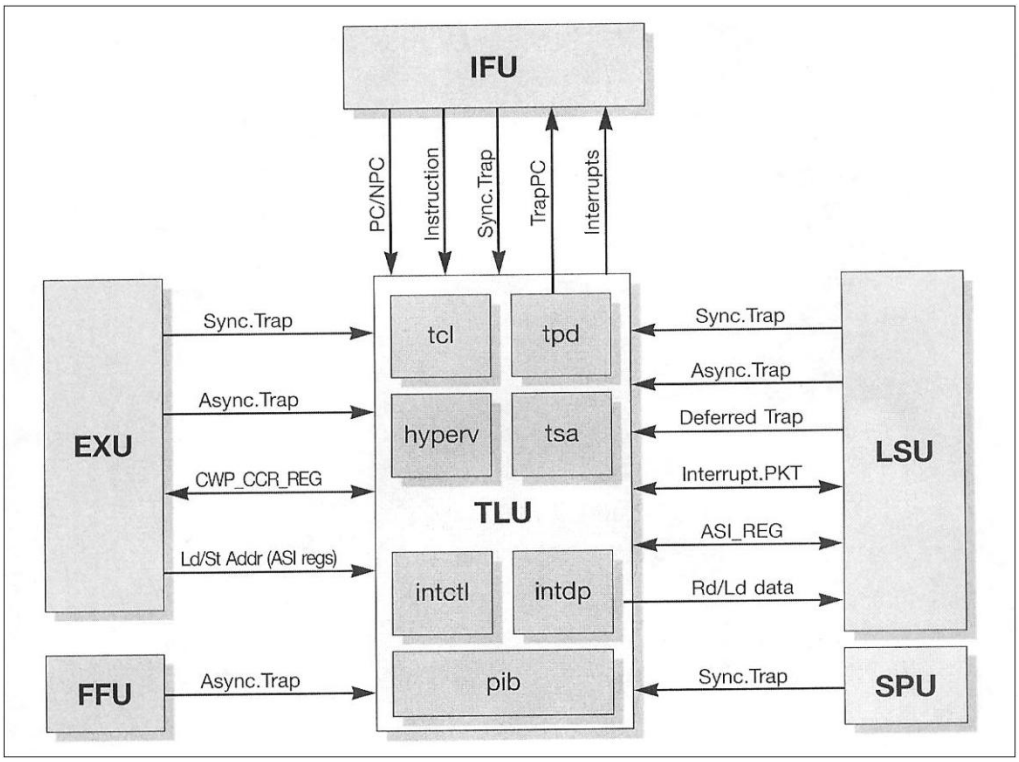

Figura 3.7-1 Diagrama de la Función TLU respecto a los demás Bloques en un Núcleo SPARC. Cortesía OpenSPARC T1.

La siguiente lista destaca las funciones de la TLU:

- Recoge las trampas de todas las unidades del núcleo SPARC.
- Detecta algunos tipos de trampas internas de la TLU.
- Resuelve la trampa con prioridad y genera el vector trampa.
- Envía el lavado de tubería hacia otras unidades SPARC mediante un conjunto de trampas LSU.
- Mantiene registros de estado del procesador.
- Administra la trampa de pila.
- Restaura el estado del procesador de la trampa de la pila y retira instrucciones.
- Implementa un hilo (thread) interno para la entrega.
- Recibe y procesa todos los tipos de interrupciones.
- Mantiene y compara las marcas y los registros relacionados SOFTINT.
- Genera un temporizador de interrupciones y las interrupciones de software (interrupt_level-n type).
- Mantiene los contadores de rendimiento de instrumentos (PIC).

Arquitectura de los Registros de la TLU.

A continuación se mencionan los registros que mantienen la TLU:

1. Estado del procesador y registros de control.
- Registro PSTATE (processor state).
- Registro TL (Trap level).
- Registro GL (global register window level).
- Registro PIL (Processor interrupt level).
- Registro TBA (Trap base address).
- Registro HPSTATE (Hypervisor processor state).
- Registro HTBA (Hypervisor trap base address).
- Registro HINTP (Hypervisor interrupt pending).
- Registro HSTICK_CMPR_REG (Hypervisor system tick compare).

2. Trampa de pila.
- Registro TPC (trap PC).
- Registro TNPC (Trap next PC).
- Registro TTYPE (Trap type).
- Registro TSTATE (Trap state).
- Registro HTSTATE (Hypervisor trap state).

3. Registros auxiliaries de estado.
- Registro TICK_REG (tick).
- Registro STICK_REG (system tick).
- Registro TICK_CMPR_REG (Tick compare).
- Registro STICK_CMPR_REG (system tick compare).
- Registro SOFTINT_CMPR_REG (software interrupt).
- Registro SET_SOFTINT (set software interrupt register).
- Registro CLEAR_SOFTINT (clear software interrupt register).
- Registro PERF_CONTROL_REG (performance control register).
- Registro PERF_COUNTER (performance counter.

4. Mapa de registros ASI.
- Registros Scratch-pad (ocho de ellos).
- Registros de CPU y dispositivos.
- Punteros de cabeza y cola.
- Registro de interrupción de CPU.
- Registro de interrupción.
- Registro del vector entrante.
- Registro de expedición de interrupciones (para llamadas cruzadas).

Tipos de Trampa (TRAPS).

Las trampas pueden ser generadas por el código de usuario, código supervisor o código hipervisor. Una trampa es entregada a diferentes niveles del controlador de trampa para ser procesada, el nivel supervisor (SV), también conocido como el

nivel privilegiado, o el nivel hipervisor (HV). La manera en que las trampas se generan puede ayudar a categorizar en síncrona (para la tubería del núcleo SPARC) o asíncrona.

Existen tres categorías definidas de las trampas: precisa, diferida e interrumpida. En los párrafos siguientes se describen brevemente la naturaleza de cada una de ellas.

1. Trampa precisa.

Una trampa precisa es introducida por una instrucción particular y ocurre antes de que cualquier estado de programa visible haya cambiado por la instrucción. Cuando una trampa precisa ocurre, se debe cumplir las siguientes condiciones:

- Haber guardado puntos en TCP a la instrucción que indujo la trampa y guardarlos en la NTPC para la instrucción que será ejecutada a continuación.
- Todas las instrucciones emitidas antes de que induzcan la trampa deben tener completada su ejecución.
- Las instrucciones emitidas después de que la indujo la trampa deben seguir siendo ejecutadas.

2. Trampa Diferida.

Una trampa diferida es inducida por una instrucción particular. Sin embargo la trampa puede ocurrir después de que estado de programa visible haya cambiado la ejecución de cualquiera de las otras trampas o más instrucciones. Si una instrucción induce una trampa en diferido, y la trampa ocurre simultáneamente, la trampa diferida no podrá aplazarse más allá de la trampa precisa.

3. Trampa Interrumpida.

La trampa interrumpida o perturbada se causa por una condición (por ejemplo, una interrupción), en lugar de ser causada por una instrucción. Cuando una trampa interrumpida ha sido servida, la ejecución del programa se reanuda donde se quedó anteriormente. La interrupción de las trampas es controlada por una combinación del nivel de interrupción del procesador (PIL) y la habilitación de la interrupción (IE). La condición de ruptura o de interrupción se ignora cuando las interrupciones están deshabilitadas (PSTATE.IE=0) o la condición es inferior a la especificada en el PIL.

La siguiente tabla ilustra los tipos de trampas del procesador OpenSPARC T1.

Trap Type	Deferred	Disrupting	Precise
Asynchronous	*None*	None	Spill traps, FPU traps, DTLB parity error on loads, SPU-MA Memory error return on load to SYNC reg
Synchronous	DTLB parity error on stores (precise to SW)	Interrupts and some error traps	All other traps

Tabla 3.7 Tipos de trampas (Traps) en el OpenSPARC T1. Cortesía OpenSPARC T1.

Flujo de Trampas.

Una trampa asíncrona se asocia normalmente con las instrucciones de longitud de latencia y de guarda/restaura, por lo que la aparición de una trampa no es sincrónico con la operación de tubería del núcleo SPARC. Todas esas trampas son precisas en el procesador OpenSPARC T1.

La interrupción de las trampas está asociada con ciertas condiciones particulares. El TLU las recoge y las reenvía a la IFU, la cual las hace bajar por la tubería como interrupciones en lugar de enviar las instrucciones desde abajo.

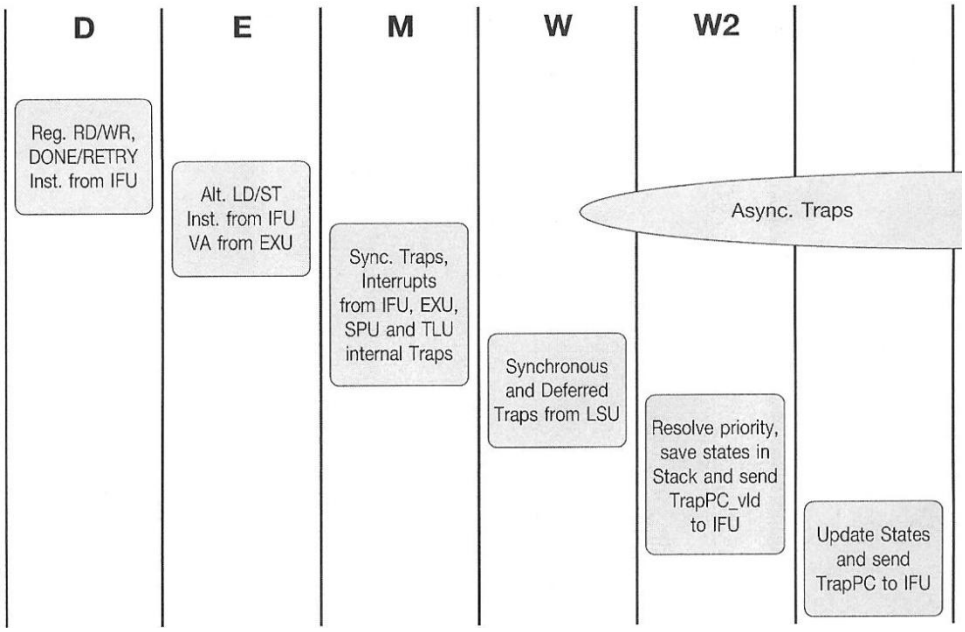

Figura 3.7-2 Secuencia del Flujo de Trampas. Cortesía OpenSPARC T1.

Todas las trampas de la IFU, EXU, SPU, LSU y de la TLU serán ordenadas por tipo para resolver la prioridad de cada una y para determinar el tipo de trampa y

tipo de vector. Después de que estos se resuelven, la dirección base de la trampa (TBA) será seleccionada para viajar por la tubería para su ejecución.

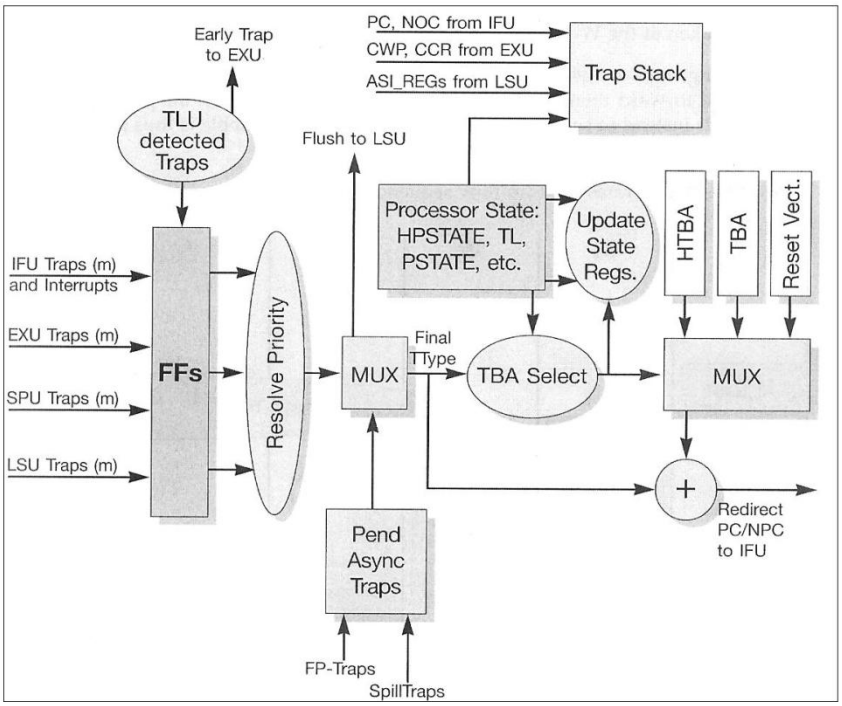

Figura 3.7-3 Diagrama del Flujo de trampas con respecto al bloque de hardware. Cortesía OpenSPARC T1.

Construcción del Programa de Contador de Trampas.

Se describirá el algoritmo para la construcción del programa de contador de Trampas (TPC):

- Supervisor trap (SV trap)
  Redirect PC <= {TBA[47:15], (TL>0), TTYPE[8:0], 5'b00000}
- Hypervisor trap (HV trap)
  Redirect PC <= {TBA[47:14], TTYPE[8:0], 5'b00000}
- Traps in non-split mode
  Redirect PC <= {TBA[47:15], (TL>0), TTYPE[8:0], 5'b00000}
- Reset trap
  Redirect PC <= {RSTVAddr[47:8], (TL>0), RST_TYPE[2:0], 5'b00000}
- RSTVAddr = 0xFFFFFFFFF0000000
- Done instruction
  Redirect PC <= TNPC[TL]
- Retry instruction
  Redirect PC <= TPC[TL]
  Redirect NPC <= TNPC[TL]

Interrupciones.

Las interrupciones de software se entregan a cada núcleo virtual mediante el "interrupt_level_n" y trampas (0x41-0x4f) a través del registro SOFTINT_REG. Llamadas cruzadas (cross-call) de CPU y Entradas/Salidas interrupciones se entregan a cada núcleo virtual usando la trampa "interrupt_vector" (0x60).

La trampa "interrupt_vector" para interrupciones de software tiene un registro de 64 bits ASI_SWVR_INTR_RECEIVE.

Dispositivos de entrada y salida, y llamadas cruzadas de CPU contienen un identificador de 6 bits, el cual determina qué vector de interrupción (nivel) en el registro ASI_SWVR_INTR_RECEIVE la interrupción será destinada.

Cada cadena de registro ASI_SWVR_INTR_RECEIVE puede tener en cola hasta 64 interrupciones pendientes, uno para cada vector de interrupción. Vectores de interrupción tienen prioridad con el vector 63 siendo la máxima prioridad y el vector 0 será la prioridad más baja.

Cada fuente de interrupción de Entrada/Salida tiene gran cableado de interrupciones que se utiliza como índice de una tabla de información del vector de interrupción (INT_MAN) en el puente de la unidad de Entrada/Salida. Generalmente a cada fuente de interrupción de Entrada/Salida se le asignará un único objetivo y su nivel de vector.

Flujo de Interrupción.

La siguiente figura ilustra el flujo de interrupciones y el vector de interrupciones:

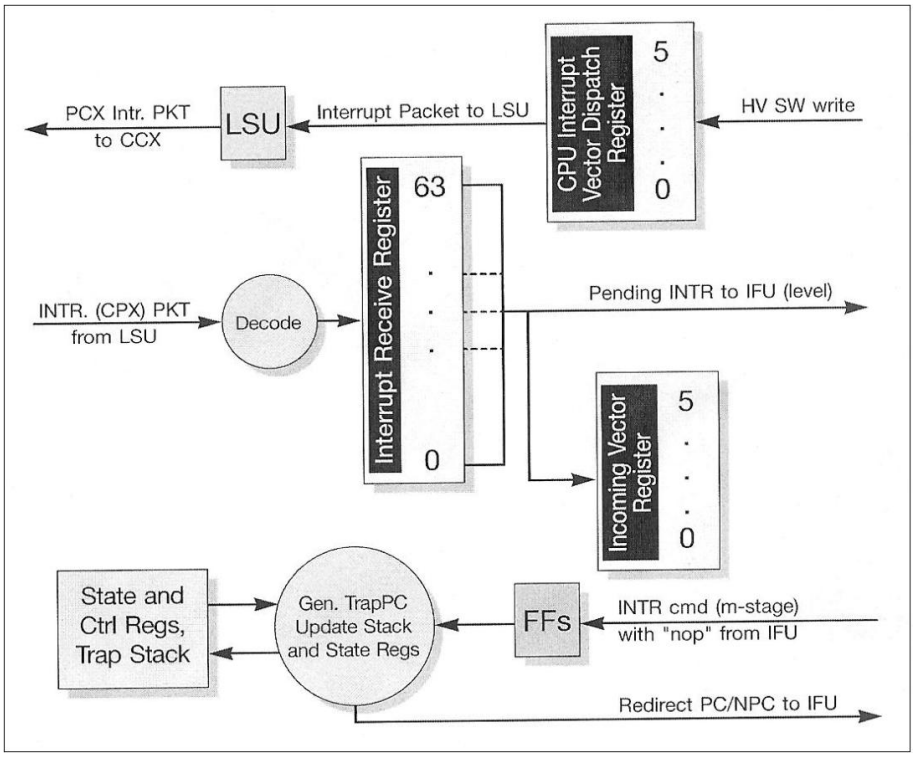

Figura 3.7-4 Flujo de Hardware y Vectores de Interrupción. Cortesía OpenSPARC T1.

Comportamiento de Interrupciones y enmascaramiento de Interrupciones.

En la siguiente lista se describe el comportamiento y enmascaramiento de interrupciones.

1. Interrupciones del Hipervisor no pueden ser enmascaradas por el supervisor ni tampoco por el usuario, sólo pueden enmascararse por el Hipervisor mediante el bit PSTATE.IE. Tales interrupciones incluyen interrupciones de hardware HINTP.
2. Interrupciones normales inter-core o inter-thread como las llamadas cruzadas (cross-calls) pueden ser mandadas por un software de escritura al registro CPU INT_VEC_DIS_REG.
3. Interrupciones especiales como “reset”, “idle”, o “resume” sólo pueden ser enviadas por software a través del puente Entrada/Salida (IOB) por escrito al registro INT_VEC_DIS_REG.
4. Hipervisor siempre suspenderá interrupciones del Supervisor.
5. Algunas interrupciones de supervisor como Mondo-Qs sólo pueden ser enmascaradas por el bit PSTATE.IE.
6. Interrupciones tipo “Interrupt_level_n” sólo podrán ser enmascaradas por el PIL y el bit PSTATE.IE en el nivel supervisor o usuario.

Modos de Transición de Trampas.

La figura 3.7-5 muestra el modo de transición desde los diferentes niveles de trampas:

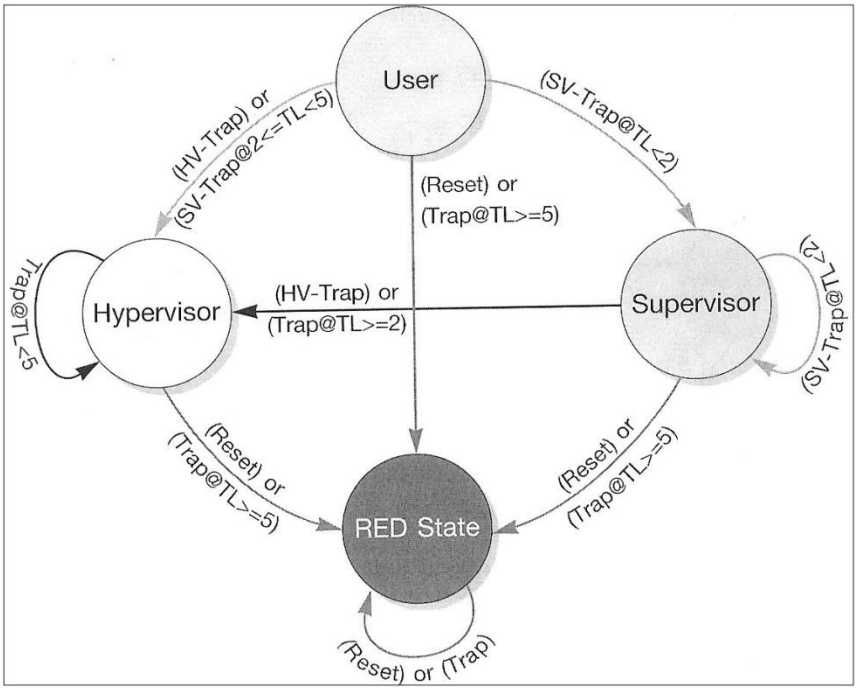

Figura 3.7-5 Modos de Transición de Trampas (Traps). Cortesía OpenSPARC T1.

3.8 CPU CACHE CROSSBAR.

Descripción Funcional.

La barra de cruce (crossbar-CCX) se encarga de manejar la comunicación desde los ocho núcleos, los cuatro bancos cache L2, el puente de Entrada/Salida y la Unidad de Punto Flotante (FPU). Todas estas unidades funcionales se comunican por el envío de paquetes y la CCX se encarga de administrar esa entrega de paquetes.

Cada núcleo SPARC de la CPU puede enviar un paquete a cualquiera de: los bancos cache L2, el puente de Entrada/Salida o a la FPU. Pero también pueden ser enviados en la dirección inversa, es decir, cualquiera de estas unidades pueden enviar paquetes a los ocho núcleos de la CPU. La siguiente figura muestra que cada uno de los ocho núcleos SPARC puede comunicarse con cada uno de los bancos cache L2, el puente Entrada/Salida y la FPU.

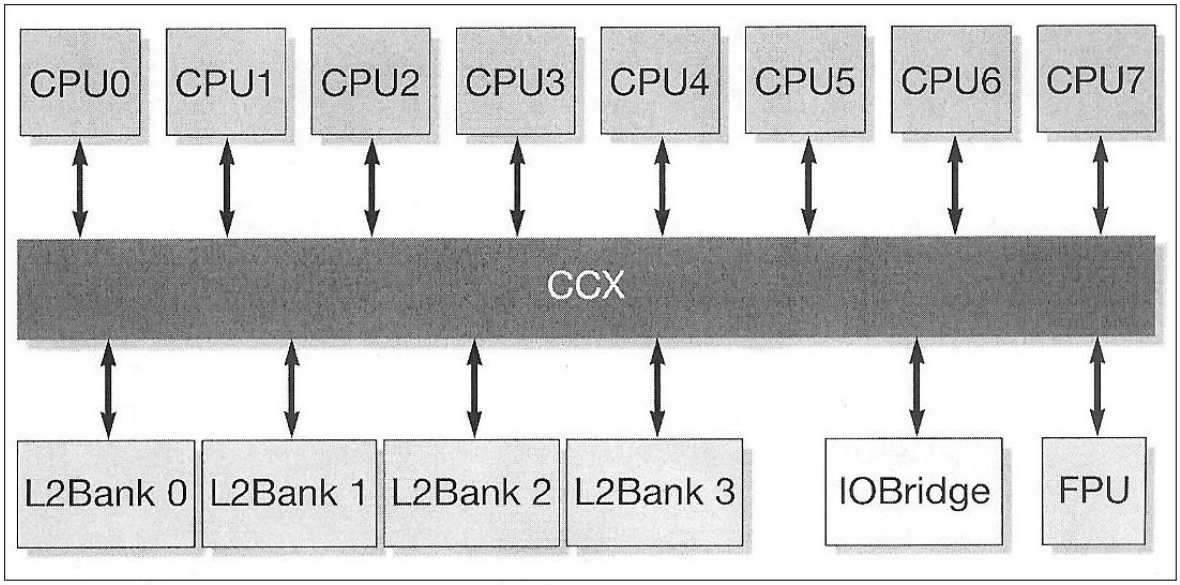

Figura 3.8-1 Interfaz del CCX. Cortesía OpenSPARC T1.

Entrega de Paquetes CCX.

La CCX consiste en dos bloques principales “processor cache crossbar” (PCX) y la “cache processor crossbar” (CPX). El bloque PCX se encarga de la comunicación de cualquiera de las ocho fuentes de CPU a cualquiera de: cuatro bancos cache L2, puente de Entrada/Salida o FPU. El CPX maneja la comunicación de los bancos cache L2, puente Entrada/Salida o FPU a cualquiera de los ocho destinos de la CPU. A continuación se ilustra las conexiones PCX y CPX.

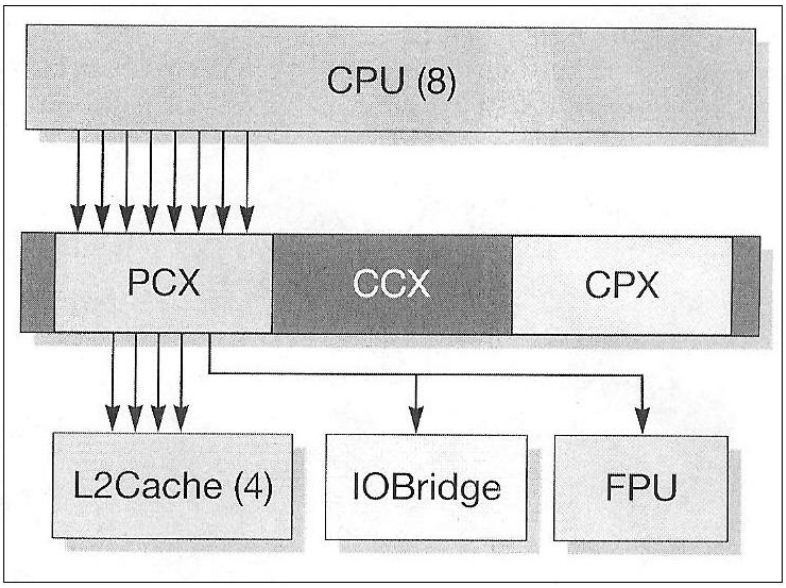

Figura 3.8-2 Interfaz del PCX. Cortesía OpenSPARC T1.

En un ciclo sólo un paquete puede ser entregado a un destino en particular. El CCX maneja dos tipos de solicitud comunicación. El primer tipo de solicitud contiene un paquete y será entregado en un ciclo. En cambio el segundo tipo de solicitud contiene dos paquetes y esos dos paquetes son entregados en dos ciclos.

El número total de ciclos requeridos de un paquete para viajar a través de la fuente hacia su destino puede ser mayor al número de ciclos de entrega del paquete.

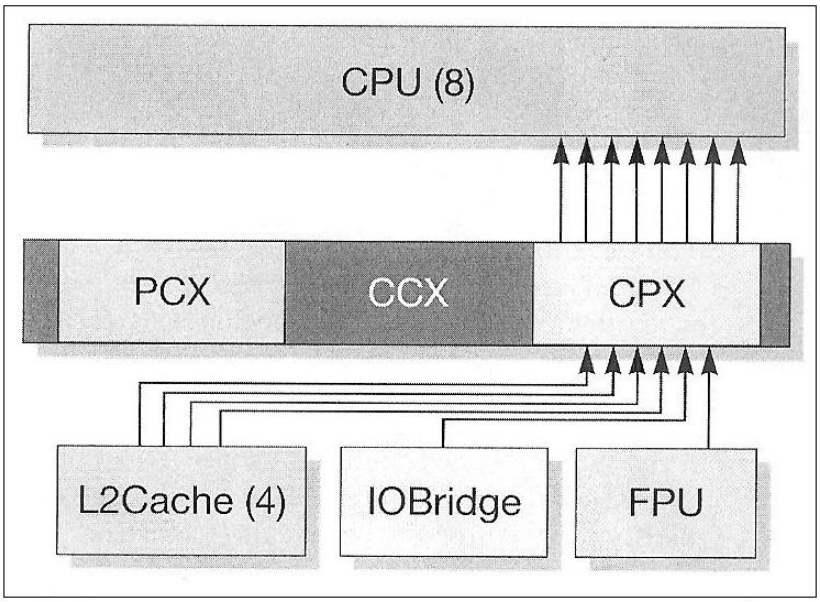

Figura 3.8-3 Interfaz del CPX. Cortesía OpenSPARC T1.

Procesamiento de Transacciones de PCX.

LOAD-CARGA

Una transacción LOAD transfiere datos desde L2 o I/O hacia el núcleo. Acceso cacheable del L2 tendrá un tamaño de 16 bits. Por lo tanto el tamaño en el paquete PCX deberá ser ignorado. 16 Bits de datos de carga se devuelven en el bus CPX.

PREFETCH-PREBUSQUEDA

Podrá ser emitida a L2. Desde la perspectiva L2 la prebúsqueda es una simple carga que no es cacheable en la L1. La L2 se hará valer en el bit PFL en el regreso del paquete CPX así el núcleo no sabrá actualizar el registro de archivos como sucedería con un “load”.

STORE-ALMACENAMIENTO

Solicitudes de Almacenamiento ocasionan que los datos sean actualizados en L2 o I/O. El núcleo SPARC enviará 64B de datos e indicará el tamaño de almacenamiento. Para almacenamientos menores a 64B los datos serán duplicados como se muestra en la siguiente tabla:

Operation	Data Fill
Stb 0x14	*0x14141414_14141414*
Sth 0x0102	0x01020102_01020102
Stw 0x01020304	0x01020304_01020304
Stx 0x01020304_05060708	0x01020304_05060708

Tabla 3.8 Campo de datos. Cortesía OpenSPARC T1.

Descripción del Bloque Funcional PCX.

El PCX contiene cinco módulos idénticos de mediadores o jueces (arbiter), uno para cada destino. El “arbiter” almacena los paquetes de las fuentes de un destino particular. El PCX después juzga y entrega los paquetes hacia su destino final. La siguiente figura muestra un diagrama de bloques de la mediación.

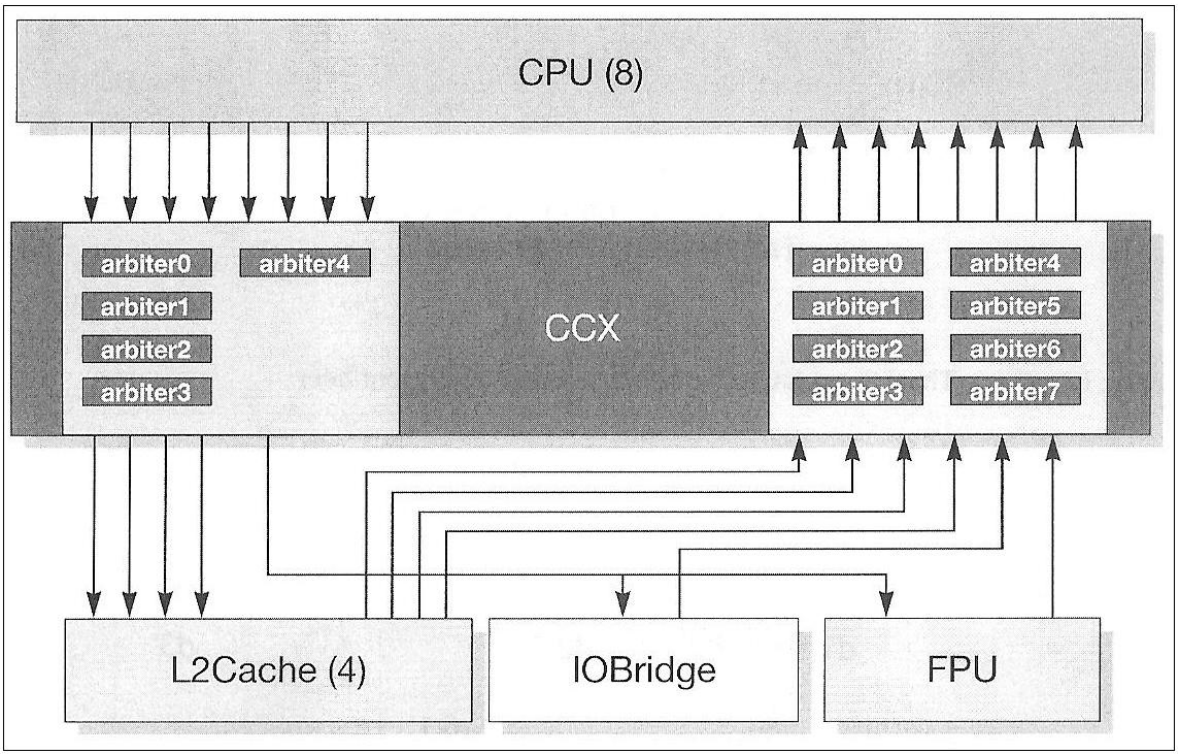

Figura 3.8-4 Bloque Interno de PCX y CPX. Cortesía OpenSPARC T1.

Flujo de Datos del Mediador PCX.

Si bien el flujo de datos es similar en el interior de los demás jueces (arbiter) se hará la descripción de sólo uno de ellos (ARB0). Se cuenta con un bus con un ancho de 124 bit por cada núcleo SPARC que se extiende a los cinco jueces (un bus por cada juez).

El ARB0 puede recibir paquetes de cualquiera de los ocho núcleos. Por lo tanto ARB0 contiene ocho colas, y cada cola es una entrada FIFO y cada entrada puede contener un paquete. Un paquete tiene un ancho de 124 bit y contiene la dirección, los datos y los bits de control. ARB0 entrega paquetes al Banco “Bank0” con un ancho de 124 bit. En la figura abajo descrita, se ilustra el flujo de datos:

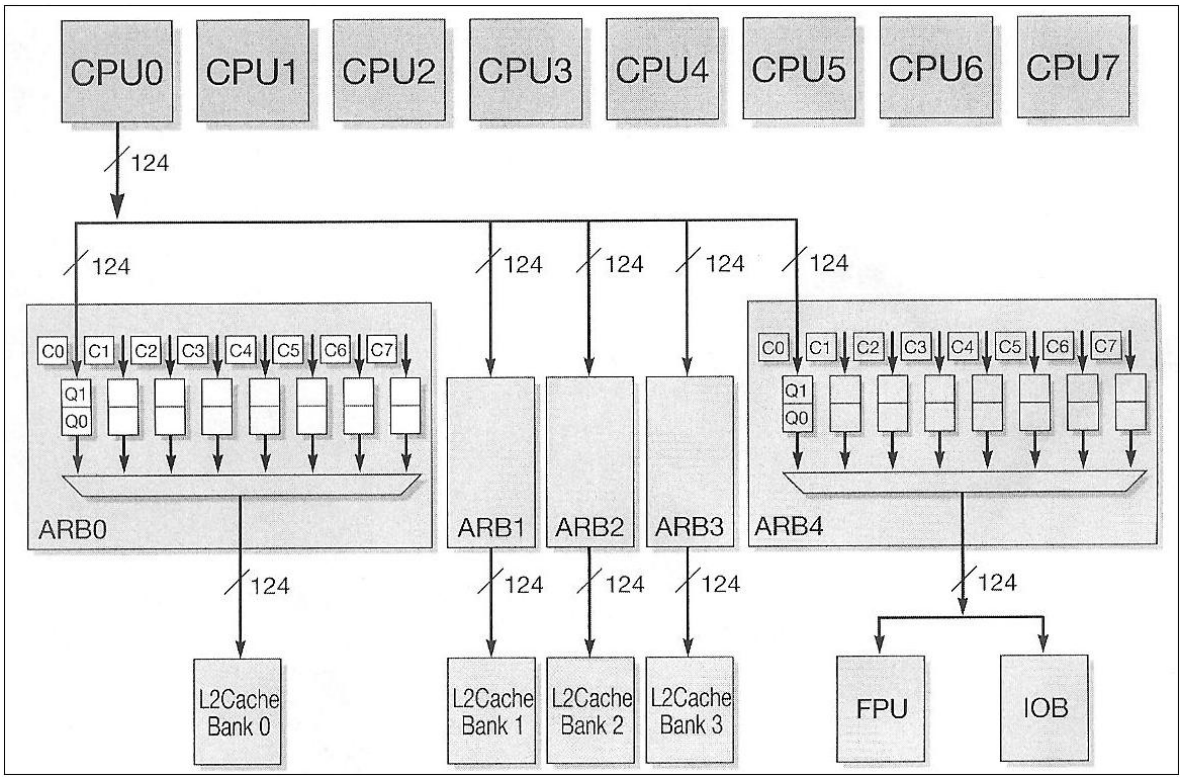

Figura 3.8-5 Flujo de Datos dentro de un "arbiter" en el PCX. Cortesía OpenSPARC T1.

ARB1, ARB2 y ARB3 reciben paquetes de la L2 cache Bank1, Bank2 y Bank3 respectivamente. ARB4 recibe paquetes tanto de la FPU y del puente I/O.

Flujo de Control del Mediador PCX.

El ARB0 gestiona paquetes en el orden en que los recibe. Por lo tanto un paquete recibido en el ciclo 4 se enviará antes de un paquete recibido en el quinto ciclo. Cuando múltiples fuentes administran un paquete en el mismo ciclo, ARB0 sigue una política "roun-robin" para arbitrar los paquetes de las fuentes.

Un bus de 5 bit por cada CPU y el bit correspondiente al destino será alto mientras todos los demás bits sean bajos.

El esquema de arbitraje del mediador se implementa mediante un tablero como se ilustra a continuación:

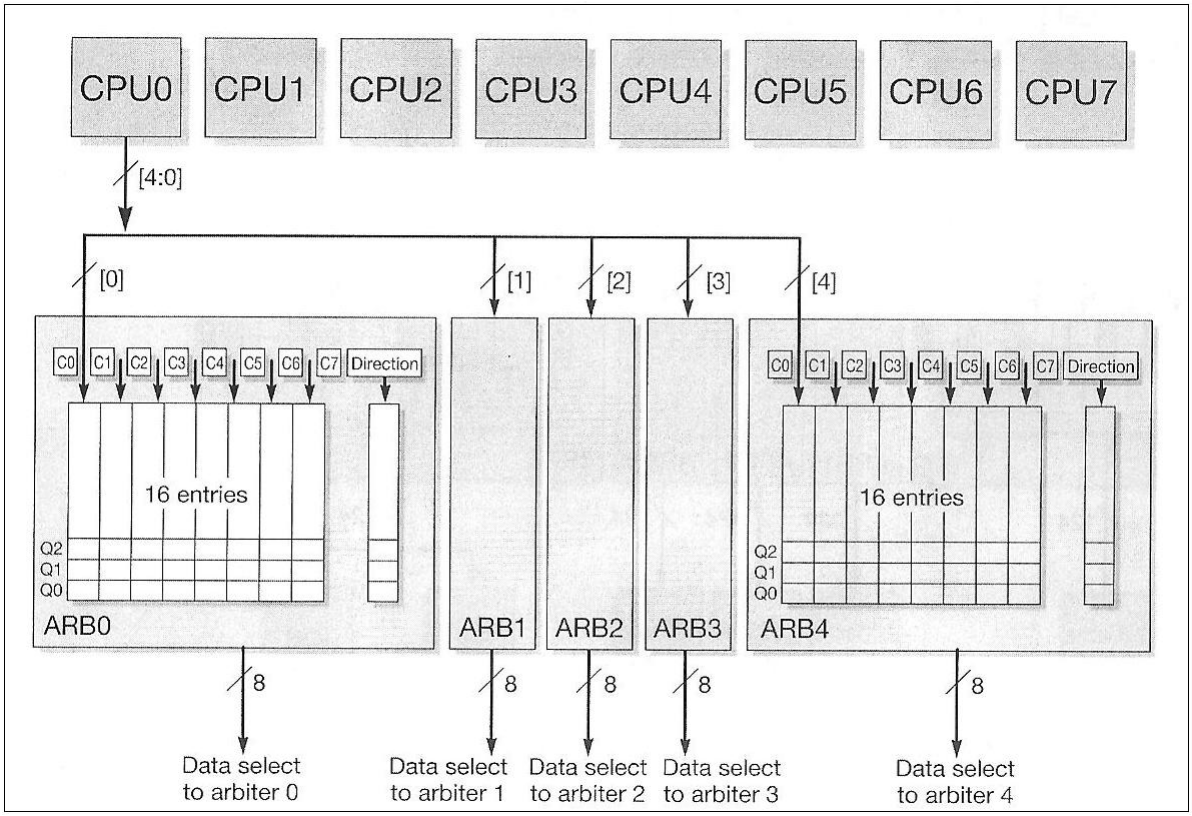

Figura 3.8-6 Flujo de Control del "arbiter" en el PCX. Cortesía OpenSPARC T1.

El tablero consta de ocho FIFO´s. Cada FIFO tiene 16 entradas y cada entrada tiene un solo bit válido recibido desde su correspondiente CPU. Cada entrada válida del FIFO representará un paquete válido de una fuente para el Bank0 cache L2. Desde cada fuente se podrá enviar como máximo dos entradas para el Bank0 cache L2, pero no puede haber dos bits válidos por cada FIFO. Por lo tanto el tablero puede tener un máximo de 16 bits válidos. El máximo caso se representa cuando el Bank0 cache L2 no es capaz de procesar nuevas entradas.

3.9 CACHE L2.

Visión General.

El cache L2 del procesador OpenSPARC T1 tiene un tamaño de 3 Mbytes y se compone de cuatro bancos simétricos que se intercalan en una frontera de 64 bytes. Cada banco opera independientemente uno de otro. Cada banco es formado por un conjunto de 12 vías asociativas con un tamaño de 768 Kbytes. El tamaño de bloque (línea) es de 64 bytes y cada banco cache L2 tiene 1024 conjuntos.

El caché L2 acepta solicitudes desde el PCX del núcleo SPARC y le responde a la CPX. La cache L2 también es responsable de mantener la coherencia en el chip en todos los chaches L1 y de mantener una copia de las etiquetas L1 en un directorio.

Descripción Funcional del Banco Cache L2.

El caché L2 está organizado en cuatro bancos idénticos, en el cual, cada banco cuenta con su propia interfaz, con la J-bus, el controlador DRAM y el larguero o barra cruzada del CPU (CCX).

Cada banco caché L2 realiza interfaz con ocho núcleos SPARC a través del larguero PCX. El PCX realiza la administración de las peticiones o solicitudes de la caché L2 como: carga, almacenamiento, accesos, etc, hacia los ocho núcleos apropiados. El PCX también acepta el regreso de datos, invalidación de paquetes, almacenamiento de paquetes de cada banco caché L2 y los reenvía al CPU apropiado.

Cada banco caché L2 consiste en los siguientes tres sub-bloques:

- sctag (secondary cache tag): el cual contiene un arreglo de etiquetas y de VUAD, un directorio cache L2 y un controlador caché.
- scbuf: contiene un buffer de escritura trasera (WBB), buffer de llenado (FB) y un buffer DMA.
- scdata: contiene el arreglo scdata.

Los cuatro estados de bits de “sctag” están organizados en un arreglo de puertos en el VUAD L2, y esos cuatro estados son: Valid (V), Used (U), Allocated (A) y Dirty (D). Este arreglo de bits cuenta con 2 puertos de lectura y dos de escritura.

La palabra “scdata” implica un arreglo de bancos en una estructura de puerto SRAM. Cada caché L2 tiene un tamaño de 768 Kbytes con su línea lógica de 64 bytes.

Los estados de tubería (pipeline) de la Caché L2 son ocho (C1 a C8).

3.10 PUENTE DE ENTRADA/SALIDA (I/O BRIDGE).

El puente de Entrada/Salida (IOB) es la interfaz entre la barra de cruce caché CPU (CCX) y los demás bloques en el procesador OpenSPARC T1. Las principales funciones se presentan a continuación:

- Decodificación de direcciones I/O.
    - Mapas o códigos del destino propio interno o externo.
    - Genera el registro de control y estado de acceso de agrupaciones.
    - Acceso programado de entrada/salida al J-Bus externo.
- Interrupciones:

    - Se encarga de recolectar las interrupciones de agrupaciones y del J-Bus.
    - Reenvía interrupciones al núcleo e hilos correctos.
- Interfaz entre la Lectura/Escritura de la SSI.
- Provee pruebas de acceso a los puertos (TAP) a la SCR´s, memoria, caché L2 y CPU ASI´s.

Interfaces del IOB.

Las principales interfaces del puente de Entrada/Salida son:

- Crossbar (CCX).
- Bus de conexión universal (UCB).
- Puertos de depuración.
- Controlador del Fusible electrónico (EFC).

La siguiente figura muestra las interfaces del IOB del resto de los bloques.

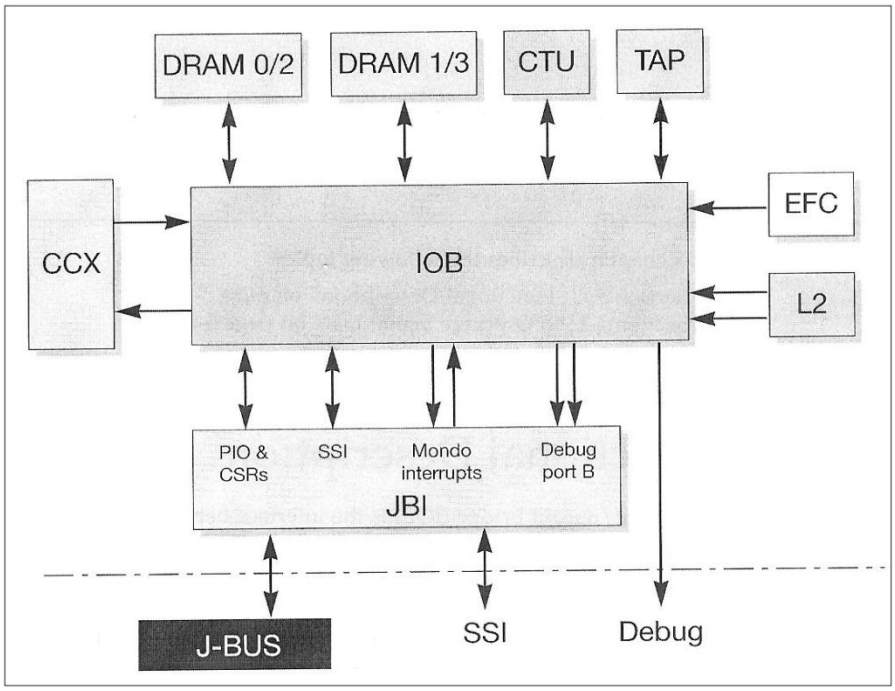

Figura 3.10-1 Interfaz IOB. Cortesía OpenSPARC T1.

Diagrama de Bloques del IOB.

En la siguiente figura se ilustra el diagrama de bloques interno del IOB. Las solicitudes PCX del CPU son procesadas por un bloque llamado “c2i” (CPU to I/O) y genera solicitudes UCB hacia varios bloques. Las solicitudes UCB desde varios bloques son procesadas por otro bloque llamado “i2c” (I/O to CPU) los cuales generarán paquetes CPX. Los registros de control/estado (CSRs) son controlados

por el bloque CSR. La depuración de bloques toma datos de la caché L2 y los manda a depurar al puerto A (un puerto externo).

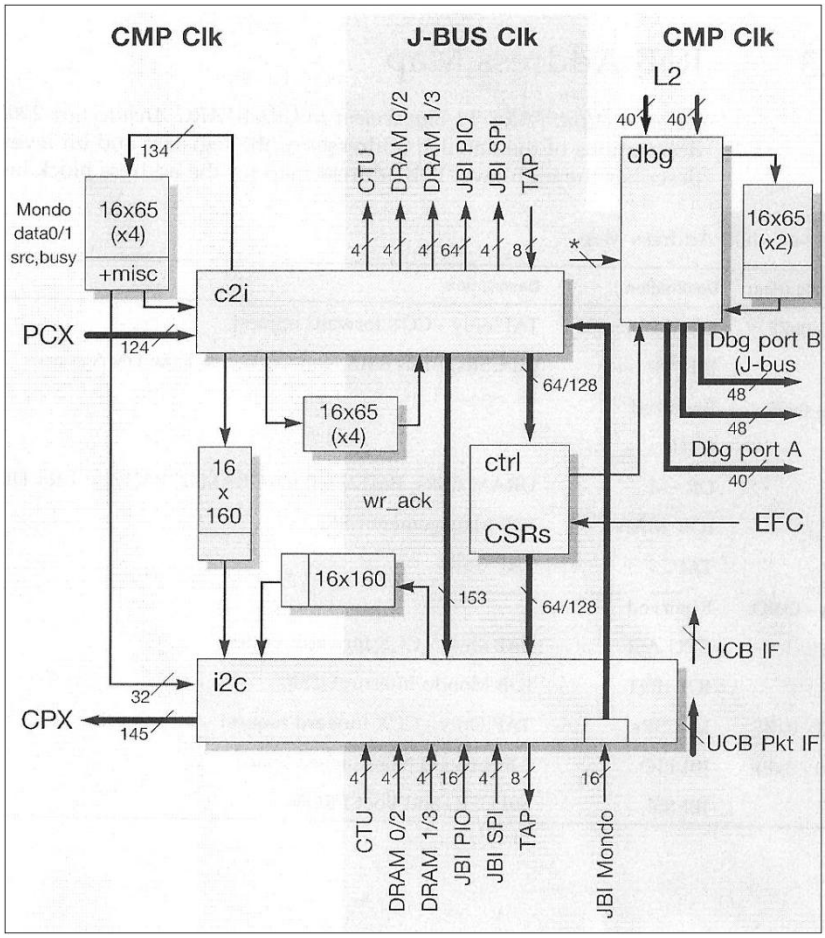

Figura 3.10-2 Diagrama de Bloques IOB. Cortesía OpenSPARC T1.

3.10.1 INTERFAZ J-BUS.

Descripción Funcional.

El bloque JBI en el procesador OpenSPARC T1 realiza interfaz con los siguientes bloques:

- Caché L2- para leer y escribir datos hacia caché L2.
- IOB- para entradas/salidas programadas (PIO), interrupciones y depuración de puertos.

Muchos de los sub-bloques JBI usan el reloj J-bus y otras partes del reloj del núcleo.

A continuación se ilustra el diagrama de bloques de JBI.

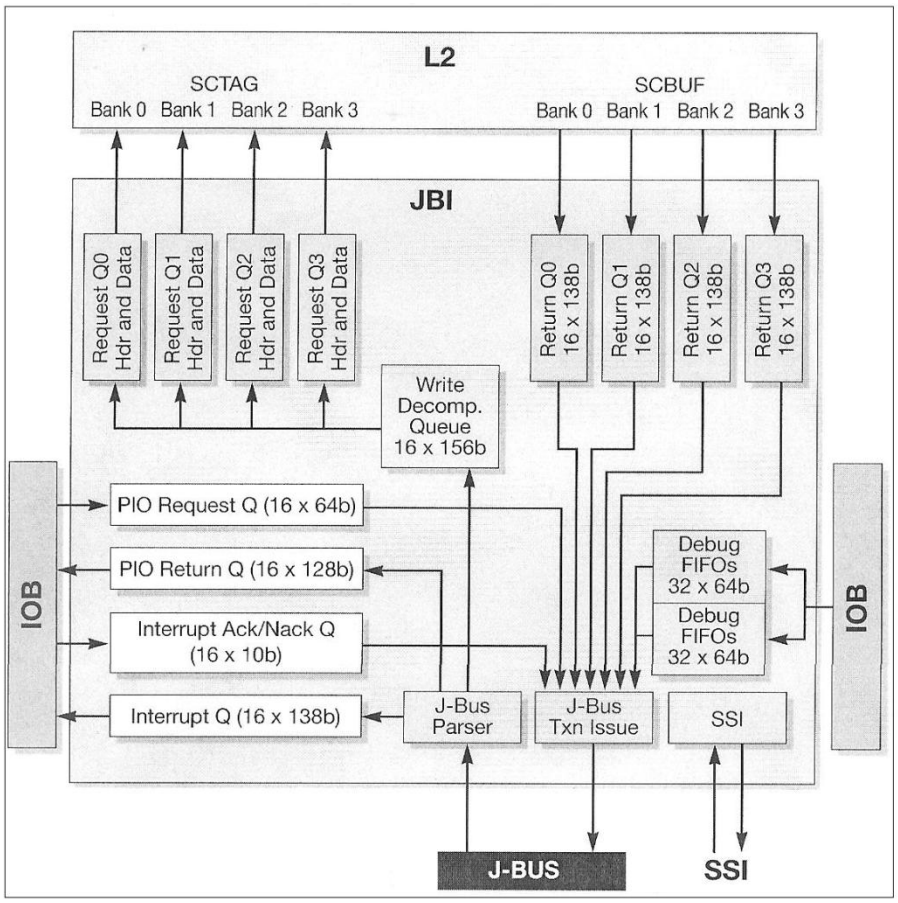

Figura 3.10-3 Diagrama de Bloques de JBI. Cortesía OpenSPARC T1.

3.11 UNIDAD DE PUNTO FLOTANTE (FPU).

Descripción Funcional.

La unidad de punto flotante del OpenSPARC T1 tiene las siguientes características:

- Implementa el conjunto de instrucciones de punto flotante del SPARC V9.
- La FPU no apoya el conjunto de instrucción visual (VIS).
- La FPU es una fuente compartida del procesador OpenSPARC T1. Cada uno de sus ocho núcleos puede tener un máximo de una instrucción FPU.
- El archivo de registro de punto flotante (FRF) y estado de registro de punto flotante (FSR) no están situados físicamente dentro de la FPU.
- Incluye tres tuberías independientes de ejecución:
    - Floating Point Adder (FPA)- Realiza suma, resta, comparación y conversión.
    - Floating Point Multiplier (FPM)- Realiza multiplicación.
    - Floating Point Divider (FPD)- Realiza División.
- Una instrucción por ciclo puede ser emitida.

- Una instrucción por ciclo puede ser completada y liberada de la FPU.

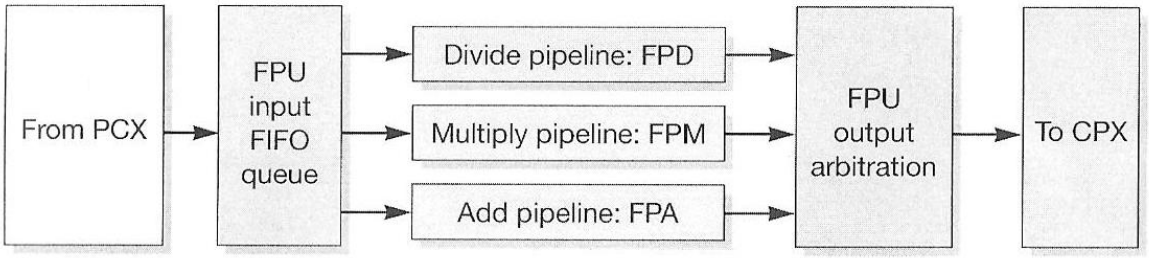

Figura 3.11-1 Diagrama de Bloques de la Función de la FPU. Cortesía OpenSPARC T1.

CONTROLADOR DRAM.

Descripción Funcional.

El controlador DDR-II DRAM del OpenSPARC T1 tiene las siguientes características:

- Contiene cuatro controladores DRAM independientes, y cada controlador está conectado a un Banco Caché L2 y a un canal de memoria DDR-II
- Tiene un máximo espacio de dirección de 37 bits para un tamaño de memoria máximo de 128 Gbytes.
- Las líneas caché de 64 bytes se intercalan a través de cuatro canales.
- Su rango de operación es de 125 MHz hasta 200 MHz con una velocidad de datos de 250 hasta 400 MT/sec.
- Su pico de ancho de banda es de 23 Gbyte/sec a 200 MHz.
- El controlador DRAM opera en dos modos: modo de cuatro canales y modo de dos canales, el cual es programable.

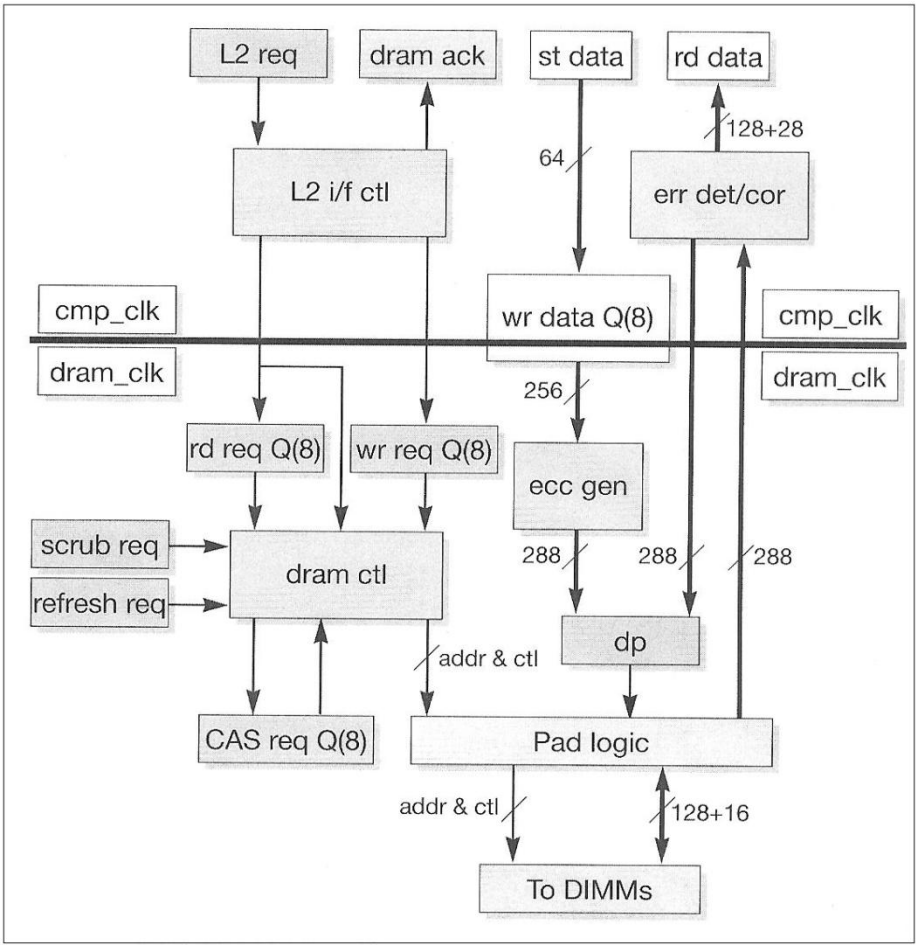

Figura 3.11-2 Diagrama de Bloques del Controlador DDR-II DRAM. Cortesía OpenSPARC T1.

Prioridad de Arbitraje.

Las solicitudes leídas cuentan con mayor prioridad sobre las peticiones de escritura, pero existe un contador que habilita la escritura. El controlador DRAM nunca verá solicitudes leídas seguidas de solicitudes escritas y el orden de prioridad será como se describe a continuación:

1. Actualizar solicitudes.
2. En espera de la columna de dirección de estrobos (CAS) y solicitudes (round-robin).
3. Solicitudes de fila de dirección de estrobos (RAS).
4. Espera de solicitudes escritas RAS que coinciden con direcciones.
5. Solicitudes de lectura o escritura RAS.

Diagramas de estado del Controlador DRAM.

La siguiente figura presenta el nivel superior del diagrama de estado del controlador DRAM. Un software deberá iniciar el controlador.

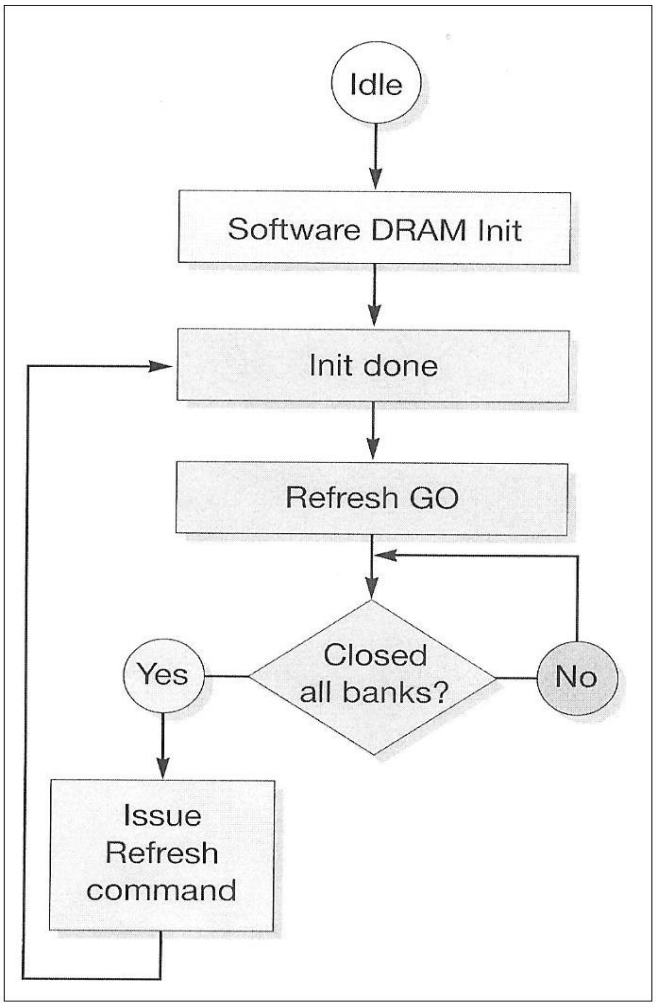

Figura 3.11-3 Diagrama de nivel superior del Controlador DDR-II DRAM. Cortesía OpenSPARC T1.

CONTROL DE ERRORES.

El procesador OpenSPARC T1 detecta, registra y reporta toda serie de errores al software. A continuación se describirán brevemente los tipos de errores y cómo se detectan, registran y reportan dichos errores.

Existen tres tipos de errores en el procesador OpenSPARC T1:

1. Errores Corregibles (CE).

Son corregidos por el hardware y él mismo puede generar trampas e interrumpir para dar un seguimiento de la frecuencia de errores.

2. Errores Incorregibles (UE).

Estos tipos de errores no pueden ser corregidos por el hardware y deberán ser corregidos por el software.

3. Errores Fatales (FE).

Los errores fatales pueden crear potencialmente daños y se tendrá que realizar un restablecimiento.