Fernando Pardo Carpio
DECV Diseño Electrónico y Circuitos VLSI
C/ Hugo de Moncada, 4 Entl.-Izq.
46010 Valencia
Telf. 96 360 9177
Fax. 96 361 6198

Futurebus+ y Tolerancia a Fallos

Descripción de las posibilidades de Futurebus+ como bus en un sistema tolerante a fallos

Fernando Pardo/Jose A. Boluda/Rafael Martínez/Carlos Pérez

DECV Diseño Electrónico y Circuitos VLSI

Abstract

Futurebus+ es un bus poco conocido todavía pero por sus altas prestaciones está llamado para sustituir a los populares buses VME, EISA, Multibus, etc. Sus características lo hacen ideal para cualquier tipo de sistema, desde sistemas multiprocesadores con soporte para cache, hasta sistemas con funcionalidad de tiempo-real, pasando por los sistemas tolerantes a fallos que son el objeto del presente artículo.

Introducción

La evolución de los computadores hacia sistemas multiprocesadores, tolerantes a fallos, con funcionalidad de tiempo-real, coherencia de cache y tamaño de palabra cada vez

mayor, ha echo que los populares buses de 32 bits como VME o Multibus estén empezando a ser obsoletos. Futurebus+ representa el mayor cambio de mentalidad en la industria de los computadores. Es una de las primeras arquitecturas de bus diseñada a priori como estándar y pensada para servir de plataforma para las múltiples generaciones futuras de computadoras derivadas de los avances en la tecnología informática. Futurebus+ no está diseñado para una arquitectura específica o procesador concreto como los buses utilizados hasta ahora, por lo que puede ser aplicado a cualquier sistema.

No se van a enumerar todas las características técnicas que este bus ofrece, pero si vale la pena resaltar alguna de ellas como su alta velocidad de transmisión de datos, disponiendo de protocolos síncronos y asíncronos para que el bus no se quede anticuado por mucho que progrese la tecnología. Otra característica que pocos buses ofrecen es el soporte para snoopy-cache cualquiera que sea el número de buses interconectados. Hay otras muchas características, como sus completos protocolos de arbitraje que lo hacen ideal para sistemas de tiempo real, y otras, más en la línea de la tolerancia a fallos, que se explicarán más adelante. En general, se puede decir que este bus ha tomado las principales características de los buses más modernos, las ha mejorado y además ha incorporado otras nuevas. De todas estas características se han escogido y analizado aquellas que más cerca están de los sistemas tolerantes a fallos y de alta disponibilidad.

Protocolos distribuidos

Una manera de obtener sistemas tolerantes a fallos es evitar los puntos críticos de fallo, es decir, puntos donde si el componente falla el sistema falla también. Supongamos un módulo procesador compuesto por tres procesadores cuya salida se "vota". En este caso, si cualquiera de ellos falla, la salida será correcta siempre que sea uno sólo el procesador

que falló. Este sencillo sistema es tolerante a fallos puesto que se permiten fallos en los procesadores siempre que sea uno solo el procesador que falla. Pero, si analizamos bien el circuito, encontramos que existe un punto crítico donde, si se produce una avería, el sistema fallará. Este es el caso en el que el propio circuito que vota las salidas falla, en este sistema, el votador, es un punto crítico de fallo.

En un sistema multiprocesador, cuyos procesadores comparten un único bus, es necesario controlar que un solo procesador posea el bus a un tiempo. A esta operación se le llama arbitraje del bus. En un bus con arbitraje centralizado existe un módulo en el sistema que se ocupa de arbitrar el bus. Este módulo decide quien será el siguiente en tomar el bus atendiendo a una prioridad o a una cola donde los distintos módulos hacen sus peticiones. En un caso como éste, el sistema tendría un punto de fallo crítico en el módulo de arbitraje, puesto que una avería en este módulo produce una fallo general en el sistema. La forma de solucionar este problema en un sistema tolerante a fallos es haciendo que el arbitraje sea distribuido.

Futurebus+ es un bus que permite ambos tipos de arbitraje, tanto centralizado como distribuido [1]. En el arbitraje distribuido, cada módulo en el sistema posee un numero de arbitraje compuesto por dos partes, una parte alta con la prioridad del módulo, y una parte baja, única para cada módulo que, en el caso que dos o más módulos tengan la misma prioridad, sirve para saber el lugar que ocupa el módulo peticionario dentro de una cola. El arbitraje del bus tiene lugar al mismo tiempo que las transacciones, y consiste en que todos los módulos que desean competir por el bus ponen su número de arbitraje en unas líneas que comparten todos los módulos gracias a una OR cableada. Después de la competición, el módulo con el número de arbitraje más alto gana el bus y espera a que el actual maestro acabe para poder coger el bus. La forma en que los módulos averiguan cuál es el módulo vencedor o con el número de arbitraje más alto es la siguiente: para empezar,

todos los módulos competidores ponen su número en las líneas de bus AB[7..0] (Arbitration Bus). Estas líneas forman una OR cableada de manera que su contenido real es un OR lógico de todos los números de arbitraje de los módulos que compiten en ese instante. Si el número de competición de un módulo tiene uno de sus bits a cero y otro módulo tiene un uno en este mismo bit, entonces los que tengan cero en esta posición deben poner a cero las líneas de bus que sean menos significativos que este bit. De esta manera nos aseguramos de que en AB[7..0] sólo quedará el número más alto de todos los de arbitraje. Una simple comparación por parte de cada módulo servirá para detectar el módulo que ganó el bus.

La sincronización de estas operaciones se realiza mediante un protocolo de tres líneas (AP, AQ y AR) de forma totalmente asíncrona, lo que es especialmente interesante para la detección y la tolerancia de fallos. Por un lado, el no disponer de un reloj que sincronice la operación de arbitraje sirve para que no haya un punto crítico de fallo, que sería el propio reloj, y por otro, el hecho de que la sincronización sea distribuida sirve para detectar fallos en un módulo mediante la implementación de "time-outs" que serían detectados por el resto de módulos. Por ejemplo, el paso de una fase a otra dentro del arbitraje se caracteriza por la puesta a cero de una de las tres líneas de sincronismo. Si un módulo se avería es posible que nunca ponga a cero estas líneas, en este caso, un circuito que detecte un time-out en cualquiera de estas tres líneas detectará que un módulo no está bien y podrá iniciar una operación de aislamiento del módulo averiado para que el sistema pueda seguir funcionando. Los chips disponibles en el mercado para la implementación del arbitraje en Futurebus+ vienen con contadores programables de "time-out" de manera que pueden generar una interrupción al módulo e iniciar una rutina de aislamiento y reconfiguración del sistema en caso de error.

El arbitraje centralizado funciona de una manera diferente. Los módulos que quieren hacerse con el bus mandan su petición al árbitro central mediante dos líneas del bus (RQ0 y RQ1) y programan su prioridad mediante la escritura de los registros del chip que hace de árbitro central. La ventaja que presenta este modo frente al distribuido es que las operaciones de petición y cesión del bus se realizan a más velocidad, además de que sólo un chip de arbitraje es necesario en todo el sistema siempre que los demás módulos no implementen paso de mensajes de arbitraje, que consiste en utilizar el bus de arbitraje para mandar mensajes de 8 bits de un módulo a otro y que por tanto necesitan un chip dedicado. La desventaja es, por supuesto, que este módulo o circuito es un punto crítico de fallo.

La mejor opción, y la más cara, es tener un sistema con ambos modos de arbitraje. Tomemos como ejemplo los chips de arbitraje de Texas Instruments [7], el FB2010 que se ocupa de los protocolos del arbitraje distribuido, y el FB2011 que es un chip para el arbitraje centralizado que además soporta el modo distribuido. En un sistema con los dos modos tendríamos todos los módulos con el chip FB2010 salvo uno que haría de árbitro central y que tendría el FB2011. La operación normal sería en modo centralizado de manera que los chips FB2010 no participarían en el arbitraje si bien serían útiles para atender los mensajes de arbitraje que se produzcan. En el caso de producirse un fallo en el FB2011 o en el módulo que lo contiene, automáticamente se inicializaría una operación de aislamiento de este módulo y se pasaría al modo distribuido de arbitraje. Esta operación es fácil de realizar a través de los registros de programación de los chips de arbitraje. La implementación de un árbitro central junto con el distribuido para ser reconfigurado en caso de fallo se encuentra en la fig. 1. En el esquema vemos que las señales de arbitraje RQ[1:0] (petición del bus), GR (cesión del bus) y PE (preemption, cesión súbita del bus) están compartidas por el árbitro distribuido TFB2010, y el central TFB2011. En caso de pasar de un modo a otro, un conmutador cambia la procedencia de las líneas GR y PE,

que en el modo centralizado son generadas por el TFB2011, y en el modo distribuido por el TFB2010 en la propia placa. Este conmutador se implementa directamente en los transceivers.

No sólo el protocolo del arbitraje es distribuido, el protocolo del bus paralelo, o parte del bus dedicada a transacciones, también lo es. Las transacciones en el bus siguen un protocolo distribuido ya que la sincronización entre el maestro y el/los esclavos tiene lugar con la participación de ambos, de manera que si uno falla el otro puede detectar el fallo mediante time-outs.

En sistemas en tiempo real es necesario disponer de un reloj en el sistema que se ocupe de proveer un tiempo absoluto. Este reloj suele ser único por lo que puede representar un punto crítico de fallo. En Futurebus+ este reloj puede estar distribuido disponiendo cada módulo de su propio reloj de tiempo real sincronizándose entre ellos de forma periódica, de forma que si uno de los relojes falla siempre queda el resto de relojes que podrán seguir siendo usados. El reloj de tiempo real de Futurebus+ usa los registros especificados en el standard CSR (IEEE 1212) [4] y el protocolo de sincronización especificado en el IEEE 896.3 [3].

Otro ejemplo de distribución de protocolos en Futurebus+ es la elección del monarca. El monarca es el módulo que inicializa el sistema y su elección se hace al reinicializar el sistema o al arrancar maquina [3]. En muchos buses ni siquiera existe la posibilidad de elegir este módulo, sino que la placa colocada en un slot concreto del backplane arranca el sistema, lo que representa nuevamente un punto critico de fallo, si falla este módulo el sistema no arranca. En Futurebus+ la elección del monarca no depende de un módulo concreto, cualquier módulo en el sistema capaz de hacer las tareas de inicialización puede

ser monarca, de manera que si al arrancar, el que hace de monarca falla, la tarea de inicialización puede ser retomada por cualquier otro módulo en el sistema.

Múltiples buses

Normalmente la tolerancia a fallos se obtiene por redundancia de los elementos que componen el sistema, es decir, por duplicación o multiplicación de recursos, de manera que el sistema se pueda reconfigurar ante un fallo y pueda seguir funcionando. El bus es también un elemento que puede fallar en un sistema ya que un bus no sólo son los cables que unen unas placas con otras, sino también los distintos interfaces o puertos que comunican los módulos con el bus.

Futurebus+ no utiliza la lógica TTL para la transmisión de datos a través de las líneas, sino que utiliza lógica BTL (Bus Transceiver Logic) que se caracteriza porque los niveles lógicos son 1 y 2 voltios en vez de los 0 y 5 voltios de TTL [6]. Con esto, además de ciertas restricciones en la impedancia característica del backplane, se consigue una velocidad de transmisión de datos mucho mayor que en cualquier bus basado en lógica TTL. Para poder convertir la lógica TTL de los elementos del sistema (memorias, procesadores, etc.) es necesario disponer de unos transceivers que conviertan los niveles TTL a BTL. Pues bien, estos transceivers pueden fallar, y además pueden fallar de manera que impidan al bus seguir funcionando. Por ejemplo, puede ocurrir que uno de los transceiver de protocolo de arbitraje se estropee y la señal de una de las líneas de sincronismo se ponga a uno de forma permanente, como el sincronismo entre los módulos se detecta por la puesta a cero de las líneas de sincronismo, el arbitraje quedará paralizado ya que esta línea jamás se pondrá a cero, por tanto se producirá un error de "time-out" y no se podrá recuperar el sistema ya que sólo se dispone de un bus. En un caso como éste,

el sistema podría seguir funcionando si se dispusiera de otro bus sobre el cual poder seguir operando mientras se repara o sustituye el otro bus.

Los "time-outs" no son más que uno de los muchos mecanismos de detección de fallos que existen en Futurebus+. Otro de estos mecanisnos es que todas las líneas del bus están protegidas por bits de paridad, no sólo las de datos, sino también las de control. El bus a su vez, posee mecanismos de reintento en la transmisión de datos para evitar detener el sistema en caso de fallos transitorios en las líneas. Es posible implementar un contador de reintentos para fijar un número máximo y a partir de ahí notificar un fallo en el bus pasando a la fase de reconfiguración del sistema.

Futurebus+ posee los protocolos necesarios para poder realizar sistemas que tengan dos o más buses, incluso sistemas más complejos en los que varios buses están dispuestos de forma gerárquica, formando anillos, etc. Todo esto se consigue gracias a que está basado en la arquitectura de registros CSR, que impide conflictos en el mapeado de memoria de cada bus, además de que permiten una identificación unívoca de cada bus dentro del sistema. Hasta 1024 buses puede contener un sistema basado en Futurebus+. También se ha tenido en cuenta la coherencia de cache entre varios buses, de manera que no haya conflicto a pesar de que la memoria principal se halle en un bus diferente al que posee el procesador que esté realizando una operación sobre la cache. Las posibles combinaciones son múltiples y analizaremos aquí sólo el caso más simple de dos buses.

Un posible sistema de este tipo podría consistir en dos buses a los que se conectan varios módulos con doble puerto, fig. 2. En él, tanto los procesadores como la memoria y las entrada/salidas comparten los dos buses de manera que en el caso en que falle un bus se puede usar el otro. En el caso más simple ambos buses realizan la misma tarea, si bien el

mayor rendimiento se obtiene cuando ambos buses funcionan de forma independiente de manera que el número de transacciones dentro del sistema se ve multiplicado por dos.

Otro caso muy interesante de sistema basado en dos buses es la arquitectura FASST (Fault-tolerant Architecture with Stable Storage Technology) fig. 3, que a diferencia del anterior sistema los módulos procesadores están conectados a un único bus. Si un procesador falla sus tareas son tomadas por otro procesador mediante una técnica de checkpointing que consiste en guardar periódicamente las variables claves de un procesador para así poder recuperar la tarea que estuviera realizando. Si falla un bus la operación se pasa al bus que queda libre. En un sistema como este es necesario disponer de un puente o bridge que comunique ambos buses para permitir la propagación de mensajes o interrupciones entre procesadores conectados a distintos buses. Otras funciones de este Bridge serían las de aislar un bus de otro en caso de fallo y monitorizar ambos buses para detectar fallos.

Libre inserción y extracción de módulos

Dos de los objetivos de los sistemas tolerantes a fallos y de alta disponibilidad son el de presentar resultados que correspondan siempre con las especificaciones, y el que el sistema esté en funcionamiento continuo, es decir, que el sistema sea fiable y siempre disponible. Precisamente para que un sistema tolerante a fallos funcione con el mínimo de interrupciones posibles es necesario disponer de los mecanismos de mantenimiento "on line" necesarios para evitar tener que detener el sistema en caso de avería. Pongamos el ejemplo del sistema FASST comentado anteriormente. Supongamos que el sistema sólo tiene dos elementos procesadores. En el caso en que uno de ellos falle, las tareas que estuviera realizando se pasan al procesador que queda, con lo cual, el sistema pasa de

tener dos procesadores a tener sólo uno. En este caso el sistema deja de ser tolerante a fallos y se encuentra en una fase crítica en la que debe funcionar el menor tiempo posible para evitar averías que pararían el sistema. Para ello es necesario sustituir el módulo averiado por otro y, además, hacerlo de manera que no se tenga que detener el sistema. Afortunadamente, Futurebus+ permite extraer e insertar módulos con el sistema en funcionamiento [2]. Esto lo hace especialmente interesante en sistemas, no sólo tolerantes a fallos, sino también de tiempo real, ya que en general son sistemas que no deben ser interrumpidos, bien porque realicen una tarea crítica o bien porque una interrupción en el sistema supone una pérdida considerable de dinero.

La libre inserción y extracción en Futurebus+ se consigue a través de unos protocolos especiales. Estos protocolos permiten avisar al sistema de que un módulo va a ser insertado o extraído. Existen cuatro niveles de inserción/extracción. En el nivel más bajo la inserción o extracción de módulos no está permitida, bien porque no se han dispuesto los mecanismos para ello, o bien porque no se desea interferir en el funcionamiento del sistema para evitar transitorios. El siguiente nivel permite la extracción o inserción pero con la condición de que se detengan las transacciones en el bus. Un nivel superior permite ciertas transacciones no críticas, y por último, el nivel más alto, permite extraer o insertar los módulos en cualquier caso. Estos niveles han sido especificados debido al riesgo que existe de que se produzcan transitorios en el momento de insertar o extraer módulos. Así, para sistemas tolerantes a fallos, se recomienda el uso del segundo nivel donde al detener las transacciones, no hay peligro de que la extracción produzca un error. El primer nivel es sin duda el más seguro, pero dejaríamos al sistema sin una de las principales característica que Futurebus+ ofrece para sistemas de alta disponibilidad, como es, la libre inserción y extracción de módulos con el sistema en marcha.

Conclusiones

Futurebus+ es quizá el estándar que mejor se adapta a los sistemas tolerantes a fallos, si bien es un bus de propósito general. De la misma manera podríamos decir que es uno de los mejores buses para tiempo real [5]. Esto demuestra la versatilidad de este bus que está llamado a ser el bus de propósito general de los próximos años. Como ejemplo sirva el que el Pentágono anunció su elección de Futurebus+ como la base para todos los futuros computadores en misiones críticas de la armada norteamericana.

Futurebus+ es todavía un bus muy nuevo por lo que existen pocas realizaciones comerciales. Sus altas prestaciones lo hacen un bus prohibitivo para implementaciones de sistemas sencillos ya que este rendimiento superior a cualquier bus industrial tiene un precio alto. Por poner un ejemplo, Futurebus+ puede transmitir datos a 4 veces la velocidad del VME, pero esto es con la tecnología actual, dentro de unos años esta velocidad puede ser incluso mayor ya que en Futurebus+ la velocidad sólo depende de la tecnología de los componentes electrónicos.

Ya existen en el mercado conjuntos de chips para simplificar la tarea de hacer módulos procesadores, tarjetas de memorias, etc. basados en Futurebus+. En particular hay tres firmas que han apostado por Futurebus+, por un lado National Semiconductors que ofrece un controlador de arbitraje distribuido, por otro Phillips-Signetics, que presenta chips de arbitraje para el bus tanto centrales como distribuidos, un controlador de protocolo paralelo, y FIFOs para la transmisión en modo empaquetado. Por último, Texas Instruments presenta el mejor conjunto de chips para Futurebus+, permitiendo realizar en poco tiempo un interface entre el bus y memorias, procesadores, etc. La disponibilidad de estos chips es escasa todavía, especialmente en Europa ya que las empresas Norteamericanas están intentando monopolizar todos los productos basados en

Futurebus+. Algunas de ellas son FORCE computers, o SGS Thomson que ya presentan algunos productos como módulos procesadores, tarjetas de memorias y sistemas completos basados en Futurebus+. Compañías como Mupac y BICC-VERO fabrican el back-plane, y en concreto Mupac también comercializa tarjetas de prototipo que incluyen los transceivers. El grupo DECV en Valencia trabaja también en Futurebus+ en la realización de un procesador dual para sistemas tolerantes a fallos tipo FASST, una memoria central genérica con código corrector de errores para sistemas de memoria compartida, y tarjetas de prototipación que incluyen no sólo los transceivers BTL sino también los chips de protocolo de Texas Instruments.

Las especificaciones concretas de Futurebus+ se encuentran en los estándares del IEEE publicados al efecto, destacan el 896.1, 896.2 y 896.3 que hacen referencia directa a las especificaciones de Futurebus+. Otro estándar interesante es el IEEE 1212 que hace referencia a la arquitectura de registros CSR.

Referencias

[1] IEEE 896.1 "Logical Layer Specification", 1991.

[2] IEEE 896.2 "Physical Layer and Profile Specifications", 1991.

[3] IEEE P896.3 "System Design Guide", 1991.

[4] IEEE P1212 "Control and Status Register Architecture", August, 1991.

[5] Lui Sha, Ragunathan Rajkumar, John P.Lehoczky, "Real-Time Computing with IEEE Futurebus+", IEEE Micro, June 1991 pp. 30-33.

[6] R.V.Balakrishnan, "The Proposed IEEE 896 Futurebus-A Solution to the Bus Driving Problem", IEEE Micro, August 1984 pp 23-27.

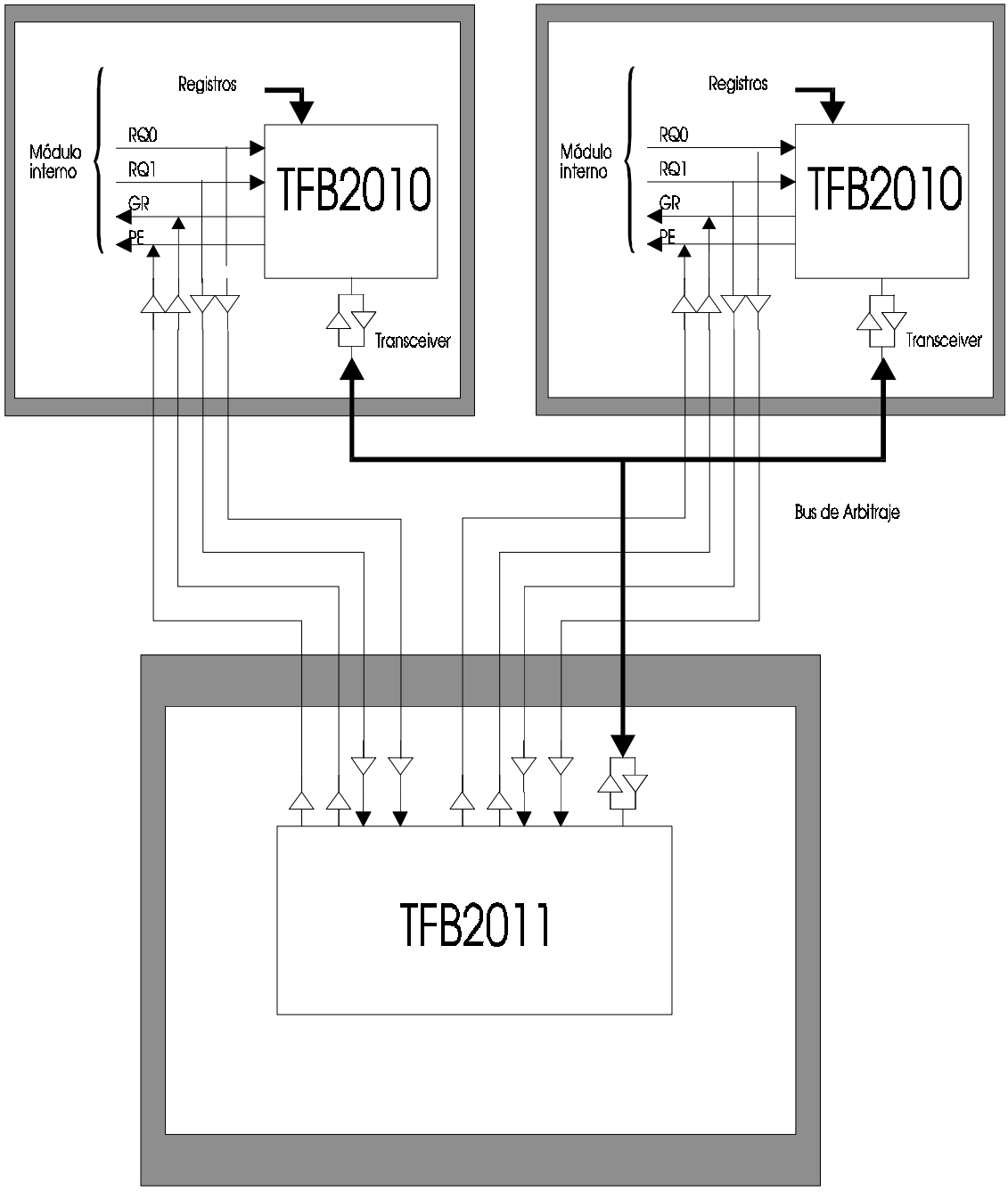
[7] Texas Instruments, "Futurebus+ Interface Family", February 1993.

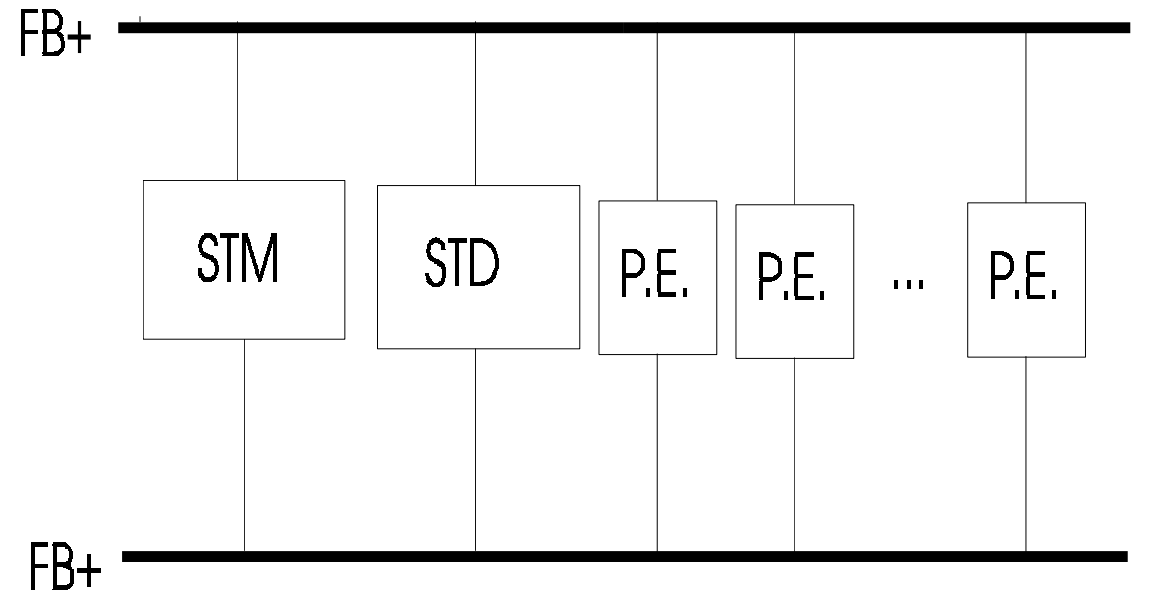
Para cualquier pregunta sobre el presente artículo dirigirse a Fernando Pardo, DECV, C/ Hugo de Moncada 4, Entrl. 46010 Valencia. Email: pardo@vm.ci.uv.es

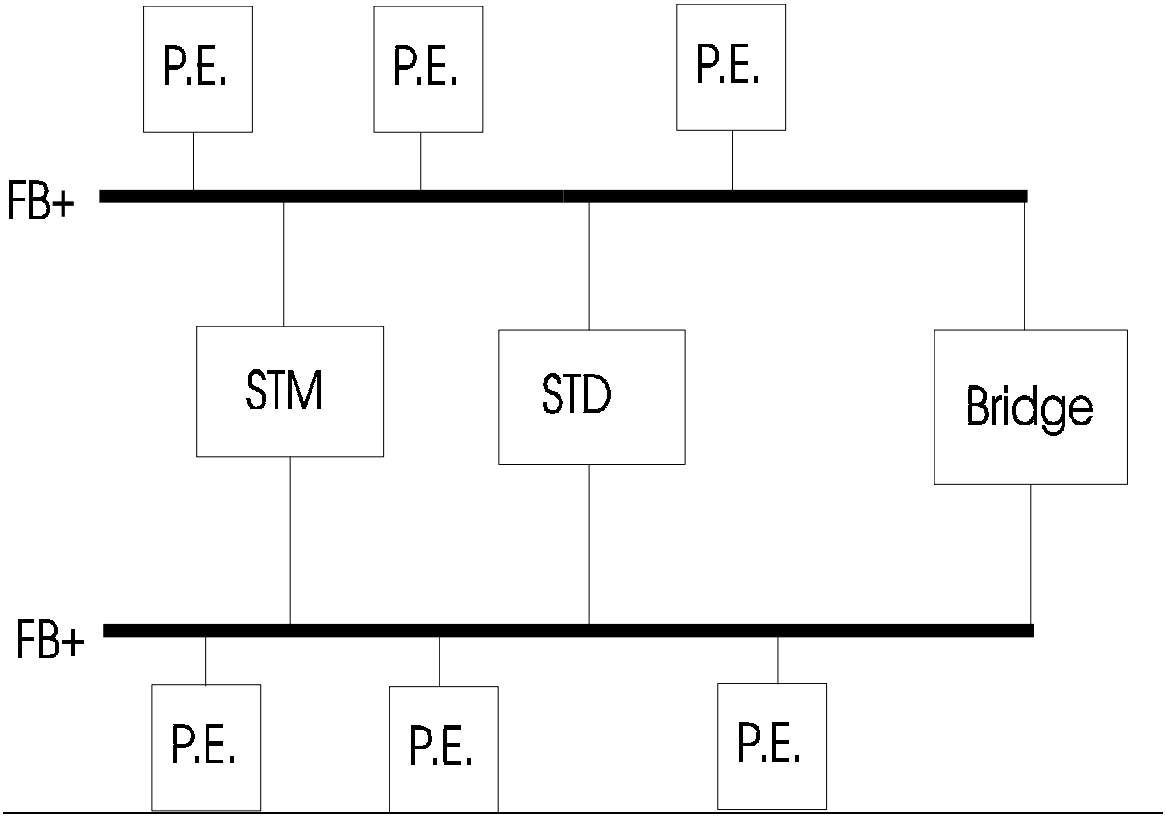
Fig. 1 Implementación del arbitraje en modos centralizado y distribuido.

Fig. 2 Arquitectura de dos buses, PE=Elementos Procesadores (Fail-stop o Fail-safe), STD=Disco estable, STM=Memoria estable.

Fig. 3 Arquitectura FASST, PE=Elementos Procesadores (Fail-stop), STD=Disco estable, STM=Memoria estable, Bridge=Puente entre los dos Futurebus+.


Registros
Módulo interno
RQ0
RQ1
GR
PE
TFB2010
Transceiver
Registros
Módulo interno
RQ0
RQ1
GR
PE
TFB2010
Transceiver
Bus de Arbitraje
TFB2011


FB+
STM
STD
P.E.
P.E.
...
P.E.
FB+


P.E.
P.E.
P.E.
FB+
STM
STD
Bridge
FB+
P.E.
P.E.
P.E.