(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 689 125**

(51) Int. CI.:

G06F 9/52 (2006.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(86) Fecha de presentación y número de la solicitud internacional: **27.11.2009 PCT/FR2009/052322**

(87) Fecha y número de publicación internacional: **24.06.2010 WO10070222**

(96) Fecha de presentación y número de la solicitud europea: **27.11.2009 E 09797109 (7)**

(97) Fecha y número de publicación de la concesión europea: **09.05.2018 EP 2366147**

(54) Título: **Gestor físico de barrera de sincronización entre procesos múltiples**

(30) Prioridad:

16.12.2008 FR 0807089

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
08.11.2018

(73) Titular/es:

BULL SAS (100.0%)
Rue Jean Jaurès
78340 Les Clayes-sous-Bois, FR

(72) Inventor/es:

SOLINAS, ANGELO;
CHICHEPORTICHE, JORDAN;
DERRADJI, SAÏD;
PAIRAULT, JEAN-JACQUES;
MENYHART, ZOLTAN;
JEAUGEY, SYLVAIN y
COUVEE, PHILIPPE

(74) Agente/Representante:

AZNÁREZ URBIETA, Pablo

DESCRIPCIÓN

Gestor físico de barrera de sincronización entre procesos múltiples

La presente invención se refiere al tratamiento de procesos ejecutados en paralelo.

Determinados *software* o programas informáticos tardan un tiempo considerable en ejecutar o realizar una tarea dada. Para ser más eficaz y reducir el tiempo de cálculo, estos programas pueden aprovechar el carácter paralelo de los ordenadores en los que son ejecutados. Por carácter paralelo de un ordenador se entiende un ordenador en el que están montados varios procesadores o al menos un procesador con varios núcleos o al menos un procesador con varias colas de ejecución ("threads").

Para aprovechar el carácter paralelo, un programa informático divide su tarea (o tarea principal) en varias subtareas cuyos cálculos pueden ser realizados en paralelo mediante diferentes procesos. Por tanto, cada proceso tendrá por objeto ejecutar y realizar una de estas subtareas. Una vez que un proceso ha terminado su subtarea en curso, se le podrá atribuir una segunda subtarea a realizar, tras la cual eventualmente le será atribuida una siguiente subtarea y así sucesivamente.

La utilización de múltiples procesos (procesamiento multiproceso) conlleva la necesidad de sincronizar los mismos. Esta sincronización tiene por objeto permitir una reestructuración ordenada de la tarea principal una vez realizadas las subtareas.

En general, una sincronización de este tipo se consigue mediante un mecanismo denominado "mecanismo de sincronización inter-proceso". Este mecanismo debe ser rápido para no anular la ventaja temporal obtenida con el uso de procesos ejecutados en paralelo.

Para realizar la sincronización arriba mencionada, es conocido un mecanismo de carácter lógico denominado "mecanismo de barrera". Este mecanismo se puede basar en diversos algoritmos que siguen un mismo esquema principal, descrito más abajo.

En un primer momento, un programa informático destinado a realizar una tarea es ejecutado mediante *n* procesos, cada uno capaz ejecutar un conjunto de subtareas. Cada subtarea se divide en bloques sucesivos destinados a realizar fases de trabajo, tales como un cálculo intermedio, por ejemplo. Así, los bloques o cálculos intermedios de los diferentes procesos se ejecutan en paralelo. Cada proceso que ha terminado un bloque se pone en espera a nivel de una barrera (barrera de sincronización) hasta que todos los otros bloques paralelos de los otros procesos han terminado y se han unido a su vez a la barrera. Únicamente cuando todos los procesos han llegado a la barrera, los bloques siguientes son ejecutados durante una fase de trabajo posterior. Este principio se describe más abajo con ayuda de un diagrama temporal.

La figura 1 muestra un mecanismo de barrera y el funcionamiento general de una barrera de sincronización. A partir de una tarea principal T, en un primer momento un gestor de procesos PM descompondrá la tarea T en *n* subtareas ST. Estas *n* subtareas ST serán ejecutadas por *n* procesos P. Dicho de otro modo, la tarea principal T compleja se descompone en varias subtareas ST simples, siendo realizada cada una de estas subtareas por un proceso independiente.

Finalmente, los resultados obtenidos de las diferentes subtareas ST ejecutadas por los procesos P se reunirán con el fin de cumplir la tarea principal T.

Se ha de señalar que el concepto de gestor de procesos PM se debe entender en un sentido amplio. Por tanto, el gestor PM no es necesariamente un elemento propio. En efecto, el gestor de procesos generalmente puede ser considerado como la capacidad de un programa informático para implementar un método de desglose pasivo o activo con el fin de permitir que los procesos se repartan las subtareas entre ellos. La capacidad puede ser implícita, determinada por uno de los procesos, o incluso corresponder a un desglose predefinido por un usuario.

Tal como se menciona más arriba, durante la descomposición de una tarea en múltiples procesos P_j, existe una necesidad de sincronización en la ejecución en paralelo de estos diferentes procesos. Para ello, los propios *n* procesos se dividen en bloques B que deben ser ejecutados sucesivamente en el tiempo. El subconjunto de los bloques B que están en ejecución al mismo tiempo (y que resultan de diferentes procesos P) constituye una fase de trabajo W. Por consiguiente, cada conjunto de bloques B de un mismo rango *i* constituye una fase de trabajo W independiente.

Los bloques B_i de la fase de trabajo de rango *i*, designada W_i, se ejecutan en paralelo. El tiempo t de ejecución de los bloques B resultantes de diferentes procesos P_j es variable. Para asegurar la sincronización arriba mencionada, los bloques B son sometidos a una barrera de sincronización BS (100). Para cada proceso P se llama a esta barrera BS (100) cuando éste ha terminado de ejecutar su bloque B_i en curso. Es la barrera de sincronización BS (100) la que autoriza un paso al bloque B_{i+1} de un siguiente rango y ello únicamente cuando todos los bloques B_i en curso se hayan "unido" a la barrera, es decir cuando hayan informado a ésta que su ejecución ha terminado.

El primer bloque B terminado, es decir el que tiene el tiempo t de ejecución más corto, informa por solicitud a la barrera de sincronización BS (100) por un lado de que ha terminado su trabajo y, por otro lado, del número de bloques en curso que quedan durante la misma fase de trabajo. En general, el número de bloques durante una fase de trabajo es equivalente al número *n* de procesos P.

Normalmente, las barreras de sincronización están provistas de un contador. El contador se inicializa cuando el primer bloque B se ha unido a la barrera. Después, el contador disminuye cada vez que otro bloque B se une a la barrera BS (100). Por tanto, la barrera BS (100) puede seguir el progreso (o avance) de una fase de trabajo y más concretamente la finalización de cada bloque B en curso. Cuando el último bloque B, es decir el que tiene un tiempo t de ejecución más alto, se une a la barrera BS (100), ésta informa a cada proceso P y autoriza a los mismos a pasar a una siguiente fase de trabajo W. De nuevo, esta siguiente fase de trabajo W está formada por bloques B ejecutados en paralelo y resultantes de los diferentes procesos P. Durante esta siguiente fase de trabajo, el mecanismo de la barrera BS (100) es análogo al precedente. Esto se repite para cada fase de trabajo y continúa hasta la finalización de los procesos P. La tarea T se realizará entonces por reconstitución de los resultados de los procesos P.

Dichos algoritmos requieren un número determinado de interacciones entre los procesos, bloques y barrera. Estas interacciones se describirán más adelante en la descripción detallada y comprenden en particular la inicialización de la barrera, la información proporcionada a la barrera cuando un bloque ha terminado su trabajo, la verificación de que todos los subprocesos han terminado su bloque en curso. Estas interacciones, cuando son gestionadas por barreras de tipo *software*, son relativamente lentas y consumen mucho ancho de banda.

La figura 2, relativa a la técnica anterior, representa una implementación de barrera de sincronización BS (100) conocida. Los mecanismos conocidos son implementados en *software*. Así, los datos que definen la barrera de sincronización BS (100) están almacenados en la memoria RAM (202) (significado de RAM en inglés: *Random Access Memory*) de un ordenador (o de otro dispositivo informático) y los diferentes procesos P acceden (por lectura/escritura R/W) a esta memoria RAM (202) para interactuar con dicha barrera BS (100). Este acceso se realiza por medio de un espacio de dirección y de una dirección ADR (detallada más abajo). El acceso comprende, tal como se describe más arriba, la inicialización de la barrera BS (100) (con inicialización del contador), el hecho de informar a la barrera BS (100) cada vez que un bloque B ha terminado su trabajo durante una misma fase de trabajo W, verificar si todos los procesos P han terminado su bloque B de la fase de trabajo W en curso, etc. El programa destinado a realizar estas funciones también está activo en la memoria de acceso aleatorio, en particular llamando a una biblioteca de funciones.

Un espacio de dirección puede estar segmentado en segmentos independientes. En general, por segmento se entiende un segmento de memoria definido por dos valores:

- la dirección en la que comienza este segmento (dirección de base), y
- el tamaño del segmento.

Por tanto, un segmento constituye un intervalo de direcciones continuas en una memoria principal (física o virtual).

La figura 2 muestra un dispositivo informático que comprende varios procesadores PZ_1 a PZ_y (200), un gestor de acceso a memoria CACHE COHER MGR (206), una memoria RAM (202) que contiene una zona de programa en la que se encuentra la barrera de sincronización BS (100) de tipo *software*. Por tanto, el dispositivo según la figura 2 comprende una unidad de procesamiento capaz de procesar multiprocesos. Los procesos se ejecutarán en diferentes procesadores, en diferentes núcleos ("cores") de procesadores y/o incluso en diferentes colas de ejecución ("threads"). La unidad de procesamiento proporciona a estos procesadores lo que se denomina como un "espacio de dirección", en particular hacia la memoria de acceso aleatorio, donde se encuentran el código y los datos que definen la barrera de sincronización BS (100) de tipo *software*, en una zona asociada a una dirección precisa ADR, que puede ser la dirección de comienzo de la zona. El dispositivo de la figura 2 comprende además un gestor de procesos (208) del tipo definido más arriba para descomponer una tarea T en n procesos P, a su vez divididos en bloques B sucesivos.

Las barreras de la técnica anterior (figura 2) permiten ejecutar una sincronización entre diferentes procesos P. Pero, como ya se ha mencionado, el carácter de *software* de una barrera hace que ésta sea lenta con respecto

a determinadas necesidades. En efecto, cada vez que un proceso P interactúa con la misma, se llama a una biblioteca de funciones de la barrera BS (100). Además, dentro de la memoria se necesitan numerosas interacciones con la memoria para leer y escribir los datos de actualización de la barrera, hasta que se detecta que todos los procesos han alcanzado el punto de encuentro ("barrera de sincronización"). Posteriormente, una vez que el proceso P ha informado a la barrera BS (100), el proceso P debe consultar regularmente a la barrera BS (100) para ver si los otros bloques B en curso han terminado su trabajo.

Todo esto, y en particular las numerosas interacciones arriba mencionadas, hace que las barreras de sincronización BS (100) de tipo *software* sean lentas y consuman ancho de banda. Esto se traduce en pérdidas de ciclos de reloj, lo que resulta molesto en especial, ya que el modo multiproceso se utiliza para ir más rápido.

Además, existe el riesgo de que diferentes bloques pertenecientes a procesos respectivos independientes informen a la barrera al mismo tiempo. De ello se derivan conflictos de acceso a memoria que generan problemas adicionales de latencia y de ancho de banda (gestión de los conflictos por el CACHE COHER MGR).

El documento "Dynamic Barrier Architecture For Multi-Mode Fine-Grain Parallelism Using Conventionnal Processors", de W. E. COHEN, H. G. DIETZ, J. B. SPONAUGLE, describe la transmisión por un procesador de una solicitud formateada como una solicitud de lectura de una memoria. Sin embargo, esta solicitud es transmitida a una barrera de sincronización que incluye un árbol lógico asociado al procesador. El árbol lógico está concebido para indicar al procesador cuándo todos los procesadores implicados en la sincronización han alcanzado la barrera de sincronización, con el fin de que el procesador continúe su ejecución.

La presente invención mejora la situación.

Con este fin, la invención proporciona un dispositivo informático de barrera de sincronización según la reivindicación 1.

En las reivindicaciones 2 a 13 se mencionan características opcionales.

La presente invención también proporciona un procedimiento de procesamiento informático según la reivindicación 14.

En la reivindicación 15 se mencionan características opcionales.

Otras características y ventajas de la invención se desprenderán del examen de la siguiente descripción detallada y de las figuras adjuntas, en las que:

- la figura 1 es un diagrama temporal que ilustra el funcionamiento general de un mecanismo de barrera,
- la figura 2 es el diagrama esquemático de una implementación de una barrera de sincronización de tipo *software* de la técnica anterior,
- la figura 3 representa un dispositivo informático que incluye una memoria y una unidad de procesamiento y que puede realizar procesamientos multiproceso en diferentes procesadores con un circuito de *hardware* que forma un gestor de barrera de sincronización,
- la figura 4 representa un circuito de *hardware* que forma un gestor de barrera de sincronización que incluye una memoria dedicada y un microprograma,
- la figura 5 representa un automatismo de barrera de sincronización según un modo de realización de la invención, y
- la figura 6 representa un organigrama de las principales operaciones según una forma de realización de la invención.

Las figuras y la siguiente descripción contienen esencialmente elementos de carácter determinado. Por tanto, no solo podrán servir para comprender mejor la invención, sino también para contribuir a su definición, dado el caso.

La Solicitante ha logrado superar los problemas de la técnica anterior mencionados y propone una barrera de carácter físico o *hardware*. Ahora se describirá en referencia a la figura 3, que supone una barrera de carácter físico o *hardware* de este tipo. Así, el dispositivo informático de la figura 3 comprende una memoria RAM (202), una unidad de procesamiento capaz de procesar multiprocesos en diferentes procesadores PZ_1 a PZ_y (200) y un gestor de acceso a memoria COHER CACHE MGR (206) entre dicha memoria RAM (202) y los procesadores PZ (200).

En la forma de realización aquí descrita, el dispositivo comprende además un circuito de *hardware* que forma un gestor de barrera de sincronización HBM (400), el cual comprende una memoria dedicada Ded_MEM (404) y un microprograma micro-Prog (402) tal como se muestra en la figura 4. En esta etapa, el gestor HBM (400)

solo necesita una salida de datos “D (unidir)”. En la práctica se trata de una entrada/salida principalmente por motivos de compatibilidad (lectura/escritura R/W) con el bus conectado.

En la forma de realización descrita, los enlaces dirección/datos hacia el circuito de *hardware* (HBM, 400) evitan el gestor de acceso a memoria COHER CACHE MGR (206).

En términos generales, el gestor de barrera de sincronización HBM (400) interactuará directamente con los procesos P que participan en la barrera BS (100). Después de la interacción puede tener lugar un almacenamiento de datos en la memoria dedicada Ded_MEM (404).

El gestor de barrera de sincronización HBM (400) se puede encontrar por ejemplo en un procesador, en un conjunto de chips («chipset» u otro) o, como se muestra en la figura 3, dentro de un componente adicional, tal como un circuito de *hardware*. El gestor HBM (400) debe estar accesible para cualquier transacción resultante de los procesos P que participan en la barrera BS (100) y que están dirigidos a este gestor. Por tanto, se puede acceder o llamar al gestor HBM (400) para cualquier solicitud dirigida a su espacio de memoria. Así es como múltiples direcciones pueden estar dirigidas a la misma barrera BS (100) ("address aliasing").

Dicho de otro modo, cada proceso P que emite una solicitud a la barrera de sincronización BS (100) porta:

- en los bits superiores de esta solicitud, la dirección de la barrera, y
- en los bits inferiores, datos adicionales.

Evidentemente, es posible organizar libremente en los bits superiores o inferiores (bits elegidos) la localización de las informaciones arriba mencionadas (dirección y datos). Así, los bits superiores de la solicitud pueden portar dichos datos adicionales y los bits inferiores la dirección de la barrera.

Un ejemplo de datos adicionales puede ser el número de procesos P que participan en la barrera BS (100). De este modo, cada uno de los procesos P se puede dirigir a una misma barrera BS (100) comunicándole informaciones necesarias para la sincronización. Estas informaciones pueden ser almacenadas por el microprograma micro-Prog (402) en su memoria dedicada Ded_MEM (404) y a continuación procesadas por el microprograma micro-Prog (402) del gestor de barrera de sincronización HBM (400).

Aplicando este principio, el gestor de barrera de sincronización HBM (400) puede gestionar varias barreras de sincronización BS (100) a la vez. Esta posibilidad es importante en determinadas aplicaciones.

Consideremos ahora un grupo de n procesos P que utilizan para su sincronización una barrera física. En una primera etapa, la barrera BS (100) está en su estado inicial y ninguno de los n procesos P ha accedido a la misma. Los procesos P se encuentran en una primera fase de trabajo W y cada uno ejecuta sus primeros bloques B (véase la figura 1). De modo similar al arriba descrito, cuando el primer proceso P ha terminado su bloque B, informa a la barrera BS (100) por medio de una solicitud. Esta solicitud incluye en sus bits inferiores el número *n* de procesos P que participan en la barrera, lo que permite la inicialización de un contador CNT (406) de la barrera BS (100) una vez recibida la primera solicitud. Cuando la barrera de sincronización BS (100) recibe esta primera solicitud, pasa al modo (o estado) activado. A partir de ahí, cada vez que se dirige una solicitud a la barrera BS (100), el gestor de barrera de sincronización HBM (400) disminuye (cuenta atrás) el contador CNT (406). Solo se responde a las solicitudes sobre los datos D cuando el gestor de barrera de sincronización HBM (400) ha recibido todas las solicitudes procedentes de los n procesos P que participan en la barrera BS (100). En ese momento, la sincronización se considera efectiva. Entonces se autoriza que el conjunto de los procesos P pase a la siguiente fase de trabajo W.

Obsérvese que, una vez terminado un bloque, el proceso correspondiente solo consulta una vez a la barrera BS para determinar el avance de la fase de trabajo W. Esto se debe al hecho de que la barrera BS puede almacenar en su espacio de memoria Ded_MEM (40) propio el número de solicitudes ya recibidas. Cada proceso permanecerá en espera hasta la recepción de la respuesta procedente de la barrera BS. Por tanto, no se produce ninguna consulta múltiple (regular o no) de los procesos a la barrera. Además, cada consulta es menos costosa en términos de ancho de banda. Esta es la causa de la ganancia de ancho de banda lograda por la invención.

Por cierto, en este contexto se ha de señalar que el tiempo de ejecución t de un bloque B no está relacionado necesariamente con la llegada de éste a la barrera de sincronización BS. En efecto, por motivos de competencia, de no ordenamiento de las vías de comunicación, de conflictos o incluso de arbitraje, una segunda solicitud que haya salido más tarde que una primera solicitud puede llegar a la barrera BS antes que dicha primera solicitud. Sin embargo, esto no cambia nada en el funcionamiento de la barrera según la invención. Por motivos de simplicidad, en la presente descripción se considera que una solicitud emitida por un primer

proceso que tiene un tiempo de ejecución t más corto que un segundo proceso llegará a la barrera BS antes que la solicitud emitida por el segundo proceso.

En una forma de realización de la invención, el espacio de memoria de la barrera de sincronización BS (100) está implementado en el espacio de memoria destinado al bus PCI del ordenador.

En este ejemplo, lo que se ha denominado "solicitud" procede de una instrucción "load" del procesador con una dirección del espacio de memoria del bus PCI. Esta solicitud es un mensaje sobre el bus de sistema. Este espacio de memoria permite una interacción rápida entre los procesos P y/o la solicitud y la barrera de sincronización BS (100).

Si se requieren varias barreras, puede resultar ventajoso que el gestor de barrera de sincronización gestione estas barreras en relación con segmentos de memoria, por ejemplo páginas de memoria. Estas múltiples barreras pueden estar ramificadas en un mismo circuito o en circuitos independientes.

Así, la memoria PCI ofrece espacio suficiente para prever un tamaño predeterminado de página de memoria para cada barrera mientras se permite suministrar un acceso protegido entre barreras.

Por ejemplo, para páginas de 64 KB (*kilobytes*), esto permite utilizar los 16 bits inferiores de una solicitud (llamada)) para transmitir datos (en particular ADR). Por tanto, el gestor de barrera de sincronización HBM (400) puede albergar M·64KB páginas, donde M es el número de barreras BS (100) físicas implementadas en el gestor de barrera de sincronización HBM (400). M puede tener en particular un valor de 512, con lo que se alcanza un espacio de memoria total de 32 MB (*megabytes*). Evidentemente, estos 32 MB corresponden a una memoria de tipo virtual que, por tanto, no deben considerarse como MB "verdaderos", sino que simplemente son vistos como tales por la aplicación a sincronizar. A continuación se muestra un ejemplo de una composición de una solicitud que puede emplearse para acceder a la memoria (R [J..I] = bits de la solicitud I a J). Esta solicitud incluye en particular la dirección de la barrera (100), una instrucción en ejecución (detallada más adelante), la indicación de si se trata de una sincronización a un nivel o a varios niveles (detallada más abajo) y el número de procesos que participan en la sincronización y, por tanto, en la barrera.

R[63..16]	R[15..9]	R[8]	R[7..0]
Dirección (ADR) de la barrera BS	Instrucción	Nivel(es) de sincronización; bit: 1 → 2 niveles 0 → 1 nivel	Número de procesos que participan en la barrera

En el bit R[8], los valores 0 o 1 corresponden respectivamente a una sincronización a un nivel y una sincronización a dos niveles. En el ejemplo de realización mostrado más abajo se detalla un nivel de sincronización superior.

La figura 5 se refiere a un ejemplo de realización de un gestor de sincronización HBM (400) capaz de gestionar una sincronización de nivel superior, más concretamente de dos niveles en este caso. Puede emplearse por ejemplo una sincronización de dos niveles cuando se deben sincronizar varios grupos independientes de procesos P, teniendo cada uno de los grupos una barrera BS (100) física (o de *hardware*). En este caso, el gestor de sincronización HBM (400) debe gestionar el caso en que cada grupo debe ser sincronizado por sí solo y después los grupos del conjunto deben ser sincronizados entre sí.

La primera solicitud recibida por la barrera BS (100) tiene un estado listo PRE, que incluye en sus bits inferiores una información que indica si se trata de una sincronización de uno o de dos niveles. Si se trata de una sincronización de un nivel, ésta va a ser gestionada por una barrera de un nivel, o más concretamente por un estado activo ACT de la barrera concebida para un nivel (estado ACT_1_N). Por el contrario, si se trata de una sincronización de dos niveles, esta misma barrera entrará en un estado activo ACT concebido para dos niveles (estado ACT_2_N), en cuyo caso su comportamiento será tal como se describe a continuación:

Cuando todas las solicitudes han sido recibidas por la barrera BS (100), ésta elige uno de los procesos P como maestro M entre todos los procesos P que participan en la barrera BS (100). En un primer momento, únicamente la solicitud del maestro M responderá mediante un dato D especial que indica que es el maestro del grupo. A partir de ahí, el maestro tiene libertad para realizar el segundo nivel de sincronización. Este segundo nivel de sincronización puede ser por ejemplo una barrera BS (100) de tipo *software*. Cuando el maestro M ha terminado este segundo nivel de sincronización, transmite una última solicitud a la barrera BS (100). En respuesta a esta última solicitud, la barrera responde a todas las demás solicitudes procedentes de los otros procesos P que participan en la barrera BS (100) (incluyendo el maestro M) y vuelve a su estado listo PRE. El maestro M es dinámico y se puede redefinir en cada sincronización.

Ejemplo de realización

Los diferentes estados del automatismo de barrera mostrados en la figura 5 son los siguientes:

- estado de espera INACT,
- estado listo PRE,
- estado activo con sincronización a un nivel ACT_1_N,
- estado activo con sincronización a dos niveles ACT_2_N,
- estado de sincronización SYNC y
- estado de anulación ANN.

Más abajo se detalla cada uno de estos estados.

• **INACT**

La barrera física está en estado de espera e inactiva. La única transición posible es la transición T0. Esta transición corresponde a la recepción de la barrera de una solicitud con una instrucción de paso al estado listo PRE, llamada PREPA (instrucción = PREPA) para activar la barrera. La barrera pasa al estado listo PRE.

• **PRE**

La barrera física está lista para recibir solicitudes de los procesos que participan en la barrera.

De acuerdo con la forma de realización descrita pueden tener lugar tres transiciones: T1, T2 o T13. De acuerdo con la solicitud, la barrera elegirá cuál de las transiciones se debe efectuar.

- Transición T1: esta transición corresponde a la recepción por la barrera de una solicitud que contiene una instrucción de registro ENREGISTRER (instrucción = ENREGISTRER) para inicializar la barrera. La solicitud comprende en sus bits inferiores la información de que se requiere una sincronización a un solo nivel (SYNC_1_N). La barrera se activa y se encuentra entonces en estado activo con sincronización a un nivel ACT_1_N.
- Transición T2: de modo similar a T1, esta transición corresponde a la recepción por la barrera de una solicitud con una instrucción de registro ENREGISTRER para inicializar la barrera. En cambio, la solicitud comprende en sus bits inferiores la información de que se requiere una sincronización a dos niveles (SYNC_2_N). La barrera se activa y pasa a en estado activo con sincronización a dos niveles ACT_2_N.
- Transición T13: esta transición corresponde a la recepción por la barrera de una solicitud con una instrucción de apagado ETEINDRE (instrucción = ETEINDRE) para inactivar la barrera y pasar al estado de espera INACT (véase más arriba).

• **ACT_1_N**

La barrera realiza una sincronización a un solo nivel. A partir de este estado existen varias transiciones.

- Transición T3: esta transición tiene lugar cada vez que la barrera BS (100) recibe una solicitud de un proceso P con una instrucción de registro ENREGISTRER (instrucción = ENREGISTRER), y ello antes de un tiempo límite predeterminado (detallado más abajo). El contador interno CNT (406) disminuye en cada transición T3 (CNT > valor umbral). T3 corresponde esencialmente a cada terminación de los bloques actuales B durante una misma fase de trabajo W. Los bloques B "se acumulan" (T3) a nivel de la barrera BS (100) hasta que el contador CNT (406) indica que todos los bloques B actuales han sido ejecutados (CNT = valor umbral).
- Transición T4: esta transición tiene lugar cuando el contador CNT (406) indica que todos los bloques B actuales han sido ejecutados: la barrera recibe una última instrucción de registro ENREGISTRER (instrucción = ENREGISTRER) y el contador disminuye a su valor umbral (CNT = valor umbral). Se da respuesta a las solicitudes procedentes de los procesos P que participan en la barrera BS (100). La respuesta indica el éxito de la sincronización. La barrera BS (100) vuelve al estado listo PRE.
- Transición T5: El contador debe llegar a su valor umbral antes de un tiempo límite predeterminado. La elección del umbral de un tiempo límite es variable y se realiza en función de la aplicación. Si se sobrepasa este tiempo límite predeterminado, la transición T5 permite anular la sincronización, opcionalmente devolviendo un mensaje de error o una orden de aumento del tiempo límite, por ejemplo. El tiempo límite se puede registrar previamente en una unidad de control provista de un contador de tiempo (“contador crónico”) capaz de efectuar una cuenta atrás en unidades de tiempo (por ejemplo µ).

- Transición T14: esta transición corresponde a la recepción por la barrera de una solicitud con una instrucción de apagado ETEINDRE (instrucción = ETEINDRE) para inactivar la barrera y pasar al estado de espera INACT

• **ACT_2_N**

La barrera realiza una sincronización a dos niveles. A partir de este estado existen varias transiciones.

- Transición T6: análoga a la transición T3 (véase más arriba).
- Transición T7: en un primer momento, la transición T7 es análoga a T4. En efecto, T7 tiene lugar cuando el contador CNT (406) indica que todos los bloques B actuales han sido ejecutados: La barrera recibe una última instrucción de registro ENREGISTRER (instrucción = ENREGISTRER) y el contador disminuye más y alcanza su valor umbral (CNT = valor umbral). A diferencia de la transición T4, aquí no se da respuesta al conjunto de las solicitudes procedentes de los procesos P, sino solamente a una de ellas. La respuesta consiste en elegir uno cualquiera de los procesos P como maestro M. La barrera pasa entonces al estado de sincronización SYNC (detallado más adelante). El tiempo límite predeterminado se reinicializa en el momento de T7.
- Transición T8: Análoga a la transición T5 (véase más arriba).
- Transición T15: Análoga a la transición T13 (véase más arriba).

• **SYNC**

Tres transiciones posibles:

- Transición T9: el maestro M recibe una solicitud con una instrucción de registro ENREGISTRER (instrucción = ENREGISTRER) antes del tiempo límite predeterminado (con: CNT = valor umbral). Se da respuesta al conjunto de los procesos P. La respuesta indica el éxito de la sincronización. La barrera BS (100) vuelve al estado listo PRE.
- Transición T10: Análoga a la transición T5 (véase más arriba).
- Transición T16: Análoga a la transición T13 (véase más arriba).

• **ANN**

Para fijar un tiempo óptimo (máximo aceptable) para la realización de una sincronización, la barrera BS (100) está provista de un contador de tiempo, también llamado contador crónico. El contador se puede configurar y puede describir un tiempo límite. El contador inicia una cuenta atrás (generalmente en unidades µs) al recibir la primera solicitud. A partir de ese momento comienza a correr el tiempo. Si se sobrepasa el tiempo límite predeterminado antes de la recepción de la última solicitud en la barrera BS (100), ésta pasa al estado de anulación ANN.

El tiempo límite puede variar en función de las barreras, y más concretamente en función de los diferentes estados de una barrera, en particular: ACT_1_N, ACT_2_N, SYNC.

Dicho de otro modo, si la barrera BS (100) entra en estado de anulación ANN, ello se debe a que se ha sobrepasado el límite de tiempo en el estado precedente, antes de la recepción de todas las solicitudes. La barrera responde entonces a las solicitudes ya recibidas mediante un mensaje de fallo de sincronización.

En la práctica, este tiempo límite es programable. Su límite superior se puede fijar en función del contexto, en particular para evitar interferencias con los "*time-out"* del procesador.

En el estado de anulación ANN existen tres transiciones:

- Transición T11: se recibe una solicitud con una instrucción de registro ENREGISTRER (instrucción = ENREGISTRER), en cuyo caso se da respuesta a las solicitudes mediante un mensaje de error tal como se describe más arriba.
- Transición T12: se recibe una solicitud con una instrucción que indica el retorno al estado listo PRE (instrucción = PREPA). La barrera vuelve al estado listo PRE (véase más arriba). Éste, por ejemplo, pide al conjunto de los procesos (P) que vuelvan al final de la ejecución de una fase de trabajo (W) anterior.
- Transición T17: análoga a la transición T13 (véase más arriba).

El organigrama de la figura 6 recoge las principales operaciones de una barrera de sincronización BS según una forma de realización de la invención. El organigrama muestra la barrera BS (100) en su estado PRE

(operación 700). Una vez que el primer proceso P ha terminado su bloque B, informa (mediante una llamada) a la barrera BS (100) por medio de una solicitud dirigida al gestor HBM (400) (operación 702). La solicitud comprende información sobre el o los niveles de sincronización (instrucción = ENREGISTRER para ACT_1_N o ACT_2_N, por ejemplo). El contador CNT (406) se inicializa (generalmente a n = número de procesos P) y la barrera BS (100) almacena por una parte un identificador SVE_ID_Req correspondiente a dicho primer proceso P y, por otra parte, la información sobre el o los niveles de sincronización SVE_N (operación 704). A partir de ese momento, la barrera BS (100) activada espera las siguientes llamadas de los otros procesos P (operación 706). Si se sobrepasa el tiempo límite predeterminado o si la barrera BS (100) recibe una solicitud con una instrucción de apagado (instrucción = ETEINDRE) (operación 714), la barrera pasa respectivamente al estado de anulación ANN o de espera INACT (operación 716). En cambio, si otro proceso P informa a la barrera de que ha terminado de su bloque en curso (sin sobrepasar el tiempo límite t_Lim y sin instrucción = ETEINDRE), el contador CNT (406) disminuye (operación 708, con m = número de procesos que todavía no han terminado su bloque B durante la fase de trabajo W en curso). Paralelamente a esta disminución, la barrera almacena el identificador SVE_ID_Req que corresponde al proceso P que recientemente ha informado a la barrera BS (100) (operación 708). La barrera BS (100) verifica a continuación si el contador CNT (406) ha alcanzado su valor umbral (operaciones 710 y 712). En caso negativo (operación 710; CNT > 0), la barrera vuelve al estado de espera (operación 706). En caso afirmativo (operación 712; CNT = 0), la barrera avanza con el fin de realizar la sincronización (según el nivel fijado en la operación 704). Una vez fijada una sincronización a un nivel (operación 720; ACT_1_N), la barrera responde a cada proceso P mediante los datos D, que comprenden por ejemplo una instrucción de avance a una próxima fase de trabajo W (operación 740). Cuando se ha fijado el nivel de sincronización a dos niveles (operación 730; ACT_1_N), es decir, por ejemplo para varios grupos de procesos P (véase más arriba), la barrera BS (100) elige un maestro M entre los procesos P en curso (operación 732; CH_M) y efectúa una segunda sincronización (operación 734; SYNC) antes de la respuesta mediante datos D (operación 740). La sincronización finaliza (operación 750) con la vuelta al estado listo PRE (instrucción = PREPA) o con una activación de la barrera (instrucción = ETEINDRE).

Evidentemente, la invención no se limita a las formas de realización anteriormente descritas, sino que abarca todas las realizaciones que puedan prever los expertos en la técnica en el marco de las reivindicaciones adjuntas.

Así, en la forma de realización descrita, se utiliza una sola barrera BS para la sincronización de los procesos. Puede ser útil integrar varias barreras de sincronización BS en un sistema informático, en particular para permitir la sincronización de varios grupos de procesos, contribuyendo cada grupo a la ejecución de una tarea diferente. Por ejemplo, en cálculo científico en una máquina de 16 núcleos se puede prever que se realicen 2 cálculos independientes utilizando cada uno 8 núcleos, con lo que se tendrán 2 grupos de 8 procesos, ejecutándose cada proceso en un núcleo diferente. En este ejemplo se requerirán 2 barreras.

Evidentemente, cuando se utilizan varias barreras de sincronización BS, éstas se pueden implementar en un mismo componente o también en componentes diferentes. En efecto, el dispositivo puede incluir varios circuitos de *hardware*, a cuyos los espacios de direcciones se accede mediante segmentos sacados de dichos datos de cada llamada. En este caso se puede prever que cada uno de los circuitos de *hardware* esté ramificado bien en un mismo circuito, bien en circuitos independientes.

También se señala que se puede prever fácilmente una combinación entre barreras de tipo *software* y barreras según la invención, en concreto con un circuito de *hardware*. Por tanto, el dispositivo informático aquí descrito puede comprender además una barrera de sincronización de tipo *software* operando en combinación con dicho circuito de *hardware*.

Reivindicaciones

1. Dispositivo informático con barrera de sincronización, que comprende:

- una memoria (RAM, 202),
- una unidad de tratamiento que es capaz de procesar multiprocesos en diferentes procesadores (PZ, 200) y que permite una ejecución en paralelo de los bloques (B) mediante procesos (P), estando asociados dichos bloques (B) por grupos en fases de trabajo (W) sucesivos,
- un circuito de *hardware* (HBM, 400) con un espacio de dirección utilizable para la memoria (RAM, 202), que puede recibir una llamada de cada proceso (P) que indica la finalización de la ejecución de un bloque (B) en curso, y cada llamada incluye datos,

incluyendo el proceso (P) un primer grupo de procesos y un segundo grupo de procesos, independientes entre sí,

estando configurado dicho circuito de *hardware* (HBM, 400) para realizar un primer nivel de sincronización de los procesos (P) del primer grupo, respectivamente del segundo grupo, realizando las siguientes etapas:

- extraer el número de procesos del primer grupo, respectivamente del segundo grupo, a partir de una primera llamada de un proceso del primer grupo, respectivamente del segundo grupo,
- realizar una cuenta atrás de este número a partir de otras llamadas,

siendo sincronizados los procesos del primer grupo, respectivamente del segundo grupo, cuando la cuenta atrás indica que se ha ejecutado el conjunto de los bloques (B) del proceso del primer grupo, respectivamente del segundo grupo, de la fase de trabajo (W) en curso,

estando configurado dicho circuito de *hardware* (HBM, 400) además para:

- cuando se ha realizado el primer nivel de sincronización para el primer grupo, respectivamente para el segundo grupo, elegir como maestro uno de los procesos (P) del primer grupo, respectivamente del segundo grupo,
- enviar una respuesta al proceso maestro del primer grupo, respectivamente del segundo grupo, indicando que este proceso es el maestro del primer grupo, respectivamente del segundo grupo, de modo que el proceso maestro del primer grupo y el proceso maestro del segundo grupo realizan un segundo nivel de sincronización para sincronizar el primer grupo y el segundo grupo entre sí,
- recibir una llamada del proceso maestro del primer grupo, respectivamente del segundo grupo, indicando que el proceso maestro ha realizado el segundo nivel de sincronización,
- después de la recepción de la llamada del proceso maestro del primer grupo y de la llamada del proceso maestro del segundo grupo indicando que el proceso maestro del primer grupo y el proceso maestro del segundo grupo han realizado el segundo nivel de sincronización, enviar respuestas a los otros procesos del primer grupo y del segundo grupo para autorizar la ejecución de los bloques (B) de una fase de trabajo posterior,

teniendo lugar el acceso al espacio de dirección del circuito de *hardware* por segmentos sacados de dichos datos de cada llamada, incluyendo estos datos en particular la dirección del circuito de *hardware* en el espacio de dirección.

2. Dispositivo según la reivindicación 1, donde el segundo nivel de sincronización es una barrera de sincronización de tipo *software*.

3. Dispositivo según la reivindicación 1 o 2, donde el circuito de *hardware* comprende un microprograma (micro-Prog, 402).

4. Dispositivo informático según la reivindicación 3, donde el dispositivo comprende además una memoria dedicada (Ded_MEM, 404) en conexión con dicho microprograma (micro-Prog, 402).

5. Dispositivo informático según cualquiera de las reivindicaciones 1 a 4, donde el circuito de *hardware* está configurado para suspender las respuestas a cada llamada hasta verificar una condición de finalización que indica que todos los procesos (P) han señalado la finalización de la ejecución del bloque (B) de la fase de trabajo (W) en curso.

6. Dispositivo informático según cualquiera de las reivindicaciones 1 a 5, donde el circuito de *hardware* está configurado para responder a cada llamada mediante una salida de datos (D) y para autorizar a los procesos (P) a pasar a la fase de trabajo (W) posterior cuando todos los procesos han señalado la finalización de la ejecución del bloque (B) de la fase de trabajo (W) en curso.

7. Dispositivo informático según cualquiera de las reivindicaciones 1 a 6, donde cada llamada indica dicho número de procesos.

8. Dispositivo informático según cualquiera de las reivindicaciones 1 a 7, donde la totalidad de las llamadas son del mismo tipo (ENREGISTRER) definido por los datos de cada llamada.

9. Dispositivo informático según una de las reivindicaciones 1 a 8, donde el dispositivo comprende varios circuitos de *hardware*, a cuyos espacios de direcciones se accede mediante segmentos sacados de dichos datos de cada llamada.

10. Dispositivo informático según la reivindicación 9, en el que cada uno de dichos circuitos de *hardware* está ramificado en un mismo circuito.

11. Dispositivo informático según la reivindicación 9, donde cada uno de dichos circuitos de *hardware* está ramificado en un circuito independiente.

12. Dispositivo informático según una de las reivindicaciones 1 a 11, donde el dispositivo comprende además una barrera de sincronización de tipo *software*, que opera en combinación con dicho circuito material.

13. Dispositivo informático según una de las reivindicaciones 1 a 12, donde el dispositivo comprende además un gestor de acceso a memoria (CACHE COHER MGR, 206) y donde dichas llamadas al circuito de *hardware* (HBM, 400) son directas, evitando el gestor de acceso a memoria.

14. Procedimiento de tratamiento informático a nivel de proceso, del tipo que comprende las siguientes etapas:

a. descomponer una tarea (T) en subtareas ejecutadas como procesos (P) compuestos por bloques (B) sucesivos, incluyendo los procesos (P) un primer grupo de procesos y un segundo grupo de procesos, independientes entre sí;
b. prever una primera barrera de sincronización (BS, 100), respectivamente una segunda barrera de sincronización (BS, 100), provista de un contador (CNT, 406) con respecto al número de procesos (P) del primer grupo, respectivamente del segundo grupo, en un gestor físico de barrera (HBM, 400) que se encuentra en un circuito de *hardware* con un espacio de dirección;
c. en cada proceso (P) del primer grupo, respectivamente del segundo grupo, definir un primer bloque (B) como bloque en curso y ejecutarlo, mientras se accede a dicha primera barrera de sincronización (BS, 100), respectivamente segunda barrera de sincronización, para disminuir dicho contador (CNT, 406) cuando finaliza la ejecución de este bloque (B) en curso, realizándose el acceso por medio de una llamada, incluyendo cada llamada datos, e incluyendo estos datos en particular la dirección del circuito de *hardware* en el espacio de dirección;

d. en la barrera de sincronización:

- realizar un primer nivel de sincronización de los procesos (P) de los procesos del primer grupo, respectivamente del segundo grupo, realizando las siguientes etapas:
  - extraer el número de procesos del primer grupo, respectivamente del segundo grupo, a partir de una primera llamada de un proceso del primer grupo, respectivamente del segundo grupo,
  - inicializar el contador a partir de este número,
  - realizar una cuenta atrás de este número a partir de otras llamadas,

  siendo sincronizados los procesos del primer grupo, respectivamente del segundo grupo, cuando la cuenta atrás indica que se ha ejecutado el conjunto de los bloques (B) de la fase de trabajo (W) en curso,
- cuando se ha realizado el primer nivel de sincronización para el primer grupo, respectivamente para el segundo grupo, elegir como maestro uno de los procesos (P) del primer grupo, respectivamente del segundo grupo,

- enviar una respuesta al procesador maestro del primer grupo, respectivamente del segundo grupo, indicando que este proceso es el maestro del primer grupo, respectivamente del segundo grupo,

e. en el proceso maestro del primer grupo y en el proceso maestro de segundo grupo:

- participar en un segundo nivel de sincronización para sincronizar el primer grupo y el segundo grupo entre sí,
- enviar una llamada a la barrera de sincronización indicando que el proceso maestro del primer grupo, respectivamente del segundo grupo, ha realizado el segundo nivel de sincronización,

f. en la barrera de sincronización:

- tras la recepción de la llamada del proceso maestro del primer grupo y de la llamada del proceso maestro del segundo grupo indicando que el proceso maestro del primer grupo y el proceso maestro del segundo grupo han realizado el segundo nivel de sincronización, enviar respuestas a los otros procesos del primer grupo y del segundo grupo para autorizar la ejecución de bloques (B) de una fase de trabajo posterior,

g. en cada proceso (P) en el que se ha finalizado la ejecución del bloque (B) en curso, esperar una respuesta de dicha barrera de sincronización (BS, 100), estando la respuesta relacionada directamente con el contador (CNT, 406) y siendo emitida la misma cuando éste indica que se han ejecutado todos los bloques (B) actuales,

h. cuando se han ejecutado todos los bloques (B) actuales, definir nuevos bloques (B) en curso a partir del bloque siguiente de cada uno de los procesos (P) y repetir las etapas c. a f. con estos nuevos bloques (B) en curso.

15. Procedimiento según la reivindicación 14, donde el segundo nivel de sincronización se realiza mediante una barrera de sincronización de tipo *software*.

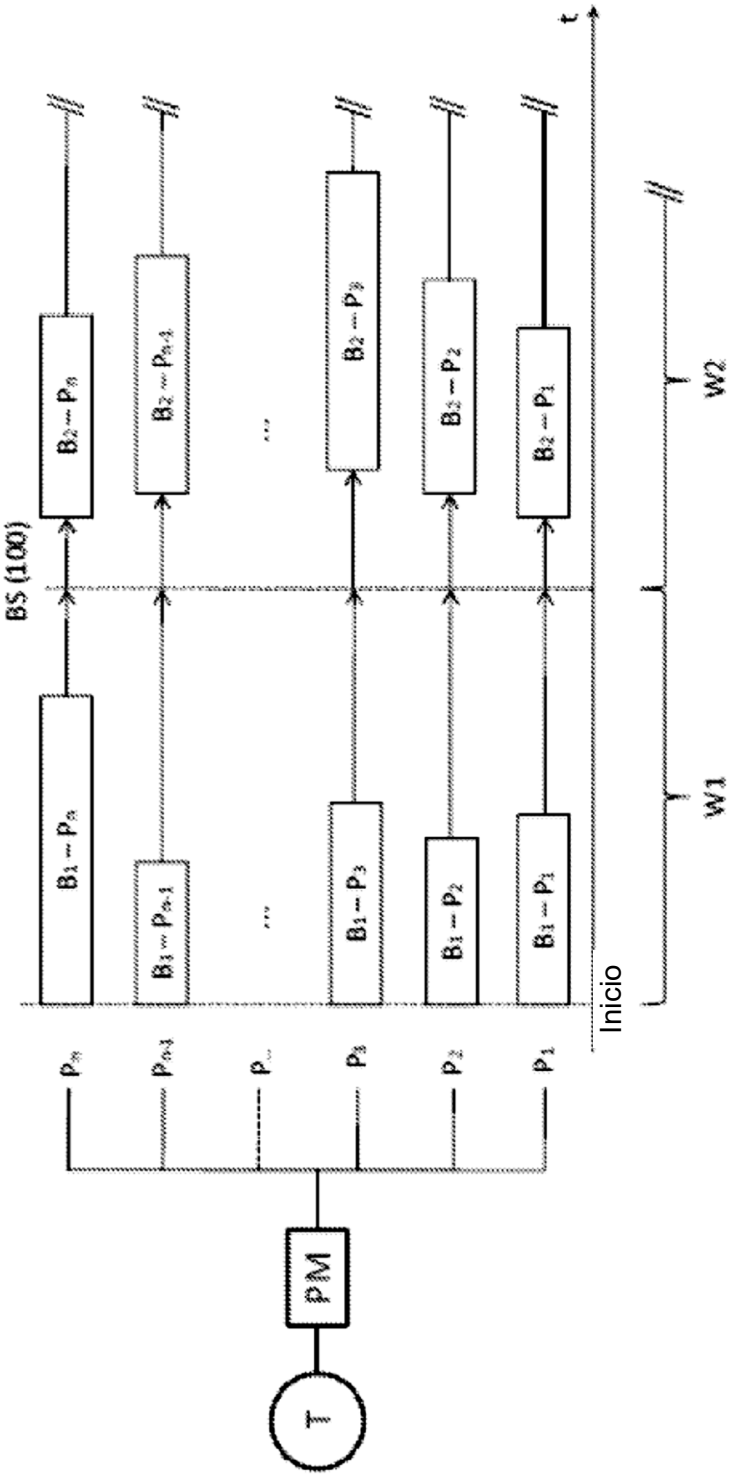
t
BS (100)
B2 – Pn
B2 – Pn-1
B2 – P3
B2 – P2
B2 – P1
B1 – Pn
B1 – Pn-1
B1 – P3
B1 – P2
B1 – P1
W2
W1
Inicio
Pn
Pn-1
P3
P2
P1
PM
T

Figura 1

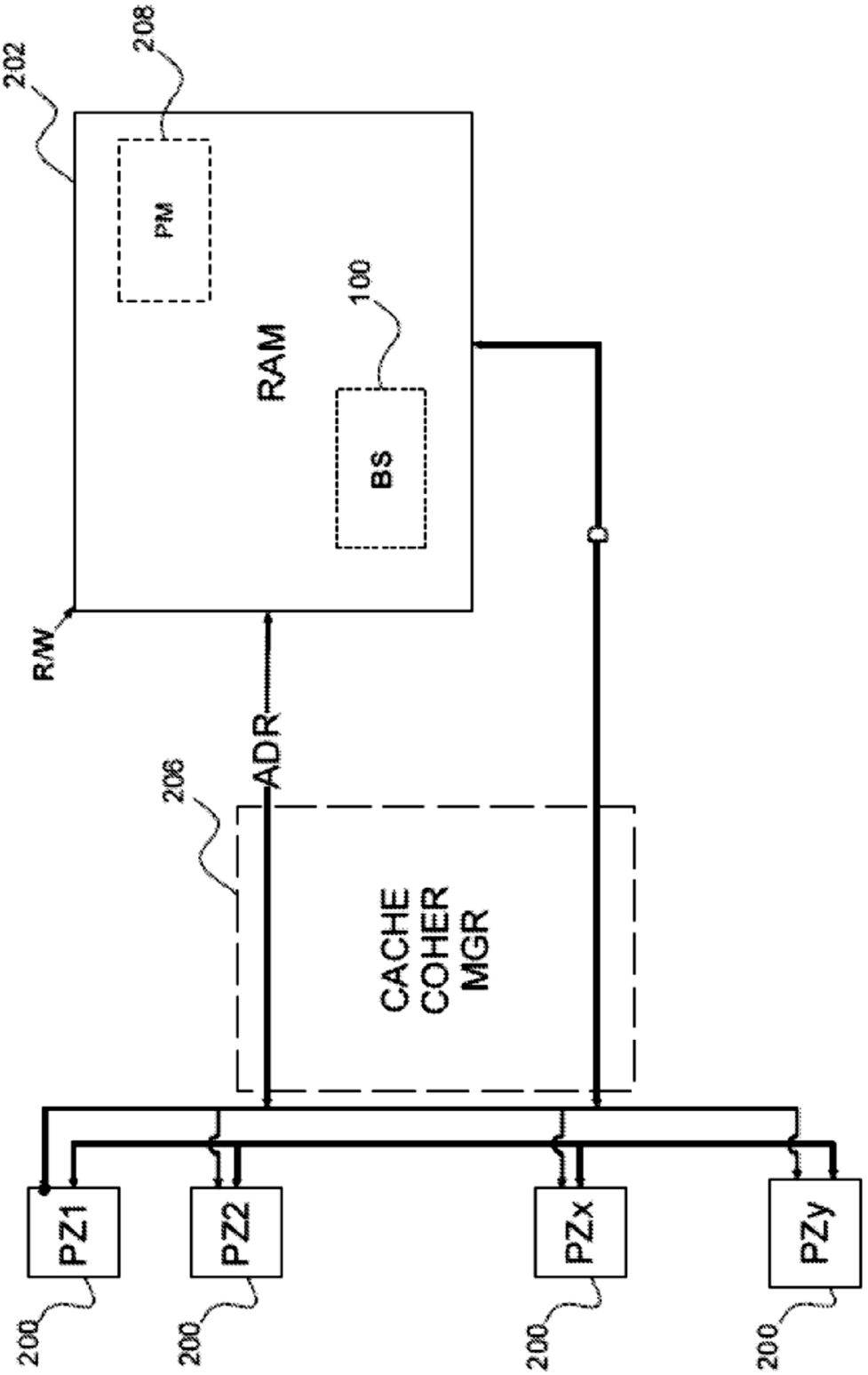
202
208
PM
RAM
100
BS
R/W
ADR
206
CACHE
COHER
MGR
PZ1
PZ2
PZx
PZy
200
200
200
200

Figura 2

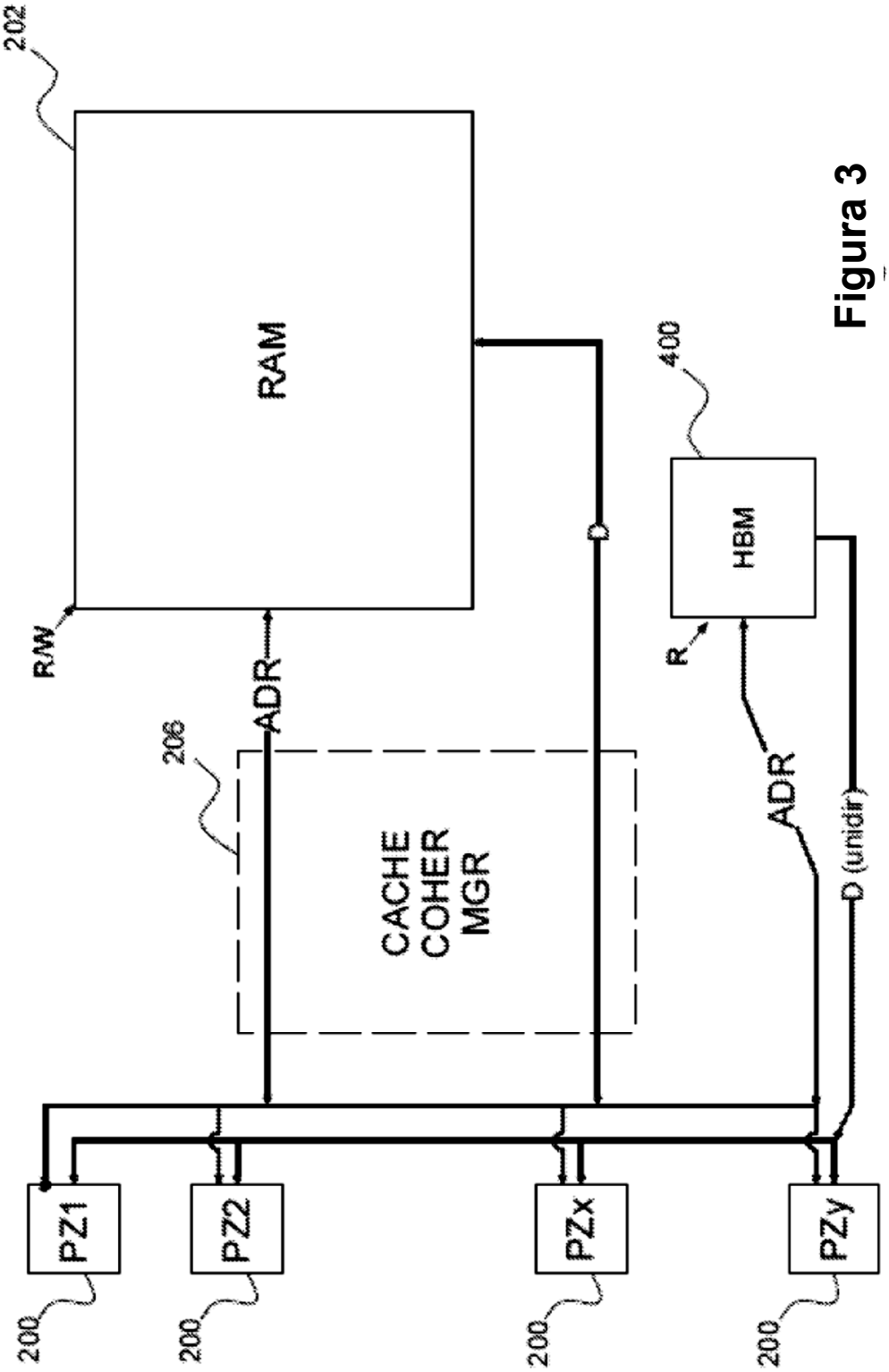
202
RAM
R/W
ADR
D
206
CACHE
COHER
MGR
400
HBM
R
ADR
D (unidir)
PZ1
PZ2
PZx
PZy
200
200
200
200

Figura 3

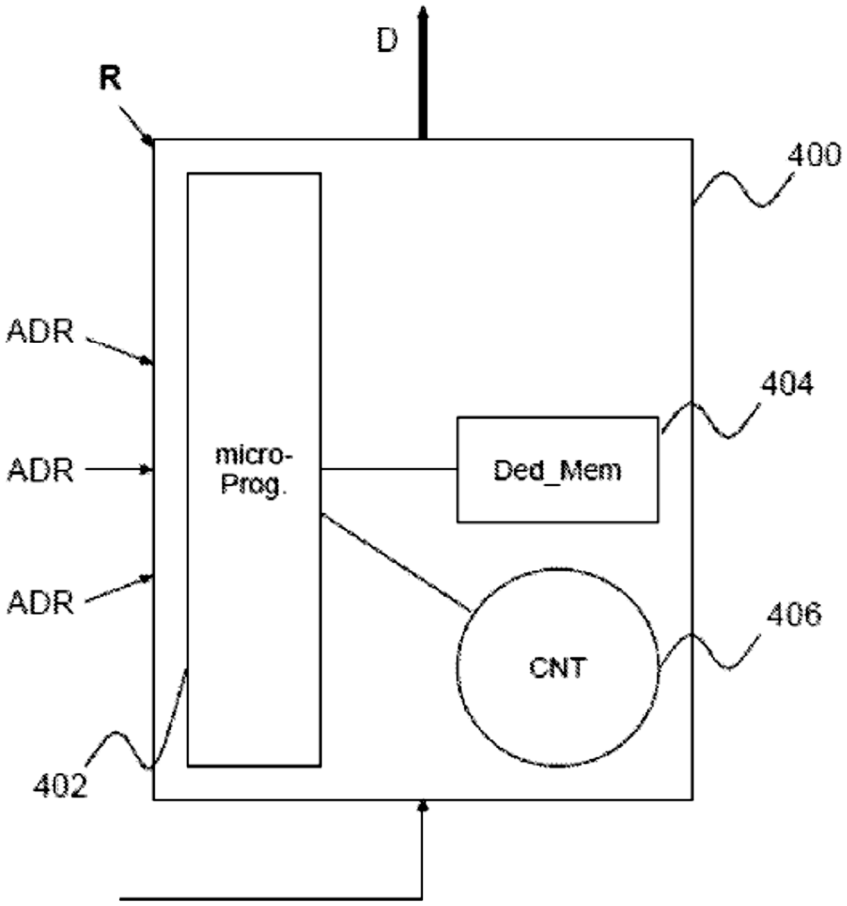

Figura 4

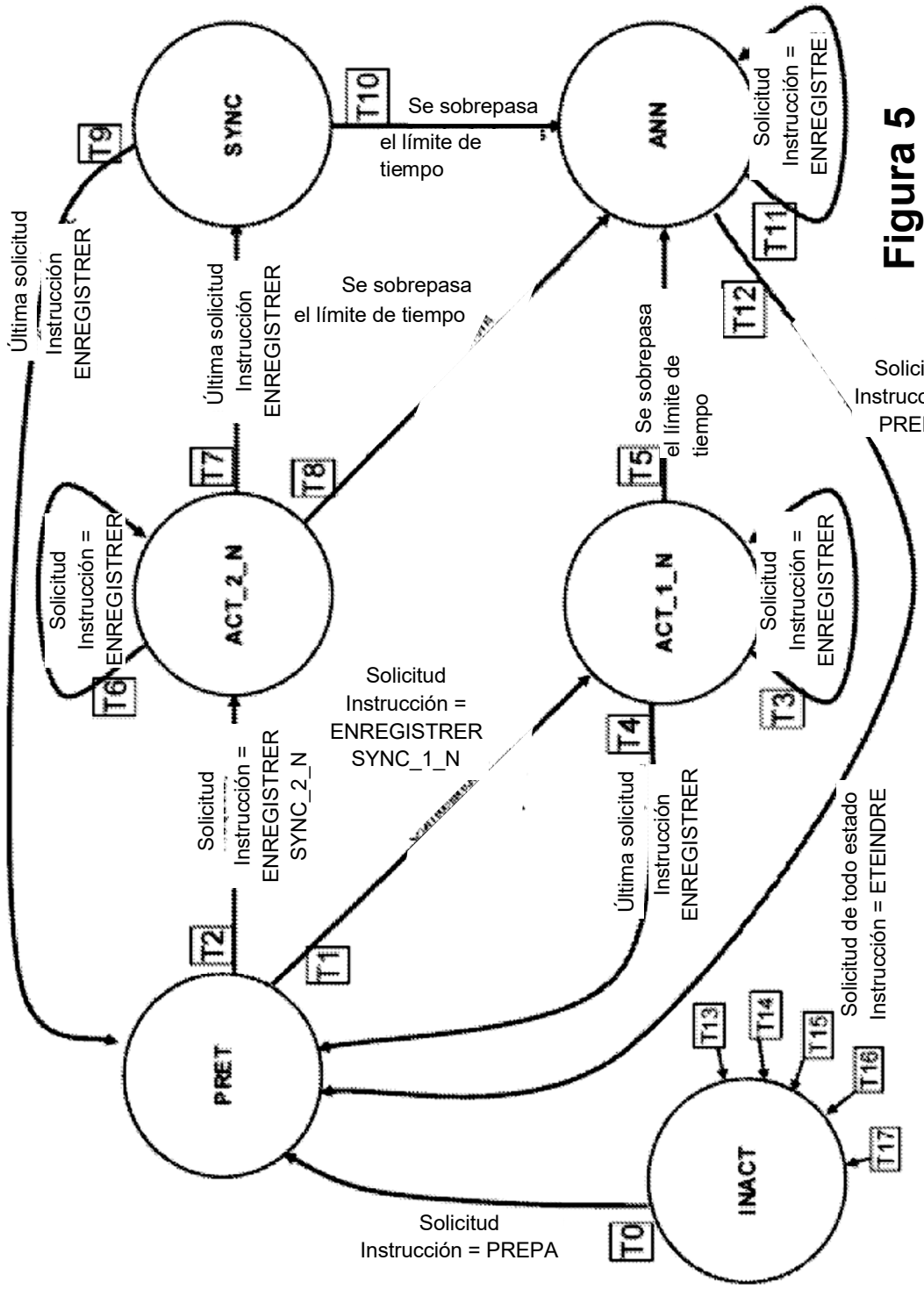
SYNC
ANN
ACT_2_N
ACT_1_N
PRET
INACT
T0
T1
T2
T3
T4
T5
T6
T7
T8
T9
T10
T11
T12
T13
T14
T15
T16
T17
Se sobrepasa el límite de tiempo
Solicitud Instrucción = ENREGISTRE
Última solicitud Instrucción ENREGISTRER
Solicitud Instrucción = ENREGISTRER
Solicitud Instrucción = ENREGISTRER SYNC_2_N
Solicitud Instrucción = ENREGISTRER SYNC_1_N
Solicitud Instrucción = PREPA
Solicitud de todo estado Instrucción = ETEINDRE

Figura 5

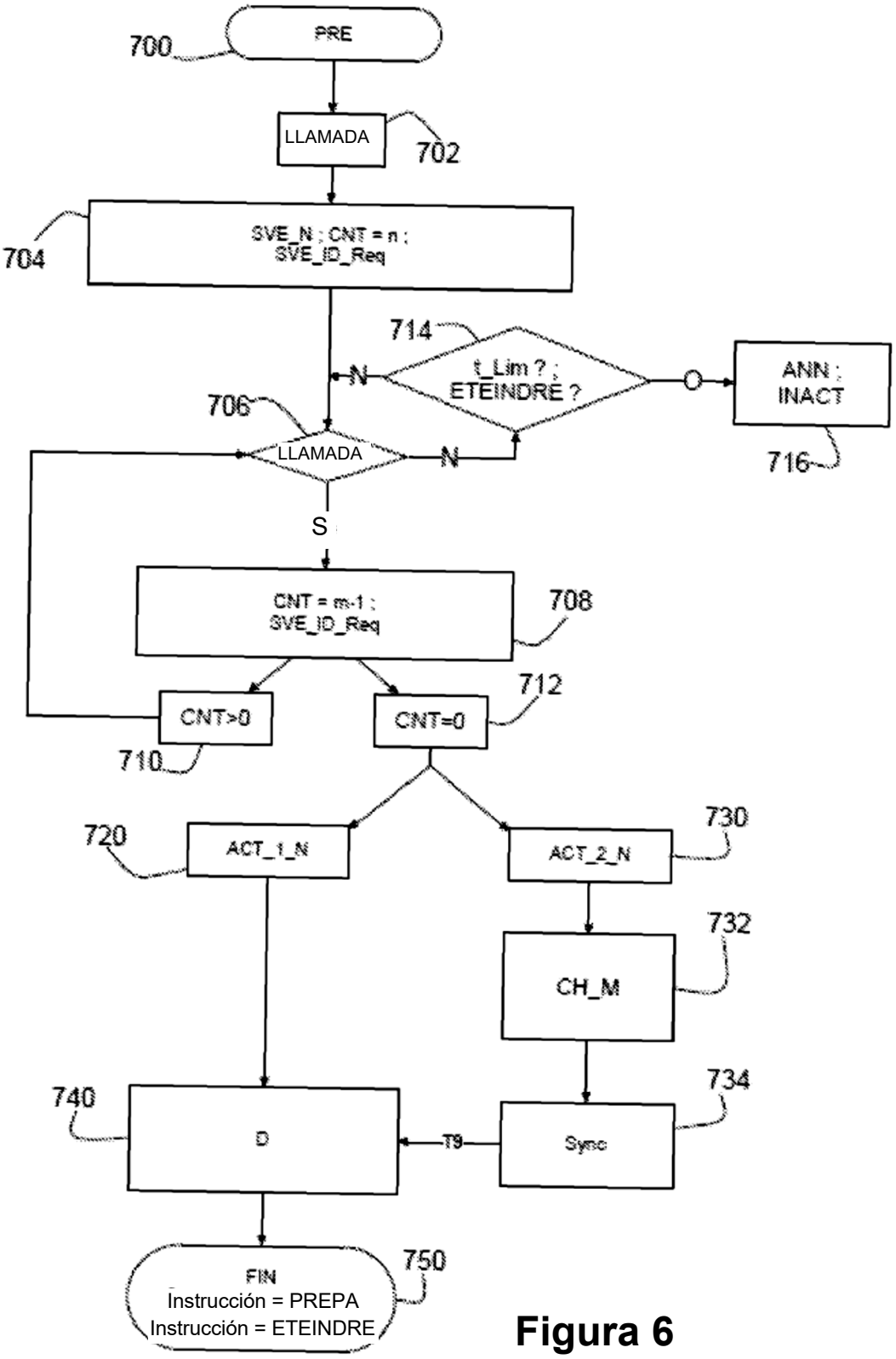
700
PRE
LLAMADA
702
704
SVE_N ; CNT = n ;
SVE_ID_Req
714
N
t_Lim ? ;
ETEINDRE ?
O
ANN ;
INACT
716
706
LLAMADA
N
S
CNT = m-1 ;
SVE_ID_Req
708
CNT>0
710
CNT=0
712
720
ACT_1_N
730
ACT_2_N
732
CH_M
734
740
D
T9
Sync
FIN
Instrucción = PREPA
Instrucción = ETEINDRE
750

Figura 6