(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 694 803**

(51) Int. CI.:

G06F 11/07 (2006.01)
G06F 11/16 (2006.01)
G06F 11/18 (2006.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(86) Fecha de presentación y número de la solicitud internacional: **28.10.2011 PCT/US2011/058394**

(87) Fecha y número de publicación internacional: **03.05.2012 WO12058597**

(96) Fecha de presentación y número de la solicitud europea: **28.10.2011 E 11837198 (8)**

(97) Fecha y número de publicación de la concesión europea: **22.08.2018 EP 2633408**

(54) Título: **Sistema, método y aparato para la corrección de errores en sistemas multiprocesador**

(30) Prioridad:

28.10.2010 US 407770 P

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
27.12.2018

(73) Titular/es:

DATA DEVICE CORPORATION (100.0%)
105 Wilbur Place
Bohemia, NY 11716, US

(72) Inventor/es:

HILLMAN, ROBERT y
WILLIAMSON, GALE

(74) Agente/Representante:

ISERN JARA, Jorge

DESCRIPCIÓN

Sistema, método y aparato para la corrección de errores en sistemas multiprocesador

Referencia cruzada a aplicaciones relacionadas

Esta solicitud reivindica prioridad a la solicitud de patente provisional n.º 61/407,770, presentada el 28 de octubre de 2010, titulada “Self Correcting Computing with Partial Scrubbing-Re-sync”.

Campo técnico

Esta divulgación se refiere a la corrección de errores de los dispositivos con diversos módulos informáticos. Específicamente, esta divulgación está relacionada con la corrección de errores en un dispositivo que incluye dos o más procesadores. Algunas implementaciones también incluyen múltiples estructuras de memoria asociadas.

Descripción de la tecnología relacionada

Ciertos entornos requieren que los sistemas informáticos sean extremadamente fiables. Al mismo tiempo, algunos de estos entornos pueden ser extremadamente severos, exponiendo los componentes informáticos a elementos potencialmente catastróficos.

Uno de tales entornos es el entorno espacial. Los sistemas informáticos que pueden colocarse en el espacio, tales como en la órbita terrestre, no están disponibles para el mantenimiento regular y, por lo tanto, deben estar diseñados para funcionar durante la vida útil de la nave espacial. Por lo tanto, un sistema informático montado en una nave espacial debe ser altamente confiable y robusto en su tolerancia a fallos, ya sea interiores o exteriores.

Además, los objetos colocados en el espacio están sujetos a diversos tipos de radiación que pueden ser extremadamente perjudiciales para ciertos componentes informáticos. Por ejemplo, el sol puede producir elementos de radiación que afectan a los sistemas electrónicos. Un único elemento de radiación puede provocar un trastorno, conocido como un evento único de cambio de estado (SEU), ya sea de un procesador o una memoria en un sistema informático al cambiar el estado de los transistores dentro de estos componentes informáticos. Sería deseable que un ordenador en el entorno espacial fuese tolerante a tales eventos únicos de cambio de estado debido a que se producen con frecuencia en el espacio.

Desarrollar componentes informáticos que sean individualmente tolerantes a estos trastornos puede ser extremadamente caro e ineficiente. Por encima de todo, debido a los largos ciclos de desarrollo, tales componentes carecen en general del rendimiento de los componentes del estado de la técnica. Por ejemplo, un procesador diseñado para ser tolerante a la radiación puede tener dos años para cuando el desarrollo esté completo. En esos dos años, los procesadores pueden tener más del doble de velocidad o rendimiento. Además, el reforzamiento de dichos componentes contra fallos puede hacer que los componentes sean mucho más caros que los componentes comparables sin el reforzamiento.

Una forma de protegerse contra los SEU es el uso de sistemas con múltiples procesadores redundantes. Por ejemplo, la patente de Estados Unidos n.º 5.903.717, desvela un sistema informático para detectar y corregir errores de los SEU. El sistema incluye una pluralidad de procesadores (CPU) cuyas salidas se votan en cada ciclo de reloj. Cualquier señal de salida de CPU que no coincida con la mayoría de las señales de salida de CPU produce una señal de error. El sistema reacciona a las señales de error generando una interrupción de gestión de sistema. Como reacción a la interrupción de gestión de sistema que resulta de un error detectado, el software del sistema inicia una resincronización de la pluralidad de CPU cuando el error está provocado por un evento único de cambio de estado.

La patente de Estados Unidos 7.467.326 desvela un sistema informático con un módulo de depuración para resincronizar los procesadores después de un intervalo de tiempo predeterminado o cualquier otro evento que pueda definirse por un usuario. El módulo de depuración mejora la confiabilidad de un sistema depurando los componentes en una programación regular, en lugar de esperar a que se detecte un error. Por lo tanto, los errores que pueden pasar desapercibidos durante un período prolongado no pueden propagarse y dañar aún más el sistema. El documento US-B2-7 797 575 desvela un método para sincronizar el estado de una pluralidad de módulos informáticos en un sistema electrónico, teniendo cada módulo informático un procesador, que comprende: verificar los datos de estado de procesador para cada uno de la pluralidad de módulos informáticos; comparar las verificaciones de procesador para los datos de estado de procesador y resincronizar la pluralidad de módulos informáticos basándose al menos en las verificaciones de procesador comparadas.

Sumario

Los sistemas, métodos y dispositivos de la divulgación tienen cada uno varios aspectos innovadores, no solo uno de los cuales es el único responsable de los atributos deseables desvelados en el presente documento.

Los métodos, aparatos, y sistemas descritos en el presente documento se refieren a unos sistemas informáticos que son robustos en su tolerancia a los efectos de un solo evento que pueden encontrarse, por ejemplo, en el entorno espacial o por las aeronaves de gran altitud. Estos métodos, aparatos y sistemas se relacionan con un sistema informático que está provisto de diversos conjuntos de protecciones contra fallos que pueden provocarse, por ejemplo, por el espacio o la radiación solar. Dichas protecciones incluyen una o más de las siguientes: múltiples procesadores, múltiples módulos de memoria, detección de errores y lógica de corrección, y blindaje mecánico de los componentes del sistema. Los métodos, aparatos y sistemas desvelados proporcionan un rendimiento mejorado con respecto a los sistemas de la técnica anterior en diversos aspectos.

Un aspecto innovador del objeto descrito en la presente divulgación puede implementarse en un método para sincronizar el estado de una pluralidad de módulos informáticos en un sistema electrónico, teniendo cada módulo informático un procesador. El método puede incluir verificar los datos de estado de procesador para cada uno de la pluralidad de módulos informáticos, comparar las verificaciones de procesador para los datos de estado de procesador y resincronizar la pluralidad de módulos informáticos basándose al menos en las verificaciones de procesador comparadas.

Otro aspecto innovador del objeto descrito en la presente divulgación puede implementarse como un aparato informático tolerante a fallos. El aparato informático tolerante a fallos puede incluir una pluralidad de módulos informáticos, en el que cada módulo informático comprende un procesador que tiene datos de estado de procesador, un módulo de verificación configurado para generar los valores de verificación de los datos de estado de procesador y una unidad de comprobación de tolerancia a fallos configurada para recibir la pluralidad de valores de verificación y determinar si cada módulo informático está sincronizado con los otros módulos informáticos.

El objeto también puede implementarse como un aparato informático tolerante a fallos, que incluye una pluralidad de módulos informáticos, incluyendo cada módulo informático un procesador que tiene datos de estado de procesador, medios para que la verificación configurada genere unos valores de verificación de los datos de estado de procesador, medios para comparar la pluralidad de valores de verificación, y medios para determinar si el procesador dentro de cada módulo informático está sincronizado con los procesadores de los otros módulos informáticos.

En algunas implementaciones, el objeto de la presente divulgación puede implementarse como un medio de almacenamiento legible por ordenador, no transitorio, que tiene unas instrucciones almacenadas en el mismo que provocan que un circuito de procesamiento realice un método. El método puede incluir verificar los datos de estado de procesador para cada uno de una pluralidad de módulos informáticos, comparar las verificaciones de procesador para los datos de estado de procesador y resincronizar la pluralidad de módulos informáticos basándose al menos en las verificaciones de procesador comparadas.

Los detalles de una o más implementaciones del objeto descrito en esta memoria descriptiva se exponen en los dibujos adjuntos y la descripción siguiente. Otras características, aspectos y ventajas se harán evidentes a partir de la descripción, los dibujos y las reivindicaciones. Obsérvese que las dimensiones relativas de las siguientes figuras pueden no estar dibujas a escala.

De acuerdo con la invención de acuerdo con la reivindicación 1, se propone un método para sincronizar el estado de una pluralidad de módulos informáticos en un sistema electrónico, un aparato informático tolerante a fallos de acuerdo con la reivindicación 8, y de acuerdo con la reivindicación 12 un medio de almacenamiento legible por ordenador, no transitorio que tiene unas instrucciones almacenadas en el mismo que hacen que un circuito de procesamiento realice el método de acuerdo con la reivindicación 1.

Las reivindicaciones dependientes se refieren a las realizaciones preferidas de la presente invención.

Breve descripción de los dibujos

La figura 1 ilustra un diagrama de bloques de una realización de un satélite que incluye un aparato informático tolerante a fallos.
La figura 2 ilustra un diagrama de bloques de una realización de un sistema de procesamiento tolerante a fallos que tiene múltiples procesadores.
La figura 3 ilustra un diagrama de bloques de una segunda realización de un sistema de procesamiento tolerante a fallos a modo de ejemplo.
La figura 4 es un diagrama de flujo que ilustra un proceso a modo de ejemplo para verificar las verificaciones en una pluralidad de módulos informáticos.
La figura 5 muestra un diagrama de flujo de un proceso de verificación a modo de ejemplo.
La figura 6 muestra un diagrama de flujo de un proceso a modo de ejemplo que resincroniza un módulo informático.
La figura 7 muestra un diagrama de flujo que ilustra una implementación a modo de ejemplo de un método de verificación.
La figura 8 muestra un diagrama de flujo que ilustra un proceso de recepción de memoria a modo de ejemplo.

La figura 9 muestra un diagrama de flujo que ilustra un proceso de envío de memoria a modo de ejemplo.
La figura 10 muestra un diagrama de flujo que ilustra un proceso de sincronización de memoria a modo de ejemplo.

Los números y las designaciones de referencia similares en los diversos dibujos indican elementos similares.

Descripción detallada

La siguiente descripción está dirigida a ciertas implementaciones con el fin de describir los aspectos innovadores de la presente divulgación. Sin embargo, un experto en la materia reconocerá fácilmente que las enseñanzas del presente documento pueden aplicarse de muchas maneras diferentes. Por lo tanto, no se pretende que las enseñanzas se limiten a las implementaciones representadas únicamente en las figuras, sino que tienen una amplia aplicabilidad como será fácilmente evidente para un experto en la materia.

Una realización es un sistema de procesamiento tolerante a fallos que tiene una capacidad y un rendimiento mejorados. El sistema mantiene una alta tolerancia a los errores, tales como los eventos únicos de cambio de estado (SEU) que pueden encontrarse, por ejemplo, en el entorno espacial o en aeronaves de gran altitud. En esta realización, el sistema de procesamiento tolerante a fallos incorpora al menos un módulo de procesamiento que actúa para controlar y procesar datos para el sistema en general. Cada módulo de procesamiento es tolerante a fallos al incluir al menos tres módulos informáticos que están configurados para ejecutarse en paralelo entre sí para evitar las SEU, pero puede incluir un número mayor de módulos informáticos para mejorar aún más la confiabilidad del sistema. En más realizaciones, cada módulo informático opera independientemente de, y en paralelo con, los otros módulos informáticos.

Mientras que los sistemas anteriores han empleado sistemas con componentes redundantes que pueden resincronizarse, el rendimiento de estos sistemas de la técnica anterior puede ser limitado debido a la naturaleza de su sincronización. Por ejemplo, los sistemas de la técnica anterior pueden tener acoplados una pluralidad de procesadores a un controlador de coherencia que comparaba las salidas de múltiples procesadores síncronos para garantizar una operación tolerante a fallos. Debido a que cada uno de la pluralidad de procesadores en estos sistemas no estaba reforzado en general contra los efectos de los SEU, el controlador de coherencia garantizaba que las operaciones del procesador seguían siendo tolerantes a fallos. Esta comparación de salidas puede haberse realizado con cada ciclo de bus y puede haber comparado las señales en direcciones o líneas de datos asociadas con los procesadores incluidos en los módulos informáticos. Este proceso puede haber ralentizado la operación de los procesadores debido a que las velocidades del bus del controlador de coherencia pueden haber sido más lentas que la velocidad máxima del bus de los procesadores.

El acceso a la memoria de estos sistemas de la técnica anterior puede haber tenido también un rendimiento limitado. En estos sistemas, el espacio de memoria para los procesadores puede haberse proporcionado por un chip de memoria reforzado contra la radiación accesible a los procesadores a través del controlador de coherencia. En algunos casos, acceder a la memoria reforzada a través de un controlador de coherencia podría ralentizar la operación de los procesadores. En algunos sistemas, las velocidades de bus entre los procesadores y el controlador de coherencia pueden ser más lentas que las velocidades de bus máximas admitidas por los procesadores. Los módulos de memoria reforzados utilizados en estos sistemas también pueden proporcionar velocidades de acceso a los datos más lentas, ya que sus diseños personalizados y reforzados contra la radiación pueden haber disminuido su comercialización, lo que hace que se basen en diseños de memoria más antiguos.

Por consiguiente, algunas realizaciones descritas en el presente documento mejoran el rendimiento del dispositivo reduciendo la dependencia de cada módulo informático en la interacción con el controlador de coherencia. Por ejemplo, los procesadores modernos pueden diseñarse con múltiples direcciones y buses de datos. Usando estos procesadores modernos, puede acoplarse operativamente un primer conjunto de buses a un controlador de coherencia, mientras que un segundo conjunto de buses puede acoplarse a otros componentes de hardware capaces de operar a una velocidad de bus más alta. Entonces, los procesadores pueden realizar ciertas operaciones que no requieren un control de coherencia estricto usando el segundo conjunto de buses.

Para proporcionar una detección y una corrección de errores adicionales de la pluralidad de módulos informáticos en esta configuración, las verificaciones de estado del procesador pueden realizarse en las realizaciones de la invención. Las verificaciones del estado de estado de procesador pueden compararse y los módulos informáticos resincronizarse de acuerdo con los resultados de la comparación.

Algunas implementaciones descritas en el presente documento, mejoran el rendimiento de la memoria del sistema, proporcionando a cada procesador una memoria local conectada directamente para almacenar los datos de estado de memoria o de programa. En algunas realizaciones, esta memoria local puede acoplarse al procesador mediante el segundo conjunto de buses descritos anteriormente. Esta arquitectura aumenta la velocidad de acceso a memoria y reduce la necesidad de interactuar con un controlador de coherencia para cada acceso de memoria desde el procesador. Al menos una parte del espacio de direcciones del procesador puede asignarse a esta memoria conectada directamente. Ya que la memoria conectada directamente puede ser una memoria de grado comercial no

reforzada, puede beneficiarse de densidades más altas y velocidades más altas disponibles en el mercado comercial. En algunas realizaciones, la memoria local conectada directamente no está diseñada específicamente para reforzarse contra los efectos de la radiación del espacio exterior, y por lo tanto también puede ser susceptible a los SEU. En esta realización, el sistema comprobaría también la coherencia de la memoria junto con la coherencia de comprobación de los procesadores.

En una implementación, cada procesador dentro de un módulo informático incluye un módulo de verificación que tiene unas instrucciones que configuran el procesador para generar un valor de verificación a partir de los datos de estado de procesador, la memoria local, o una combinación de los dos. Debido a que el espacio de memoria dentro de cada módulo informático puede ser relativamente grande, el espacio de memoria puede dividirse en segmentos o partes predeterminados, y el módulo de verificación configurarse para crear solo un valor de verificación a partir de un segmento o parte específica de la memoria. La misma parte de memoria de las memorias locales en cada módulo informático se verificaría entonces por el módulo de verificación para crear un valor de verificación que podría ser comparable a través de múltiples módulos informáticos.

En algunas realizaciones, el sistema crea unos valores de verificación a intervalos temporales regulares durante la operación del sistema. Por ejemplo, el sistema usa un temporizador de tal manera que, cada 1, 2, 5, 10, 20, 30 o más milisegundos, inicia una verificación para garantizar que los valores de verificación procedentes de la memoria o los datos de estado de procesador procedentes de cada módulo informático coincidan entre sí. Debido a que la memoria local puede ser una cantidad relativamente grande de datos a verificar, en algunas realizaciones, el sistema usa un proceso de interrogación secuencial de todo el espacio de memoria de la memoria local de cada módulo informático, creando unos valores de verificación para diferentes partes de la memoria en cada intervalo temporal secuencial. En esta realización, el sistema divide la memoria en partes o segmentos, y en un momento t = 1, crea y compara un valor de verificación de la primera parte de memoria de cada módulo informático. A continuación, en el momento t = 2, el sistema crea y compara un valor de verificación de la segunda parte de la memoria de cada módulo informático, y así sucesivamente hasta que se interroga todo el espacio de memoria local de cada procesador. Esto permite que el sistema compruebe la consistencia de la memoria cada pocos milisegundos, debido a que el proceso de creación de una verificación de las partes de memoria relativamente más pequeñas no es tan intensivo en cómputo como si se estuviera verificando toda la memoria.

En algunas implementaciones, después de que se hayan creado las verificaciones por cada procesador, las verificaciones se comunican a una unidad de comprobación de tolerancia a fallos. La unidad de comprobación de tolerancia a fallos puede implementarse como un circuito integrado de aplicación específica (ASIC) o una matriz de puertas programable en campo (FPGA) u otro dispositivo lógico programable. Si la unidad de comprobación de tolerancia a fallos determina que las verificaciones no coinciden, la unidad de comprobación de tolerancia a fallos puede identificar una verificación mayoritaria y una verificación minoritaria. Una verificación mayoritaria puede ser una verificación producida por la mayoría de los módulos informáticos, y una verificación minoritaria puede ser una verificación producida por una minoría o uno de los módulos informáticos.

En implementaciones que verifican solo una parte de una memoria con cada intervalo temporal tal como se ha descrito anteriormente, pueden verificarse otras partes de la memoria para cada módulo informático en respuesta a la detección de verificaciones de memoria no coincidentes entre los módulos informáticos. Las verificaciones de cada parte de memoria para cada módulo informático pueden compararse a continuación para identificar cualquier parte de memoria adicional que pueda tener errores. Este diseño reconoce que una corrupción de memoria en una parte de la memoria puede indicar una mayor probabilidad de corrupción de memoria en otras partes de la memoria.

En algunas otras implementaciones, un procesador que ejecuta las instrucciones incluidas en el módulo de verificación puede generar una verificación que combine los datos de estado de procesador con una parte de los datos de estado de memoria local. En algunas implementaciones, la parte de la memoria que se incluye en la verificación puede variar con cada generación de verificación.

En algunas implementaciones, cada espacio de direcciones del procesador puede también asignarse a una dirección y a las líneas de datos controladas por la unidad de comprobación de tolerancia a fallos. Otras implementaciones pueden acoplar un bus separado a una unidad de comprobación de tolerancia a fallos. La unidad de comprobación de tolerancia a fallos puede determinar si uno o más de los procesadores producen una salida que es diferente de la mayoría de los procesadores para detectar un error. Tras la detección, el error puede registrarse y la operación del procesador que ha generado el error puede suspenderse. En algunas implementaciones, puede permitirse que el procesador continúe la operación cuando se detecta un error. Si el procesador se suspende o se le permite continuar la operación puede depender de la naturaleza del error.

En algunas implementaciones, un módulo de procesamiento puede no responder. Por ejemplo, puede dejar de señalar la dirección y las líneas de datos controladas por la unidad de comprobación de tolerancia a fallos, o puede dejar de producir verificaciones de sus datos de estado de procesador o los datos de estado de memoria. En estas circunstancias, pueden proporcionarse unos medios para hacer que el procesador se reinicie. Por ejemplo, el módulo de comprobación de tolerancia a fallos puede estar acoplado operativamente a la línea de señalización de reinicio del procesador. Cuando se reinicia un procesador, sus datos de estado de procesador y los datos de estado

de memoria pueden reinicializarse con los datos de estado de procesador y los datos de estado de memoria procedentes de otro módulo de procesamiento.

Algunas implementaciones tienen una o más de las siguientes ventajas potenciales. Debido a que los métodos, aparatos y sistemas desvelados pueden hacer uso de módulos de memoria disponibles comercialmente, puede proporcionarse un mayor tamaño de memoria para cada módulo informático mientras se mantienen los parámetros de coste. Este aumento del tamaño de la memoria puede permitir capacidades más rápidas o más potentes. Por ejemplo, un software más potente con capacidades mejoradas puede almacenarse dentro de la memoria disponible más grande. Como alternativa, el aumento de memoria puede usarse para aumentar el tamaño de la memoria caché, proporcionando un acceso más rápido a los datos y/o al código. En segundo lugar, el sistema puede diseñarse de manera tal que la mayoría del acceso a los datos se realiza a través de la memoria conectada directamente a mayor velocidad, y no a los módulos de memoria reforzados más lentos accesibles a través del controlador de coherencia. Por lo tanto, las velocidades de acceso a la memoria pueden mejorarse en general, lo que aumenta aún más el rendimiento.

La figura 1 ilustra un diagrama de bloques de un satélite 100 que incluye un sistema de procesamiento tolerante a fallos 110 para controlar diversas funciones del satélite 100. Por ejemplo, el sistema de procesamiento tolerante a fallos 110 está conectado a un módulo de control de propulsor 115 que supervisa la operación de los propulsores satelitales. Además, el sistema de procesamiento tolerante a fallos 110 está conectado a un módulo de despliegue de matriz solar 120 que incluye una programación para controlar el despliegue y la colocación de la matriz solar en el satélite. Un módulo de comunicación 125 también puede conectarse al sistema de procesamiento tolerante a fallos 110 para proporcionar unos canales de comunicación entre el satélite 100 y una estación de tierra. Al usar el sistema informático tolerante a fallos 110, el satélite 100 está provisto de un sistema robusto y resistente a los SEU que podrían afectar a la guía o al control del satélite 100. El satélite 100 también puede incluir otros módulos de procesamiento que no se ilustran en la figura 1 para mantener un entorno informático robusto para el satélite 100.

La figura 2 ilustra un diagrama de bloques del sistema de procesamiento tolerante a fallos 110 mostrado en la figura 1. El sistema incluye tres módulos informáticos separados 250, 260 y 270. Cada módulo informático incluye un procesador. Por ejemplo, el módulo informático 1 250 incluye el procesador 210, el módulo informático 2 260 incluye el procesador 211 y el módulo informático 3 270 incluye el procesador 212. Como se muestra, cada procesador incluye dos registros interiores. Por ejemplo, el procesador 220 incluye los registros 214 y 215. De manera similar, el procesador 211 incluye los registros 216 y 217, y el procesador 212 incluye los registros 218 y 219.

Existe una memoria 280 accesible para cada procesador. La memoria 280 puede accederse a través de la unidad de comprobación de tolerancia a fallos 230 (descrita a continuación). Cada procesador puede ejecutar las instrucciones almacenadas en una memoria 280. En una implementación, esas instrucciones pueden organizarse en un módulo de sistema operativo 285 y un módulo de verificación 290. Cada procesador puede leer las instrucciones incluidas en el módulo de sistema operativo 285 o en el módulo de verificación 290. Esas instrucciones pueden configurar los procesadores para realizar funciones. Por ejemplo, el módulo de verificación 290 puede configurar los procesadores 210, 211 y 212 para realizar una verificación de al menos una parte de los datos de estado de procesador para su procesador correspondiente. Por ejemplo, las instrucciones en el módulo de verificación 290 pueden configurar el procesador 210 para leer los datos de estado almacenados en los registros 214 o 215 para el procesador 210 y crear una verificación de esos datos de estado. Por lo tanto, las instrucciones dentro del módulo de verificación 290 que se ejecutan en cada uno de los procesadores 215, 217 y 219 representan un medio para la verificación configurada para generar los valores de verificación para una pluralidad de datos de estado de procesadores. Como se describirá con más detalle a continuación, al hacer que cada procesador proporcione los datos de estado verificados para su comparación, el sistema de procesamiento tolerante a fallos 110 puede comparar de manera rápida y eficaz los estados de cada procesador en el sistema para determinar si ha ocurrido algún fallo.

Como se muestra en la figura 2, los tres procesadores 210, 211, y 212 están acoplados operativamente a una unidad de comprobación de tolerancia a fallos 230, a través de enlaces 234, 235, y 236. La unidad de comprobación de tolerancia a fallos 230 está configurada para comparar las verificaciones generadas por los procesadores 210, 211 y 212 cuando ejecutan las instrucciones incluidas en el módulo de verificación 290. Por ejemplo, la unidad de comprobación de tolerancia a fallos 230 puede determinar si las tres verificaciones generadas por los procesadores 210, 211 y 212 son iguales en un intervalo periódico. Si las verificaciones no son iguales, la unidad de comprobación de tolerancia a fallos 230 puede determinar a continuación si al menos la mayoría de las verificaciones son iguales. Por lo tanto, la unidad de comprobación de tolerancia a fallos 230 representa un medio para comparar una pluralidad de valores de verificación. Este proceso de comparación de valores de verificación se describirá con más detalle a continuación.

Como es sabido, la creación de una verificación se refiere a un procedimiento determinista que pueden recibir unos datos de entrada de tamaño variable y emitir otros datos de un tamaño fijo. Cuando los datos de estado de procesador se utilizan como entrada para una función de verificación, cualquier pequeño cambio en los datos de estado de procesador dará lugar a un cambio grande y detectable en el valor de verificación resultante de esos datos de estado. En una implementación, la verificación devuelta por los procesadores cuando se ejecutan las

instrucciones almacenadas en el módulo de verificación 290 se usa para determinar si el estado de cada procesador es el mismo y está sincronizado con los otros procesadores.

Una función de verificación bien conocida se conoce como el algoritmo de verificación seguro o “SHA” y fue diseñado por la Agencia nacional de seguridad. Una variante, SHA-1 es una función de verificación criptográfica publicada por el NIST como un estándar de procesamiento de información federal de los Estados Unidos. SHA-1 produce un valor de verificación de 160 bits. Debido a que los datos dentro de cada valor de verificación son sustancialmente más pequeños que los datos en todos los datos de estado de cada procesador, la realización divulgada que compara los valores de verificación para cada conjunto de datos de estado de procesador es más eficaz que comparar los datos de estado reales.

La unidad de comprobación de tolerancia a fallos 230 puede estar configurada además para resincronizar los módulos informáticos 250, 260, y 270 cuando se determina que las tres verificaciones generadas por los procesadores 210, 211, y 212 no son iguales. Por lo tanto, la unidad de comprobación de tolerancia a fallos 230 representa unos medios para determinar si un procesador dentro de cada módulo informático está sincronizado con los procesadores de los otros módulos informáticos. Por ejemplo, la unidad de comprobación de tolerancia a fallos puede resincronizar los módulos informáticos 250, 260 y 270 cuando una, dos o las tres verificaciones no coinciden entre sí. En una realización, resincronizar un módulo informático incluye restaurar el estado de procesamiento del módulo informático que produjo una verificación minoritaria.

En algunas realizaciones, el estado de procesamiento de un módulo informático que produjo una verificación minoritaria puede restaurarse basándose en el estado de procesamiento de un módulo informático que produjo una verificación mayoritaria. Por ejemplo, si los módulos informáticos 250 y 260 produjeron el mismo valor de verificación, y el módulo informático 270 produjo un valor de verificación diferente, entonces el módulo informático 270 produjo la verificación minoritaria y los módulos informáticos 250 y 260 produjeron una verificación mayoritaria. En esta circunstancia, el valor de estado del procesador 210 en el módulo informático mayoritario 250 puede usarse para restaurar el valor de estado del procesador 212 en el módulo informático minoritario 270. Como alternativa, el valor de estado del procesador 211 en el módulo informático mayoritario 260 también puede usarse para restaurar el valor de estado del procesador 212 en el módulo informático minoritario 270.

La unidad de comprobación de tolerancia a fallos 230 también puede estar configurada para comparar las salidas de los módulos 250, 260, y 270. Por ejemplo, las líneas 234, 235, y 236 puede incluir las líneas de direcciones y las líneas de datos reafirmadas por los procesadores 210, 211, y 212. La unidad de comprobación de tolerancia a fallos 230 puede comparar las líneas de direcciones y de datos de los módulos informáticos 250, 260 y 270, y determinar si las líneas correspondientes entre los tres módulos informáticos están reafirmando las señales equivalentes. Esta comparación puede producirse en cada ciclo de bus para líneas de direcciones y para las líneas de datos 234, 235 y 236.

La unidad de comprobación de tolerancia a fallos 230 puede determinar una salida mayoritaria y una salida minoritaria opcional basándose en las señales de las líneas 234, 235, y 236. La unidad de comprobación de tolerancia a fallos 230 puede configurarse para pasar solamente señales mayoritarias a otros componentes de hardware, por ejemplo, la memoria 280. Aunque se muestra la unidad de comprobación de tolerancia a fallos 230 comparando las señales de salida de tres módulos informáticos 250, 260 y 270, que incluyen tres procesadores 210, 211 y 212, debería entenderse que la unidad de comprobación de tolerancia a fallos 230 podría configurarse para aceptar un mayor número de entradas. Por ejemplo, una unidad de comprobación de tolerancia a fallos 230 podría comparar la salida de 5, 7, 9 u 11 procesadores para determinar una mayoría y una señal de salida minoritaria opcional.

Como se muestra en la figura 2, la unidad de comprobación de tolerancia a fallos 230 está conectada a una memoria principal 280 que se usa por el sistema de procesamiento tolerante a fallos 110 para almacenar datos, instrucciones y programas utilizados por el sistema 110. Además, la memoria 280 puede incluir instrucciones que configuran los procesadores 210, 211 y 212, por ejemplo, leyendo los datos de un módulo de sistema operativo 285 almacenados en la memoria 280 y que carga ese sistema operativo en uno o más de los procesadores durante un proceso de restauración.

Por supuesto, debería tenerse en cuenta que cada procesador, como se muestra en la figura 2, puede ser un dispositivo de procesador único, o un núcleo de un dispositivo multiprocesador. Por ejemplo, el sistema de procesamiento tolerante a fallos puede incluir una pluralidad de procesadores separados, tales como los realizados por INTEL o AMD. A la inversa, cada procesador puede ser un núcleo único de un procesador de múltiples núcleos creado por INTEL o AMD que funciona como se ha descrito anteriormente. En algunas realizaciones, puede usarse una mezcla de dispositivos de procesador único o multiprocesador como los procesadores en el sistema 110.

La figura 3 ilustra otra realización de un sistema de procesamiento tolerante a fallos 300, pero esta realización difiere de la realización mostrada en la figura 2 por el uso de bancos de memoria separados enlazados a cada procesador. Como se muestra, el sistema 300 incluye tres módulos informáticos 360, 370 y 380. Cada módulo informático incluye

al menos un procesador. Por ejemplo, el módulo informático 360 incluye el procesador 330, mientras que el módulo informático 370 incluye el procesador 331 y el módulo informático 380 incluye el procesador 332.

Cada procesador incluye unos registros interiores para almacenar la información de estado de procesador. Por ejemplo, el procesador 330 incluye los registros interiores 385 y 386. El procesador 331 incluye los registros interiores 387 y 388, y el procesador 332 incluye los registros interiores 389 y 390. Cada procesador también incluye un controlador de memoria asociado con el procesador y para controlar el acceso a un almacenamiento de memoria local. En una realización, el almacenamiento de memoria local es la memoria caché. Por ejemplo, el procesador 330 incluye el controlador de memoria 320 para gestionar el acceso a la memoria local 310. De manera similar, el controlador de memoria 321 del procesador 331 gestiona el acceso a la memoria local 311, y el controlador de memoria 322 del procesador 332 gestiona el acceso a la memoria local 312.

Debería tenerse en cuenta que cada memoria local puede particionarse en una pluralidad de segmentos de memoria. Por ejemplo, la memoria local 312 del módulo informático 380 se particiona en cuatro segmentos de memoria en el ejemplo ilustrado. Estos segmentos de memoria se identifican como los artículos 318a-d.

También debería tenerse en cuenta que la memoria local puede almacenarse exterior al procesador, o como memoria integrada almacenada en el chip del procesador. En algunas realizaciones, la memoria local es una memoria caché de acceso rápido, y los segmentos de memoria son segmentos de memoria caché. En otras realizaciones, la memoria local es una memoria de acceso aleatorio convencional y está configurada para almacenar datos del programa además de los datos específicos para un procesador en particular.

De manera similar a la realización descrita para la figura 2, cada módulo informático puede ejecutar las instrucciones incluidas en un módulo de verificación 345. Las instrucciones en el módulo de verificación 345 pueden configurar los procesadores para calcular un valor de verificación a partir de los datos de estado almacenados en los registros del procesador. Mientras que el módulo de verificación 345 se ilustra como que se almacena en la memoria 390, que puede accederse por los módulos informáticos 360, 370 y 380 a través de la unidad de comprobación de tolerancia a fallos 350, el módulo de verificación 345 puede en su lugar almacenarse en las memorias locales 310, 311 y 312. Estas memorias locales pueden proporcionar un acceso más rápido a los segmentos de código tales como los proporcionados por el módulo de verificación 345. En otras realizaciones, las copias del módulo de verificación 345 pueden almacenarse en la memoria 390 y en las memorias locales 310, 311 y 312.

En algunas realizaciones, el módulo de verificación 345 configura los procesadores para crear una verificación, que también incluye los datos de la memoria local asociada a cada procesador. Por ejemplo, cuando las instrucciones incluidas en el módulo de verificación 345 se ejecutan por el procesador 330, el procesador 330 puede crear una verificación para representar el estado del procesador 330 y de la memoria local 310. Del mismo modo, cuando las instrucciones en el módulo de verificación 345 se ejecutan por el procesador 331, el procesador 331 puede crear una verificación para representar el estado del procesador 331 y de la memoria local 311. Cuando las instrucciones en el módulo de verificación 345 se ejecutan por el procesador 332, el procesador 332 puede crear una verificación para representar el estado del procesador 332 y la memoria local 312. Por lo tanto, las instrucciones incluidas en el módulo de verificación 345 que se ejecutan en cada uno de los procesadores 330, 331 y 332 pueden representar un medio para que la verificación configurada genere unos valores de verificación para los datos de estado de procesador de una pluralidad de módulos informáticos.

Como se ha tratado anteriormente, una operación de verificación puede incluir cualquier operación que asigne de manera exclusiva un gran conjunto de datos a un conjunto de datos más pequeño. Por ejemplo, una verificación puede asignar múltiples bytes que comprenden los datos de estado de procesamiento almacenados en los registros 385 y 386 en una cantidad de cuatro bytes. Una verificación puede asignar además tanto un estado de procesamiento como un estado de memoria a un conjunto de datos más pequeño, tal como una cantidad de cuatro bytes. Por ejemplo, una verificación puede asignar los contenidos del estado de procesamiento almacenado en los registros 385 y 386, junto con los contenidos de al menos una parte de la memoria local 310, en una cantidad de cuatro bytes. En algunas implementaciones, una verificación puede ser una suma de comprobación producida por el algoritmo de verificación SHA-1.

Las instrucciones dentro del módulo de verificación 345 pueden configurar más procesadores 330, 331 y 332 para variar los datos de estado de memoria usados para crear una verificación. Por ejemplo, cuando las instrucciones dentro del módulo de verificación 345 se ejecutan dentro del módulo informático 1 en el procesador 330, una primera verificación puede asignar los registros de estado de procesamiento 385 y 386, junto con los datos de estado de memoria del segmento de memoria 316a, a una verificación. En un ejemplo, la verificación es una verificación de cuatro bytes. Cuando se produce una segunda verificación con el procesador 330, el módulo de verificación 345 puede asignar los registros de estado de procesamiento 385 y 386, junto con el segmento de memoria 316b a una verificación, por ejemplo, otra cantidad de cuatro bytes. Después de que el módulo de verificación 345 que se ejecuta en el procesador 330 haya incluido todos los segmentos de la memoria 310 en una verificación, las instrucciones del módulo de verificación 345 pueden configurar el procesador 330 para que regrese al segmento de memoria 316a y repita el ciclo. Por lo tanto, las instrucciones dentro del módulo de verificación 345, que se ejecutan

en los procesadores 330, 331 o 332, representan unos medios para la verificación combinada de los datos de estado de procesador y los datos de estado de memoria para cada módulo informático.

En algunas implementaciones, el módulo de verificación 345 no puede incluir instrucciones que configuren los procesadores 330, 331, y 332 para crear verificaciones para las memorias locales 310, 311, y 312 como se ha descrito anteriormente. En estas realizaciones, puede proporcionarse una unidad de comprobación de coherencia de memoria separada 352. La unidad de comprobación de coherencia de memoria 352 puede configurarse para monitorizar la dirección, los datos y las líneas de control entre los procesadores y sus respectivas memorias locales. Como se muestra, la unidad de comprobación de coherencia de memoria 352 está configurada para monitorizar las líneas entre el procesador 330 y la memoria local 310 a través de las líneas 354. La unidad de comprobación de coherencia de memoria 352 también puede monitorizar la dirección, los datos y las líneas de control entre el procesador 331 y la memoria 311 a través de las líneas 356, y el procesador 332 y la memoria 312 a través de las líneas 358. Al monitorizar estas direcciones, datos y líneas de control, la unidad de comprobación de coherencia de memoria 352 puede configurarse para detectar errores en las memorias 310, 311 y 312 identificando las inconsistencias entre los datos de las memorias. La unidad de comprobación de coherencia de memoria puede configurarse además para señalar a la unidad de comprobación de tolerancia a fallos a través de las líneas de señalización separadas (no mostradas) tras detectar una condición de error de este tipo.

En algunas otras realizaciones, puede usarse tanto una unidad de comprobación de coherencia de memoria 352 como un módulo de verificación 345. Por ejemplo, algunas de estas realizaciones pueden comenzar un proceso de resincronización basándose en un error detectado por la unidad de comprobación de coherencia de memoria 352. El proceso de resincronización puede sincronizarse en sí entre los módulos de procesamiento, ya que los módulos de procesamiento minoritario y mayoritario pueden haberse retirado de la operación de bloqueo. Esta sincronización puede basarse en un evento o en un período de tiempo específico que transcurre después del inicio de una resincronización. Para coordinar el proceso de resincronización entre los módulos de procesamiento, el módulo de verificación 345 puede configurar los procesadores para verificar partes de los datos de estado de procesador o los datos de estado de memoria. Estas verificaciones pueden usarse por la unidad de comprobación de tolerancia a fallos 350 para identificar diferencias en estos estados entre los módulos de procesamiento. Esta información puede ayudar a los módulos de procesamiento 360, 370 y 380 para volver a la operación de bloqueo.

En otras realizaciones, el sistema 300 puede incluir unas capacidades de acceso directo a memoria (DMA). Estas capacidades de DMA pueden proporcionar la capacidad de leer y escribir en las localizaciones de las memorias locales 310, 311 y 312, sin cargar cada palabra de la memoria procesada en los procesadores 330, 331 y 332. Esto puede permitir la recopilación de datos de las memorias para el cálculo de verificación para que se produzca sustancialmente en paralelo con la operación de los procesadores 330, 331 y 332. En algunas realizaciones, los procesadores 330, 331 y 332 pueden incluir capacidades de DMA integradas.

Algunas realizaciones pueden proporcionar componentes de hardware adicionales para controlar el proceso de verificación. En algunas de estas realizaciones, los procesadores o componentes de hardware dedicados pueden controlar un proceso de DMA que lee datos de las memorias. Estos datos pueden usarse para crear verificaciones. Algunas de estas realizaciones también pueden recuperar los datos de estado de procesador de los procesadores 330, 331 y 332 usando unos componentes de hardware especializados. A continuación, estos componentes de hardware pueden calcular las verificaciones según sea apropiado basándose en los datos recopilados y enviar la información de verificación a la unidad de comprobación de tolerancia a fallos 350. Estas realizaciones pueden proporcionar un rendimiento mejorado en comparación con las realizaciones que utilizan la capacidad de procesador existente para calcular las verificaciones periódicamente. Estas realizaciones también pueden proporcionar un aumento del coste.

Cada procesador 330, 331, y 332, cuando se ejecuta el módulo de verificación 345 puede crear verificaciones como se ha descrito anteriormente para su módulo informático respectivo. Las verificaciones de cada procesador pueden pasarse a continuación a una unidad de comprobación de tolerancia a fallos 350. En una realización, la unidad de comprobación de tolerancia a fallos 350 está configurada para comparar las verificaciones generadas por los procesadores 330, 331 y 332 para determinar una verificación mayoritaria y una verificación minoritaria opcional. La unidad de comprobación de tolerancia a fallos 350 puede iniciar una resincronización de los módulos informáticos 360, 370 y 380 cuando difieren las verificaciones proporcionadas por los procesadores 330, 331 y 332. Un módulo de comprobación de tolerancia a fallos 350 puede representar un medio para determinar si el procesador dentro de cada módulo informático está sincronizado con el procesador de los otros módulos informáticos.

La unidad de comprobación de tolerancia a fallos 350 350 puede iniciar una resincronización enviando una señal a un módulo de resincronización 349 que se ejecuta en cada procesador 330, 331, y 332. El módulo de resincronización 349 incluye unas instrucciones que configuran cada procesador para realizar una resincronización. El módulo de resincronización 349 puede almacenarse como parte de una memoria 390.

La resincronización de los módulos informáticos 360, 370, y 380 puede controlarse mediante las instrucciones localizadas en el módulo de resincronización 349 de la figura 3. Una unidad de comprobación de tolerancia a fallos 350 junto con las instrucciones dentro de un módulo de resincronización que se ejecuta en cada uno de los

procesadores 330, 331 y 332 representan un medio para resincronizar un módulo informático basándose en las verificaciones creadas por los procesadores 330, 331 y 332.

El módulo de resincronización 349 puede incluir unas instrucciones que configuren cada uno de los procesadores 330, 331, y 332 para actualizar los datos de estado de procesamiento o los datos de estado de memoria de un módulo informático que produjo una verificación minoritaria. En algunas realizaciones, el estado de procesamiento de un módulo informático que produjo una verificación minoritaria puede actualizarse basándose en el estado de procesamiento de un módulo informático que produjo una verificación mayoritaria. Por ejemplo, si los módulos informáticos 360 y 370 produjeron el mismo valor de verificación, y el módulo informático 380 produjo un valor de verificación diferente, entonces el módulo informático 380 produjo la verificación minoritaria y los procesadores 360 y 370 produjeron una verificación mayoritaria. En este ejemplo, los datos de estado de procesamiento y/o los datos de estado del módulo informático 380 pueden actualizarse basándose en los datos de estado de procesamiento y los datos de estado de memoria del módulo informático 360 o del módulo informático 370.

Cuando se produce una verificación minoritaria, el módulo informático 380 puede recibir nuevo estado desde o el módulo informático 360 o el módulo informático 370. Por ejemplo, el módulo informático 380 puede recibir un nuevo estado para los registros 389 y 390 del módulo informático 360. El nuevo estado puede recibirse desde los registros 385 y 386 del procesador 330. El nuevo estado puede transmitirse directamente desde el módulo informático 360 al módulo informático 380 a través de un enlace 361 que conecta los procesadores 330, 331 y 332. En algunas implementaciones, el enlace 361 puede ser un Bus PCI Express u otro bus de alta velocidad con una conexión directa a cada procesador del sistema 300.

Los nuevos datos de estado recibidos desde o el módulo informático 360 o el módulo informático 370 también pueden incluir los datos de estado de memoria. Por ejemplo, el módulo informático 380 puede recibir nuevos datos de estado de memoria para partes o para la totalidad de la memoria 312 desde el módulo informático 360. Los nuevos datos de estado de memoria pueden recibirse desde la memoria 310. Por ejemplo, uno o más de los segmentos de memoria 316a-d pueden transferirse a través del enlace 361 al módulo informático 380. El módulo informático 380 puede reemplazar a continuación los contenidos de la memoria 312 con los datos de estado de memoria recibidos desde el módulo informático 360. Por ejemplo, el módulo informático 380 puede reemplazar los datos de estado de memoria para uno o más segmentos de memoria 318a-d al recibir uno o más segmentos de memoria correspondientes 316a-d desde el módulo informático 360.

Antes de que se complete una resincronización, existe la probabilidad de que un módulo informático mayoritario pueda experimentar un SEU. Esto puede resultar en la transferencia de datos de estado de procesador o datos de estado de memoria incorrectos al módulo informático minoritario. Este escenario de error puede hacer que los múltiples módulos informáticos generen resultados desiguales después de que se complete la resincronización. Sin embargo, si el SEU se produjo en los datos de estado antes de que se transfirieran desde un módulo informático mayoritario a un módulo informático minoritario, un módulo informático mayoritario y el módulo informático minoritario pueden generar resultados equivalentes pero incorrectos después de que se complete la resincronización. Debido a que estas dos unidades pueden formar una mayoría, por ejemplo, en las realizaciones que utilizan solo tres módulos informáticos, estas salidas incorrectas pueden aceptarse como una salida mayoritaria a menos que se proporcionen protecciones adicionales.

Para evitar este escenario, algunas realizaciones pueden realizar una verificación adicional después de que se complete un proceso de resincronización. Por ejemplo, estas realizaciones pueden realizar una resincronización parcial adicional para determinar si coinciden las verificaciones generadas por los módulos de procesamiento. Si los módulos informáticos que formaron una mayoría durante la resincronización anterior no generan unas verificaciones coincidentes, las etapas de recuperación pueden ser apropiadas. Como alternativa, si la verificación del módulo informático minoritario no coincide con la verificación de un módulo informático mayoritario desde el que el módulo informático minoritario no ha recibido el estado durante la resincronización anterior, las etapas de recuperación también pueden ser apropiadas. Por ejemplo, un error puede marcarse como resultado de la comparación de estas verificaciones. Como resultado del error, puede realizarse un reinicio de la placa de sistema de procesamiento tolerante a fallos. Si bien esto puede interrumpir las operaciones, puede hacer que el sistema vuelva a una operación correcta de manera verificable.

Cuando los datos de estado se transfieren entre el procesador 330 y el procesador 331, puede usarse un enlace 362 que se conecta directamente estos procesadores entre sí. De manera similar, cuando se transfieren nuevos datos de estado entre el procesador 331 y el procesador 332, puede usarse un enlace 364 que conecta directamente estos dos procesadores entre sí.

Cada procesador también puede tener acceso al banco de memoria 390 a través de la unidad de comprobación de tolerancia a fallos 350. Por ejemplo, cada procesador puede acceder a la memoria 390 después de una comprobación de coherencia de las salidas de los procesadores, como se ha descrito anteriormente con respecto a la figura 2, se realiza mediante la unidad de comprobación de tolerancia a fallos 350. Las memorias locales 310, 311 y 312, o las memorias accesibles a través de la unidad de comprobación de tolerancia a fallos 350, tal como la memoria 390, pueden almacenar unas instrucciones que configuran los procesadores 330, 331 y 332 para realizar

funciones. Por ejemplo, los procesadores 330, 331 y 332 pueden leer las localizaciones de la memoria en la memoria 390 a través de la unidad de comprobación de tolerancia a fallos 350. La memoria 390 puede incluir unas instrucciones que configuran los procesadores 330, 331 y 332. La memoria 390 también puede incluir unas instrucciones para un sistema operativo (no mostrado), o para el módulo de resincronización 349. Como alternativa, cada procesador 330, 331 y 332 puede leer las instrucciones de la memoria local dentro de su módulo informático respectivo. Por ejemplo, el procesador 330 puede leer las instrucciones de la memoria local 310. El procesador 331 puede leer las instrucciones de la memoria 311 y el procesador 332 puede leer las instrucciones de la memoria local 312.

La figura 4 es un diagrama de flujo que ilustra un proceso a modo de ejemplo 400 que puede ejecutarse en una unidad de comprobación de tolerancia a fallos para comparar las verificaciones recibidas desde una pluralidad de módulos informáticos. En algunas implementaciones, el proceso 400 puede implementarse mediante unas instrucciones dentro de la unidad de comprobación de tolerancia a fallos 350 de la figura 3, o de la unidad de comprobación de tolerancia a fallos 230 de la figura 2. El proceso 400 comienza en el bloque de inicio 405 y a continuación pasa al bloque de procesamiento 410, donde el proceso 400 espera para iniciar una resincronización parcial. En algunas implementaciones, puede realizarse una resincronización parcial de los procesadores periódicamente para garantizar que cada procesador esté sincronizado con los otros procesadores del sistema. Algunas realizaciones pueden realizar una resincronización parcial cada 1, 2, 5, 10, 20 o más milisegundos.

Una vez que el proceso 400 determina que es el momento de realizar una resincronización parcial, el proceso 400 se mueve al bloque de procesamiento 415, donde los procesadores se señalizan para realizar una resincronización parcial. Al recibir una señal de resincronización parcial, los procesadores pueden verificar sus datos de estado de procesamiento o sus datos de estado de memoria. La sincronización parcial se describe en la figura 5 a continuación. A continuación, el proceso 400 se mueve al bloque de procesamiento 420, donde se reciben las verificaciones de todos los módulos de procesamiento. A continuación, el proceso 400 se mueve al bloque de decisión 425, donde se evalúan las verificaciones para determinar si son todas equivalentes. En algunas implementaciones, las verificaciones pueden incluir verificaciones separadas tanto de los datos de estado de procesador como de los datos de estado de memoria de cada módulo informático. Algunas implementaciones pueden combinar las verificaciones de los datos de estado de procesamiento y los datos de estado de memoria en un único valor de verificación. En algunas implementaciones, la parte de la verificación que representa los datos de estado de memoria puede representar solo una parte de los datos de estado de memoria de los módulos informáticos. Otras implementaciones pueden incluir una representación de todos los datos de estado de memoria de los módulos informáticos en la verificación.

Si las verificaciones son equivalentes, el proceso 400 se mueve al bloque final 450 y el proceso 400 termina. Si todas las verificaciones no coinciden, puede haber una verificación minoritaria y al menos dos verificaciones mayoritarias. A continuación, el proceso 400 se mueve desde el bloque de decisión 425 al bloque de procesamiento 430, donde los módulos de procesamiento están señalizados para realizar una resincronización completa.

Una resincronización completa sincronizará los datos de estado de procesador o los datos de estado de memoria para un módulo informático que genera una verificación minoritaria con los datos de estado de un módulo informático que genera una verificación mayoritaria. El proceso de resincronización se describe en la figura 6 a continuación. A continuación, el proceso 400 se mueve al bloque 435, donde el proceso 400 envía una señal a los módulos informáticos minoritarios que indica que recibirán un nuevo estado. A continuación, el proceso 400 se mueve al bloque de procesamiento 440, donde el proceso 400 envía una señal a al menos un módulo informático mayoritario que le indica al módulo informático mayoritario que debería enviar un nuevo estado al módulo informático minoritario. A continuación, el proceso 400 se mueve al bloque 445, donde espera a que todos los módulos informáticos completen la resincronización. A continuación, el proceso 400 se mueve al estado final 450.

La figura 5 muestra un diagrama de flujo de un proceso de verificación a modo de ejemplo. En algunas implementaciones, el proceso 500 puede implementarse de acuerdo con las instrucciones incluidas en el módulo de verificación 290 de la figura 2. Como alternativa, el proceso 500 puede implementarse mediante las instrucciones incluidas en el módulo de verificación 345 de la figura 3. El proceso 550 comienza en el estado de inicio 555 y a continuación se mueve al bloque 560, donde espera una señal de resincronización parcial. La señal de resincronización parcial puede enviarse por el proceso 400, como se ha explicado anteriormente. Cuando se recibe una señal de resincronización parcial, el proceso 550 se mueve al bloque de procesamiento 565, donde se verifican los datos de estado de procesador. Para implementar el bloque 565, el procesador 330 del módulo informático 360 puede configurarse por las instrucciones incluidas en el módulo de verificación 345 para verificar sus estados de procesador 385 y 386. A continuación, el proceso 550 se mueve al bloque de procesamiento 570, donde se verifican los datos de estado de memoria. Para implementar el bloque 570, el procesador 330 del módulo informático 360 puede configurarse por las instrucciones incluidas en el módulo de verificación 345, ilustrado en la figura 3, para verificar una o más de las partes de memoria 316a-d. A continuación, el proceso 550 se mueve al bloque 575, donde la verificación o verificaciones resultantes de la verificación combinada de los datos de estado de procesador en el bloque 565 y los datos de estado de memoria en el bloque 570 se envían a la unidad de comprobación de tolerancia a fallos. Por ejemplo, el procesador 330 puede enviar una verificación a la unidad de comprobación de tolerancia a

fallos 350 para implementar el bloque de procesamiento 570. A continuación, el proceso 550 se mueve al bloque final 580.

La figura 6 muestra un diagrama de flujo de un proceso a modo de ejemplo 600 que realiza una resincronización completa de un módulo informático. El proceso 600 puede implementarse mediante las instrucciones incluidas en el módulo de resincronización 349, ilustrado en la figura 3. Como alternativa, el proceso 600 puede implementarse mediante un módulo de resincronización almacenado en una memoria local, tal como la memoria 310, ilustrada en la figura 3. Estas instrucciones pueden ejecutarse en un procesador dentro de un módulo informático. Por ejemplo, las instrucciones pueden ejecutarse en el procesador 330 del módulo informático 360.

El proceso 600 comienza en el bloque de inicio 605 y a continuación se mueve al bloque de procesamiento 610, donde el proceso 600 espera una solicitud de resincronización completa. Cuando se recibe una solicitud de resincronización, el proceso 600 se mueve al bloque 615 donde se guardan los datos de estado de procesador. Por ejemplo, si el procesador 330 del módulo informático 360 recibe una solicitud de resincronización, puede guardar los registros 385 y 386 al implementar el bloque de procesamiento 615 del proceso 600. A continuación, el proceso 600 se mueve al bloque de decisión 620, donde determina si recibirá nuevos estados. Si recibe un nuevo estado, el proceso 600 se mueve al bloque 650, donde se reciben los nuevos datos de estado de procesador. Estos nuevos datos de estado sobrescriben cualquier dato guardado en el bloque de procesamiento 615. A continuación, el proceso 600 se mueve al bloque 655, donde se reciben los nuevos datos de estado de memoria. En algunas implementaciones, este estado de memoria puede incluir el estado de una memoria local completa, por ejemplo, la memoria local 310, 311 o 312, ilustrada en la figura 3. En otras implementaciones, este estado de memoria puede incluir solo una parte de una memoria local. Por ejemplo, puede incluir solo un segmento de memoria, por ejemplo, el segmento de memoria 316a, 317a o 318a, también ilustrado en la figura 3. Después de que se hayan recibido los datos de estado de memoria, el proceso 600 se mueve al bloque de procesamiento 640, donde se envía una señal de que se ha completado la resincronización.

Volviendo al bloque de decisión 620, si no se recibe un nuevo estado por el módulo informático que ejecuta el proceso 600, el proceso 600 se mueve al bloque de decisión 625, donde se determina si debería enviarse el estado. Si no se debe enviar ningún estado, el proceso 600 se mueve al bloque 640, donde indica que se ha completado su proceso de resincronización. Si en el bloque de decisión 625 se determina que el módulo de procesamiento que ejecuta el proceso 600 debería enviar el estado, el proceso 600 se mueve al bloque de procesamiento 630, donde se envían los datos de estado de procesador guardados. A continuación, el proceso 600 se mueve al bloque de procesamiento 635, donde se envían nuevos datos de estado de memoria. Los datos de estado de memoria enviados en el bloque 635 pueden incluir una parte o toda la memoria local. Por ejemplo, puede incluir una parte de la memoria 310, por ejemplo, el segmento de memoria 316a. Como alternativa, puede incluir dos segmentos de memoria, por ejemplo, los segmentos de memoria 316a-b. Todavía otras implementaciones pueden enviar la memoria completa 310, para incluir los segmentos de memoria 316a-d. A continuación, el proceso 600 se mueve al bloque de procesamiento 640, donde señala que su parte de la resincronización está completa. Después de procesar el bloque 640, el proceso 600 se mueve al bloque 645, donde el proceso 600 espera a que todos los módulos de procesamiento completen sus eventos de resincronización. Cuando todos los módulos de procesamiento han señalado (a través del bloque de procesamiento 640) que sus eventos de resincronización están completos, el proceso 600 se mueve al bloque 670, donde se restauran los datos de estado de procesamiento guardados del bloque de procesamiento 615 y el proceso 600 regresa del evento. A continuación, el proceso 600 se mueve al bloque final 680.

La figura 7 muestra un diagrama de flujo que ilustra una implementación a modo de ejemplo de un método de verificación. El proceso 700 puede implementarse mediante las instrucciones incluidas en el módulo de verificación 345 ilustrado en la figura 3. El proceso 700 comienza en el bloque de inicio 710 y a continuación se mueve al bloque de procesamiento 720, donde se inicializa una verificación. A continuación, el proceso 700 se mueve al bloque de procesamiento 730, donde se verifican los datos de estado de procesamiento. Por ejemplo, el procesador 330 puede configurarse por las instrucciones en el módulo de verificación 345 para verificar sus estados de procesador 385 y 386 para implementar el bloque de procesamiento 723. A continuación, el proceso 700 se mueve al bloque de decisión 740, donde determina si se realizará una verificación de memoria completa o una verificación de memoria parcial. El proceso 700 se mueve al bloque de procesamiento 750 si se realiza una verificación de memoria completa. El bloque de procesamiento 750 puede incluir todo el contenido de la memoria al determinar el valor de verificación. Si el bloque de decisión 740 determina que solo se incluirá una parte de la memoria en la verificación, el proceso 700 se mueve al bloque de procesamiento 780, donde se incrementa un identificador de segmentos de memoria local. A continuación, el proceso 700 se mueve al bloque 790, donde se continúa con la verificación con el segmento de memoria identificado por el identificador de segmentos.

Un identificador de segmentos de memoria local puede realizar un seguimiento de un segmento de memoria que debería incluirse en la verificación. Por ejemplo, algunas implementaciones pueden variar el segmento de memoria incluido en la verificación para cada resincronización parcial. Algunas implementaciones pueden incluir el primer segmento de memoria en la primera verificación que se crea. Cada sincronización parcial posterior puede incluir un segmento de memoria diferente como parte del valor de verificación. Por ejemplo, la segunda resincronización parcial puede incluir un segundo segmento de memoria, y una tercera resincronización parcial puede incluir un tercer

segmento de memoria. Cuando todos los segmentos de memoria de una memoria se han incluido como parte de una resincronización parcial, el proceso 700 puede usar nuevamente los contenidos del primer segmento de memoria en una verificación. A continuación, puede repetirse esta inclusión incremental de diferentes segmentos de memoria.

Después de que el segmento de memoria identificado se haya incluido en la verificación, el proceso 700 se mueve al bloque 760, donde se proporciona la verificación. Por ejemplo, la verificación puede proporcionarse a la unidad de comprobación de tolerancia a fallos 350, ilustrada en la figura 3. A continuación, el proceso 700 se mueve al bloque final 770.

La figura 8 muestra un diagrama de flujo que ilustra un proceso de memoria recibida a modo de ejemplo 655. El proceso 655 puede implementarse mediante unas instrucciones incluidas en el módulo de resincronización 349, que se ilustra en la figura 3. Como alternativa, el proceso 655 puede implementarse mediante las instrucciones almacenadas en las memorias locales 310, 311 o 312, también ilustradas en la figura 3. El proceso 655 comienza en el bloque de inicio 805 y a continuación se mueve al bloque de decisión 810, donde determina si el módulo informático que ejecuta el proceso 655 recibirá una imagen de memoria completa o una imagen de memoria parcial. Si recibe una imagen de memoria completa, el proceso 655 se mueve al bloque de procesamiento 820, donde se recibe la imagen de memoria completa. Por ejemplo, el módulo informático 360 puede recibir una imagen de memoria para la memoria local 310. Esta imagen puede incluir nuevos datos de estado de memoria para cada segmento de memoria de la memoria local 310, para incluir los segmentos de memoria 316a-d. A continuación, el proceso 655 se mueve al bloque 850. Volviendo al bloque de decisión 810, el proceso 655 determina que solo puede recibirse una imagen de memoria parcial, el proceso 655 se mueve al bloque 830, donde se recibe un segmento de memoria. Por ejemplo, el módulo informático 360 puede recibir una imagen para un segmento de memoria de la memoria local 310. Por ejemplo, puede recibirse una imagen solo para el segmento de memoria 316a en el bloque de procesamiento 830. A continuación, el proceso 655 se mueve al bloque 840, donde el segmento de memoria local recibido reemplaza a un segmento de memoria basándose en el ID del segmento. El ID del segmento referenciado en el bloque 840 puede ser el mismo ID del segmento que la referenciada en el bloque 780 del proceso 700, ilustrada en la figura 7. A continuación, el proceso 655 se mueve al bloque final 850.

La figura 9 muestra un diagrama de flujo que ilustra un proceso de envío de memoria a modo de ejemplo 635. El proceso 635 puede implementarse mediante unas instrucciones incluidas en el módulo de resincronización 349, ilustrado en la figura 3. El proceso 635 comienza en el bloque de inicio 905 y a continuación se mueve al bloque de decisión 910, donde se realiza una determinación de si se enviará un estado de memoria completa o si se enviará un estado de memoria parcial. Si se envía un estado de memoria completa, el proceso 635 se mueve al bloque 920 donde se envía el estado de memoria completa. A continuación, el proceso 635 se mueve al bloque final 950. Si se envía una imagen de memoria parcial, el proceso 635 se mueve al bloque 935, donde el segmento de memoria se identifica basándose en un identificador de segmentos. El identificador de segmentos referenciado en el bloque 930 puede ser el mismo identificador referenciado en el bloque 780, ilustrado en la figura 7. A continuación, el proceso 635 se mueve al bloque 940, donde se envía el segmento de memoria identificado por el identificador de segmentos. A continuación, el proceso 635 se mueve al bloque final 950.

La figura 10 muestra un diagrama de flujo que ilustra un proceso de envío y recepción de memoria a modo de ejemplo. El proceso 1000 puede ser una implementación alternativa a los procesos 635 o 655, tratados anteriormente. El proceso 1000 puede implementarse mediante las instrucciones incluidas en el módulo de verificación 345 o en el módulo de sincronización 349, de una combinación de los dos módulos, ilustrada en la figura 3. Múltiples módulos de procesamiento, tales como los módulos de procesamiento 360, 370 y 380, ilustrados en la figura 3, pueden realizar el proceso 1000 de manera sustancialmente simultánea con el fin de coordinar una resincronización de los contenidos de la memoria local entre los módulos de procesamiento.

El proceso 1000 comienza en el bloque de inicio 1005 y a continuación se mueve al bloque 1010 donde se inicializa un identificador de partes. Este identificador de partes puede usarse para identificar una parte de una memoria, tal como las partes de las memorias locales 310, 311 o 312. En algunas realizaciones, una parte de una memoria puede ser un segmento de memoria. A continuación, el proceso 1000 se mueve al bloque 1015, donde se verifica una parte de la memoria basándose en el identificador. A continuación, el proceso 1000 se mueve al bloque 1020, donde la verificación se envía a una unidad de comprobación de tolerancia a fallos, por ejemplo, la unidad de comprobación de tolerancia a fallos 350 de la figura 3.

A continuación, el proceso 1000 se mueve al bloque 1025 y espera una respuesta. El proceso 1000 puede estar esperando una respuesta de una unidad de comprobación de tolerancia a fallos. Por ejemplo, la unidad de comprobación de tolerancia a fallos puede comparar las verificaciones enviadas por los módulos informáticos 360, 370 y 380 que realizan el proceso 1000. Después de recibir una respuesta, el proceso 1000 se mueve al bloque de decisión 1030 donde se determina si coinciden las verificaciones recibidas. Por ejemplo, el módulo de comprobación de tolerancia a fallos 350 puede enviar una respuesta a los módulos de procesamiento que ejecutan el proceso 1000, indicando si la verificación calculada en el bloque de procesamiento 1015 ha coincidido con las verificaciones creadas por otros módulos de procesamiento, que también ejecutan el proceso 1000. Si las verificaciones coinciden, el proceso 1000 se mueve al bloque de decisión 1050.

Si los valores de verificación no coinciden, el proceso 1000 se mueve a bloque de decisión 1035, donde determina si esta instancia específica del proceso 1000 enviará o recibirá datos. Si las verificaciones no coinciden, un módulo de procesamiento puede haber creado una verificación minoritaria. Un módulo de procesamiento que crea una verificación minoritaria y ejecuta el proceso 1000 puede recibir una nueva parte de la memoria, con el fin de resincronizarse con los módulos de procesamiento que crearon las verificaciones mayoritarias. El módulo de procesamiento minoritario puede moverse desde el bloque de decisión 1035 al bloque 1045. En el bloque 1045, el proceso 1000 puede recibir una nueva parte de memoria desde otro módulo de procesamiento, por ejemplo, un módulo de procesamiento que ha creado una verificación mayoritaria.

Otro módulo informático que realiza el proceso 1000 pueden haber creado una verificación mayoritaria. Este módulo de procesamiento puede determinar en el bloque de decisión 1035 que enviará una parte de la memoria a un módulo de procesamiento que ha creado una verificación minoritaria. En este caso, este módulo de procesamiento puede moverse desde el bloque de decisión 1035 al bloque 1040.

Después de que una parte de la memoria se haya enviado a través del bloque 1040 o recibido a través del bloque 1045, el proceso 1000 se mueve al bloque de decisión 1050, donde determina si hay partes de memoria adicionales que deben sincronizarse. Si no hay partes restantes que deben sincronizarse, el proceso 1000 se mueve al estado final 1060. Si hay partes restantes que deben sincronizarse, el proceso 1000 se mueve al bloque 1055, donde se aumenta el identificador de partes de memoria. A continuación, el proceso 1000 regresa al bloque 1015 y se verifica la parte de memoria identificada por el identificador aumentado en el bloque 1055. A continuación, el proceso 1000 se repite como se ha descrito anteriormente.

Las diversas lógicas ilustrativas, bloques lógicos, módulos, circuitos y etapas de algoritmo descritos junto con las implementaciones desveladas en el presente documento pueden implementarse como hardware electrónico, software informático, o combinaciones de ambos. La capacidad de intercambio del hardware y el software se ha descrito en general, en términos de funcionalidad, y se ilustra en los diversos componentes ilustrativos, bloques, módulos, circuitos y etapas descritos anteriormente. Si dicha funcionalidad se implementa en hardware o software depende de la aplicación específica y las restricciones de diseño impuestas en el sistema global.

El hardware y el aparato de procesamiento de datos utilizados para implementar las diversas lógicas, bloques lógicos, módulos y circuitos descritos junto con los aspectos desvelados en el presente documento pueden implementarse o realizarse con un procesador de uno o varios chips de fin general, un procesador de señal digital (DSP), un circuito integrado de aplicación específica (ASIC), una matriz de puertas programable en campo (FPGA) u otro dispositivo lógico programable, puerta discreta o lógica de transistor, componentes de hardware discretos, o cualquier combinación de los mismos diseñados para realizar las funciones descritas en el presente documento. Un procesador de fin general puede ser un microprocesador o cualquier procesador convencional, controlador, microcontrolador o máquina de estados. Un procesador también puede implementarse como una combinación de módulos informáticos, por ejemplo, una combinación de un DSP y un microprocesador, una pluralidad de microprocesadores, uno o más microprocesadores junto con un núcleo DSP, o cualquier otra configuración de este tipo. En algunas implementaciones, pueden realizarse etapas y métodos específicos mediante una circuitería que sea específica de una función determinada.

En uno o más aspectos, las funciones descritas pueden implementarse en hardware, circuitos electrónicos digitales, software informático, firmware, incluyendo las estructuras descritas en esta memoria descriptiva y sus equivalentes estructurales, o en cualquier combinación de los mismos. Las implementaciones del objeto descrito en esta memoria descriptiva también pueden implementarse como uno o más programas informáticos, es decir, uno o más módulos de instrucciones de programa informático, codificados en un medio de almacenamiento informático para ejecutarse por, o para controlar la operación de, el aparato de procesamiento de datos.

Diversas modificaciones a las implementaciones descritas en esta divulgación pueden ser fácilmente evidentes para los expertos en la materia, y los principios genéricos definidos en el presente documento pueden aplicarse a otras implementaciones, sin alejarse del alcance de esta divulgación. Por lo tanto, no se pretende que las reivindicaciones se limiten a las implementaciones que se muestran en el presente documento, sino que se les debe otorgar el alcance más amplio compatible con esta divulgación, los principios y las características novedosas descritas en el presente documento. La palabra “ejemplar” se usa exclusivamente en el presente documento para significar “que sirve como ejemplo, instancia o ilustración”. Cualquier implementación descrita en el presente documento como “ejemplar” no debe interpretarse necesariamente como preferida o ventajosa sobre otras implementaciones. Además, un experto en la materia apreciará fácilmente que, los términos “superior” e “inferior” se utilizan a veces para facilitar la descripción de las figuras, e indican las posiciones relativas correspondientes a la orientación de la figura en una página orientada correctamente, y puede no reflejar la orientación correcta del dispositivo tal como se implementó.

Algunas de las funciones que se describen en esta memoria descriptiva en el contexto de implementaciones separadas también pueden implementarse en combinación en una única aplicación. Por el contrario, varias características que se describen en el contexto de una única implementación también pueden implementarse en implementaciones múltiples por separado o en cualquier subcombinación adecuada. Además, aunque las

características pueden describirse anteriormente como actuando en ciertas combinaciones e incluso inicialmente reivindicadas como tales, una o más características de una combinación reivindicada pueden en algunos casos eliminarse de la combinación, y la combinación reivindicada puede dirigirse a una subcombinación o variación de una subcombinación.

Del mismo modo, mientras que las operaciones se representan en los dibujos en un orden específico, esto no debería entenderse como que se requiere que tales operaciones se realicen en el orden específico mostrado o en orden secuencial, o que pueden realizarse todas las operaciones ilustradas, para lograr los resultados deseables. Además, los dibujos pueden representar esquemáticamente uno o más procesos de ejemplo en la forma de un diagrama de flujo. Sin embargo, otras operaciones que no están representadas pueden incorporarse en los procesos de ejemplo que se ilustran esquemáticamente. Por ejemplo, una o más operaciones adicionales pueden realizarse antes, después, simultáneamente o entre cualquiera de las operaciones ilustradas. En ciertas circunstancias, la multitarea y el procesamiento paralelo pueden ser ventajosos. Además, la separación de diversos componentes del sistema en las implementaciones descritas anteriormente no debería entenderse como que se requiere dicha separación en todas las implementaciones, y debería entenderse que los componentes de programa y los sistemas descritos pueden integrarse en general juntos en un único producto de software o empaquetarse en múltiples productos de software. Además, otras implementaciones están dentro del alcance de las siguientes reivindicaciones. En algunos casos, las acciones mencionadas en las reivindicaciones pueden realizarse en un orden diferente y aun así lograr los resultados deseables.

REIVINDICACIONES

1. Un método para sincronizar el estado de una pluralidad de módulos informáticos en un sistema electrónico, teniendo cada módulo informático un procesador, que comprende:

guardar al menos una parte de los datos de estado de procesador para cada uno de la pluralidad de módulos informáticos;
verificar al menos una parte de los datos de estado de procesador guardados para cada uno de la pluralidad de módulos informáticos;
comparar las verificaciones de procesador para los datos de estado de procesador;
resincronizar la pluralidad de módulos informáticos si se determina que la mayoría de los módulos informáticos tienen los mismos datos de estado de procesador, y se determina que la minoría de los módulos informáticos tienen diferentes datos de estado de procesador, caracterizado por que la resincronización comprende:

enviar los datos de estado de procesador guardados desde un primer módulo informático mayoritario a un primer módulo informático minoritario;
confirmar que los datos de estado de un segundo módulo informático mayoritario son los mismos que los datos de estado guardados del primer módulo informático mayoritario o los datos de estado guardados del primer módulo informático minoritario después de enviar los datos de estado de procesador guardados desde el primer módulo informático mayoritario al módulo informático minoritario;
marcar un error si los datos de estado no son los mismos; y
restaurar los datos de estado de procesador de un módulo informático mayoritario basándose en los datos de estado de procesador guardados del módulo informático mayoritario en respuesta a la finalización de la resincronización.

2. El método de la reivindicación 1, que comprende, además:

verificar los datos de estado de memoria para cada uno de la pluralidad de módulos informáticos;
comparar las verificaciones de memoria para los datos de estado de memoria; y
resincronizar la pluralidad de módulos informáticos basándose al menos en las verificaciones de memoria comparadas.

3. El método de la reivindicación 1, en el que la resincronización comprende enviar los datos de estado de procesador desde un primer módulo informático a un segundo módulo informático.

4. El método de la reivindicación 3, en el que enviar los datos de estado de procesador comprende enviar los datos de estado de procesador a través de un bus de datos que conecta directamente el primer módulo informático con el segundo módulo informático.

5. El método de la reivindicación 2, en el que el método se realiza a intervalos regulares durante la operación del sistema electrónico, y durante cada intervalo regular, el método crea valores de verificación para solo una parte predeterminada de los datos de estado de memoria en cada módulo informático.

6. El método de la reivindicación 5, en el que el método compara secuencialmente los valores de verificación para todos los datos de estado de memoria en una pluralidad de intervalos regulares.

7. El método de la reivindicación 1, en el que cada uno de los módulos informáticos comprende un núcleo de procesador de un procesador de múltiples núcleos.

8. Un aparato informático tolerante a fallos, que comprende:

una pluralidad de módulos informáticos, en el que cada módulo informático comprende un procesador de hardware que tiene datos de estado de procesador;
medios para que la verificación configurada genere unos valores de verificación de los datos de estado de procesador;
medios para guardar al menos una parte de los datos de estado de procesador para cada uno de la pluralidad de módulos informáticos;
medios para que la verificación configurada genere unos valores de verificación de al menos los datos de estado de procesador guardados;
medios para comparar la pluralidad de valores de verificación;
medios para determinar una mayoría de módulos informáticos que tengan los mismos datos de estado de procesador y al menos una minoría de módulos informáticos que tengan diferentes datos de estado de procesador; y
medios para resincronizar la pluralidad de módulos informáticos basándose en la determinación, caracterizado por que los medios de resincronización están configurados para:

enviar los datos de estado de procesador guardados desde un primer módulo informático mayoritario a un primer módulo informático minoritario;
confirmar que los datos de estado de un segundo módulo informático mayoritario son los mismos que los datos de estado guardados del primer módulo informático mayoritario o los datos de estado guardados del primer módulo informático minoritario después de enviar los datos de estado de procesador guardados desde el primer módulo informático mayoritario al módulo informático minoritario, y marcar un error si los datos de estado no son los mismos; y

medios para restaurar los datos de estado de procesador de un módulo informático mayoritario basándose en los datos de estado de procesador guardados del módulo informático mayoritario en respuesta a la finalización de la resincronización.

9. El aparato informático tolerante a fallos de la reivindicación 8, en el que los medios para comparar comprenden una matriz de puerta programable en campo (FPGA) programada para comparar los valores de verificación de los datos de estado de procesador.

10. El aparato informático tolerante a fallos de la reivindicación 8, en el que los medios para comparar incluyen una unidad de comprobación de tolerancia a fallos.

11. El aparato informático tolerante a fallos de la reivindicación 8, en el que los medios para determinar incluyen uno o más módulos de resincronización.

12. Un medio de almacenamiento legible por ordenador, no transitorio, que tiene unas instrucciones almacenadas en el mismo que hacen que un circuito de procesamiento realice un método que comprende:

guardar al menos una parte de los datos de estado de procesador para cada uno de la pluralidad de módulos informáticos;
verificar al menos los datos de estado de procesador guardados para cada uno de una pluralidad de módulos informáticos;
comparar las verificaciones de procesador para los datos de estado de procesador; y
determinar una mayoría de módulos informáticos que tengan los mismos datos de estado de procesador y al menos una minoría de módulos informáticos que tengan diferentes datos de estado de procesador;
resincronizar la pluralidad de módulos informáticos basándose al menos en la determinación, caracterizado por que la resincronización comprende:

enviar datos de estado de procesador desde un primer módulo informático mayoritario a un primer módulo informático minoritario;
confirmar que los datos de estado de un segundo módulo informático mayoritario son los mismos que los datos de estado guardados del primer módulo informático mayoritario o los datos de estado guardados del primer módulo informático minoritario después de enviar los datos de estado de procesador guardados desde el primer módulo informático mayoritario al módulo informático minoritario, y
marcar un error si los datos de estado no son los mismos; y

restaurar los datos de estado de procesador de un módulo informático mayoritario basándose en los datos de estado de procesador guardados del módulo informático mayoritario en respuesta a la finalización de la resincronización.

13. El medio de almacenamiento legible por ordenador de la reivindicación 12, que incluye además unas instrucciones que hacen que un circuito de procesamiento:

verifique los datos de estado de memoria para cada uno de la pluralidad de módulos informáticos;
compare las verificaciones de memoria para los datos de estado de memoria; y
resincronice la pluralidad de módulos informáticos basándose al menos en las verificaciones de memoria comparadas.

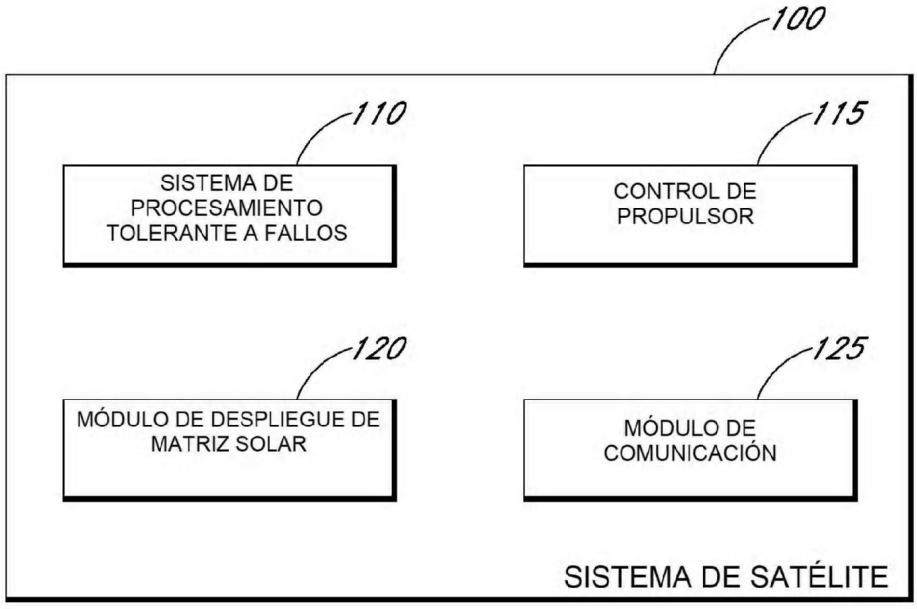
100
110
SISTEMA DE PROCESAMIENTO TOLERANTE A FALLOS
115
CONTROL DE PROPULSOR
120
MÓDULO DE DESPLIEGUE DE MATRIZ SOLAR
125
MÓDULO DE COMUNICACIÓN
SISTEMA DE SATÉLITE

FIG. 1

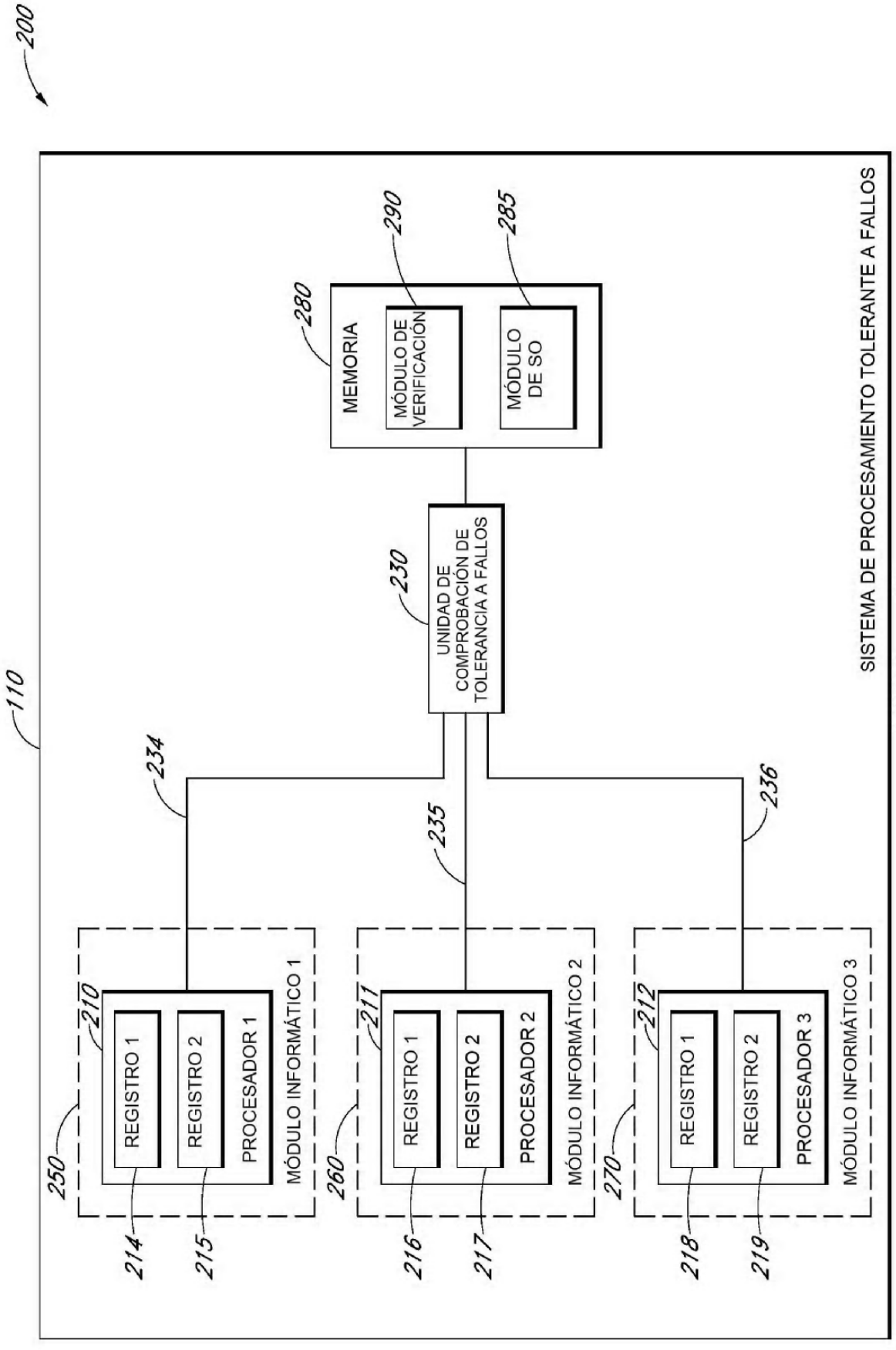
200
110
SISTEMA DE PROCESAMIENTO TOLERANTE A FALLOS
MEMORIA
280
MÓDULO DE VERIFICACIÓN
290
MÓDULO DE SO
285
UNIDAD DE COMPROBACIÓN DE TOLERANCIA A FALLOS
230
234
235
236
MÓDULO INFORMÁTICO 1
250
210
REGISTRO 1
REGISTRO 2
PROCESADOR 1
214
215
MÓDULO INFORMÁTICO 2
260
211
REGISTRO 1
REGISTRO 2
PROCESADOR 2
216
217
MÓDULO INFORMÁTICO 3
270
212
REGISTRO 1
REGISTRO 2
PROCESADOR 3
218
219

FIG. 2

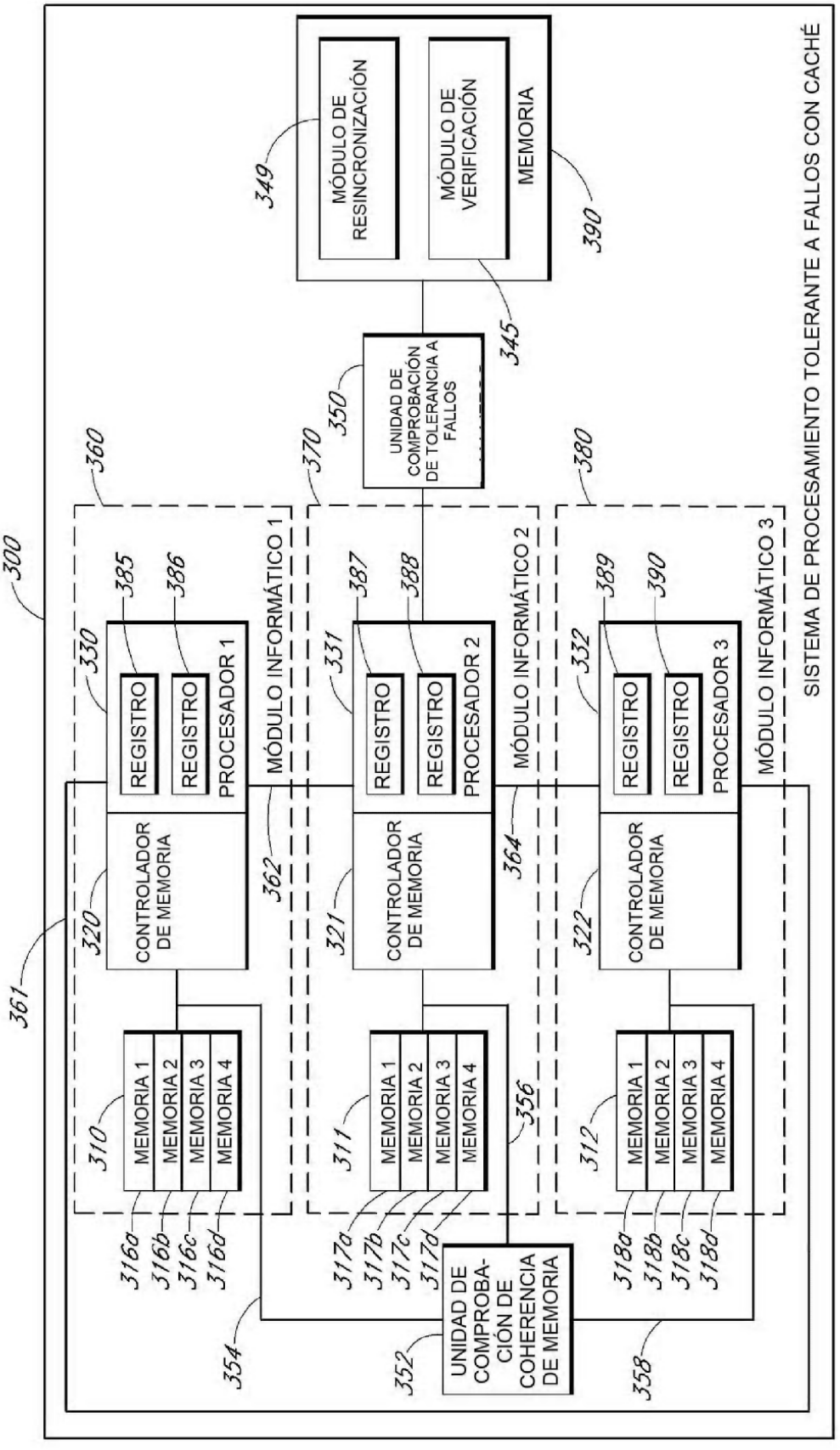
SISTEMA DE PROCESAMIENTO TOLERANTE A FALLOS CON CACHÉ
MÓDULO INFORMÁTICO 1
MÓDULO INFORMÁTICO 2
MÓDULO INFORMÁTICO 3
PROCESADOR 1
PROCESADOR 2
PROCESADOR 3
REGISTRO
CONTROLADOR DE MEMORIA
MEMORIA 1
MEMORIA 2
MEMORIA 3
MEMORIA 4
UNIDAD DE COMPROBACIÓN DE COHERENCIA DE MEMORIA
UNIDAD DE COMPROBACIÓN DE TOLERANCIA A FALLOS
MÓDULO DE RESINCRONIZACIÓN
MÓDULO DE VERIFICACIÓN
MEMORIA
300
310
311
312
316a
316b
316c
316d
317a
317b
317c
317d
318a
318b
318c
318d
320
321
322
330
331
332
345
349
350
352
354
356
358
360
361
362
364
370
380
385
386
387
388
389
390

FIG. 3

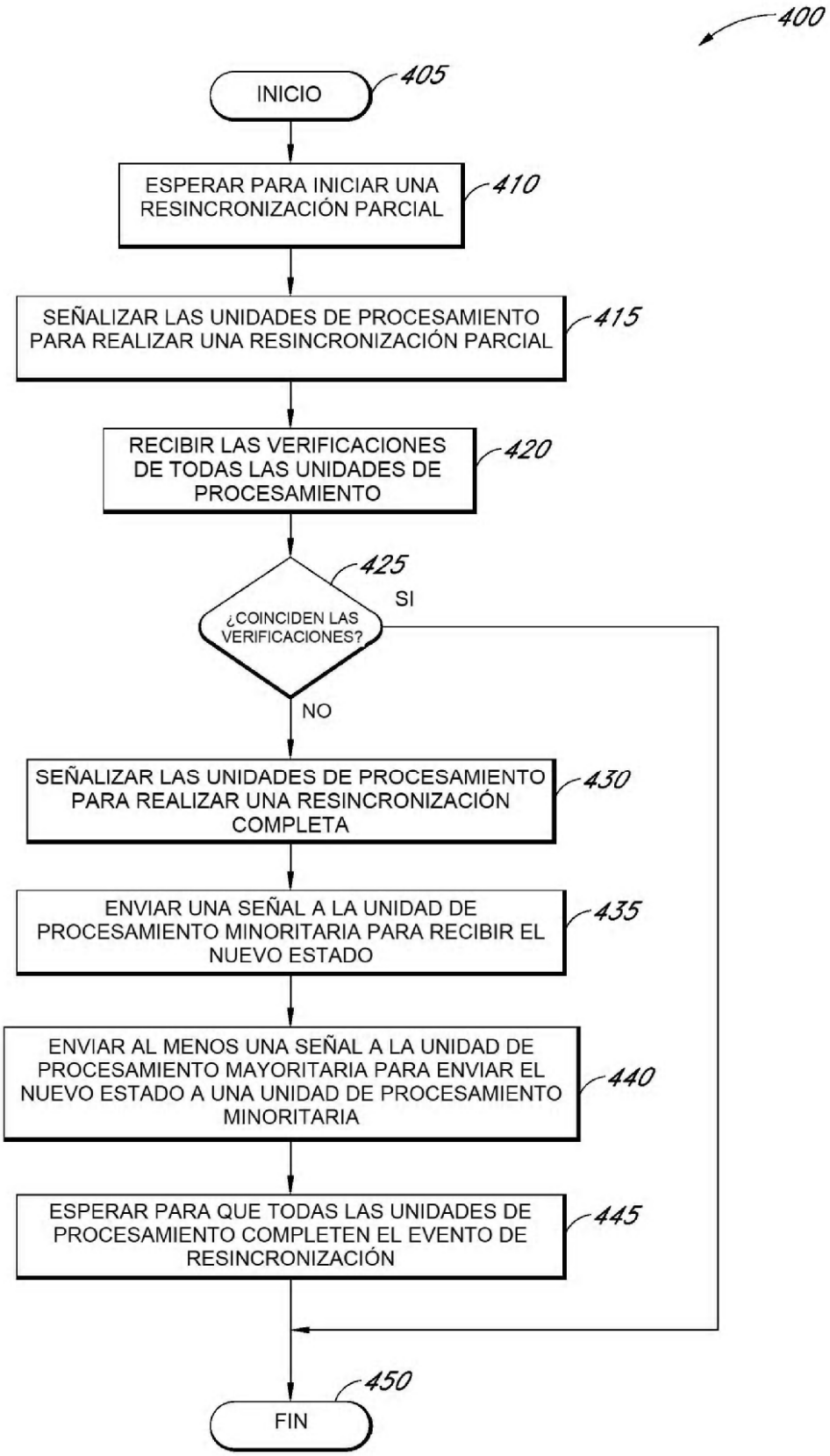
400
INICIO
405
ESPERAR PARA INICIAR UNA RESINCRONIZACIÓN PARCIAL
410
SEÑALIZAR LAS UNIDADES DE PROCESAMIENTO PARA REALIZAR UNA RESINCRONIZACIÓN PARCIAL
415
RECIBIR LAS VERIFICACIONES DE TODAS LAS UNIDADES DE PROCESAMIENTO
420
¿COINCIDEN LAS VERIFICACIONES?
425
SI
NO
SEÑALIZAR LAS UNIDADES DE PROCESAMIENTO PARA REALIZAR UNA RESINCRONIZACIÓN COMPLETA
430
ENVIAR UNA SEÑAL A LA UNIDAD DE PROCESAMIENTO MINORITARIA PARA RECIBIR EL NUEVO ESTADO
435
ENVIAR AL MENOS UNA SEÑAL A LA UNIDAD DE PROCESAMIENTO MAYORITARIA PARA ENVIAR EL NUEVO ESTADO A UNA UNIDAD DE PROCESAMIENTO MINORITARIA
440
ESPERAR PARA QUE TODAS LAS UNIDADES DE PROCESAMIENTO COMPLETEN EL EVENTO DE RESINCRONIZACIÓN
445
FIN
450

FIG. 4

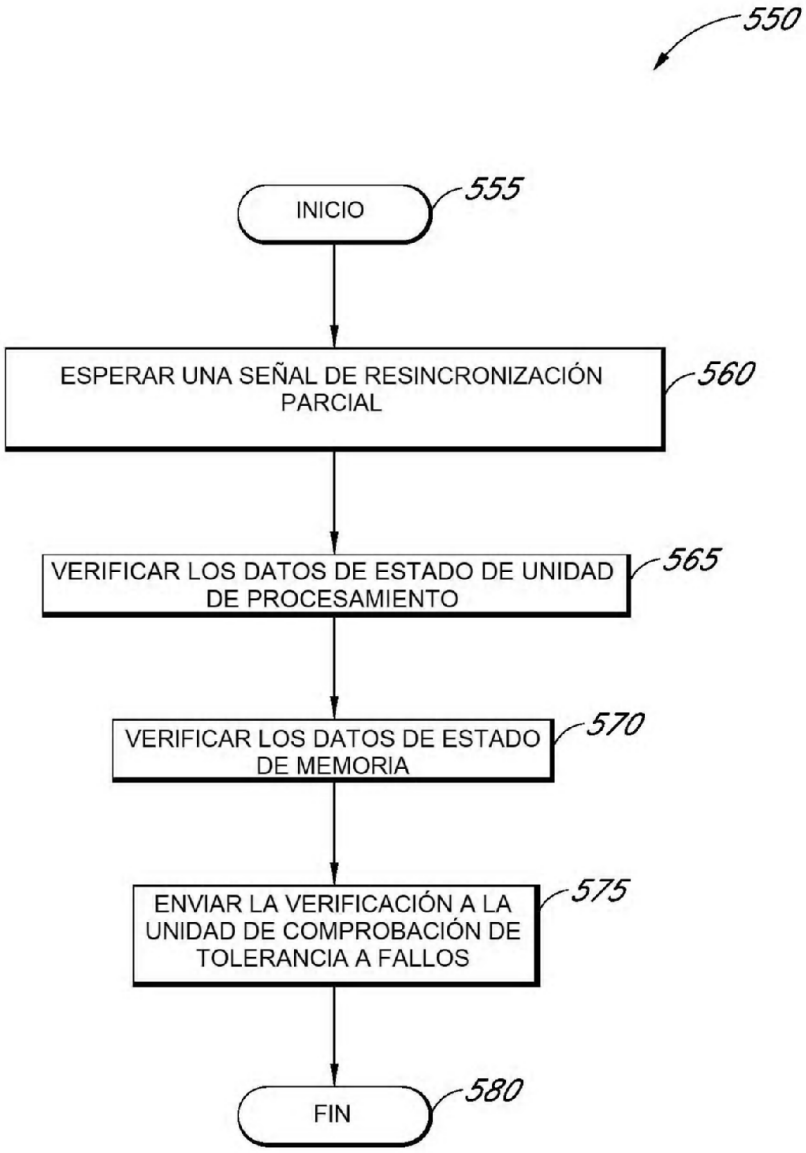
550
INICIO
555
ESPERAR UNA SEÑAL DE RESINCRONIZACIÓN PARCIAL
560
VERIFICAR LOS DATOS DE ESTADO DE UNIDAD DE PROCESAMIENTO
565
VERIFICAR LOS DATOS DE ESTADO DE MEMORIA
570
ENVIAR LA VERIFICACIÓN A LA UNIDAD DE COMPROBACIÓN DE TOLERANCIA A FALLOS
575
FIN
580

FIG. 5

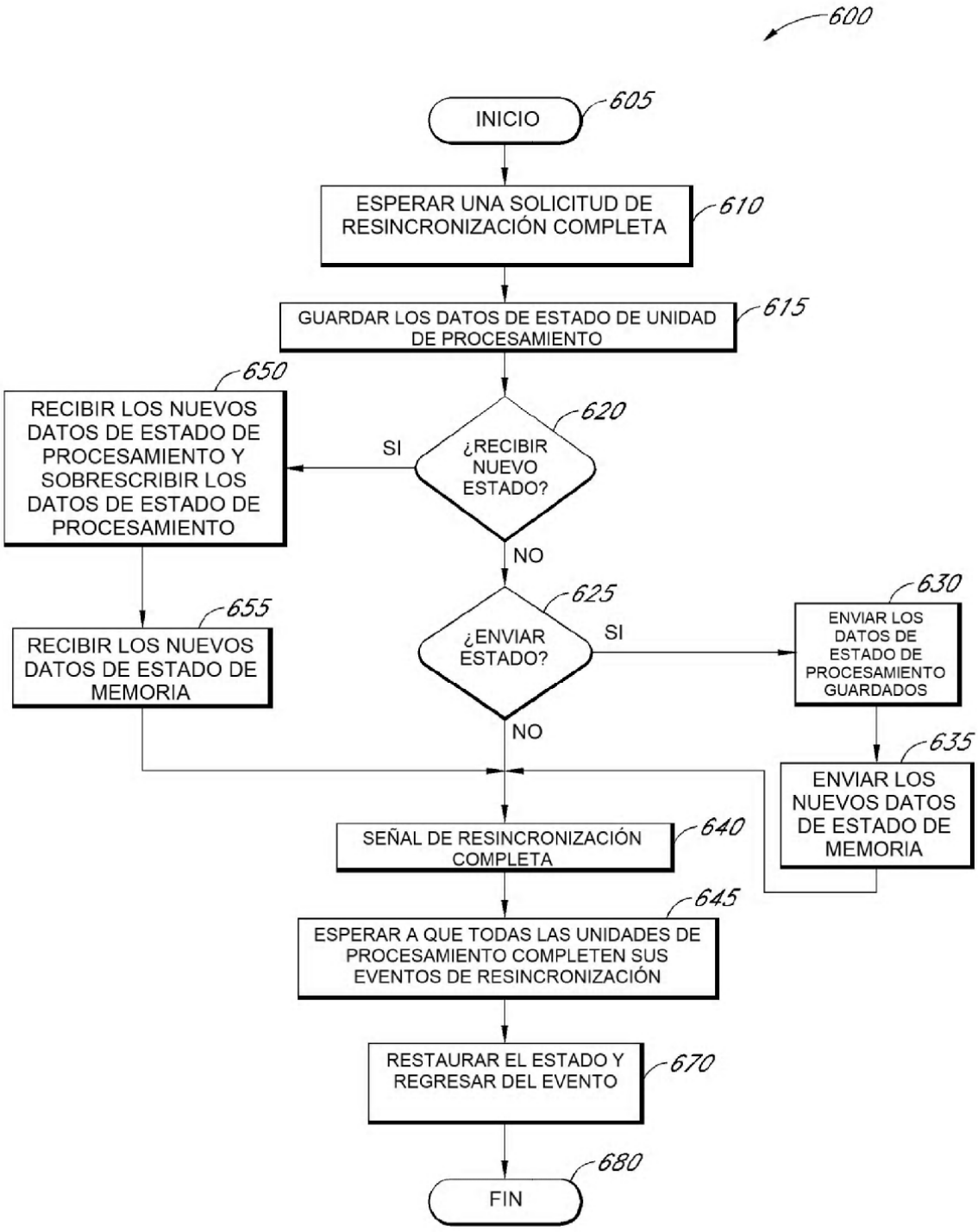
600
INICIO
605
ESPERAR UNA SOLICITUD DE RESINCRONIZACIÓN COMPLETA
610
GUARDAR LOS DATOS DE ESTADO DE UNIDAD DE PROCESAMIENTO
615
¿RECIBIR NUEVO ESTADO?
620
SI
RECIBIR LOS NUEVOS DATOS DE ESTADO DE PROCESAMIENTO Y SOBRESCRIBIR LOS DATOS DE ESTADO DE PROCESAMIENTO
650
NO
¿ENVIAR ESTADO?
625
SI
ENVIAR LOS DATOS DE ESTADO DE PROCESAMIENTO GUARDADOS
630
NO
RECIBIR LOS NUEVOS DATOS DE ESTADO DE MEMORIA
655
ENVIAR LOS NUEVOS DATOS DE ESTADO DE MEMORIA
635
SEÑAL DE RESINCRONIZACIÓN COMPLETA
640
ESPERAR A QUE TODAS LAS UNIDADES DE PROCESAMIENTO COMPLETEN SUS EVENTOS DE RESINCRONIZACIÓN
645
RESTAURAR EL ESTADO Y REGRESAR DEL EVENTO
670
FIN
680

FIG. 6

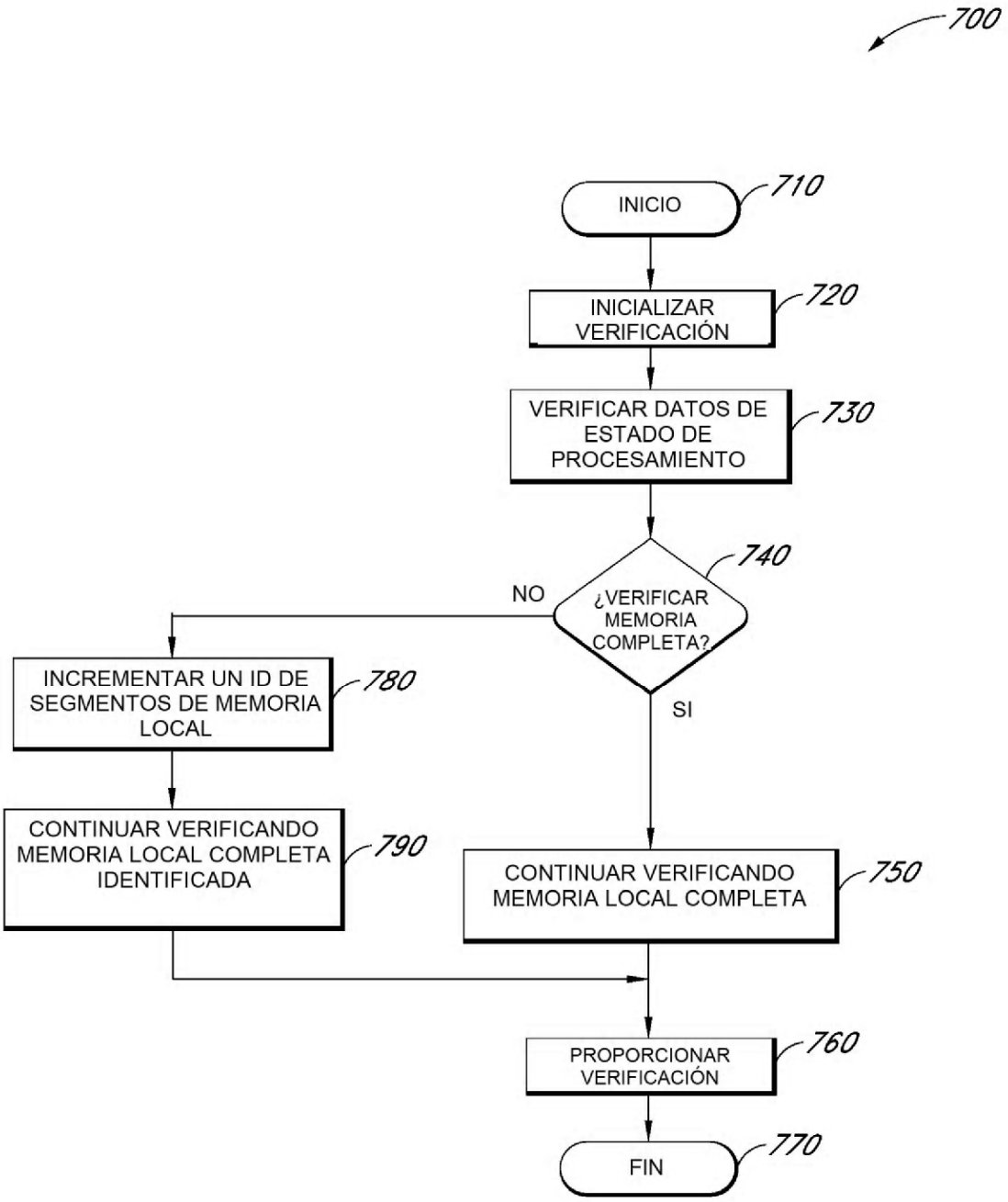
700
INICIO
710
INICIALIZAR VERIFICACIÓN
720
VERIFICAR DATOS DE ESTADO DE PROCESAMIENTO
730
¿VERIFICAR MEMORIA COMPLETA?
740
NO
SI
INCREMENTAR UN ID DE SEGMENTOS DE MEMORIA LOCAL
780
CONTINUAR VERIFICANDO MEMORIA LOCAL COMPLETA IDENTIFICADA
790
CONTINUAR VERIFICANDO MEMORIA LOCAL COMPLETA
750
PROPORCIONAR VERIFICACIÓN
760
FIN
770

FIG. 7

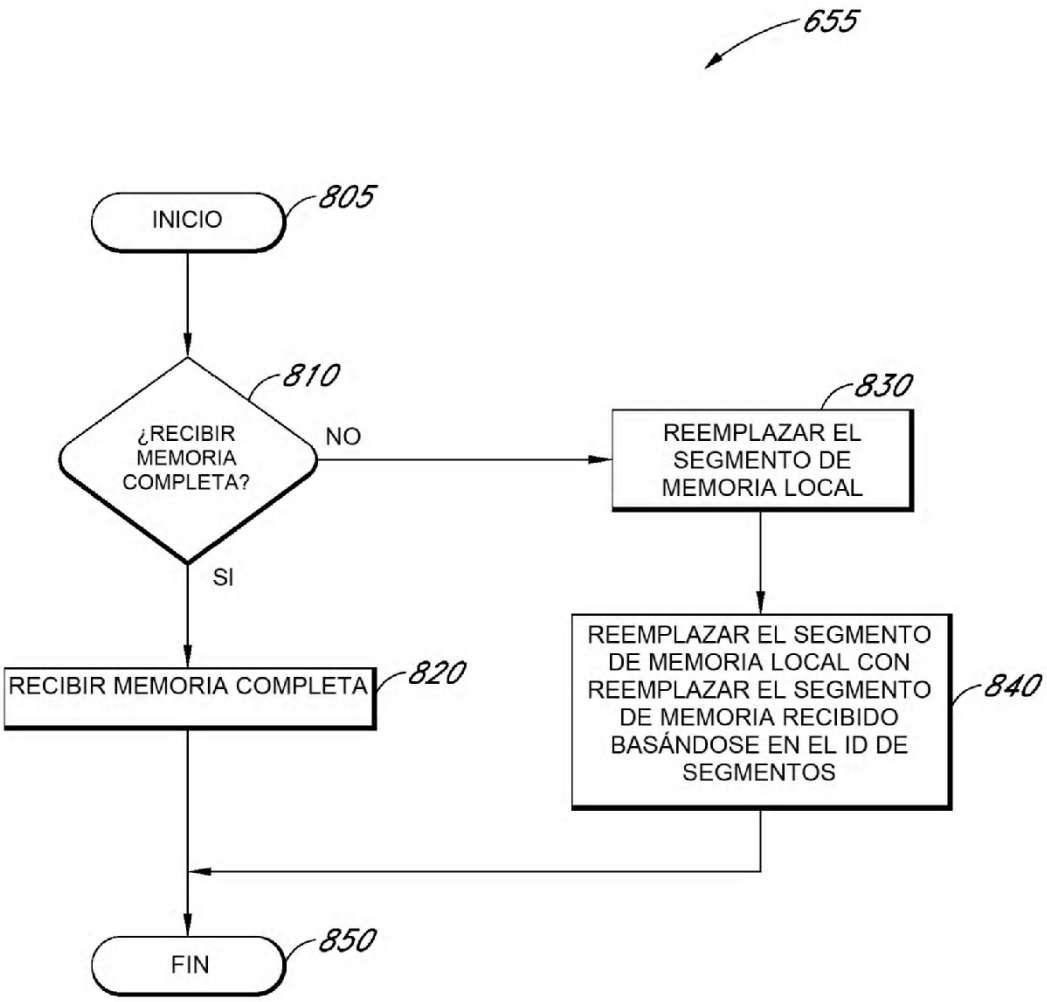
655
INICIO
805
¿RECIBIR MEMORIA COMPLETA?
810
NO
SI
REEMPLAZAR EL SEGMENTO DE MEMORIA LOCAL
830
RECIBIR MEMORIA COMPLETA
820
REEMPLAZAR EL SEGMENTO DE MEMORIA LOCAL CON REEMPLAZAR EL SEGMENTO DE MEMORIA RECIBIDO BASÁNDOSE EN EL ID DE SEGMENTOS
840
FIN
850

FIG. 8

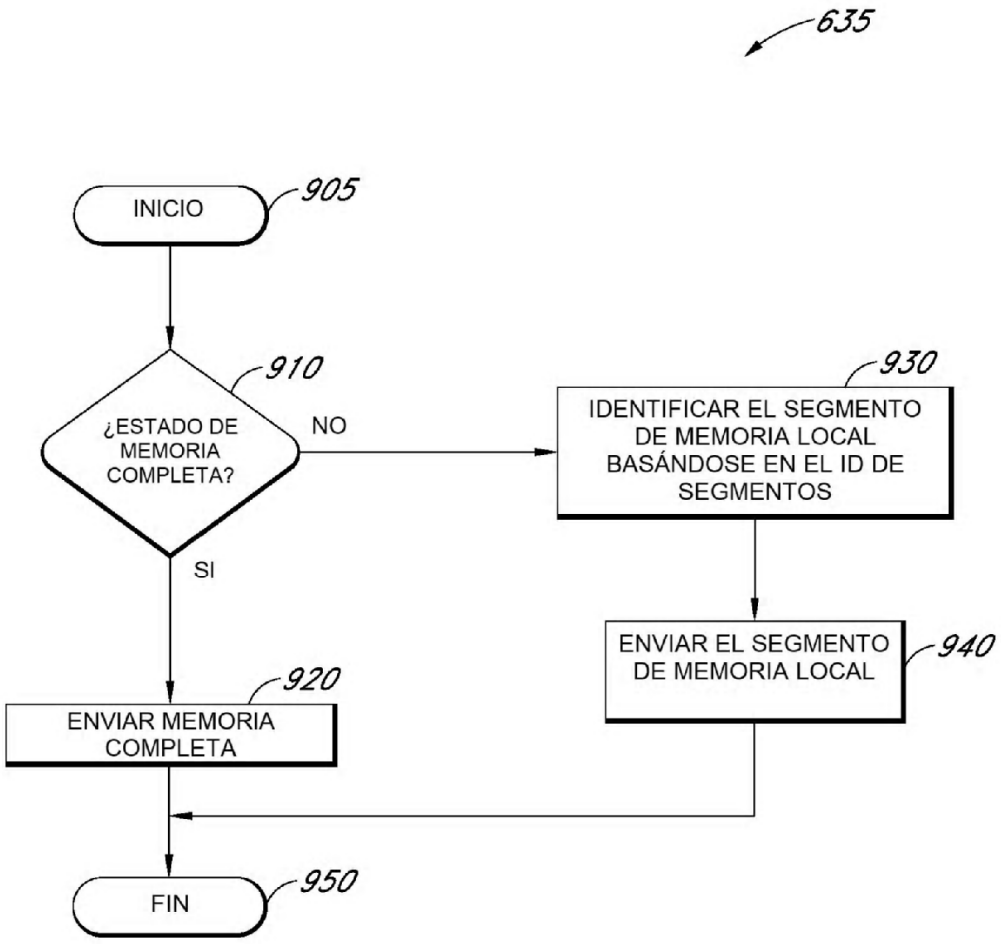
635
INICIO
905
¿ESTADO DE MEMORIA COMPLETA?
910
NO
SI
IDENTIFICAR EL SEGMENTO DE MEMORIA LOCAL BASÁNDOSE EN EL ID DE SEGMENTOS
930
ENVIAR EL SEGMENTO DE MEMORIA LOCAL
940
ENVIAR MEMORIA COMPLETA
920
FIN
950

FIG. 9

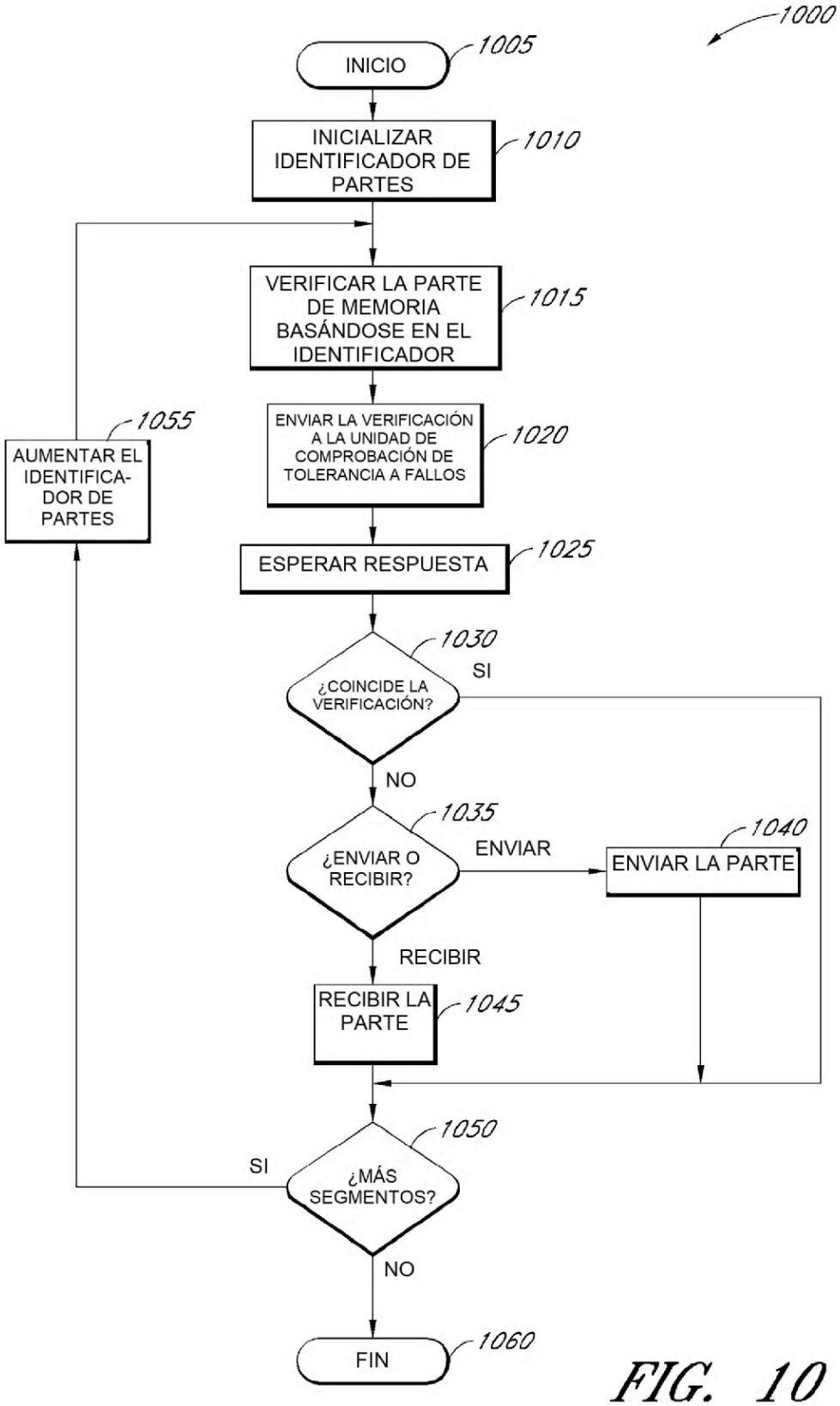
1000
INICIO
1005
INICIALIZAR IDENTIFICADOR DE PARTES
1010
VERIFICAR LA PARTE DE MEMORIA BASÁNDOSE EN EL IDENTIFICADOR
1015
ENVIAR LA VERIFICACIÓN A LA UNIDAD DE COMPROBACIÓN DE TOLERANCIA A FALLOS
1020
ESPERAR RESPUESTA
1025
¿COINCIDE LA VERIFICACIÓN?
1030
SI
NO
¿ENVIAR O RECIBIR?
1035
ENVIAR
ENVIAR LA PARTE
1040
RECIBIR
RECIBIR LA PARTE
1045
¿MÁS SEGMENTOS?
1050
SI
NO
FIN
1060
AUMENTAR EL IDENTIFICA-DOR DE PARTES
1055

FIG. 10