(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 696 510**

(51) Int. CI.:

G06F 8/35 (2008.01)
G06F 8/36 (2008.01)
G06F 9/44 (2008.01)
G05B 19/042 (2006.01)
G05B 17/00 (2006.01)
G06F 9/48 (2006.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Fecha de presentación y número de la solicitud europea: **23.07.2009 E 09166167 (8)**

(97) Fecha y número de publicación de la concesión europea: **29.08.2018 EP 2154606**

(54) Título: **Generación de una configuración ejecutable**

(30) Prioridad:

04.08.2008 AT 42008 U

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
16.01.2019

(73) Titular/es:

AVL LIST GMBH (100.0%)
Hans-List-Platz 1
8020 Graz, AT

(72) Inventor/es:

GRUBER, WERNER;
PEINSIPP, DIETMAR;
PRILLER, PETER y
STIEGLBAUER, GERALD

(74) Agente/Representante:

ELZABURU, S.L.P

DESCRIPCIÓN

Generación de una configuración ejecutable

La invención que nos ocupa se refiere a un procedimiento para generar una configuración ejecutable en un sistema de destino, preferiblemente en un equipo de automatización para desarrollar un vehículo o un componente de un vehículo, para llevar a cabo una tarea de automatización en el sistema de destino, así como a un dispositivo para generar y ejecutar una configuración ejecutable en un sistema de destino, preferiblemente en un equipo de automatización para desarrollar un vehículo o un componente de un vehículo, para llevar a cabo una tarea de automatización en el sistema de destino.

Los bancos de pruebas modernos para desarrollar un vehículo o un componente de un vehículo han de satisfacer una serie de exigencias y han de cubrir un gran número de tareas de automatización, como por ejemplo el mando y la regulación de diversos componentes del banco de pruebas y de diversas pruebas de funcionamiento, la realización de diversas mediciones en la pieza de prueba, la visualización del estado de prueba y de los datos, etc. Sin embargo, el equipo de automatización de los bancos de pruebas actuales es, principalmente por motivos históricos, un sistema central en el que los distintos componentes del banco de pruebas se enlazan mediante diferentes interfaces. De este modo se forma en el equipo de automatización un conjunto central de información y de datos al que pueden incorporarse todos los servicios. Sin embargo, de este modo se forma también un sistema poco flexible con una escalabilidad limitada, dado que para integrar nuevos componentes (SW y HW) se requieren importantes cambios en el equipo de automatización (por ejemplo una nueva interfaz, un procesamiento modificado de los datos del/al componente, una adaptación de la integración en el sistema existente, etc.). Esto hace que los cambios en el equipo de automatización o una ampliación del mismo o la aplicación de deseos especiales del cliente resulten muy costosos. Además, debido a esto han de conservarse necesariamente abundantes configuraciones diferentes, lo que hace que el mantenimiento de los sistemas existentes resulte costoso. Otro problema de los equipos centrales de automatización existentes consiste en la falta de una posibilidad sencilla de integrar en los equipos de automatización una funcionalidad externa, como por ejemplo un determinado regulador de un cliente o una técnica de medición especial. Los sistemas centrales deben además ejecutar también centralmente todas las tareas de cálculo, con lo que se alcanza pronto el límite de potencia de la CPU.

Por el documento US 2007/009394 A1 se conoce un procedimiento para la configuración de un *hardware* programable (FPGA). En éste se interconectan gráficamente unidades de *software* para obtener un modelo gráfico del *software*, que se transforma en una lista de red. Después, la lista de red se compila en un archivo de programa FPGA, que luego se transfiere al *hardware* programable. Así pues, la representación gráfica del *software* se transforma en un circuito de unidades lógicas, como por ejemplo puertas Y, multivibradores biestables (*flip-flop*), etc., resolviéndose la estructura de los componentes del *software*. En el FPGA programado ya no es posible identificar componentes individuales de *software* ni actuar sobre éstos de forma aislada. Por el contrario, un cambio del *software* requiere una nueva creación de la configuración en el FPGA. La configuración también ha de crearse de nuevo cuando cambia el sistema de destino, o sea el *hardware* programable.

Por lo tanto, un objetivo de la invención que nos ocupa es indicar un dispositivo y un procedimiento para generar un programa ejecutable en un sistema de destino, en particular en un equipo de automatización para desarrollar un vehículo o un componente de un vehículo, que por una parte sean flexibles y por otra parte puedan escalarse y adaptarse a las necesidades del explotador del sistema de destino fácilmente y rápidamente. Otro objetivo consiste en configurar el equipo de manera que sea posible integrar fácilmente también funcionalidades externas en el sistema de destino.

Este objetivo se logra mediante el procedimiento definido en las reivindicaciones independientes y mediante el dispositivo correspondiente. Utilizando componentes de *software* abstraídos es posible aplicar de manera fácil y flexible una determinada tarea de automatización. En este contexto, los componentes de *software* pueden crearse según los requisitos, con lo que se posibilita una adaptación fácil al sistema de destino y también una integración fácil de funcionalidades externas. En relación con un sistema central, como por ejemplo en el caso de los equipos de automatización existentes hasta ahora, en el que debido a la disposición rígida de todas las etapas de procesamiento resultaba muy difícil una adaptación al curso real de las señales y por lo tanto un procesamiento adaptado al flujo de señales, en el enfoque según la invención orientado a los componentes puede asegurarse muy fácilmente el procesamiento óptimo adaptado al flujo de señales. Además, el sistema puede adaptarse y escalarse fácilmente en el nivel de componentes, por ejemplo cambiando sólo componentes individuales sin tener que modificar todo el sistema de destino. De este modo se logra un procedimiento sumamente flexible para crear configuraciones ejecutables. Además, los distintos componentes pueden desarrollarse independientemente y por separado, lo que por ejemplo simplifica mucho la ampliación de la funcionalidad. De este modo se hace asimismo posible que un componente pueda estar presente en el sistema de destino en varias versiones de forma paralela, lo que aumenta la flexibilidad. Además, mediante esta estructura es posible una aptitud para una repartición fácil y eficaz entre varias CPU/varios núcleos.

Mediante el tipo de creación según la invención de la configuración ejecutable pueden tomarse ventajosamente en consideración particularidades o especificaciones del sistema de destino ya durante la creación de la configuración, lo que posibilita una configuración o un orden de procesamiento optimizada u optimizado para el sistema de destino

respectivo. Además, puede preverse que automáticamente se proponga o se emplee en la configuración ejecutable un filtro y/o un elemento de exploración, en caso de que se produzcan variaciones de frecuencia en el flujo de señales. Los componentes de *software* pueden procesarse en el sistema de destino al menos parcialmente en distintos hilos (*threads*), pudiendo distintos hilos sincronizarse también con distintas frecuencias. Esto posibilita un procesamiento de alto rendimiento de la configuración en el sistema de destino. Adicionalmente se ejecutan en el sistema de destino ventajosamente procesos genéricos, que ponen a disposición un espacio de proceso, buzones e hilos, en los que los componentes de *software* se cargan con este fin en forma de una librería que puede cargarse de forma dinámica y se instancian cuantas veces se desee. Además, un componente de *software* puede presentar también un disparador *(trigger*), pudiendo la información del disparador encaminarse de manera automática y transparente a lo largo del orden de procesamiento, lo que posibilita la reacción del componente de *software* a determinados sucesos. Puede lograrse una vigilancia del procesamiento si se asigna a un disparador de un componente de *software* un presupuesto de tiempo, y un componente de *software* que rebase este presupuesto de tiempo es suspendido o envía un aviso. También es ventajoso procesar un componente de *software* más de una vez en un ciclo en el sistema de destino, para por ejemplo realizar tareas que no sean necesarias hasta más tarde cuando existan recursos para ello, lo que puede aumentar la velocidad de reacción.

Para ser particularmente flexible, puede estar previsto ventajosamente conmutar o modificar el orden de procesamiento durante el tiempo de ejecución entre dos ciclos completos, preparándose la nueva ejecución y realizándose la conmutación/modificación entonces de forma sincrónica entre el final de un ciclo y el comienzo del siguiente prácticamente sin sobrecarga.

Una modificación de una configuración ejecutable puede llevarse a cabo con una gran facilidad, flexibilidad y rapidez mediante listas de red delta, describiendo una lista de red delta las partes de red que se han de eliminar y las partes de red añadidas y realizándose el cambio en un momento favorable según el orden de procesamiento. Por lo tanto, los componentes de *software* no afectados por esto pueden seguir ejecutándose sin impedimentos en el sistema de tiempo de ejecución. Así se hacen posibles modificaciones parciales en el sistema sin tener que parar el sistema completo.

El rendimiento del procesamiento de la configuración ejecutable puede aumentarse aún más si se ejecutan componentes de *software* en el sistema destino en distintas CPU o en distintos núcleos de una CPU. Mediante la estructura de componentes de *software* finamente granular es posible también realizar una repartición fina en el sistema de destino. Esto posibilita también un procesamiento optimizado de la configuración en el sistema de destino respectivo.

A continuación se describe la invención que nos ocupa por medio de las Figuras esquemáticas 1 a 3, que muestran configuraciones ejemplares y no restrictivas y ventajosas. Muestran:

Fig. 1 un diagrama de bloques de un banco de pruebas con un equipo de automatización,

Fig. 2 un esquema de proceso para generar un código de programa ejecutable según la invención

y

Fig. 3 un ejemplo de un modelo sencillo de una tarea de automatización.

En la Fig. 1 está representado esquemáticamente un banco 100 de pruebas, en el que ha de probarse en cuanto a determinadas funciones una pieza 10 de prueba, por ejemplo un motor, un tren propulsor o todo un vehículo. Con este fin, por regla general la pieza 10 de prueba tiene conectada también una máquina 11 de solicitación para poder generar determinados estados de carga. Tanto la pieza 10 de prueba como la máquina 11 de solicitación son reguladas en este contexto por un equipo 1 de automatización (sistema de destino), que está implementado como *hardware* en el banco de pruebas y en el que se ejecuta el *software* del banco de pruebas, según la tarea de prueba que se haya de llevar a cabo. Con este fin, en el equipo 1 de automatización está prevista una unidad 4 de regulación, que se comunica con la pieza 10 de prueba y la máquina 11 de solicitación mediante una interfaz 2 para un bus 7 de datos. Adicionalmente pueden estar previstas unidades externas 13, como por ejemplo una unidad de adquisición de datos (DAQ) externa por ejemplo para mediciones rápidas (en tiempo real) (indicadas mediante las líneas de trazos) o un aparato de mando externo por ejemplo para una simulación o regulación distribuida, que sean necesarias para una determinada tarea de prueba y que también están conectadas mediante el bus 7 de datos al equipo 1 de automatización o a la pieza 10 de prueba o a la máquina 11 de solicitación. Para las tareas de regulación está previsto por regla general un bus de datos rápido, como por ejemplo EtherCAT, Profibus, CAN-bus, Ethernet Powerlink, etc., con el fin de poder garantizar los tiempos de reacción correspondientes. Aparte del bus de datos rápido puede estar previsto adicionalmente otro bus 8 de datos más lento, como por ejemplo Ethernet, Profibus, CAN-bus, mediante el cual pueden transportarse datos menos críticos en cuanto al tiempo, como por ejemplo datos de medición. Naturalmente, también pueden estar presentes varios buses de datos (rápidos y/o lentos). Con este fin está prevista en el equipo 1 de automatización una interfaz 3, mediante la cual unas unidades 5 de procesamiento correspondientes del equipo 1 de automatización pueden comunicarse con unos componentes 12-1 a 12-n del banco de pruebas, por ejemplo con aparatos de medición (por ejemplo mediciones de gases del cárter (*BlowBy*), consumo de combustible, sensores de agua, aceite, aire o combustible, mediciones de gases de escape,

etc.) o actuadores (por ejemplo palanca de cambio, embrague, pedal acelerador, etc.).

El equipo 1 de automatización de un banco de pruebas puede también estar conectado a una unidad 6 de gestión de bancos de pruebas de orden superior, que por ejemplo controle y administre varios bancos de pruebas de una instalación de pruebas o de una fábrica de pruebas, o se ocupe de tareas de gestión de datos.

En el equipo 1 de automatización se ejecuta el *software* del banco de pruebas, que se compone de distintas partes de programa, por ejemplo programas para la gestión de datos, programas para llevar a cabo determinadas tareas de automatización (por ejemplo la regulación de la máquina de solicitación, el control del desarrollo de procedimientos de medición, la especificación de determinados perfiles de valor de ajuste, la vigilancia de la pieza de prueba, la reacción automática a situaciones excepcionales, etc.), programas para la comunicación con unidades de gestión de bancos de pruebas de orden superior, etc. Estos distintos programas pueden ejecutarse en este contexto de manera ya conocida en función de las exigencias planteadas a un sistema operativo en tiempo real o a un sistema operativo convencional.

Está prevista además una unidad 30 de desarrollo, que está conectada al equipo 1 de automatización, por ejemplo directamente o a través de una red, y con la que pueden generarse y cargarse en el equipo 1 de automatización determinadas configuraciones ejecutables, sobre todo configuraciones para llevar a cabo una tarea de automatización. En este contexto, una configuración ejecutable consiste en esencia en los componentes de *software* y *hardware* implicados, el modo en que éstos cooperan y cómo ha de procesarse tal configuración para llevar a cabo la tarea de automatización. Para ello está instalado en el equipo 1 de automatización también un *software* que puede interpretar y ejecutar tales configuraciones. Sin embargo, la carga de la configuración puede naturalmente realizarse también mediante la unidad 6 de gestión de bancos de pruebas o mediante un bus 7, 8 de datos. Asimismo, la unidad 30 de desarrollo puede ser parte integrante del equipo 1 de automatización.

Para poder crear tal configuración ejecutable está prevista en la unidad 30 de desarrollo, como está representado esquemáticamente en la Fig. 2, una biblioteca (memoria) para componentes 20-1 a 20-n de *software* abstraídos, que aplican distintas funciones del equipo 1 de automatización, por ejemplo un filtro, una interfaz de E/S, una función matemática, una ley de regulación o una funcionalidad externa, etc. “Abstraídos” significa aquí que los componentes 20-1 a 20-n de *software* almacenados aún no tienen ninguna instancia concreta (dependiente del sistema de destino), sino que están descritos sólo por medio de su función general, o sea por ejemplo como interfaz de E/S y no como entrada CAN o salida CAN. Estos componentes 20-1 a 20-n de *software* pueden estar predefinidos o, en caso necesario, pueden crearse nuevos mediante unas interfaces de usuario correspondientes. Cada componente 20-1 a 20-n de *software* está realizado con una interfaz 22 de comandos predefinida e igual y una interfaz 21 predefinida e igual para un canal de datos en el equipo 1 de automatización. Un canal de datos transporta por lo general datos escalares u organizados de manera vectorial en distintos formatos (por ejemplo número entero, coma flotante, etc.), cíclicamente o de forma orientada a los sucesos, de un productor de datos a uno o varios consumidores de datos y puede estar realizado por ejemplo como un bus. Los componentes de *software* no tienen dependencias mutuas directas y pueden por lo tanto someterse a mantenimiento en gran parte independientemente unos de otros. Estos componentes de *software* pueden estar presentes también paralelamente en varias instancias. En un caso ideal, existe una descripción en un lenguaje de modelización abstracto en una instancia independiente del sistema de destino. Asociada a ésta (mapeada) puede existir también ya una descripción específica para un determinado sistema de destino (o sea por ejemplo una determinada unidad 1 de automatización). Ambas pueden estar presentes ya en el sistema de destino, o en un sistema independiente. Entonces, los componentes de *software* realmente ejecutados pueden también existir compilados como ejecutables en el sistema de destino mismo.

Sin embargo, al final todos los componentes de *software* han de estar presentes en el sistema de destino como ejecutables. En este contexto, estos ejecutables pueden estar ya presentes en el sistema de destino o pueden transferirse al sistema de destino en un momento posterior, por ejemplo durante la instalación de nuevos componentes de *software*.

A partir de esta biblioteca 20 se seleccionan, según la tarea de automatización que se haya de aplicar (por ejemplo pruebas de funcionamiento ejecutadas de manera automatizada, en las que se especifica para la pieza de prueba un determinado perfil de requisitos (por ejemplo número de revoluciones/par) de forma controlada por tiempo o por desplazamiento y en las que el comportamiento de la pieza de prueba se determina en valores característicos e instantes significativos, o la terminación reiterada de operaciones de cambio en una caja de cambios para determinar los fenómenos de desgaste, o la determinación de curvas características de máquinas de combustión interna y la optimización automática en relación con criterios de rendimiento o de gases de escape, etc.), mediante una interfaz 26 de usuario adecuada, una serie de componentes 20-a a 20-m de *software* y se interconectan éstos mediante los canales 21 de datos para obtener un modelo 23, aún abstraído, de la tarea de automatización. En la Fig. 3 está representado un modelo 23 sencillo de un regulador. El modelo 23 se compone en este contexto de cuatro componentes 20-a a 20-d de *software*, siendo dos de ellos entradas, realizando un componente de *software* la ley de regulación de un regulador PID con las dos entradas y emitiéndose a través de una salida las magnitudes de salida calculadas. En este contexto, un modelo 23 puede naturalmente, en función de la tarea de automatización que se haya de llevar a cabo, hacerse prácticamente todo lo complejo que se desee.

Por medio de una unidad 27 de transformación de modelo, el modelo 23 se convierte, mediante una transformación

de modelo, en una lista 24 de red consistente en nodos, atributos y bordes, constituyendo los componentes 20-a a 20-m de *software* los nodos, constituyendo los parámetros iniciales de los componentes de *software* los atributos y constituyendo los canales 21 de datos los bordes de la lista 24 de red. Tales transformaciones de modelo son en sí conocidas y por regla general se componen de varias etapas de transformación sucesivas. En este contexto es posible también, mediante una identificación inequívoca del tipo de componente de *software* y de la versión, hacer funcionar paralelamente en un mismo sistema varias versiones del mismo componente de *software*. La lista 24 de red puede estar presente en un lenguaje de descripción estructurado, como por ejemplo XML. Tal lista 24 de red describe, preferiblemente en forma abstraída, la totalidad de los componentes 20-a a 20-m de *software* utilizados, así como su interconexión completa. Mediante esta sencilla representación es posible construir la funcionalidad deseada en el sistema de tiempo de ejecución muy rápidamente y además independientemente del sistema de destino.

La lista 24 de red es una descripción abstracta de la tarea de automatización que se ha de aplicar y puede ser independiente del sistema de destino. Por lo tanto, en caso necesario, la lista 24 de red puede traducirse a configuraciones 25 ejecutables en distintos sistemas de destino. Para generar una configuración 25 ejecutable a partir de la lista 24 de red en una unidad 28 de transformación de sistema de destino, se mapean los componentes de *software* no específicos del sistema de destino a componentes ahora específicos del sistema de destino, es decir que una “entrada” general se convierte ahora por ejemplo en una “entrada CAN” concreta. Con este fin, la información necesaria para ello se halla ya en la unidad 30 de desarrollo, por ejemplo cuando ya se conocen previamente el sistema de destino y sus componentes y recursos, o la unidad 30 de desarrollo se conecta al sistema de destino (por ejemplo con un determinado banco de pruebas) y consulta toda la información necesaria, o sea qué componentes y recursos concretos existen en el sistema de destino, cómo están éstos relacionados, etc. En este contexto son posibles m:n relaciones, es decir que un componente puede dividirse también en varios componentes específicos del sistema de destino o reunirse, buscándose 1:1 relaciones, es decir que un componente independiente del sistema de destino se mapee a un componente dependiente del sistema de destino. Además, la unidad 28 de transformación de sistema de destino separa los recursos según los canales disponibles, según los componentes con sus puertos de entrada y salida, parámetros, así como información de interconexión y orden de procesamiento, y crea a partir de ello la lista de red dependiente del sistema de destino, o sea la configuración ejecutable 25. El diseño estructural de esta lista de red dependiente del sistema de destino está configurado ventajosamente para una instanciación de alto rendimiento y paralelizable.

Además, los componentes 20-a a 20-m de *software* utilizados se instancian en el sistema de destino y se parametrizan mediante sus interfaces 22 de comandos según las especificaciones.

El orden de procesamiento de los componentes 20-a a 20-m de *software* interconectados se crea en la unidad 28 de transformación de sistema de destino a partir de la lista 24 de red, por ejemplo mediante un análisis ya conocido de gráficos de flujo de señales de la teoría de los grafos. Sin embargo, para ello puede por supuesto aplicarse cualquier otro método adecuado, por ejemplo determinar y rastrear de forma recursiva dependencias en la lista 24 de red. El orden de procesamiento para el procesamiento adaptado al flujo de señales de la lista 24 de red puede realizarse en este contexto ya fuera de línea, por ejemplo en la unidad 30 de desarrollo, y también antes de la conversión de la lista 24 de red en una lista de red dependiente del sistema, o ya en el sistema de destino mismo. En este contexto, el procesamiento adaptado al flujo de señales significa que se agrupan secuencias de pasos de cálculo que se han de efectuar unos tras otros, que contienen la instancia en cuestión de un componente que se aplica al cálculo/procesamiento y en caso dado también una información adicional como por ejemplo “fase” (disparador/posdisparador, etc.). Dos secuencias separadas se distinguen ventajosamente por que no tienen dependencias de datos y por lo tanto también pueden calcularse o pasarse independientemente una de otra, lo que constituye una condición previa para la paralelización en sistemas multinúcleo/multi-CPU.

Dado que una lista 24 de red o el modelo 23 puede presentar una estructura del todo compleja, en esta etapa del procesamiento adaptado al flujo de señales pueden tenerse ya ventajosamente algunas cosas en cuenta. Por ejemplo pueden proponerse o emplearse automáticamente filtros o elementos de exploración, en caso de que se produzcan variaciones de frecuencia en el flujo de señales. También es posible un procesamiento con diferentes frecuencias, por ejemplo generando distintos hilos cuya prioridad se fije, por ejemplo, según una *rate monotonic scheduling* (asignación de mayor prioridad a las tareas de menor periodo). Con este fin, puede estar previsto también que los componentes de *software* mismos no tengan ningún hilo, sino sólo el entorno. Es decir que un entorno genérico ponga a disposición un espacio de proceso, una gestión de hilo, una gestión DLL/RSL conocida y uno o varios buzones. En este contexto está disponible un conjunto de hilos, del que se bloquea en cada caso un hilo en un buzón y, en caso de un mensaje a un componente que se halle en el espacio de proceso, se acepta el mensaje y se asigna éste al componente para la ejecución. A continuación el siguiente hilo se “desliza” del conjunto y espera ahora en el buzón. Cada componente de *software* o distintos componentes de *software* puede o pueden presentar también un disparador, pudiendo la información del disparador encaminarse de manera automática y transparente a lo largo del orden de procesamiento (secuencia de disparador), por ejemplo mediante CPU - proceso - componente. En la secuencia de disparador puede darse también conjuntamente un presupuesto de tiempo a cada registro. Si un componente rebasa este presupuesto de tiempo puede suspenderse el mismo, o puede enviarse un aviso. También puede efectuarse una división inteligente en más de una fase, es decir que a un componente de *software* puede tocarle el turno más de una vez en un ciclo (por ejemplo un paso en un sistema cíclico. Ejemplo paso 100 Hz --> un ciclo dura por lo tanto 10 ms). En principio, un componente de *software* hace en un ciclo siempre

sólo aquello que sea directamente e inmediatamente necesario para el siguiente componente de *software* y pasa ya el turno. Mediante la división inteligente, las cosas que sean necesarias más tarde pueden realizarse más tarde, pero aún en el mismo ciclo. Asimismo, el orden de procesamiento puede también conmutarse o modificarse aún en el tiempo de ejecución entre dos ciclos completos prácticamente sin sobrecarga. Con este fin se prepara la ejecución, realizándose la conmutación/modificación entonces de manera sincrónica entre el final de un ciclo y el comienzo del siguiente.

Gracias a la forma uniforme de todos los componentes de *software*, en el momento de la instanciación un algoritmo adecuado puede decidir en qué CPU o en qué núcleo de una CPU del sistema de destino ha de ejecutarse qué componente de *software*. Sin embargo, esta repartición del procesamiento de los componentes de *software* puede también realizarse ya en el sistema de destino mismo. En este contexto, la repartición se realiza ventajosamente por componentes, es decir que no se colocan partes de un componente de *software* en distintos núcleos/distintas CPU, sino que todo el componente de *software* se asigna como una unidad a un núcleo/a una CPU. Mediante la estructura de componentes de *software* finamente granular es posible también realizar una repartición fina. Con este fin existen distintos enfoques en sí conocidos: Cada componente trae sus propias restricciones, es decir que cada componente de *software* tiene sus propias necesidades definidas. Entonces, un solucionador soluciona este problema de limitación-satisfacción. Si hay varias soluciones válidas, una simulación puede descubrir entonces la solución óptima o comprobar si son siquiera factibles. Otro método sería por ejemplo un algoritmo genético, también con una simulación subsiguiente de la repartición y una evaluación de la “aptitud” (*fitness*). Por lo tanto, cada lista 24 de red compleja puede paralelizarse fácilmente y dividirse óptimamente en sistemas multi-CPU o multinúcleo. Esto se ve favorecido también por la naturaleza de los canales digitales como trayectos de comunicación accesibles globalmente de manera equivalente por todas las CPU.

Así pues, mediante listas de red delta también son fácilmente posibles modificaciones de una configuración modelo. Tales listas de red delta describen las partes de red (o los componentes de *software* y su interconexión) que se han de eliminar, seguidas de las partes de red (o los componentes de *software* y su interconexión) que se añaden. El cambio puede realizarse entonces en un momento favorable según el orden de procesamiento, por ejemplo entre dos ciclos. Por lo tanto, los componentes de *software* no afectados por esto pueden seguir ejecutándose sin impedimentos en el sistema de tiempo de ejecución. Así se hacen posibles modificaciones parciales en el sistema sin tener que parar el sistema completo.

REIVINDICACIONES

1. Procedimiento para generar una configuración (25) ejecutable en un sistema de destino, preferiblemente en un equipo (1) de automatización para desarrollar un vehículo o un componente de un vehículo, para llevar a cabo una tarea de automatización en el sistema de destino, en donde

- a partir de una biblioteca (20) se eligen una serie de componentes (20-a, 20-b … 20-m) de *software* abstraídos, independientes del sistema de destino, que aplican distintas funciones del sistema de destino y presentan una interfaz (22) de comandos predefinida y una interfaz (21) predefinida para canales de datos,

- los componentes (20-a, 20-b … 20-m) de *software* independientes del sistema de destino elegidos se interconectan mediante sus canales de datos para formar un modelo (23) con el fin de realizar la tarea de automatización,

- el modelo (23) se transforma automáticamente, mediante una transformación de modelo, en una lista (24) de red consistente en nodos, atributos y bordes, constituyendo los componentes (20-a, 20-b … 20-m) de *software* los nodos, constituyendo los parámetros iniciales los atributos y constituyendo los canales de datos los bordes de la lista de red, **caracterizado por que**

- los componentes (20-a, 20-b … 20-m) de *software* de la lista (24) de red independientes del sistema de destino se transforman en componentes de *software* de la configuración ejecutable (25) dependientes del sistema de destino mapeando a componentes de *software* dependientes del sistema de destino los componentes (20-a, 20-b … 20-m) de *software* independientes del sistema de destino en forma de relaciones m:n, preferiblemente relaciones 1:1, teniendo en cuenta componentes y recursos concretos presentes en el sistema de destino,

- los componentes de *software* de la configuración ejecutable (25) dependientes del sistema de destino se instancian en el sistema de destino y se parametrizan mediante su interfaz de comandos para llevar a cabo la tarea de automatización y

- a partir de la lista (24) de red se determina en el sistema de destino el orden de procesamiento de los componentes de *software* interconectados para un procesamiento adaptado al flujo de señales.

2. Procedimiento según la reivindicación 1, **caracterizado por que** automáticamente se propone o se emplea en la configuración ejecutable (25) un filtro y/o un elemento de exploración, en caso de que se produzcan variaciones de frecuencia en el flujo de señales.

3. Procedimiento según la reivindicación 1 o 2, **caracterizado por que** los componentes de *software* se procesan en el sistema de destino al menos parcialmente en distintos hilos (*threads*).

4. Procedimiento según la reivindicación 3, **caracterizado por que** se sincronizan distintos hilos al menos parcialmente con distintas frecuencias.

5. Procedimiento según la reivindicación 3 o 4, **caracterizado por que** en el sistema de destino se ejecutan procesos genéricos, que ponen a disposición un espacio de proceso, buzones e hilos, en los que los componentes de *software* se cargan con este fin en forma de una librería que puede cargarse de forma dinámica y se instancian cuantas veces se desee.

6. Procedimiento según la reivindicación 3, **caracterizado por que** un componente de *software* presenta un disparador *(trigger*), encaminándose la información del disparador de manera automática y transparente a lo largo del orden de procesamiento.

7. Procedimiento según la reivindicación 6, **caracterizado por que** a un disparador de un componente de *software* se le asigna un presupuesto de tiempo, y un componente de *software* que rebase este presupuesto de tiempo es suspendido o envía un aviso.

8. Procedimiento según una de las reivindicaciones 1 a 7, **caracterizado por que** un componente de *software* se procesa más de una vez en un ciclo en el sistema de destino.

9. Procedimiento según una de las reivindicaciones 1 a 8, **caracterizado por que** el orden de procesamiento se conmuta o se modifica durante el tiempo de ejecución entre dos ciclos completos preparando la nueva ejecución y realizando la conmutación/modificación entonces de forma sincrónica entre el final de un ciclo y el comienzo del siguiente.

10. Procedimiento según una de las reivindicaciones 1 a 9, **caracterizado por que** se llevan a cabo modificaciones de una configuración ejecutable (25) mediante listas de red delta, describiendo una lista de red delta las partes de red que se han de eliminar y las partes de red añadidas y realizándose el cambio en un momento favorable según el orden de procesamiento.

11. Procedimiento según una de las reivindicaciones 1 a 10, **caracterizado por que** se ejecutan componentes de *software* en el sistema de destino en distintas CPU o distintos núcleos de una CPU.

12. Dispositivo para generar y ejecutar una configuración ejecutable en un sistema de destino, preferiblemente en un equipo (1) de automatización para desarrollar un vehículo o un componente de un vehículo, para llevar a cabo una tarea de automatización en el sistema de destino, en donde está prevista una unidad (30) de desarrollo que está conectada al sistema de destino y en la que

- está prevista una biblioteca (20) a partir de la cual pueden seleccionarse, mediante una interfaz (26) de usuario, un serie de componentes de *software* abstraídos, independientes del sistema de destino, que aplican distintas funciones del sistema de destino y que presentan una interfaz (22) de comandos predefinida y una interfaz (21) predefinida para canales de datos, y los componentes (20-a, 20-b, … 20-m) de *software* independientes del sistema de destino seleccionados pueden interconectarse mediante sus canales de datos para formar un modelo (23) con el fin de realizar la tarea de automatización,

- está prevista una unidad (27) de transformación de modelo (23) que transforma automáticamente el modelo, mediante una transformación de modelo, en una lista (24) de red consistente en nodos, atributos y bordes, constituyendo los componentes de *software* los nodos, constituyendo los parámetros iniciales los atributos y constituyendo los canales de datos los bordes de la lista (24) de red, **caracterizado por que**

- está prevista una unidad (28) de transformación de sistema de destino que transforma los componentes (20-a, 20-b, … 20-m) de *software* de la lista (24) de red independientes del sistema de destino en componentes de *software* de la configuración ejecutable (25) dependientes del sistema de destino, por el método de que la unidad (28) de transformación de sistema de destino mapea a componentes de *software* dependientes del sistema de destino los componentes (20-a, 20-b … 20-m) de *software* independientes del sistema de destino en forma de relaciones m:n, preferiblemente relaciones 1:1, teniendo en cuenta componentes y recursos concretos presentes en el sistema de destino, instancia los componentes de *software* de la configuración ejecutable (25) dependientes del sistema de destino en el sistema de destino para llevar a cabo la tarea de automatización y los parametriza mediante su interfaz de comandos y determina a partir de la lista (24) de red en el sistema de destino el orden de procesamiento de los componentes de *software* interconectados para un procesamiento adaptado al flujo de señales.

13. Dispositivo según la reivindicación 12, **caracterizado por que** el sistema de destino está realizado como un sistema multi-CPU o multinúcleo y en el sistema de destino se ejecutan componentes de *software* en distintas CPU o distintos núcleos de una CPU.

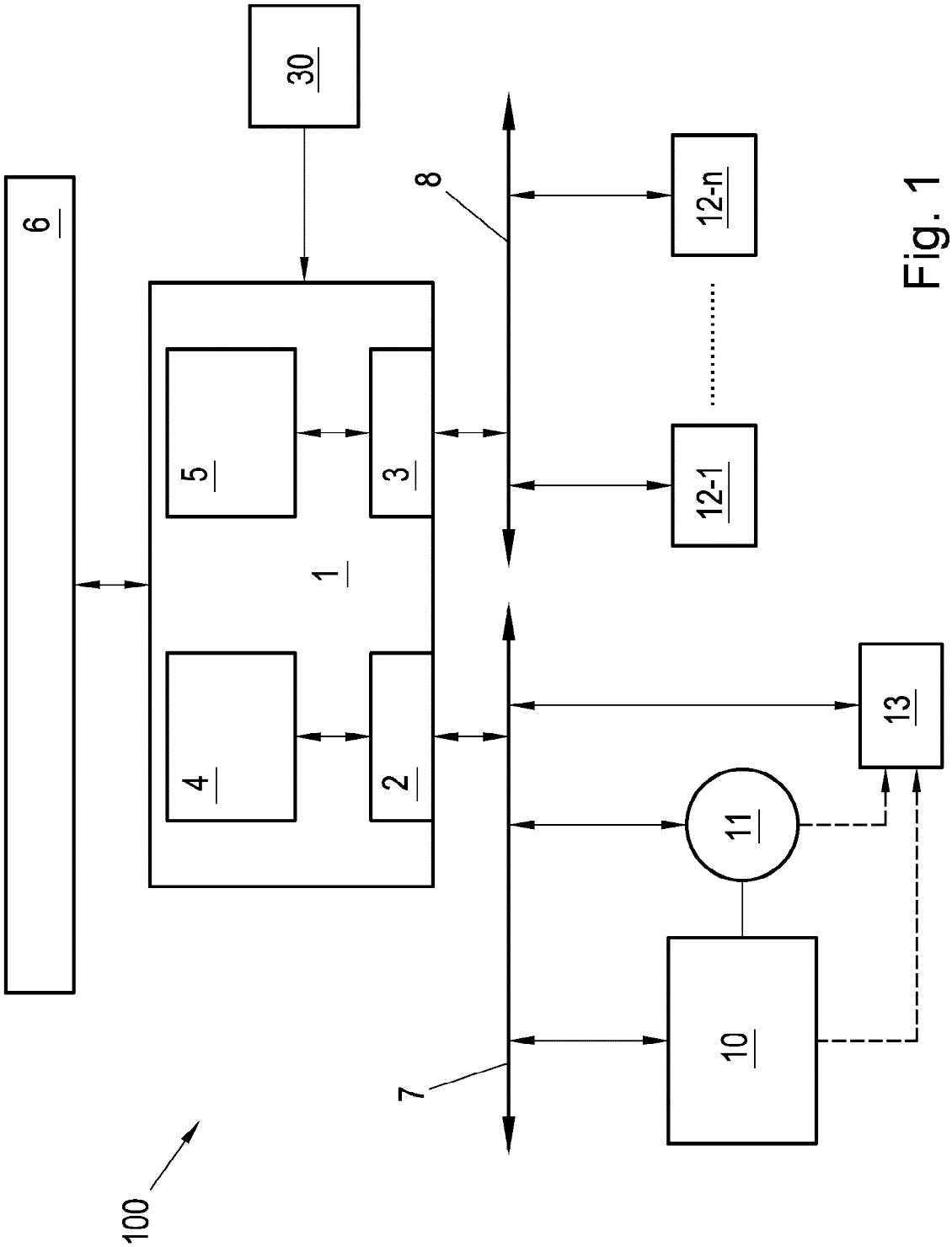

Fig. 1

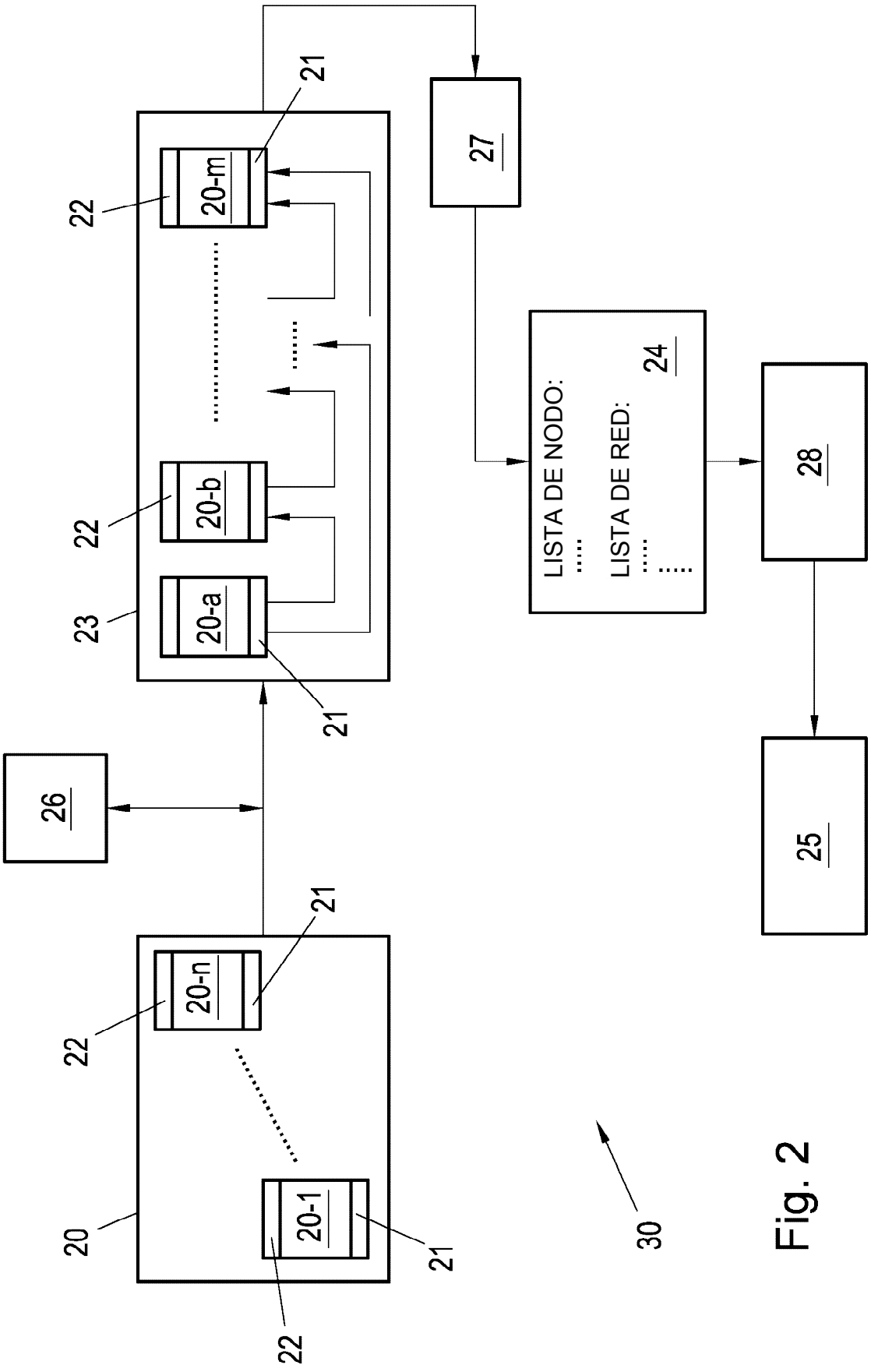
20
20-1
20-n
20-a
20-b
20-m
21
22
23
24
25
26
27
28
30
LISTA DE NODO:
LISTA DE RED:

Fig. 2

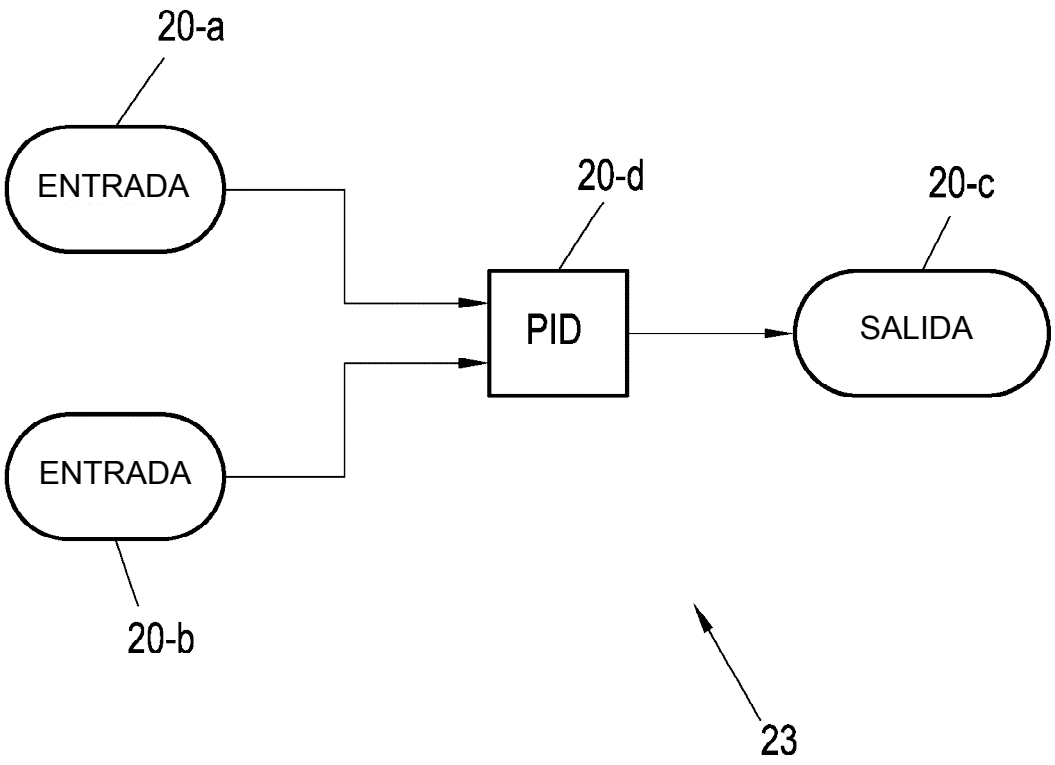

Fig. 3