(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 374 483**

(51) Int. CI.:
G06F 11/26 (2006.01)
G01R 31/3183 (2006.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Número de solicitud europea: **08858340 .6**
(96) Fecha de presentación: **25.11.2008**
(97) Número de publicación de la solicitud: **2232373**
(97) Fecha de publicación de la solicitud: **29.09.2010**

(54) Título: **PROCEDIMIENTO Y APARATO PARA PROBAR UN SISTEMA EN CHIP QUE IMPLICA ACCESOS EN PARALELO Y EN SERIE.**

(30) Prioridad:
04.12.2007 US 950138

(45) Fecha de publicación de la mención BOPI:
17.02.2012

(45) Fecha de la publicación del folleto de la patente:
17.02.2012

(73) Titular/es:
Alcatel Lucent
3 avenue Octave Gréard
75007 Paris, FR

(72) Inventor/es:
CHAKRABORTY, Tapan, Jyoti;
CHIANG, Chen-Huan;
GOYAL, Suresh;
PORTOLAN, Michele y
VAN TREUREN, Bradford, Gene

(74) Agente: **Carpintero López, Mario**

DESCRIPCIÓN

Procedimiento y aparato para probar un sistema en chip que implica accesos en paralelo y en serie

Campo de la invención

La invención se refiere al campo de las placas de circuitos impresos y, más específicamente, a la prueba de placas de circuitos impresos

Antecedentes de la invención

El grupo de acción común de prueba (JTAG) se refiere a la norma IEEE 1149 para someter a prueba puertos de acceso para probar placas de circuitos impresos usando la exploración de límites. El JTAG es usado por herramientas de generación automatizada de prueba (ATG) para probar placas de circuitos impresos. El lenguaje de descripción de exploración de límites (BSDL) se ha desarrollado como parte de la norma IEEE 1149.1 para el JTAG al nivel de la placa y asimismo, se ha desarrollado el Lenguaje de Descripción Jerárquica de Exploración (HSDL) como una extensión de BSDL. BSDUHSDL describe recursos disponibles sobre una placa o un componente de una placa (donde HSDL describe componentes compuestos por otros componentes). Mientras que BSDUHSDL es eficiente para el JTAG al nivel de placa, el paso del JTAG al nivel de la placa al JTAG al nivel del chip pone de manifiesto las limitaciones de BSDUHSDL.

JTAG instrucción (IJTAG) se está normalizando (denominado como la norma P1687) para solucionar las limitaciones de JTAG existentes asociadas al cambio de JPAG al nivel de la placa al JTAG al nivel del chip; sin embargo, el trabajo en curso asociado al IJTAG ha revelado que BSDUHSDL es incapaz de satisfacer los requisitos de descripción para la prueba de JTAG al nivel del chip. BSDUHSDL se basa en una lista ordenada de células que compone el registro de exploración de límites, sin embargo, tal descripción estática no es apropiada para describir cadenas complejas de exploración dinámica necesarias en IJTAG. Asimismo, BSDUHSDL falla en la provisión de cualquier espacio para describir procedimientos de prueba necesarios para cada componente del sistema.

El documento WO 2005/078465 A1 divulga la prueba de núcleos funcionales o IP que forman parte de un sistema en chip, SOC. Los medios de prueba comprenden una capa en la cual se empotra el núcleo.

Sumario de la invención

La presente invención proporciona un nuevo lenguaje de descripción de hardware para la prueba de JTAG al nivel del chip. Este nuevo lenguaje de descripción de hardware, denominado como Nuevo BSDL (NSDL), permite probar recursos de un sistema en chip a describir, permitiendo de este modo que el sistema en chip sea descrito para facilitar la prueba del sistema en chip. La presente invención proporciona un enfoque de abajo hacia arriba para describir un sistema en chip. La presente invención soporta descripciones algorítmicas de cada uno de los componentes del sistema en chip, y soporta una descripción algorítmica de interconexiones entre los componentes del sistema en chip, permitiendo de este modo la generación de una descripción algorítmica de todo el sistema en chip o partes del sistema en chip.

En una realización, se proporciona un procedimiento para probar un sistema en chip como se define en la reivindicación 1.

Se proporcionan mejoras adicionales mediante las reivindicaciones dependientes 2-8.

En una realización; se proporciona un aparato como se define en la reivindicación 9.

En una realización; se proporciona un soporte de almacenamiento legible por ordenador como se define en la reivindicación 10.

Breve descripción de los dibujos

Las enseñanzas de la presente invención se pueden entender fácilmente considerando la siguiente descripción detallada en combinación con los dibujos anexos, en los cuales:

La figura 1 ilustra un diagrama de bloques de alto nivel de un entorno de prueba;

La figura 2 ilustra un diagrama de bloques de alto nivel del sistema en chip del entorno de prueba de la figura 1.

La figura 3 ilustra un conocimiento de entrada-salida de un componente “sin acceso”;

La figura 4 ilustra un conocimiento de entrada-salida de un componente de “acceso limitado” o “pleno acceso”.

La figura 5 ilustra una referencia explícita de rodajas de una trayectoria interna de exploración de un componente;

La figura 6 ilustra un diagrama de bloques de alto nivel de una representación de un dispositivo de cruce;

La figura 7 ilustra un diagrama de bloque de alto nivel del uso de un dispositivo genérico de cruce para modificar dinámicamente la trayectoria de exploración de un sistema en chip.

La figura 8 ilustra un diagrama de bloques de alto nivel de un dispositivo de cruce que se puede describir usando NSDL;

La figura 9 ilustra un diagrama de bloques de alto nivel de un dispositivo de cruce que se puede describir usando NSDL;

La figura 10 ilustra un diagrama de bloques de alto nivel de un dispositivo de cruce que se puede describir usando NSDL;

La figura 11 ilustra un diagrama de bloques de alto nivel del sistema de prueba del entorno de prueba de la figura 1;

La figura 12 ilustra un procedimiento ejemplar ejecutado por el sistema de prueba de la figura 1 para probar un sistema mediante una conexión JTAG;

La figura 13 ilustra un procedimiento ejemplar ejecutado por el sistema de prueba de la figura 1 para probar un sistema mediante una conexión JTAG.

La figura 14 ilustra un procedimiento ejemplar ejecutado por el sistema de prueba de la figura 1 para probar un sistema a través de una conexión JTAG.

La figura 15 ilustra el uso de una descripción de uno de los componentes del sistema en chip de la figura 2 para determinar los valores de los bits del registro para un procedimiento de prueba para probar ese componente.

La figura 16 ilustra el uso de una descripción de la composición del sistema en chip de la figura 2 para determinar los flujos de bits para un procedimiento de prueba para probar uno de los componentes del sistema en chip de la figura 2;

La figura 17 ilustra un procedimiento ejemplar ejecutado por el sistema de prueba de la figura 1 para probar un componente de un sistema en una estructura IJTAG/NSDL;

La figura 18 ilustra un diagrama de bloques de alto nivel de un sistema en chip ejemplar;

La figura 19 ilustra el uso de una descripción de uno de los componentes del sistema en chip de la figura 18 para determinar los valores de los bits del registro para un procedimiento de prueba para probar ese componente.

La figura 20 ilustra el uso de descripciones de los componentes del sistema en chip de la figura 18 para determinar una descripción de la composición del sistema en chip de la figura 18;

La figura 21 ilustra un diagrama de bloques de alto nivel de un esquema general de conexión de una interfaz de acceso en paralelo;

La figura 22 ilustra un diagrama de bloques de alto nivel que ilustra dos esquemas ejemplares de conexión de acceso en paralelo;

La figura 23A ilustra un diagrama de bloques de alto nivel de un entorno ejemplar de prueba;

La figura 23B ilustra un diagrama de bloques de alto nivel de flujo de datos dentro del entorno ejemplar de prueba de la figura 23A.

La figura 24 ilustra un diagrama de bloques de alto nivel de una conexión ejemplar de una conexión ejemplar entre un puerto paralelo y el núcleo de un sistema en chip;

La figura 25 ilustra un diagrama de bloques de alto nivel de una conexión ejemplar entre un puerto paralelo y el núcleo de un sistema en chip;

La figura 26 ilustra un diagrama de bloques de alto nivel de una conexión ejemplar de una conexión ejemplar entre un puerto paralelo y el núcleo de un sistema en chip;

La figura 27 ilustra un diagrama de bloques de alto nivel de una conexión ejemplar de una conexión ejemplar entre un puerto paralelo y el núcleo de un sistema en chip;

La figura 28 ilustra un diagrama de bloques de alto nivel de un esquema interno de conexión de una interfaz de acceso en paralelo;

La figura 29 ilustra un procedimiento para describir los recursos de prueba de un sistema en chip; y

La figura 30 ilustra un diagrama de bloques de alto nivel de un ordenador estándar apropiado para su uso en la ejecución de las funciones descritas en el presente documento.

Para facilitar la comprensión, se han usado números de referencia idénticos, allí donde es posible, para designar elementos idénticos que son comunes a las figuras.

Descripción detallada de la invención

Como se describe en la presente invención, se ha normalizado la Instrucción JTAG (IJTAG) (denominado como la norma P1687, o alternativamente, IJTAG) para resolver las limitaciones JTAG existentes asociadas al cambio de la prueba de JTAG al nivel de la placa a la prueba de JTAG al nivel del chip; sin embargo, el trabajo en curso asociado a IJTAG ha revelado que BSDUHSDL es incapaz de satisfacer los requisitos de descripción para la exploración de límites al nivel del chip. La presente invención proporciona un nuevo lenguaje de descripción de hardware que resuelve las limitaciones de BSDUHSDL para la prueba de JTAG al nivel del chip. Este nuevo lenguaje de descripción de hardware se denomina en el presente documento Nuevo BSDL (NSDL). El lenguaje NSDL permite la descripción de recursos de prueba de un sistema en chip, facilitando de este modo la prueba del sistema en chip. Las numerosas ventajas del lenguaje de descripción de NSDL se pueden ver en la siguiente descripción de NSDL.

Como se describe en el presente documento, el nuevo lenguaje de descripción de hardware, NSDL, también permite avances adicionales en la prueba basado en el chip. El lenguaje de descripción NSDL permite el uso de denominados “dispositivos de uso” para facilitar la prueba de dispositivos de sistema en chip. Un dispositivo de curce permite la modificación dinámica de la trayectoria de exploración de sistema del sistema en chip. El lenguaje de descripción NSDL también permite el uso de acceso en paralelo para facilitar la prueba de dispositivos de sistema en chip. El acceso en paralelo los dispositivos de sistema en chip se puede proporcionar de muchas maneras. Asimismo, cabe señalar que, aunque tales avances han sido permitidos por el lenguaje de descripción NSDL, tales avances se pueden utilizar también en combinación con otros lenguajes de descripción que se puedan desarrollar en el futuro.

En el proceso de desarrollo de hardware, hay tres actores principales: proveedores de dispositivos, arquitectos de sistemas e ingenieros de prueba. Un proveedor de dispositivos produce dispositivos específicos. Un arquitecto de sistemas usa los dispositivos proporcionados por el proveedor de dispositivos para componer un sistema. Un ingeniero de prueba prueba el sistema para garantizar que el sistema funciona adecuadamente (por ejemplo, probando interconexiones entre dispositivos del sistema, funciones de los dispositivos, funciones del sistema, y similar). El lenguaje NSDL puede ser utilizado por el proveedor de dispositivos (por ejemplo, para describir sus dispositivos), el arquitecto de sistemas (por ejemplo en la composición del sistema), y por el probador de sistemas (por ejemplo probando el sistema). De este modo, se espera que el lenguaje NSDL se use a lo largo de todo el proceso de desarrollo de hardware.

En el proceso de desarrollo de sistema en chip, los dispositivos de que se compone el sistema pueden ser dispositivos de software, es decir, una descripción del dispositivo en algún lenguaje de descripción de hardware. En este proceso, el arquitecto de sistemas integra los dispositivos de software con el código de nivel del sistema en un flujo de desarrollo del nivel de sistema para obtener el sistema en chip que finalmente se prueba por el ingeniero de prueba. A medida que complejidad del sistema en chip aumenta (por ejemplo en términos de número de dispositivos, interconexiones entre dispositivos, dependencias intradispositivos, y similar), aumenta la complejidad de la prueba del sistema en chip. El lenguaje NSDL permite que los dispositivos de sistema en chip de cualquier complejidad sea fácil de describir y por lo tanto de probar.

La figura 1 ilustra un diagrama de bloques de alto nivel de un entorno de prueba. Específicamente, el entorno de prueba 100 incluye un sistema en chip (S-o-C) 110 y un sistema de prueba (ST) 120. El ST 120 prueba S-o-C 110 (por ejemplo componentes individuales de prueba de S-o-C 110 (que incluye funciones de componentes), interconexiones entre dispositivos en S-o-C 110, funciones de nivel de sistema de S-o-C 110, y similar, así como varias combinaciones de los mismos.) El ST 120 prueba el S-o-C 110 mediante un Puerto de acceso de prueba (TAP) 115, que incluye un puerto de entrada 115_I Denominado puerto TDI) y un puerto de salida 115_O (denominado puerto TDO).

En una realización, en un entorno P1687, el TAP 115 se describe mediante la norma IEEE 1149.1. Aunque principalmente se ilustra y describe en el presente documento usando el puerto TDI 115, y el puerto TDO 115O el TAP 115 puede incluir otros puertos de control tales como un puerto TCK, un puerto TMS, y opcionalmente, un puerto TRST (que se ha omitido por motivos de claridad). Asimismo, aunque principalmente se ilustra y describe respecto de un TAP descrito por la norma IEEE 1149.1, el TAP 115 puede utilizar varios puertos (por ejemplo, puerto descritos por otras normas y similar, así como las varias combinaciones de los mismos).

El ST 120 lleva a cabo la prueba sobre S-o-C 110 usando procedimientos de prueba. El ST 120 puede llevar a cabo una o más pruebas usando uno o más procedimientos de prueba. Se puede usar un procedimiento de prueba para

probar una parte de un componente (por ejemplo, una función de un componente, un conjunto de funciones de un componente dependencias intracomponentes, y similar), un componente, un grupo de componentes(por ejemplo, interconexiones entre componentes, dependencias intercomponentes, y similar), una o más funciones del nivel de sistema, y similar, así como las varias combinaciones de los mismos).

El ST 120 genera un procedimiento de prueba para probar el S-o-C 110. El procedimiento de prueba especifica la información necesaria para probar el S-o-C 110. Un procedimiento de prueba para el S-o-C 110 puede especificar una descripción del S-o-C 110 (incluyendo descripciones de cada uno de los componentes individuales del S-o-C 110, así como una descripción del nivel de sistema del S-o-C 110). Un procedimiento de prueba puede especificar un vector de entrada de prueba y un vector de salida de prueba esperado. Un procedimiento de prueba puede incluir otra información asociada a una prueba, tal como un tiempo estimado necesario para la prueba, el procesamiento de datos de salida para la prueba (por ejemplo, acciones de registro, error de activación, recuperación, etc.), y similar, así como varias combinaciones de los mismos.

El ST 120 genera el procedimiento de prueba para probar el S-o-C 110 usando una descripción del S-o-C 110 (incluyendo descripciones de cada uno de los componentes individuales del S-o-C 110, así como una descripción del nivel del sistema del S-o-C 110). Las descripciones de los componentes individuales del S-o-C 110 se pueden especificar usando NSDL. La descripción de un componente individual puede describir una trayectoria de exploración interna del componente. La descripción del nivel de sistema del S-o-C 110 se puede especificar usando NSDL. Las descripciones del nivel de sistema del S-o-C 110 pueden describir una topología del S-o-C 110 (por ejemplo interconexiones entre componentes, dependencias intercomponentes, y similar).

La información de descripción para S-o-C 110 (incluyendo descripciones de componentes individuales, la descripción del nivel de sistema, y similar) incluye información adaptada para su uso en la generación de procedimientos de prueba sobre S-o-C 110. Por ejemplo, la información de descripción incluye información de trayectoria de exploración de componentes, información de topología del sistema, y similar, que se puede procesar para determinar la información de la longitud de la trayectoria de exploración, información de jerarquía de trayectoria de exploración y similar, así como varias combinaciones de las mismas. La información de descripción para S-o-C 110 puede incluir cualquier información de descripción descrita en el presente documento.

El ST 120 prueba el S-o-C 110 ejecutando uno o más procedimientos de prueba sobre S-o-C 110. El ST 120 genera los flujos de bits de entrada y los resultados de prueba esperados (por ejemplo los valores de bits de salida o flujos de bits de salida esperados) para cada prueba a realizar. El ST 120 proporciona los flujos de bits de entrada (denominados vectores de entrada de prueba) al puerto TDI 115 y recibe los correspondientes flujos de bits de salida (denominados vectores de salida de prueba) desde el puerto TDO 115O. El ST 120 compara los flujos de bits de salida efectivos con los flujos de bits de salida esperados con el fin de determinar los resultados de la prueba. El ST 120 puede almacenar los resultados de la prueba.

El ST 120 puede ejecutar uno o más procedimientos de prueba para probar el S-o-C 110. El ST 120 puede organizar la ejecución de múltiples procedimientos de prueba para de este modo minimizar un tiempo total de prueba (ya que diferentes decisiones de programación darán como resultado diferentes tiempos de realización de la prueba para el mismo conjunto de procedimientos de prueba). El ST 120 puede especificar una programación de prueba (es decir, una programación que especifica un orden según el cual se deben ejecutar los procedimientos de prueba). El ST 120 puede llevar a cabo otras varias funciones asociadas a la prueba de un sistema en chip.

La figura 2 ilustra un diagrama de bloques de alto nivel del sistema en chip del entorno de prueba de la figura 1. Como se ilustra en la figura 2, el S-o-C 110 incluye una pluralidad de componentes 210_A – 210_E (colectivamente, componentes 210) que se interconectan con una pluralidad de interconexiones 220 de componentes (colectivamente, interconexiones de componentes 220).

Los componentes 210 pueden incluir cualesquiera componentes que puedan incluirse en un sistema de S-o-C. Los componentes 210 se pueden describir usando NSDL. Los componentes 210 se pueden denominar también en el presente documento como recursos de prueba.

En una realización, en una sistema según la norma P1687, los componentes 210 pueden incluir IPs, instrumentos y/o seleccionar bits de instrumento (SIBs).

Un dispositivo de propiedad intelectual (IP) es un dispositivo normal que requiere una prueba.

Un instrumento es una dispositivo que, aparte de necesitar una prueba, ofrece una funcionalidad adaptada para ayudar a probar (por ejemplo leer valor, monitorizar valores, proporcionar información útil y similar, así como varias combinaciones de los mismos). Por ejemplo, un instrumento puede ser una salida de un sensor de temperatura que se ha de usar para parametrizar la prueba de aceleración de la vida útil. Por ejemplo, un instrumento puede ser el valor de referencia de un sensor usado para calibrar un filtro sintonizable para la etapa de adquisición de radiofrecuencia definida por software. Dicho de otro modo, los instrumentos pueden ayudar a realizar pruebas tanto

durante la prueba del sistema inicial, como a lo largo de toda la vida útil del sistema.

Ya que lo IPs/instrumentos pueden ser bastante similares, se pueden usar los dos términos de manera intercambiable en el presente documento. Asimismo, puesto que los IPs y los instrumentos se pueden usar como componentes de un sistema en chip, los IPs y los instrumentos se pueden denominar más generalmente en el presente documento como componentes.

Un IP/instrumento puede incluir una trayectoria de exploración jerárquica. Un componente con una trayectoria de exploración jerárquica incluye una trayectoria de exploración interna que entra a formar parte de la trayectoria de exploración del sistema cuando el componente se introduce dentro del sistema. Un SIB es una célula de trayectoria de exploración jerárquica que permite que partes de la trayectoria de exploración se incluyan o eliminen dinámicamente en o de la trayectoria de exploración (dependiendo de que dispositivo o dispositivos se usarán en la prueba). La SIB forma parte de la Propuesta de Hardware del actual borrador de la norma P1687.

En general, la jerarquía mejora la prueba de componentes de un sistema en chip. Por ejemplo la jerarquía permite minimizar la trayectoria activa de exploración del sistema y el aislamiento de los componentes durante la prueba, reduciendo de este modo el tiempo de acceso a los componentes de un sistema en chip.

En otras realizaciones, en los sistemas según otras normas, los componentes 210 pueden incluir otros tipos de componentes.

En otra realización, los componentes 210 pueden incluir uno o más dispositivos de cruce. Se permite el uso de dispositivos de cruce en la prueba de un sistema en chip usando el lenguaje NSDL (es decir, la mayoría de tales dispositivos no son capaces de ser descritos por BSDUHSDL). El uso de dispositivos de cruce en la prueba de sistema en chip se puede entender mejor respecto de las figuras 6 – 10.

Como se ilustra en la figura 2, cada uno de los componentes 210 incluye una pluralidad de registros internos. Especialmente, el componente 210_A incluye tres registros (A_0, A_1, A_2), el componente 210_B incluye seis registros (B_0, B_1, B_2, B_3, B_4, B_5), el componente 210_C incluye cinco registro (C_0, C_1, C_2, C_3, C_4), el componente 210_D incluye tres registros (D_0, D_1, D_2), y el componente 210_E incluye cuatro registros (E_0, E_1, E_2, E_3). Los registros de cada componente 210 forman una trayectoria de exploración interna para ese componente 210. La trayectoria de exploración interna de cada componente 210 se puede describir usando NSDL.

Como se ilustra en la figura 2, cada uno de los componentes 210 soporta al menos una función. Especialmente, el componente 210_A soporta tres funciones, el componente 210_B soporta cuatro funciones, el componente 210_C soporta tres funciones, el componente 210_D soporta dos funciones, y el componente 210_E soporta una función. Las funciones soportadas por cada uno de los componentes 210 utilizan los registros (es decir, las trayectorias de exploración internas) de cada uno de los componentes 210, respectivamente. De este modo, las funciones soportadas por cada uno de los componentes 210 se puede describir usando NSDL.

Como se ilustra en la figura 2, los componentes 210 de S-o-C 110 se conectan mediante interconexiones 220 de componentes de S-o-C 110. Los componentes 210 (es decir, las trayectorias de exploración interna 210) y las interconexiones de componentes 220 entre los componentes 210 forman una trayectoria (o trayectorias) de exploración interna desde el puerto de prueba de entrada (TDI) de S-o-C 110 al puerto de prueba de salida (TDO) de S-o-C 110. La trayectoria de exploración interna se puede definir usando NSDL (por ejemplo definiendo cada uno de los componentes individuales usando NSDL y definiendo la composición del sistema usando NSDL, para formar por lo tanto una descripción global del sistema basada en NSDL).

Como se describe en este documento, la descripción global del sistema de S-o-C 110 basada en NSDL es una descripción algorítmica (es decir, debido a que cada uno de los componentes 210, interconexiones de componentes 220, dependencias de intracomponentes e intercomponentes, y similar, se describe usando una colección de algoritmos interrelacionados). La descripción algorítmica de S-o-C 110 incluye información adaptada para su uso por ST 120 probando el S-o-C 110 (por ejemplo información de longitud de trayectoria de exploración, información de jerarquía de trayectoria de exploración, y similar, así como varias combinaciones de las mismas).

Se puede describir un sistema en chip describiendo cada uno de los componentes del sistema en chip (por ejemplo, descripciones de IPs, instrumentos, dispositivos de cruce (allí donde se usan), y similar, así como carias combinaciones de los mismos) y describiendo la topología del sistema en chip (incluyendo descripciones de las interconexiones entre cada una de los componentes del sistema en chip, dependencias componentes y componentes, y similar, así como varias combinaciones de los mismos).

En una realización, en un sistema según la norma P1687, donde los componentes puede incluir IPs, instrumentos, y/o seleccionar bits de instrumentos (SIBs), la descripción del sistema en chip requiere una descripción de cada IP/instrumento (por ejemplo, incluir el significado de los registros internos, conjuntos de procedimientos/flujos de bits para aplicar/observar, y similar), una descripción de cada SIB (donde se usan SIB), una descripción de la

composición de la trayectoria de exploración del sistema (es decir, la manera en que la trayectoria de exploración pasa a través del sistema en chip, incluyendo la manera en que la trayectoria de exploración interna de cada componente pasa a través del componente, y similar), y similar, así como varias combinaciones de los mismos.

La inserción de un IP dentro de una cadena de exploración permite probar el IP a través de la cadena de exploración. El uso de un IP dentro de un sistema en chip varía según el nivel de privilegio de acceso (APL) para el IP (por ejemplo, sin acceso, acceso limitado pleno acceso).

Si el APL para un IP es “sin acceso” la herramienta de prueba no tiene conocimiento de los componentes internos del IP, de este modo, debe basarse en la información proporcionada por el proveedor del IP. Por ejemplo, se debe proporcionar el conjunto de flujos de bits para el IP (por ejemplo, los flujos de bits de entrada y los flujos de bits de salida esperados). En este caso, los flujos de bits se consideran flujos de bits estáticos. La herramienta de prueba debe insertar los flujos de bits estáticos para el IP dentro del flujo de bits del sistema para el sistema en chip y el procesar los flujos de bits de salida correspondientes como se especifica el proveedor del IP.

Si el APL para un IP es “pleno acceso” la herramienta de prueba tiene pleno conocimiento de los componentes internos del IP, incluyendo tanto la cadena de exploración interna del IP descrito en NSDL y las fuentes del IP en una lenguaje de descripción elegido. En este caso, la herramienta de prueba puede calcular directamente los flujos de bits de entrada requeridos y los flujos de bits de salida esperados para el IP (por ejemplo, usando su propio algoritmo), o el proveedor del IP puede proporcionar un conjunto de flujos de bits precalculados (por ejemplo, como flujos de bits estáticos y/o dinámicos).

Si el APL para un IP es “acceso limitado” la herramienta de prueba tiene solo un conocimiento limitado de los componentes internos del IP. Se proporciona una descripción NSDL del IP. La descripción NSDL del IP incluye una descripción de la cadena de exploración interna del IP y un conjunto de procedimiento que se puede usar para probar el IP. En este caso, la herramienta de prueba utiliza la descripción del IP con el fin de generar flujos de bits (por ejemplo flujos de bits de entrada y flujos de bits de salida esperados) para su uso en la prueba del IP.

La inserción de un instrumento dentro de un sistema en chip permite probar el sistema en chip por la inspección de algunos valores o algunas condiciones. Un instrumento puede soportar una o más funciones que se pueden usar con fines de prueba. De este modo, mientras la descripción de un IP incluye solo un conjunto de procedimientos necesarios para probar el IP, la descripción de un instrumento también incluye un conjunto de procedimiento (y/o flujos de bits) que se pueden usar para acceder a las funciones del instrumento. La descripción del instrumento puede incluir una descripción de funciones del instrumento en términos de los valores de registro de los registros de los que se compone el instrumento. De este modo, usando NSDL, la única diferencia entre los IPs y los instrumentos es el conjunto de procedimientos.

Como se describe en el presente documento respecto de los IPs e instrumentos, se pueden especificar descripciones de IPs e instrumentos usando los procedimientos especificados. Se puede considerar que un procedimiento es una concatenación de instrucciones atómicas a ejecutar cada vez que se solicita el procedimiento.

Las descripciones de procedimientos para un IP/instrumento pueden depender del APL del IP/instrumento.

Si el APL de un componente es “sin acceso”, los procedimientos se pueden representar como valores de flujos de bits (es decir, valores a escribir dentro de la trayectoria de exploración y valores a leer a partir de la trayectoria de exploración).

Si el APL de un componente es “acceso limitado” o “pleno acceso”, el conocimiento de la trayectoria de exploración del IP/instrumento proporciona libertad adicional respecto de la representación de los procedimientos para el IP/instrumento, de manera que los procedimientos se puedan componer. Una composición espacia de procedimientos indica el modo en que las entradas/salidas de diferentes procedimientos se pueden usar para componer las entradas/salidas de la trayectoria de exploración del sistema del sistema en chip. Una composición temporal indica la manera en que los procedimientos se pueden aplicar de manera secuencial al mismo componente (por ejemplo, IP/instrumento) para llevar a cabo operaciones específicas. Asimismo, se pueden anidar procedimientos a y/o desde procedimientos mayores y/o desde múltiples procedimientos menores.

Un procedimiento incluye atributos de procedimiento (es decir, una descripción del procedimiento) y un cuerpo de procedimiento. Las descripciones de procedimientos pueden incluir información tal como longitudes, indicaciones de modo ocupado, condiciones de entrada, condiciones de salida, dependencias, descripciones de trayectoria de exploración interna, y similar, así como varias combinaciones de los mismos.

Un procedimiento de longitud fija se puede definir como un procedimiento que siempre toma la misma cantidad de tiempo en ser ejecutado. Un procedimiento de longitud variable se puede definir como un procedimiento que toma una cantidad variable de tiempo en ejecutarse. Un procedimiento de longitud variable se puede definir usando otros valores de tiempo (por ejemplo, tiempos de mejor y peor caso, tiempos medios y similar, así como varias

combinaciones de los mismos). En una realización, se debe proporcionar al menos una condición de salida para cada procedimiento de longitud variable.

La longitud de procedimiento se puede expresar en términos de ciclos (o, cuando no es posible su expresión en términos de ciclos, en algunos otros términos absolutos, tales como segundos o alguna otra medida de tiempo). Por ejemplo, puede que no sea posible expresar la longitud de procedimiento en términos de ciclos para instrumentos que toman medidas físicas, instrumentos asíncronos, instrumentos que están funcionando en dominios de reloj no sincronizado, y similar. Si la longitud de procedimiento se expresa en tiempo, la herramienta de prueba puede determinar el recuento de ciclos (por ejemplo, con conocimiento del periodo efectivo de rejo de prueba) o estimar el recuento de ciclos (por ejemplo usando un periodo de reloj de referencia).

El modo ocupado del procedimiento se debería declarar mediante el procedimiento. El modo ocupado de un procedimiento es “retenido” si, durante la ejecución del procedimiento, el valor de la cadena de exploración no debe variar (es decir, cada acceso de exploración debe restablecerlo a mismo valor). Por ejemplo, un instrumento “retenido” puede ser un dispositivo combinatorio (es decir, conde cada modificación afecta al resultado). El modo ocupado de un procedimiento es “no importa”si, durante la ejecución del procedimiento, el valor de la cadena de exploración no es importante. Por ejemplo, un instrumento “no importa” puede ser cualquier dispositivo que muestrea entradas solo cuando se activan.

La descripción de un sistema en chip incluye también descripciones de dependencias asociadas con el sistema en chip. Las dependencias de un sistema en chip incluyen dependencias intracomponentes (entre las funciones o procedimientos de un componente del sistema en chip) y dependencia intercomponentes (entre las funciones o procedimientos de diferentes componentes del sistema en chip). Las descripciones de dependencias se pueden especificar de muchas maneras (por ejemplo, escuchándolas, por enlace basado en el nombre, y similar, así como varias combinaciones de las mismas.).

Por ejemplo se puede describir una función X a través de una indicación de que la “función X depende de la conclusión de las funciones X_1, X_2, …, X_n”, lo cual significa que la función X no puede empezar su ejecución hasta que concluya cada una de las subfunciones listadas. Por ejemplo, se puede describir un procedimiento X a través de una indicación de que “la función X depende del procedimiento Y” significa que, para cualesquiera razones, el procedimiento Y debe concluir antes de que se ejecute el procedimiento X. Las dependencias de un sistema en chip se puede describir de varias maneras.

Se debería declarar las dependencias intracomponentes e intercomponentes de un sistema en chip. La declaración de dependencias permite la herramienta de prueba para llevar a cabo una programación de prueba. En una realización, se puede declarar una dependencia como parte de la parte declarativa del o los procedimientos de la dependencia. En una realización, se puede declarar una dependencia usando denominación explícita. En tal realización, se pueden utilizar parámetros genéricos para enlazar dependencias externas con el o los nombres simbólicos de los componentes con los cuales se asocian las dependencias externas.

Usando NSDL, se pueden llevar a cabo la descripción y la declaración de dependencias de varias maneras.

La construcción de un procedimiento (o procedimientos) para un componente (o grupo de componentes) cambia basándose en los APLs del o los componentes, lo cual afecta al conocimiento de información de entrada/salida asociada al componente.

La figura 3 ilustra el conocimiento de entrada-salida de un componente 300 “sin acceso”. Si el APL de un componente es “sin acceso”, para cada función soportada por el componente, el cuerpo de la función se compondrá de información de flujo de bits de entrada (a usar para componer el flujo de bits de entrada), información de longitud de función y de longitud de trayectoria de exploración, e información de flujo de bits de salida (por ejemplo, el flujo de bits de salida esperado, procesamiento de flujo de bits de salida requerido, y similar, así como varias combinaciones de las mismas).

La figura 4 ilustra el conocimiento de entrada-salida de un componente de“acceso limitado” o “pleno acceso” 400. Si el APL de un componente es “acceso limitado” o “pleno acceso”, se conoce la trayectoria de exploración interna del componente(es decir, se conoce cada uno de los registros del componente, así como la topología de los registros). La trayectoria de exploración interna del componente se puede dividir en múltiples rodajas, que se pueden distribuir en uno o más niveles jerárquicos. El acceso a diferentes niveles jerárquicos se puede controlar de cualquier manera (por ejemplo, usando uno o más registros del nivel adyacente). La partición de una trayectoria de exploración interna de un componente en múltiples niveles jerárquicos se puede entender mejor respecto del componente ilustrado y descrito respecto de la figura 4.

Como se ilustra en la figura 4, la trayectoria de exploración interna se divide en cinco rodajas (denominadas rodajas 1, 2, 3, 4, 5). Las cinco rodajas de la trayectoria de exploración interna se distribuyen en dos niveles jerárquicos (denominados niveles 0, 1). Las rodajas 1, 2, 3, 4 y 5 se componen de cuatro, cuatro, tres, tres y dos registros,

respectivamente. Las rodajas 1, 2 y 5 se sitúan en el nivel 0 de la jerarquía. Las rodajas 3 y 4 se sitúan en el nivel 1 de la jerarquía. Un registro (denominado registro H1) controla el acceso entre el nivel 0 y el nivel 1 de la jerarquía de trayectoria de exploración interna. El registro H1 controla la trayectoria de exploración interna de manera que se evitan bien las rodajas 3 y 4 (es decir, se excluyen de la trayectoria de exploración interna) o bien no se evitan las rodajas 3 y 4 (es decir se incluyen en la trayectoria de exploración interna).

Como se ilustra en la figura 4, la trayectoria de exploración interna se compone como sigue. La entrada de la rodaja 1 es TDI y la salida de la rodaja 1 es la entrada de la rodaja 2. La entrada de la rodaja 2 es la salida de la rodaja 1 y la salida de la rodaja 2 es una primera entrada a H1 (entrada de nivel 0). En el nivel 0, la entrada de H1 es la salida de la rodaja 2 y la salida de H1 es la entrada de la rodaja 5. En el nivel 1, la salida de H1 es la entrada de la rodaja 3 y la entrada de H1 es la salida de la rodaja 4. La entrada de la rodaja 2 es una salida de H1 (salid de nivel 1) y la salida de la rodaja 3 es la entrada de la rodaja 4. LA entrada de la rodaja 4 es la salida de la rodaja 3 y la salida de la rodaja 4 es una entrada de H1 (entrada de nivel 1). La rodaja de entrada 5 es una salida de H1 (salida de nivel 0) y la salida de la rodaja 5 es TDO.

En una realización , la partición de una trayectoria de exploración interna de un componente en múltiples rodajas es funcional, es decir, cada rodaja de la trayectoria de exploración interna tiene una o más funciones que operan en la misma. Cada una de las cuales se puede probar usando uno o más procedimientos de rodaja, cada uno de los cuales se puede programar independientemente por la herramienta de prueba. En esta realización, el cuerpo de una función que opera sobre una rodaja dada de la trayectoria de exploración interna se puede componer de una manera similar a la composición del cuerpo de una función de un componente “sin acceso” (es decir, el cuerpo de la función se describe usando información de flujo de bits de entrada, información de longitud de función y de longitud de trayectoria de exploración, información de flujo de bits de salida, y similar, así como varias combinaciones de las mismas).

De este modo utilizando diferentes procedimientos que operan con diferentes rodajas o combinaciones de rodajas de la trayectoria de exploración interna, la partición de la trayectoria de exploración interna de un componente se puede usar de varias maneras. Por ejemplo (con referencia a la figura 4), se podría ejecutar un primer procedimiento P1 asociado a la rodaja 1 para escribir datos en la rodaja 1, activando de este modo la ejecución de alguna función en el componente que da como resultado datos de almacenamiento o de resultados en la rodaja 2 y datos de error en la rodaja 5. A continuación, se podrían ejecutar dos procedimientos adicionales P2 y P5 (que operan sobre las rodajas 2 y 5, respectivamente) para leer os valores de las rodajas 2 y 5, respectivamente. De este modo, la partición de la trayectoria de exploración interna dentro de las rodajas (incluyendo la partición a través de múltiples niveles jerárquicos) proporciona gran flexibilidad en la prueba de sistema en chip. El ST 120 puede convertir fácilmente el procedimiento algorítmico en flujos de bits de prueba en serie (por ejemplo flujos de bits de entrada y flujos de bits de salida esperados).

Puesto que se pueden desarrollar diferentes procedimientos para utilizar las rodajas de una trayectoria de exploración interna de un componente, los procedimientos que utilizan rodajas de una trayectoria de exploración interna deben ser capaces de referenciar las rodajas (incluyendo las señales en el interior de las mismas). Un procedimiento puede referenciar una rodaja de cualquier manera. En la realización, por ejemplo, un procedimiento puede referenciar una rodaja usando denominación explícita. En tal realización, cada rodaja (es decir, un registro o grupo de registros) se asigna a un único nombre, permitiendo de este modo que se pueda acceder a rodajas instanciadas jerárquicamente como ficheros. Un ejemplo del uso de denominación explícita para identificar rodajas de un componente se ilustra y describe respecto de la figura 5.

La figura 5 ilustra la referencia de rodajas de una trayectoria de exploración interna de un componente 500. Como se ilustra en la figura 5, el componente 500 incluye una trayectoria de exploración de 12 registros que tiene seis rodajas, cada una de las cuales se denomina con un único nombre. La primera rodaja es un único registro denominado BS_0. La segunda rodaja es un único registro denominado BS_1. La tercera rodaja es una cadena en serie de cuatro registros denominada SCAN_4_BIT_0, que incluye cuatro registros denominados BS_0, BS_1, B_2 y BS_3. La cuarta rodaja es una cadena en serie de cuatro registros denominada Another_SCAN_4_BIT, que incluye cuatro registros denominados BS_0, BS_1, BS_2 y BS_3. La quinta rodaja es un único registro denominado BS_2. La sexta rodaja es un único registro denominado BS_3.

Como se describe en el presente documento, se puede acceder a los registros jerárquicamente instanciados como ficheros. Por ejemplo, el único nombre del registro indicado por la flecha 1 es BS_1 (aunque dos otros registros se denominan BS_1, se verá la manera en que el uso de la denominación jerárquica garantiza la única denominación de registros). Por ejemplo, el único nombre de la rodaja (es decir, grupo de registros) indicado por la flecha 2 SCAN_4_BIT_0. Por ejemplo el único nombre del registro indicado por la flecha 3 es another_SCAN-4_BIT-BS_1. Dicho de otro modo, cada rodaja de la trayectoria de exploración interna, así como cada registro de cada rodaja, para cualquier número de niveles jerárquicos de rodajas y registros, es capaz de referenciarse usando un único nombre.

Aunque en el presente documento se ilustra y describe principalmente atributos de procedimiento específico que se

pueden utilizar para definir un procedimiento o un conjunto de procedimientos para un componente (por ejemplo longitudes de procedimiento, dependencias de función y de procedimiento, descripciones de trayectoria de exploración interna para un componente, descripciones de rodaja, atributos de referencia de rodajas, y similar), se pueden especificar otros varios atributos de procedimiento. Los atributos de procedimiento usados para describir un procedimiento pueden incluir cualquier información que se pueda usar para describir el procedimiento de tal manera que soporta la prueba del sistema en chip.

Además de los atributos de procedimiento, un procedimiento incluye un cuerpo de procedimiento. El cuerpo de procedimiento incluye los detalles del procedimiento. El cuerpo de procedimiento se puede aplicar de cualquier manera. El cuerpo de procedimiento no tiene un concepto de la trayectoria de exploración (en su lugar, los accesos se realizan en paralelo sobre las rodajas y la herramienta de prueba fusiona estas operaciones en los accesos a la trayectoria de exploración del sistema). Este cuerpo de procedimiento puede usar varias instrucciones, aserciones, procedimientos anidados y similar, así como varias combinaciones de los mismos.

En una realización, en la cual se describe un sistema en chip usando NSDL aplicado usando el lenguaje VHDL (y de este modo, cada uno de los procedimientos asociados al sistema en chip se describe usando NSDL aplicado que usa el lenguaje VHDL), el cuerpo de procedimiento se puede expresar usando la sintaxis de VDHL. El cuerpo de procedimiento de un procedimiento se puede expresar de otras maneras, dependiendo de la aplicación de la descripción del sistema en chip.

Una descripción de una trayectoria de exploración interna de un componente se proporciona fácilmente con NSDL. En NSDL, cada célula de registro es considerada una entidad, y las células de registro se pueden agrupar en paquetes (por ejemplo, usando reglas de VHDL donde NSDL se aplica como un superconjunto de VHDL). En NDL, cada entidad de registro se describe usando dos parámetros genéricos: “precedente” y “siguiente”, que se pueden usar para expresar la trayectoria de exploración referenciando de manera explícita instancias de registro. El parámetro “precedente” de una entidad de registro especifica la fuente de la entrada a esa entidad de registro. El parámetro “siguiente” de una entidad de registro especifica el destino de la salida de esta entidad de registro. De este modo, una entidad de registro puede incluir un único registro, un grupo de registros (incluyendo agrupaciones jerárquicas), y similar.

La descripción de una trayectoria de exploración interna de un componente se puede entender mejor en relación con el siguiente código de muestra que corresponde a la cadena de exploración interna ilustrada y descrita respecto de la figura 5.

El siguiente código de muestra muestra una única trayectoria de exploración que incluye cuatro registros básicos de exploración de límites dispuestos en serie allí donde el primer registro de exploración de límites muestra recibe la entrada del puerto TDI y el cuartoregistro de exploración de límites proporciona la salida al puerto TDO:

```
BS_0: BS generic map (precedent => "TDI", following => "BS_1");
BS_1: BS generic map (precedent => "BS-0", following => "BS_2");
BS_2: BS generic map (precedent => "BS_1 ", following => "BS_3");
BS_3: BS generic map (precedent => "BS_2", following => "TDO");
```

Esta cadena de exploración de cuatro registros se puede encapsular en una entidad e instanciar jerárquicamente. El siguiente código de muestra muestra exactamente la manera en que esta cadena de exploración de cuatro registros se puede encapsular en una entidad e instanciar jerárquicamente (para su fin último en la descripción de las entidades de registro SCAN_4_BIT_0 y otro_SCAN-4_BIT):

```
La entidad SCAN_4_Bit es:
        Generic (precedent: string := "TDI"; following : string := TDO");
End entity;
La arquitectura A de SCAN_4_BIT es
begin
        BS_0: BS generic map (precedent => precedent, following => "BS_1");
        BS_1: BS generic map (precedent => "BS_0", following => "BS_2");
        BS_2: BS generic map (precedent => "BS_1 ", following => "BS_3");
        BS_3: BS generic map (precedent => "BS_2", following => following);
end;
```

El siguiente código de muestra muestra la representación (descripción) de la trayectoria de exploración de 12 registros de la figura 5 (donde la trayectoria de exploración de 12 registros se representa usando descripciones de seis entidades de registro, incluyendo cuatro registros individuales y dos entidades de registro compuestas por cadenas de exploración de cuatro registros, como se ilustra y describe respecto de la figura 5.

```
BS_0: BS generic map (precedent => "TDI", following => "BS_1");
```

```
BS_1: BS generic map (precedent => "BS-0", following => "SCAN_4_BIT_0");
SCAN_4_BIT_0: BS generic map (precedent => "BS_1", following => "another-SCAN_4_BIT");
another_SCAN_4_BIT: BS generic map (precedent => "SCAN_4_BIT_0", following => "BS_2");
BS_2: BS generic map (precedent => "another_SCAN_4_BIT", following => "BS_3");
BS_3: BS generic map (precedent => "BS_2", following => "TDO");
```

Usando este tipo de descripción, información acerca de la de la trayectoria de exploración (por ejemplo su longitud, su estructura jerárquica, y similar, así como varias combinaciones de las mismas) se puede calcular automáticamente con la herramienta de prueba en el tiempo de compilación (por ejemplo, usando verificación contextual). Asimismo, las descripciones de las entidades de registro son altamente reutilizables (por ejemplo, para su uso en otros diseños). Ya que la información de descripción para entidades de registro se puede recoger, almacenar y verificar de muchas maneras, (es decir ya que la aplicación de la tabla de símbolos puede variar considerablemente entre compiladores, por ejemplo en términos de tablas de dispersión, bases de datos y similar), tales detalles se omiten por motivos de claridad y generalidad.

Asimismo, usando este tipo de descripción, ya que cada componente del sistema en chip se describe fácilmente, se puede obtener una descripción global de sistema del sistema en chip combinando descripciones de componentes individuales (incluyendo descripciones de interconexiones entre componentes, descripciones de dependencias intercomponentes, y similar; así como varias combinaciones de los mismos). Dicho de otro modo, NSDL proporciona una flexibilidad considerable en la descripción (y por lo tanto en la prueba) de las configuraciones de sistema en chip.

Como se describe en el presente documento, además de poder describir componentes típicos que forman parte de la trayectoria de exploración de un sistema en chip, NSDL es también capaz de describir un dispositivo que permite cambios dinámicos en la trayectoria de exploración de un sistema en chip. Este tipo de dispositivo se denomina “dispositivo de cruce” en términos de NSDL. Una representación genérica de un dispositivo de cruce se ilustra y describe respecto de la figura 6.

La figura 6 ilustra un diagrama de bloques de alto nivel de una representación de un dispositivo de cruce. Específicamente, la representación 600 de dispositivo de cruce es representativa de un dispositivo de cruce capaz de modificar dinámicamente la trayectoria de exploración de un sistema en chip. Un dispositivo de cruce encamina una o más entradas (denominadas afluentes) a una o más salidas (denominadas tributarios). Como se ilustra en la figura 6, la representación 600 de dispositivo de cruce incluye una pluralidad de afluentes (indicados como path_in_0 a path_in_m) y una pluralidad de tributarios (indicados como path_out_0 a path_out_n).

Con el fin de utilizar un dispositivo de cruce en la prueba de un sistema en chip, se describe el dispositivo de cruce. La descripción de la o las trayectorias de exploración dinámicamente variable de un dispositivo de cruce es bastante difícil (sino imposible) de conseguir usando BSDL/HSDL, que se diseñan específicamente para procesar trayectorias de exploración estática: Por el contrario, la descripción de trayectoria(s) de exploración dinámicamente variable de un dispositivo de cruce se proporciona fácilmente con NSDL., lo cual proporciona una descripción algorítmica, del dispositivo de cruce que es fácilmente entendida por la herramienta de prueba.

A continuación se muestra la descripción genérica de un dispositivo de cruce que usa NSDL:

```
IP <entity name> es
    Generic (
            precedent : string := "TDI";
            following : string := "TDO";
            affluent : string := "deselected";
            tributary : string := "deselected";
            );
End <entity name>;
```

Como se ve en la descripción genérica del dispositivo de cruce proporcionada anteriormente, los parámetros genéricos “afluente” y “tributario” se pueden usar como trayectoria de exploración jerárquica. Los parámetros genéricos “afluente” y “tributario” se establecen inicialmente como "deseleccionado”, de manera que el dispositivo de cruce opera meramente como un componente pasante dentro de la trayectoria de exploración (es decir, el afluente y el tributario no están activos, y de este modo, no modifican dinámicamente la trayectoria de exploración). El dispositivo de cruce se puede activar para modificar dinámicamente la trayectoria de exploración usando los parámetros “afluente” y “tributario” para requerir funciones de selección relativa.

Si hay más de un afluente y/o más de un tributario, solo se ha de identificar cada uno de los afluentes y/o tributario. En una realización, por ejemplo, se puede usar una única numeración ordenada (por ejemplo afluente_0, afluente_1, etc. para los afluentes; tributario_0, tributario_1, etc. para los tributarios). Usando esta descripción basada en NSDL, la herramienta de prueba es capaz de entender las conexiones a través del dispositivo de cruce referenciando meramente los parámetros genéricos.

Una descripción de un dispositivo de cruce se puede generar de cualquier manera. En una realización, se genera una descripción de un dispositivo de cruce usando una descripción de un conjunto de conexiones de entrada y un conjunto de conexiones de salida interconectadas mediante una arquitectura adaptada para controlar dinámicamente el acceso a un componente(o componentes) de un sistema en chip. La descripción del dispositivo de cruce se puede almacenar para su uso en la realización de una prueba. Un procesador puede recibir o recuperar la descripción del dispositivo de cruce (por ejemplo, a partir de la memoria, otro sistema, o a partir de cualquier otra fuente de tales descripciones) para su uso en la realización de una prueba.

Como se describe en el presente documento, el conjunto de conexiones de entrada incluye una conexión de entrada de trayectoria de exploración (conectada a la trayectoria de exploración del sistema en chip, en una dirección procedente de TDI) y al menos una conexión hacia el componente (indicado como conexión de entrada de acceso de componente) y el conjunto de conexiones de salida incluye una conexión de salida de trayectoria de exploración (conectada a la trayectoria de exploración del sistema en chip, en dirección hacia TDO) y al menos una conexión al componente (indicada como una conexión de salida de conexión de componente).

En una dirección, el conjunto de conexiones de entrada se especifica usando el parámetro “precedente” (para la conexión de entrada de trayectoria de exploración) y uno o más parámetros “afluente” (para una o más conexiones de entrada de acceso de componente). En una realización, el conjunto de conexiones de salida se especifica usando el parámetro “siguiente” (para la conexión de salida de trayectoria de exploración) y uno o más parámetros “afluente” (para una o más conexiones de salida de acceso de componente).

El dispositivo de cruce puede modificar dinámicamente la trayectoria de exploración del sistema de cualquier manera. La arquitectura puede ser cualquier arquitectura para conectar dinámicamente algunas conexiones del conjunto de conexiones de entrada a algunas de las conexiones del conjunto de conexiones de salida mediante el componente donde se selecciona el dispositivo de cruce para añadir el componente a la trayectoria de exploración de sistema. Por ejemplo, la arquitectura puede ser una arquitectura de conmutador, una arquitectura de bus, una arquitectura de red, y similar.

En una realización, la descripción de un dispositivo de cruce es una descripción algorítmica que incluye al menos una regla de composición adaptada para ser entendida por una herramienta de prueba. La descripción del dispositivo de cruce se puede modificar dinámicamente para añadir dinámicamente el componente a la trayectoria de exploración y eliminar dinámicamente el componente de la trayectoria de exploración (por ejemplo, modificando los parámetros “afluente” y “tributario”).

Usando NSDL para describir recursos de prueba en la prueba del sistema en chip permite que muchos dispositivos de cruce diferentes sean descritos de manera a permitir que el dispositivo de cruce se use en la prueba del sistema en chip. Los diferentes dispositivo de cruce que se pueden describir por NSDL se pueden agrupar en tres categorías generales. Especialmente, un dispositivo de cruce puede representar una conexión “alámbrica”, una conexión “de transacción” o una conexión “alámbrica-de transacción).

Un dispositivo de cruce “alámbrico” es un dispositivo de cruce que se conecta básicamente por cable a la trayectoria de exploración del sistema en chip (y se puede seleccionar o deseleccionar a voluntad). Un dispositivo de cruce “alámbrico” opera de una manera similar a un conmutador (es decir, una conexión puede programarse dinámicamente entre un afluente y un tributario, a voluntad). Una trayectoria de exploración de un dispositivo de cruce !alámbrico” necesita deseleccionarse de manera explícita porque la trayectoria de exploración del dispositivo de cruce “alámbrico” se conecta por cable a la trayectoria de exploración de sistema del sistema en chip. Un primer ejemplo de un dispositivo de cruce “alámbrico” (especialmente un componente de seleccionar Bit de Instrumento (SIB), que es parte de la proposición de hardware de P1687) se ilustra y describe en la figura 8. Un segundo ejemplo de dispositivo de cruce “alámbrico” se ilustra y describe en la figura 9.

Un dispositivo de cruce “de transacción” es un dispositivo de cruce que soporta conexiones temporales (es decir, conexiones que representan transacciones específicas). Un dispositivo de cruce “de transacción” puede operar como cualquier arquitectura (por ejemplo un bus, un sistema en chip, y similar). Las transacciones pueden ser cualesquiera transacciones que pueden ser soportadas por la arquitectura del dispositivo de cruce “de transacción” (por ejemplo, acceso por bus en arquitectura de bus, encaminamiento en arquitecturas de red, y similar). Una trayectoria de exploración de un dispositivo de cruce “de transacción” no necesita deseleccionarse de manera explícita porque la trayectoria de exploración del dispositivo de cruce “de transacción” está activa solamente durante el tiempo de la transacción. Un ejemplo de un dispositivo de cruce “de transacción” se ilustra y describe en la figura 10.

En un dispositivo de cruce “alámbrico”, la al menos una conexión de entrada de acceso de componente y la al menos una conexión de entrada de acceso de componente son conexiones alámbricas. De este modo, la o las conexiones de entrada de acceso de componente y la o las conexiones de salida de acceso de componente son diferentes conexiones físicas. En un dispositivo de cruce “de transacción” la al menos una conexión de entrada de acceso de componente y la al menos una conexión de entrada de acceso de componente son conexiones de transacción, de

manera que una conexión física se puede usar para soportar múltiples transacciones (es decir, la conexión o las conexiones de entrada de acceso de componente y la o las conexiones de salida de acceso de componente pueden compartir la misma conexión física, pero pueden considerarse conexiones de transacción diferentes).

Usando BSDUHSDL, aunque es posible describir un dispositivo de cruce “alámbrico” específico (concretamente, el componente SIB), esta descripción es bastante difícil. Además, la descripción de dispositivos de cruce “alámbricos” más complicados que usan BSDUHSDL puede no ser posible. Asimismo, la descripción de dispositivos de cruce “de transacción” y “alámbrico de transacción” que usan BSDUHSDL es más probablemente imposible. De este modo, por este motivo, el único dispositivo de cruce que se está normalizando en P1687 es el componente SIB.

Por el contrario, usando, NSDL, la descripción de cualquier dispositivo de cruce “alámbrico” dispositivo de cruce “de transacción”, y dispositivo de cruce “alámbrico de transacción”, es soportado claramente, y es soportada fácilmente por una herramienta de prueba. De hecho, virtualmente cualquier dispositivo de cruce, de cualquier complejidad, capaz de modificar dinámicamente una trayectoria de exploración de un sistema en chip se puede describir usando NSDL. Con el fin de ilustrar la potencia de NSDL, se proporciona en el presente documento unos cuantos ejemplos de dispositivos de cruce (y sus respectivas descripción expresadas usando NSDL) respecto de las figuras 7 – 10.

La figura 7 ilustra un diagrama de bloques de alto nivel de uso de un dispositivo de cruce genérico para modificar dinámicamente la trayectoria de exploración de un sistema en chip. Como se ilustra en la figura 7, el sistema en chip 700 incluye una trayectoria de exploración que incluye un dispositivo de cruce 710 genérico. La trayectoria de exploración incluye dos partes de trayectoria de exploración permanente (es decir, partes que siempre se incluyen en la trayectoria de exploración) y una parte de trayectoria de exploración opcional (es decir, una parte que se puede incluir dinámicamente en y eliminar de la trayectoria de exploración mediante un dispositivo de cruce 710).

Como se ilustra en la figura 7, la trayectoria de exploración del sistema en chip 700, la entrada de acceso de prueba (TDI) se acopla a la primera parte de trayectoria de exploración permanente (que incluye una serie de células de exploración de límites), la primera parte de trayectoria de exploración permanente se acopla a una primera entrada (es decir, la precedente) del dispositivo de cruce 710, una primera salida (es decir, la siguiente) del dispositivo de cruce 710 acoplado a la segunda parte de trayectoria de exploración permanente, y la segunda parte de trayectoria de exploración permanente se acopla a la salida de acceso de prueba (TDO).

El dispositivo de cruce 710 permite la incorporación dinámica (selectiva) de la parte de trayectoria de exploración opcional a la trayectoria de exploración del sistema en chip 700. Especialmente, el dispositivo de cruce 710 incluye una segunda salida (es decir, tributario) que se puede seleccionar para acoplar la salida de la primera parte de trayectoria de exploración permanente a una entrada de la parte de trayectoria de exploración opcional e incluye una segunda entrada (es decir, un afluente) que se puede seleccionar para acoplar una salida de la parte de trayectoria de exploración opcional a la entrada de la segunda parte de trayectoria de exploración permanente.

El dispositivo de cruce 710 permite que la trayectoria de exploración se modifique dinámicamente seleccionando/deseleccionando combinaciones específicas de entradas/salidas. Cuando la segunda salida y la segunda entrada del dispositivo de cruce 710 se deselecciona, la trayectoria de exploración es TDI, SCAN_4_BIT_0, BS_0, BS_1, el dispositivo de cruce 710, BS_2, TDO. Cuando la segunda salida y la segunda entrada del dispositivo de cruce 710 se selecciona (es decir, la primera entrada se conecta a la segunda salida y la segunda entrada se conecta a la primera salida), la trayectoria de exploración es TDI, SCAN_4_BIT_0, BS_0_, BS_1, el dispositivo de cruce, otro_SCAN_4_BIT, BS_3, el dispositivo de cruce 710, BS_2, TDO.

La figura 8 ilustra un diagrama de bloques de alto nivel de un dispositivo de cruce que se puede describir usando NSDL. Especialmente, la figura 8 ilustra un dispositivo SIB 800. El dispositivo SIB 800 permite la selección de otro componente (es decir, de manera que el componente se añade a la trayectoria de exploración).

El dispositivo SIB se controla por un bit de selección. Cuando el valor del bit de selección es “0”, la célula no está activa (es solo un bit en el interior de la trayectoria de exploración). Cuando el valor del bit de selección se establece en “1”, la trayectoria de exploración sale a través del puerto WSIo (es decir, el tributario) y entra a través del puerto WSOi (es decir el afluente), añadiendo de este modo a la trayectoria de exploración cualquier dispositivo al cual se conectan estos puertos.

A continuación se muestra una descripción basada en NSDL del dispositivo SIB 800 (con los números incluidos por motivos de claridad):

```
1 IP SIB generic (precedent : string := "TDI";
        2 following : string := "TDO";
        3 tributary : string := "deselected";
        4 affluent : string:= "deselected")
5 Begin
6 UpSIB : REG generic map (precedent =>"TDI",
```

```
        7 following=>"TDO"
        8 elements => 1);
9
10 Procedure select
11 Length 1;
12 Selection wired;
13 {
14 UpSib <= ’1’;
15 tributary := "TDI";
16 affluent := "TDO";
17 }
18
19 Procedure deselect
20 Length 1;
21 Selection wired;
22 {
23 UpSib <= ’0’;
24 tributary := "deselected";
25 affluent := "deselected";
26 }
27
28 End SIB;
```

En la descripción del dispositivo SIB 800: las líneas 6 – 9 declaran los registros internos (es decir los bits que la herramienta necesita con el fin de procesar la jerarquía, que es solo un bit en el caso del dispositivo SIB 800; y las líneas 10 – 17 muestran el procedimiento de selección y las líneas 19 – 26 muestran el procedimiento de deselección (fácilmente identificado por la herramienta de prueba que usa sus nombres respectivos).

En el cuerpo de descripción, es fácil identificar las dos funciones de NSDL: (1) la modificación de la trayectoria de exploración y, de este modo, se expresa el flujo de bits por la rodaja que procesa (véase las líneas 14 y 23); y (2) la modificación de la topología se realiza asignando los valores de las secuencias (por ejemplo de una manera similar a la asignación “precedente” y “siguiente” en la cartografía de topología).

La figura 9 ilustra un diagrama de bloques de alto nivel de un dispositivo de cruce que se puede describir usando NSDL. Especialmente, la figura 9 ilustra un dispositivo 900 de conmutación de jerarquía. El dispositivo 900 de conmutación de jerarquía incluye tres afluentes de entrada y dos tributarios de salida. Aunque se ilustra como que tiene tres afluentes y dos tributarios, se puede soportar cualquier número de afluentes y tributarios.

A continuación se ilustra una descripción basada en NSDL del dispositivo 900 de conmutación de jerarquía (con los números de línea incluidos por motivos de claridad):

```
1 IP Hierarchy_switch generic (precedent : string := "TDI";
2       following : string := "TDO";
3       tributary_0 : string "deselected";
4       tributary_1 : string :="deselected";
5       affluent_0 : string:="deselected";
6       affluent_1 : string:="deselected";
7       affluent_2 : string:="deselected";
8       )
9
10 Begin
11
12 Affl_map : Vector_REG generic map (precedent =>‘TDI",
13       following=>"Trib_map"
14       elem_size => 2;
15       elements => 3);
16 Trib_map : Vector_REG generic map (precedent =>"Affl_map",
17       following=>"TDO"
18       elem_size => 2;
19       elements => 2);
20
21 subtype name is string (1 to 11);
22 type name_vector is array (natural range <>) of name;
23 constant affluent_name : name_vector (0 to 2) :=
```

```
24        ("affluent_0 ","affluent_1 ","affluent_2 ");
25 constant tributary_name : name_vector (0 to 1) :=
26        ("tributary_0","tributary_1");
27
28
29 Procedure select(affluent_nmb : in std_logic_vector(1 downto 0);
30        tributary_nmb : in std_logic)
31 Length 1;
32 Selection wired;
33 {
34 Trib_map <= affluent_nmb;
35 Affl_map <= "0"&tributary_nmb;
36
37 case (tributary_nmb)
38        when '0' => tributary_0 :=
39        affluent_name(conv_integer(affluent_nmb));
40 when '1' => tributary_1 :=
41        affluent_name(conv_integer(affluent_nmb));
42 end case;
43
44 case (affluent_nmb)
45 when "00" => affluent_0 :=
46        tributary_name(conv_integer(tributary_nmb));
47 when "01" => affluent_1 :=
48        tributary_name(conv_integer(tributary_nmb));
49 when "10" => affluent_2 :=
50        tributary_ name(conv_integer(tributary_nmb));
51 when others => assert false report "ERROR!" severity failure;
52 end case;
53 }
54
55 Procedure deselect_affluent
56        (affluent_nmb : in std_logic_vector(1 downto 0))
57 Length 1;
58 Selection wired;
59 {
60 Affl_map <= "11";
61 case (affluent_nmb)
62        when "00" => affluent_0 := "deselected";
63        when "01" => affluent_1 := "deselected";
64        when "10" => affluent_2 : := "deselected";
65        when others => assert false report "ERROR!" severity warning;
66 end case;
67
68 }
69
70 Procedure deselect_tributary (tributary_nmb : in std_logic)
71 Length 1;
72 Selection wired;
73 {
74 Trib_map <= "11";
75 case (tributary_nmb)
76        when '0' => tributary_0 := "deselected";
77        when '1' => tributary_1 := "deselected";
78 end case;
79 }
80 End hierarchy_switch;
```

En la descripción de dispositivo 900 de conmutación de jerarquía: las líneas 3 – 7 declaran los afluentes y tributarios (donde a cada uno se le da un nombre); las líneas 12 – 19 declaran la trayectoria de exploración interna (es decir similar a la matriz de conmutación); las líneas 21 – 26 aprovechan plenamente las capacidades VHDL definiendo algunos tipos personalizados (preparando de este modo lo que significa esencialmente una matriz de conmutación para los descriptores jerárquicos).

En la descripción del dispositivo 900 de conmutación de jerarquía: el procedimiento “Select” (Seleccionar) (líneas 29-53) se ocupa de las modificaciones de flujos de bits y, asimismo, aprovecha las capacidades algorítmicas VHDL (y los tipos personalizados definidos anteriormente) con el fin de describir las modificaciones de trayectoria de exploración dinámica. Los parámetros de la función se refieren a los números ordinales de las trayectorias a conectar.

Como se observa en la descripción del dispositivo 900 de conmutación de jerarquía, hacer que una conexión a través de un dispositivo 900 de conmutación de jerarquía se vuelva inactiva no requiere la deselección de ambos extremos de la conexión, en su lugar la deselección de solo un extremo de la conexión corta la conexión. De este modo, la deselección se puede procesar mediante dos procedimientos. (1) “deselect_affluent” permite la deselección de afluentes (usando el número ordinal como parámetro) y (2) “deselect_tributary” permite la deselección de tributarios (usando el número ordinal como parámetro).

La figura 10 ilustra un diagrama de bloques de alto nivel de un dispositivo de cruce que se puede describir usando NSDL. Especialmente, la figura 10 ilustra un dispositivo 1000 de arquitectura de bus. El dispositivo 1000 de arquitectura de bus incluye un componente 1011 de pasarela maestra y cinco componentes esclavos 1013A – 1013_E (colectivamente, componentes esclavos 1013) interconectados por un bus de componente 1012.

En el dispositivo 1000 de arquitectura de bus, a cada uno de los componentes esclavos 1012 se le asigna una dirección usada por el componente de pasarela 1011 para acceder a los componentes esclavos 1012. Un protocolo determinado debe ser soportado por el dispositivo 1000 de arquitectura de bus con el fin de permitir que el componente 1011 de pasarela acceda a los componentes esclavos 1012; sin embargo, la descripción basada en NSDL del dispositivo de arquitectura de bus 1000 no requiere ninguna información acerca del protocolo.

A continuación una descripción basada en NSDL del dispositivo 1000 de arquitectura de bus (con los números de línea incluidos por motivos de claridad)

Descripción de paquete de pasarela:

```
1 Package GW_package is
2
3 Constant N_SLAVES : integer := 5;
4 Constant ADDRESS_DEPTH : integer := 3;
5 subtype slave_name_type is string (1 to 1);
6 subtype slave_address_type is std_logic_vector
7         (ADDRESS_DEPTH-1 downto 0);
8
9 Type slave_mapping_type is record
10        slave_name : slave_name_type;
11        slave_address : slave_address_typr;
12 end record;
13
14 Type network_mapping_type is array (1 to N_SLAVES) of
15        slave_mapping_type;
16
17 Constant BUS_OP_FIELDS : integer := 2;
18
19 Constant BUS_READ : std_logic_vector(BUS_OP_FIELDS-1
downto 0) := "01";
20 Constant BUS_WRITE: std_logic_vector(BUS_OP_FIELDS-1
downto 0) := "10";
21 Constant BUS_IDLE : std_logic_vector(BUS_OP_FIELDS-1 downto
0) := "00";
22
23 End GW_package;
```

Descripción de pasarela

```
1 Use GW_package.all;
2
3 IP GW generic (precedent: string := "TDI";
4         following : string := "TDO";
5         tributary : string := "deselected";
6         affluent : string := "deselected";
7         network_mapping : network_mapping_type);
8
9
```

```
10 Begin
11
12 Address_map : REG generic map (precedent =>"TDI",
13         following=>"TDO"
14         elements => ADDRESS_DEPTH);
15 Bus_operation : REG generic map (
16         precedent => "Address_map";
17         following => "TDO";
18         elements => BUS_OP_FIELDS);
19
20 Function get_slave_address(name : in slave_name_type)
21         return slave_address_type is
22 Begin
23 for k in N_SLAVES loop
24 If network_mapping(k).stave_name = name then
25 return network_mapping(k).slave_address;
26 End loop;
27 assert false report "ERROR, slave "&name&" does not exist"
28 severity failure;
29 end get_slave_address;
30
31 Procedure select_tributary(tributary_name :in slave_name_type)
32 Length 10;
33 Selection transaction;
34 {
35 address_map <= get_slave_address(tributary_name);
36 bus_operation <= BUS_WRITE;
37 tributary := tributary_name;
38 }
39
40 Procedure select_affluent(affluent_name : in slave_name_type)
41 Length 10;
42 Selection transaction;
43 {
44 address_map <= get_slave_address(affluent_name);
45 bus_operation <= BUS_READ;
46 affluent := affluent_name;
47 }
48
49 End GW;
```

La descripción basada en NSDL del dispositivo 1000 de arquitectura de bus se ha dividido en dos ficheros para una mejor legibilidad. Especialmente, la descripción basada en NSDL del dispositivo 1000 de arquitectura de bus se divide en las siguientes partes: (1) un paquete que alberga todas las declaraciones tipo para el dispositivo 1000; y (2) una descripción del dispositivo de arquitectura de bus 1000.

La descripción basada en NSDL del dispositivo 1000 de arquitectura de bus no necesita información acerca de la aplicación efectiva del protocolo de bus del dispositivo 1000 de arquitectura de bus; en su lugar, la descripción basada en NSDL solo necesita información acerca de las órdenes necesarias para iniciar las transacciones dentro del dispositivo 1000 de arquitectura de bus. Las transacciones se definen mediante las funciones “select_tributary” y “select_affluent” incluidas en la descripción basada en NSDL.

La función “select_tributary” (líneas 31 – 38) escribe la dirección correcta en el registro correcto (explotando las funciones y tipos personalizados) con el fin de ordenar una operación de “escritura” al bus. La función “select-affluent” (líneas 40 – 47) lee la dirección correcta desde el registro correcto (explotando las funciones y tipos personalizados) con el fin de ordenar una operación de “lectura” desde el bus.

Puesto que el dispositivo 1000 de arquitectura de bus es un dispositivo de cruce “de transacción”, la descripción basada en NSDL del dispositivo 1000 de arquitectura de bus no requiere procedimientos de “seleccionar” o “deseleccionar”; en su lugar, los procedimientos de “seleccionar” y “deseleccionar” se marcan como “transacciones” de manera que la herramienta de prueba sabe que la conexión estará solamente activa una vez, y entonces el afluente y el tributario volverán ajustarse como “deseleccionados”.

Como se describe en el presente documento, los dispositivos de cruce tales como el dispositivo de arquitectura de

bus 1000 (y otros dispositivos de cruce) se encuentra completamente fuera de las capacidades actuales de la norma P1687. La descripción de los dispositivos de cruce tales como el dispositivo 1000 de arquitectura de bus (y otros dispositivos de cruce) solamente es posible gracias al uso de NSDL.

Como se describe en el presente documento, se puede describir un sistema en chip usando NSDL y, asimismo, la descripción al nivel del chip del sistema en chip se puede utilizar mediante una herramienta de prueba para probar el sistema en chip. Una herramienta de prueba ejemplar adaptada para probar un sistema en chip descrito usando NSDL se ilustra y describe en la figura 11. Asimismo, la prueba de un sistema en chip descrito usando NSDL se puede entender mejor en las figuras 12 -20.

La figura 11 ilustra un diagrama de bloque de alto nivel del sistema de prueba del entorno de prueba de la figura 1. Específicamente, ST 120 incluye un procesador 1110, una memoria 1120, una interfaz de entrada/salida (E/S) 1130, y circuitos de soporte 1140. El procesador 1110 coopera con la memoria 1120, la interfaz E/S 1130, y los circuitos de soporte 1140 para proporcionar diversas funciones de prueba ilustradas y descritas en el presente documento.

Como se ha representado en la figura 11, la memoria 1120 almacena recursos adaptados para su uso en la realización de la prueba del sistema. Específicamente, la memoria 1120 almacena una herramienta de prueba 1121, descripciones de recursos de prueba 1122, y datos de prueba 1123 adaptados para su uso en la realización de la prueba del sistema. La memoria 1120 puede almacenar cualesquiera otros programas, descripciones datos y similar que se puedan usar para llevar a cabo la prueba del sistema (denominado como otros 1124).

La herramienta de prueba 1121 controla el sistema de prueba. La herramienta de prueba 1121 puede incluir uno o más procedimientos de prueba. Los procedimientos de prueba se pueden generar mediante uno o más compiladores de prueba. Los procedimientos de prueba se puede ejecutar con el fin de probar uno o más sistemas. La herramienta de prueba 1121 incluye cualesquiera otros procedimientos, programas y similar que se puedan usar para controlar la prueba del sistema.

Las descripciones de recursos de prueba 1122 pueden incluir cualesquiera descripciones adaptadas para su uso en la prueba del sistema, tales como descripciones de componentes, descripciones de sistema, y similar, así como diversas combinaciones de las mismas. Las descripciones de recursos de prueba 1122 pueden incluir descripciones de topología del sistema. Las descripciones de recursos de prueba 1122 pueden incluir cualesquiera otras descripciones que se puedan procesar para su uso en la realización de la prueba del sistema en chip.

Las descripciones de recursos de prueba 1122 pueden incluir una o más bibliotecas de manera que las plantillas de descripción se puedan mantener para diferentes tipos de recursos de prueba (y, de este modo, poder acceder a las mismas y modificarlas si fuese necesario). Las plantillas pueden ser plantillas del nivel de los componentes (por ejemplo, una plantilla para algún IP, una plantilla para algún instrumento, y similar), plantillas de topología del sistema, y similar, así como varias combinaciones de las mismas.

Los datos de prueba 1123 incluyen cualesquiera datos adaptados para su uso en la realización de la prueba de sistema. Los datos de prueba 1123 pueden incluir datos de flujos de bits de entrada, datos de flujos de bits de salida (por ejemplo flujos de bits de salida esperados determinados por el sistema de prueba y los flujos de bits de salida reales capturados a partir del sistema en chip), y similar, así como varias combinaciones de los mismos. Los datos de prueba 1123 pueden incluir cualesquiera otros datos que se puedan aplicar a un sistema que se está siendo probado y/o recuperado a partir de un sistema que se están probando.

La interfaz E/S 1130 proporciona una interfaz a partir de ST 120 al sistema en chip 110. La interfaz E/S 1130 es una interfaz basada en JTAG. La interfaz E/S 1130 soporta una interfaz TDI mediante la cual ST 120 puede aplicar flujos de bits de entrada al sistema en chip 110 en respuesta a la ejecución de un procedimiento de prueba por el procesador 1110. La interfaz E/S 1130 soporta una interfaz TDO mediante la cual ST 120 puede recuperar los flujos de bits de salida reales del sistema en chip 1110 en respuesta a la ejecución de un procedimiento de prueba por el procesador 1110.

Aunque principalmente en el presente documento la descripción e ilustración se realiza respecto de una interfaz TDI y una interfaz TDO, la interfaz E/S 1130 puede soportar cualquier número y tipo o tipos de interfaces de prueba requerido o deseado para probar diferentes configuraciones de sistema en chip. Por ejemplo, para la prueba basada en JTAG, la interfaz E/S puede también soportar interfaces para señales TCK, señales TMS y opcionalmente señales TRST.

Los circuitos de soporte 1140 incluyen cualesquiera circuitos adicionales que se puedan usar en la realización de la prueba de sistema. Por ejemplo, los circuitos de soporte 1140 pueden incluir procesadores adicionales, memoria adicional, interfaces adicionales, circuitos generadores de flujos de bits de prueba, circuitos procesadores de flujos de bits de prueba, y similar, así como varias combinaciones de los mismos. Los circuitos de soporte 1140 incluyen cualesquiera circuitos adicionales que puedan ser requeridos por el sistema de prueba 120.

El procesador 1140 coopera con la memoria 1120 , la interfaz E/S 1130 y los circuitos de soporte 1140 para proporciona varias funciones de prueba del sistema en chip descritas en el presente documento.

El procesador 1140 genera descripciones (por ejemplo descripciones al nivel de las funciones, descripciones de componentes y similar). El procesador 1140 procesa las descripciones procedentes de las descripciones de recursos de prueba 1122 con el fin de generar descripciones (por ejemplo usar descripciones de componentes para analizar interconexiones de componentes, para generar descripciones de sistema y similar). El procesador 1140 almacena las descripciones como parte de las descripciones de recursos de prueba 1122.

El procesador 1140 genera procedimientos de prueba para probar un sistema en chip y almacena los procedimientos de prueba como parte de la herramienta de prueba 1121. El procesador 1140 genera datos de prueba que usan procedimientos de prueba procedentes de la herramienta de prueba 1121 y almacena los datos de prueba como parte de los datos de prueba 1123.

El procesador puede cooperar con la memoria 1120, la interfaz E/S 1130 y los circuitos de soporte 1140 para proporcionar cualesquiera otras funciones de prueba de sistema en chip descritas en el presente documento.

La prueba de un sistema en chip que usa una descripción basada en NSDL del sistema en chip se puede entender mejor con las figura 12-14 que proporcionan procedimientos para probar un sistema en chip que usan una descripción basada en NSDL del sistema en chip. La prueba de un sistema en chip que usa una descripción basada en NSDL del sistema en chip se puede entender con las figuras 15 y 16 (que proporcionan un ejemplo que ilustra la prueba de un componente de un sistema en chip), la figura 17 (que proporciona un procedimiento para probar un componente de un sistema en chip), y las figuras 18 - 20 (que proporcionan un ejemplo que ilustra la prueba de un sistema en chip).

La figura 12, ilustra un procedimiento ejemplar ejecutado por el sistema de prueba de la figura 1, para probar un sistema a través de una conexión JTAG. Aunque la ilustración y descripción se ha realizado en serie, al menos una parte de las etapas del procedimiento 1200 de la figura 12 se puede llevar a cabo de manera simultánea o en un orden distinto del ilustrado y descrito en la figura 12. El procedimiento 1200 empieza en la etapa 1202 y continúa hasta la etapa 1204.

En la etapa 1204, se determinan los flujos de bits de prueba para el sistema en chip. Los flujos de bits de prueba incluyen un flujo de bits de entrada y un flujo de bits de salida esperado. Los flujos de bits de prueba se pueden determinar de cualquier manera descrita en el presente documento. En una realización, los flujos de bits de prueba se determinan usando el procedimiento ilustrado y descrito en la figura 13.

En la etapa 1206, el flujo de bits de entrada se aplica al sistema en chip. El flujo de bits de entrada se puede aplicar al sistema en chip por la interfaz TDI del sistema en chip. En la etapa 1208, se captura un flujo de bits de salida real procedentes del sistema en chip. El flujo de bits de salida real se puede capturar a partir del sistema en chip mediante la interfaz TDO del sistema en chip.

En la etapa 12 10 se determinan los resultados de prueba usando el flujo de bits de salida real y el flujo de bits de salida esperado. Los resultados de prueba se pueden determinar comparando el flujo de bits de salida real y el flujo de bits de salida esperado (por ejemplo, para determinar si hay o no muchos errores durante la prueba). En la etapa 1212 los resultados de prueba se almacenan.

En la etapa 1214, se termina el procedimiento 1200.

La figura 13 ilustra un procedimiento ejemplar ejecutado por el sistema de prueba de la figura 1, para probar un sistema a través de una conexión JTAG. Específicamente, el procedimiento 1300 de la figura 13 incluye un procedimiento para determinar flujos de bits de prueba para su uso en la prueba del sistema en chip. Aunque la ilustración y descripción se ha llevado a cabo en serie, al menos una parte de las etapas del procedimiento 1300 de la figura 13 se puede realizar simultáneamente, o en un orden distinto al ilustrado y descrito en la figura 13. El procedimiento 1300 empieza en la etapa 1302 y continúa hasta la etapa 1304.

En la etapa 1304, se determina una descripción del sistema en chip (indicado aquí como una descripción de sistema). La descripción de sistema se puede determinar de cualquier manera descrita en el presente documento. La descripción de sistema es una descripción basada en NSDL del sistema en chip. En una realización la descripción de sistema se determina usando el procedimiento ilustrado y descrito en la figura 14.

En la etapa 1306, los flujos de bits de prueba para probar el sistema en chip se determinan a partir de la descripción de sistema del sistema en chip. Los flujos de bits de prueba incluyen el flujo de bits de entrada a aplicar al sistema en chip y un flujo de bits de salida esperado que se puede comparar con un flujo de bits de salida real capturado en el sistema en chip

En la etapa 1308, se termina el procedimiento 1300.

La figura 14 ilustra un procedimiento ejemplar ejecutado por el sistema de prueba de la figura 1 para probar un sistema a través de una conexión JTAG. Específicamente, el procedimiento 1304 de la figura 14 incluye un procedimiento para determinar una descripción de sistema del sistema en chip. Aunque la ilustración y la descripción se ha llevado a cabo en serie, al menos una parte de las etapas del procedimiento 1304 de la figura 14 se pueden llevar a cabo simultáneamente, o en un orden distinto del ilustrado y descrito en la figura 14. El procedimiento 1304 empieza en la etapa 1402 y continúa hasta la etapa 1404.

En la etapa 1404, se identifican los componentes del sistema en chip. Los componentes del sistema en chip se pueden identificar de cualquier manera. En una realización, los componentes del sistema en chip se identifican como parte de los recursos de prueba suministrados con el sistema en chip. En una realización, los componentes del sistema en chip se identifican analizando el sistema en chip. Los componentes del sistema en chip se pueden identificar de cualquier otra manera.

En la etapa 1406, se determina una descripción de componentes para cada uno de los componentes del sistema en chip. Las descripciones de componentes se pueden determinar de cualquier manera.

En una realización, en la cual se predefinen las descripciones de componentes, las descripciones de componentes se pueden determinar leyendo simplemente las descripciones predefinidas. En una realización en la cual no se predefinen las descripciones de componentes, las descripciones de componentes se pueden definir por la vía rápida analizando cada uno de los componentes.

La descripción de un componente especifica la ruta exploración interna del componente. En una realización, la descripción de un componente representa el componente en términos de valores de registro. En una realización de este tipo, en la que el componente soporta múltiples funciones, cada una de las funciones del componente se puede determinar en términos de valores de registro. La descripción de un componente en términos de valores de registro se puede entender mejor con la figura 15. Las descripciones de componente se especifican usando NSDL.

En la etapa 1408, se determina la topología del sistema en chip. La topología del sistema en chip describe interconexiones entre componentes del sistema en chip. La topología del sistema en chip se determinan analizando las interconexiones entre los componentes del sistema en chip.

En la etapa 1410, se determina la descripción del sistema en chip. La descripción del sistema se determina usando las descripciones de componentes y la topología de sistema. La descripción de sistema del sistema en chip representa la ruta de exploración del sistema en chip, incluyendo rutas de exploración interna de cada uno de los componentes respectivos del sistema en chip y las interconexiones entre los componentes del sistema en chip. La descripción de sistema puede incluir cualquier otra información que se pueda usar para describir el sistema en chip. En la etapa 1412, se almacena la descripción de sistema del sistema en chip.

En la etapa 1414, se termina el procedimiento 1304.

La figura 15 ilustra el uso de una descripción de uno de los componentes del sistema en chip de la figura 2 para determinar valores de registro para un procedimiento de prueba para probar ese componente. Específicamente, se convierte una descripción de componente 210_A del sistema en chip 110 (omitido por razones de claridad) en un conjunto de valores de registro (indicados como valores de registro 1510) para el componente 210_A del sistema en chip 110. Los valores de registro 1510 del componente 210_A incluyen valores de registro para cada una de las tres funciones soportadas por el componente 210_A. Los valores de registro para cada una de las tres funciones soportadas por el componente 210_A se pueden procesar para determinar valores para probar flujos de bits usados para probar el componente 210_A.

Como se ilustra en la figura 15, se describen las funciones del componente 210_A en términos de valores de registro de registros de componente A_0, A_1, A_2 de los cuales se compone el componente 210_A. La primera función se define como: "000" esperar 4 ciclos, "001". La segunda función se define como "111", esperar 1 ciclo, "010", esperar 5 ciclos, "101". La tercera función se define como "000", esperar 2 ciclos, "100", esperar 10 ciclos, "000". dicho de otro modo, los valores de registro 1510 especifican representaciones desde las funciones hasta los valores de registro de componente par el componente 210_A. Esta descripción (es decir, la representación desde las funciones hasta los valores de registro) muestra la facilidad con la cual se puede procesar una descripción de un componente con el fin de determinar flujos de bits de prueba.

Como se ha ilustrado en la figura 15, para cada una de las funciones del componente 210_A, los valores de registro 1510 indican; (1) la manera en que los registros de componente se escriben y leen para la función, y (2) la manera en que los valores de registro de componente se deben interpretar para la función. De este modo, en un lenguaje de descripción, tal como NSDL, la descripción del componente incluye descripciones de las funciones del componente, lo cual puede a su vez convertirse en valores de registro para su uso en un procedimiento de prueba para probar el componente dentro del contexto del sistema en chip.

Aunque se omite por razones de claridad en la descripción de la representación de las funciones a los valores de registro, la descripción específica del componente 210_A en términos de NSDL es una descripción algorítmica, compuesta tal como se ilustra y describe en las figuras 3 - 5. Esta descripción basada en NSDL del componente 210_A permite la representación desde las funciones a los valores de registro a determinar. Aunque se omiten por razones de claridad, se pueden definir descripciones similares para cada uno de los otros componentes del sistema en chip 110 (es decir, componentes 210_B - componente 210_E).

La figura 16 ilustra el uso de una descripción de la composición del sistema en chip de la figura 2 para determinar flujos de bit de prueba para un procedimiento de prueba para probar uno de los componentes del sistema en chip de la figura 2. Como se ha ilustrado en la figura 16, se generan flujos de bits de prueba 1610 para su uso en la prueba al menos de una parte del sistema en chip 110. Específicamente, los flujos de bits de prueba 1610 incluyen un flujo de bits de entrada 1611, aplicado al puerto TDI del sistema en chip 110 y un flujo de bits de salida 1611o recibido procedente del puerto TDO del sistema en chip 110.

Los flujos de bits de prueba 1610 se generan mediante el ST 120 usando una descripción del sistema en chip 110 (especificada usando NSDL). La descripción del sistema en chip 110 (también indicada en el presente documento como descripción de sistema) incluye descripciones de los recursos de prueba del sistema en chip 110 (por ejemplo, descripciones de componentes 210, descripciones de interconexiones de componentes 220, y similar, así como varias combinaciones de los mismos). La descripción de sistema del sistema en chip 110 describe la topología del sistema en chip 110 y, de este modo, la ruta de exploración del sistema del sistema en chip 110.

Como se ha descrito en el presente documento, ST 120 genera flujos de bits de prueba 1610 basado en la descripción del sistema (que proporciona una descripción de la ruta de exploración del sistema) del sistema en chip 110. De este modo, puesto que la descripción de sistema del sistema en chip 110 proporciona una descripción de la ruta de exploración del sistema del sistema en chip 110, ST 120 es capaz de determinar qué partes de los flujos de bits de prueba 1610 generadas por el sistema en chip 110 corresponden a qué partes de la ruta de exploración del sistema del sistema en chip 110. Esto se ilustra en la figura 16.

Como se ilustra en la figura 16, se han localizado las posiciones de bits especificas dentro del flujo de bits de entrada 1611_1, y el flujo de bits de salida 1611_O (es decir las posiciones de bits que corresponden al componente 210_A). Se permite la localización de las posiciones específicas dentro del flujo de bits de entrada 1611_1 y el flujo de bits de salida 1611_O que corresponden a registros dentro del componente 210_A usando una descripción basada en NSDL del sistema en chip 110 (que proporciona información acerca de la ruta de exploración de sistema del sistema en chip 110, que incluye localizaciones respectivas de componentes 210 dentro del sistema en chip 110).

Aunque se omite por razones de claridad en la descripción de la conversión de registros de componente en flujos de bits, la descripción específica del sistema en chip 110 en términos de NSDL es una descripción algorítmica que describe la topología del sistema en chip 110 y, de este modo, la ruta de exploración de sistema del sistema en chip 110. Esta descripción basada en NSDL del sistema en chip 110 permite la realización de la conversión de valores de registro en flujos de bits.

La figura 17 ilustra un procedimiento ejemplar ejecutado por el sistema de prueba de la figura 1 para probar un componente de un sistema en un marco IJTAG/NSDL. Específicamente, el procedimiento 1700 de la figura 17 incluye un procedimiento para probar un componente del sistema en chip. Aunque se ilustran y describen en serie, al menos una parte de las etapas de del procedimiento 1700 de la figura 17 se pueden llevar a cabo simultáneamente, o en un orden distinto del ilustrado y descrito en la figura 17. El procedimiento 1700 empieza en la etapa 1702 y continúa hasta la etapa 1704.

En la etapa 1704, se selecciona un componente de un sistema en chip (es decir, se selecciona como el componente del sistema en chip que se probará).

En la etapa 1706, se obtiene una descripción del componente seleccionado.

En la etapa 1708, para cada función soportada por el componente, la función se convierte en valores de registro asociados a los registros del componente. Las funciones del componente se convierten en valores de registro usando la descripción del componente.

En la etapa 1710, se obtiene una descripción de sistema del sistema en chip. La descripción de sistema del sistema en chip especifica la ruta de exploración de sistema del sistema en chip, determinada a partir de las descripciones de componentes del sistema en chip y una descripción de la topología del sistema en chip. La descripción de sistema se puede usar para especificar flujos de bits de prueba para el sistema en chip.

En la etapa 1712, la posición del componente seleccionado dentro de los flujos de bits de prueba del sistema en chip se determina usando la topología del sistema en chip. La posición del componente seleccionado dentro de los flujos

de bits de prueba especifica una o más posiciones de bits dentro del flujo de bits de entrada y uno o más posiciones de bits dentro del flujo de bits de salida real.

En la etapa 1713 (una etapa opcional), se acciona un dispositivo de cruce (es decir, si el acceso al componente es controlado por un dispositivo de cruce). El dispositivo de cruce se acciona procesando una descripción algorítmica del dispositivo de cruce. El accionamiento del dispositivo de cruce permite seleccionar el dispositivo de cruce con el fin de añadir de manera dinámica el componente asociado a la ruta de exploración del sistema en chip. Después de acceder al componente ya no es necesario, el dispositivo de cruce se puede entonces deseleccionar para eliminar el componente asociado de la ruta de exploración del sistema en chip. Esta etapa es opcional porque el acceso a un componente puede ser controlado o no por un dispositivo de cruce.

En la etapa 1714, se insertan los valores de registro en la posición localizada del flujo de bits de entrada. El flujo de bits de entrada se pueden entonces aplicar a un puerto de acceso de prueba de entrada del sistema en chip (es decir para probar al menos una parte del sistema en chip). En la etapa 1716, se recuperan los valores resultantes de la posición localizada del flujo de bits de salida (es decir del flujo de bits de salida capturado por el sistema de prueba a partir de una parte de acceso de prueba de salida del sistema en chip). Los valores resultantes recuperados se pueden entonces procesar con el fin de determinar varios resultados de prueba.

En la etapa 1718, se termina el procedimiento 1700.

La figura 18 ilustra un diagrama de bloques de alto nivel de un sistema en chip ejemplar. Como se ilustra en la figura 18, el sistema en chip 1800 incluye un filtro y tres instrumentos, incluyendo un conversor analógico-digital (CAD), un conversor digital-analógico (CDA), y una unidad de acceso de antena (UAA). La UAA soporta la conversión analógica-digital analógica. El filtro soporta una conexión de salida al CDA (indicado como RX_out) y una conexión de entrada desde el CAD (indicada como TX_in). El filtro soporta una conexión bidireccional con la UAA (indicada AN_inout).

Como se describe en la figura 18, el acceso a los tres instrumentos es proporcionado por una ruta de exploración desde una entrada TDI a una salida TDO. La ruta de exploración incluye un conjunto de registros de accionador y tres células SIB (una para cada uno de los tres instrumentos que permiten añadir los instrumentos respectivos a la ruta de exploración). Los registros de accionador se interconectan con el filtro. Una primera célula SIB proporciona acceso al instrumento CAD (por un afluente y un tributario). Una segunda célula SIB proporciona acceso al instrumento CDA (por un afluente y un tributario). Una tercera célula proporciona acceso al instrumento UAA (por un afluente y un tributario).

A continuación se realización una descripción de la ruta de exploración. La entrada TDI se acopla a la entrada del conjunto de registro de accionador. La salida del conjunto de registros de accionador se acopla a la entrada de la primera célula SIB. La salida de la primera SIB se acopla a la entrada de la segunda célula SIB. La salida de la segunda célula SIB se acopla a la entrada de la tercera célula SIB. La salida de la tercera célula SIB se acopla a la salida TDO. Esta secuencia de la entrada TDI a la salida TDO forma la trayectoria de exploración del sistema en chip 1800 cuando se deselecciona cada una de las SIBs (es decir, si no se "selecciona" ninguna de las SIBs de manera que sus instrumento respectivos se añaden a la trayectoria de exploración).

Como se ha descrito en la figura 18, cada uno de los tres instrumentos se puede añadir fácilmente a la ruta de exploración seleccionando las interfaces de afluente y tributario de las células SIB respectivas asociadas a los tres instrumentos. Por ejemplo, la primera SIB se puede seleccionar de manera que el instrumento CAD se pueda añadir a la ruta de exploración para su prueba. Por ejemplo, la primera SIB y la tercera SIB se pueden seleccionar de manera que los instrumentos CAD y UAA se pueden añadir ambos para a la ruta de exploración para su prueba. Como se describe en el presente documento la descripción basada en NSDL del sistema en chip 1800 simplifica la prueba del sistema en chip 1800.

La figura 19 ilustra el uso de una descripción de uno de los componentes del sistema en chip de la figura 18 para determinar valores de bits de registro para un procedimiento de prueba para probar ese componente. Específicamente, la figura 19 ilustra una representación de una descripción del instrumento CAD (indicado como descripción 1910) para registrar valores asociados a registros del instrumento CAD (indicados como registros 1920). Como se ilustra en la figura 19, la descripción 1910 incluye una descripción de la ruta de exploración (indicada como descripción de ruta de exploración 1911) y una representación algorítmica de la función realizada por el instrumento CAD (indicada como función algorítmica 1912).

La descripción 1910 es una descripción (algorítmica) basad en NSDL). La descripción de ruta de exploración 1911 identifica los registros del instrumento CAD (que incluye registros de datos y registros de control), y describe la manera en que se disponen los registros (por ejemplo, identificar "precedente", siguiente, e información de longitud). La función algorítmica 1912 describe la operación de la función realizada por el instrumento CAD. A partir de la descripción 1910, una herramienta de prueba puede determinar fácilmente valores de flujo de bits que se pueden usar para probar el instrumento CAD.

La figura 20 ilustra el uso de descripciones de los componentes del sistema en chip de la figura 18 para determinar una descripción del sistema en chip de la figura 18. Específicamente, la figura 20 ilustra una descripción de la topología del sistema en chip de la figura 18 (indicada como topología 2010). La topología 2010 se especifica usando una representación genérica de cada una de las entrada y salida de cada uno de los componentes del sistema en chip 1800 (por ejemplo los registros de accionador, SIBs, e instrumentos).

Por ejemplo, la descripción de los registros de accionador indican que la entrada a los registros de accionador es la entrada TDI ("precedent"=>'TDI') y que la salida de los registros de accionador es la primera SIB, que se indica como RX_enable, porque permite el acceso al RX-register de los instrumento CAD (following => "RX:enable"). Dicho de otro modo, se describe el componente de registros de accionador en términos de los componentes con los que se interconecta.

Por ejemplo, la descripción del primer SIB (es decir, el componente RX_enable) indica que la entrada al componente RX_enable son los registros de accionador (precedent =>"actuator_registers") y la salida de la primera SIB es la segunda SIB, que se indica como TX_enable porque permite el acceso al TX_register del instrumento CAD ("following" => "Tx_enable").

Asimismo, puesto que la primera SIB permite el acceso al instrumento CAD, la descripción de la primera SIB también describe el acceso al instrumento CAD. Específicamente, la descripción de la primera SIB indica que el acceso desde la primera SIB al instrumento CAD se permite a través de una entrada (affluent =>"Rx"_register) y una salida (tributary=>"Rx_enable").

A partir de estos ejemplos, está claro que toda la topología del sistema en chip se puede describir fácilmente usando NSDL. Asimismo, las localizaciones de cada uno de los componentes del sistema en chip dentro de la topología del sistema en chip se puede determinar fácilmente ya que la topología proporciona una descripción de interconexiones entre los componentes del sistema en chip.

Como se ha descrito anteriormente, usando las descripciones de los componentes del sistema en chip (que proporcionan representaciones desde las funciones de los componentes a los valores de registro para registros de los componentes) y la descripción de la topología del sistema en chip, se pueden ejecutar varios tipos de pruebas sobre el sistema en chip. De este modo, las figuras 18-20 ilustran ventajas de la descripción de un sistema en chip usando NSDL.

A partir de las descripciones anteriores, las diversas ventajas y beneficios del uso del lenguaje NSDL para describir un sistema en chip son evidentes. La naturaleza algorítmica de NSDL permite una descripción algorítmica de un sistema en chip (no importa la complejidad de la topología) que se puede procesar para determinar flujos de bits de prueba adaptados para probar el sistema en chip. (por ejemplo, probar múltiples componentes, probar un componente, probar un subconjunto de funciones de un componente, y similar, así como varias funciones de los mismos). De este modo, usando NSDL, se puede determinar una descripción jerárquica de la ruta de exploración de sistema de un sistema en chip para su uso en la prueba del sistema en chip.

El lenguaje NSDL soporta descripciones de IJTAG de manera que BSDUHSDL no puede simplemente incluir la provisión de descripciones algorítmicas de componentes de sistema en chip (por ejemplo IPs, instrumentos, dispositivos de cruce, etc.) topología de sistema en chip (por ejemplo, interconexiones entre componentes, dependencias intercomponentes, etc.) y similar, así como varias combinaciones de los mismos, permitiendo de este modo una descripción algorítmica al nivel de sistema del sistema en chip. De este modo, el lenguaje NSDL permite la organización jerárquica de la ruta de prueba, permitiendo de este modo la conversión de valores de registro en flujos de bits de prueba que se pueden usar para probar el sistema en chip.

El lenguaje NSDL se puede aplicar de muchas maneras.

En una realización, NSDL se realiza usando características existentes de VHDL. Mientras BSDL se definió como un subconjunto de VHDL, NSDL utiliza un sobre conjunto de VHDL. Puesto que VHDL es un lenguaje de descripción principal del Nivel de Transferencia de Registros (RTL) , es muy apropiado para expresar los requisitos de prueba de los componentes que describe. Al seguir siendo compatible con VHDL, NSDL puede ser soportado por los compiladores existentes con cambios mínimos. Asimismo, el uso de VHDL garantiza que la transición a NSDL será una experiencia suave para las personas acostumbradas a VHDL y garantiza, asimismo, que la conversión y la adaptación de las fuentes y herramientas existentes será fácil. De este modo como NSDL utiliza VHDL, se minimiza el impacto sobre la comunidad existente de usuarios, simplificando de este modo la adopción de NSDL por la comunidad existente de usuarios.

Por el contrario, BSDL se desarrolló como un subconjunto de VHDL en un esfuerzo por hacer que BSDL sea compatible con versiones anteriores y posteriores de VHDL. Un subconjunto indica que no se añade nada (es decir, toda la información es llevada por normas sintácticas estructurales integradas en las normales de VHDL preexistentes, que se interpretan entonces de una manera diferente a la normal. Mientras la la retrocompatibilidad es

automática, la imposibilidad de definir nuevas construcciones hace difícil la evolución. Esta limitación estructural lleva a los constructores a un uso excesivo de las dos construcciones más genérica /atributos y secuencias de caracteres), a menudo de una manera contraintuitiva. Dicho de otro modo, BSDL se limitó efectivamente dentro de una pequeña parte de VHDL, eliminando de este modo cualquier posibilidad de que VHDL tenga significado contextual.

Como se ha descrito en el presente documento, NSDL proporciona muchas ventajas en la prueba de sistema en chip. Asimismo, usando VHDL para implementar NSDL se proporcionan muchas ventajas en la prueba del sistema en chip.

Los componentes de un sistema en chip se pueden modelizar sobre la frase de componentes de entidad VHDL, permitiendo de este modo descripciones de funciones individuales soportadas por componentes del sistema en chip. De este modo se puede proporcionar el acceso a cada componente a través de la ruta de exploración interna del componente, que puede incluir jerarquía. Asimismo, un componente puede incluir un conjunto correspondiente de procedimientos de prueba que un sistema de prueba puede usar para probar el componente, donde se especifican las propiedades de los procedimientos usando NSDL (por ejemplo, longitud, dependencias y similar) mientras que los cuerpos de los procedimientos se especifican usando VHDL. De este modo, se permite que los arquitectos de sistema, manejen descripciones al nivel de sistema de una manera similar a la manera en que los arquitectos de componente manejan descripciones al nivel de los componentes.

Se puede componer la ruta de exploración de un sistema en chip como una serie de entidades que se instancian como componentes VHDL y de este modo, se pueden agrupar fácilmente en paquetes o bibliotecas. Asimismo, se pueden manejar fácilmente representaciones mucho más complicadas (por ejemplo, múltiples instanciaciones de un tipo dado de componente, dependencias intercomponentes, y similar, así como varias combinaciones de los mismos). Asimismo, NSDL permite la construcción de la cadena de exploración del sistema de una manera que es una mejora respecto de la representación clásica de señales de VDL. Los procedimientos de prueba desarrollados para probar un sistema en chip pueden referenciar la ruta de exploración del sistema o partes de la ruta de exploración del sistema (denominados rodajas, que pueden incluir grupos de componentes o incluso un subconjunto o subconjunto de un componente.

Aunque se ilustra y describe principalmente en el presente documento respecto de las realizaciones en las que se aplica NSDL usando VHDL, NSDL se puede aplicar usando otras lenguajes de descripciones de hardware (que pueden incluir lenguajes de descripción de hardware que aun no se han desarrollado).

Como se ha descrito en el presente documento, , además de mejorar la prueba basada en JTAG, así como permitir el uso de dispositivos de cruce en la prueba basada en JTAG, el lenguaje NSDL también permite el acceso en paralelo a componentes de un sistema en chip, permitiendo de este modo mejorar en la prueba del sistema en chip (por ejemplo, programación de prueba mejorada, eficiencia de prueba mejorada y similar, así como varias combinaciones de las mismas).

En una realización, tal como P1687, el acceso en paralelo se destina principalmente como manera de optimizar el ancho de banda para la transmisión de datos, mientras el acceso en serie sigue manteniendo el control de la prueba. El lenguaje NSDL puede describir estos recursos auxiliares e insertarlos en el flujo de prueba (por ejemplo, en los procedimientos 1200 y 1700 ilustrado y descritos respecto de las figuras 12 y 17, respectivamente).

Asimismo, NSDL se puede expandir para describir Mecanismos de Acceso de Prueba (TAMs) más complejos, por ejemplo, usando dispositivos de cruce para proporcionar acceso en paralelo para proporcionar un sistema en chip, usando esquemas de factores de entrada en abanico y factores de salida en abanico para proporciona acceso en paralelo para probar un sistema en chip, y similar así como varias combinaciones de los mismos

El uso de acceso en paralelo para probar un sistema en chip (o múltiples sistemas en chip) se puede entender mejor con las figuras 21- 28.

La figura 21 ilustra un diagrama de bloques de alto nivel de un esquema general de conexiones de una interfaz de acceso en paralelo. específicamente, el esquema general de conexiones 2100 proporciona una interfaz de acceso en paralelo 2110 a un sistema en chip 2120. La interfaz de acceso en paralelo 2110 incluye un puerto paralelo interno 2111, un puerto paralelo externo 2112, y una interfaz interna 2113. El sistema en chip 2120 incluye un puerto paralelo 2121.

Como se ha ilustrado en la figura 21, se proporciona el acceso en paralelo al sistema en chip 2120 mediante una conexión entre el puerto paralelo interno 2111 y el puerto paralelo externo 2121. La explotación del puerto paralelo 2121 requiere: (1) conexión y sincronización del puerto paralelo 2121 al sistema en chip 2120 (que se puede realizar en el momento de la instanciación) y (2) proceso del puerto paralelo 2121 por un sistema de prueba (omitido por razones de claridad).

Como se ilustra en la figura 21, el puerto paralelo interno 2111 y el puerto paralelo externo permite n conexiones de entrada (n>0) y , conexiones de salida (m>0) se conecten al puerto paralelo 2121 del sistema en chip 2120.

El acceso en paralelo al sistema en chip 2120 se proporciona de dos maneras.

Se proporciona el acceso en paralelo al sistema en chip 2120 de manera externa. Se proporciona el acceso externo desde un sistema de prueba al sistema en chip 2120 usando el puerto paralelo externo 2112. El puerto paralelo externo 2112 funciona como una interfaz entre un sistema de prueba y un puerto paralelo interno 2111. El puertoparalelo externo 2112 soporta n conexiones de entrada y m conexiones de salida (que corresponden an/m connexiones de entrada/salida del puerto paralelo interno, respectivamente).

Se dispone internamente acceso en paralelo al sistema en chip 2120. Se dispone acceso interno desde un sistema de prueba al sistema en chip 2120 usando la interfaz interna 2113. En una realización, la interfaz interna 2113 se puede implementar usando uno o más registros internos conectados a la ruta de exploración del sistema. En esta realización, los registros internos se pueden usar para controlar el comportamiento de la interfaz de acceso en paralelo 2110 o para preguntar el estado de la interfaz de acceso en paralelo 2110.

Como se ha establecido anteriormente, el sistema de prueba que accedería al puerto paralelos interno de acceso 2111 por la interfaz interna 2112 o la interfaz externa 2113 se omite por motivos de claridad.

La interfaz de acceso en paralelo 2110 se describe usando NSDL.

La descripción de NSDL de interfaz de acceso en paralelo 2110 incluye : (1) una descripción del o los puertos internos (por ejemplo, información de anchura, direcciones de flujo de datos, e información similar) y (2) funciones/procedimientos de acceso en paralelo.

La descripción de NSDL de la interfaz de acceso en paralelo 2110 también puede incluir opcionalmente : (3) una descripción del puerto paralelo externo 2112 (por ejemplo información de anchura, direcciones de flujo de datos, e información similar)

La descripción de NSDL de la interfaz de acceso en paralelo 2110 también puede incluir opcionalmente: (4) una descripción de la interfaz interna 2113 (por ejemplo una descripción de registros utilizados para funciones de control y/o funciones de estado).

En una realización, el acceso en paralelo desde un sistema de prueba a un sistema en chip que tiene una pluralidad de componentes se puede describir usando una descripción de una interfaz de acceso de prueba en serie para acoplar flujos de bits de prueba entre el sistema de prueba y los componentes (donde la interfaz de acceso de prueba en serie proporciona acceso a los componentes usando una ruta de exploración en serie del sistema en chip), y una descripción de una interfaz de acceso de prueba en paralelo para acoplar flujos de bits de prueba entre el sistema de prueba y los componentes (donde la interfaz de acceso de prueba en paralelo proporciona acceso a los componentes sin usar una ruta de exploración en serie del sistema en chip, es decir, no directamente por la ruta de exploración en serie, aunque el acceso se puede controlar por uno o más valores de la ruta de exploración en serie). Las descripciones de la interfaz de acceso de prueba en serie y la interfaz de acceso de prueba en paralelo se pueden almacenar para su uso con fines de prueba.

En una realización, el acceso en paralelo de un sistema de prueba a un sistema en chip se puede describir usando la descripción de un módulo de interfaz en paralelo adaptado para acoplar el sistema de prueba a un módulo de núcleo del sistema en chip y almacenar la descripción del módulo de interfaz en paralelo, donde la interfaz en paralelo incluye al menos un registro en serie adaptado para acceder al módulo de núcleo usando una ruta de exploración del sistema en chip y al menos un registro en paralelo adaptado para acceder al módulo de núcleo sin usar la ruta de exploración del sistema en chip. La descripción se puede almacenar para su uso con fines de prueba.

En una realización, el acceso en paralelo desde un sistema de prueba a un sistema en chip se puede describir usando una descripción de un puerto de acceso de prueba en serie adaptado para acoplar flujos de bits de prueba entre el sistema de prueba y los componentes, una descripción de un puerto de acceso de prueba en paralelo adaptado para acoplar flujos de bits de prueba entre el sistema de prueba y los componentes, y una descripción de un puerto de interfaz adaptado para acoplar el puerto de acceso de prueba en serie y el puerto de acceso de prueba en paralelo a al menos una parte de los componentes del sistema en chip. Las descripciones se pueden almacenar para su uso en la prueba del sistema en chip.

Como se ha descrito en el presente documento, se almacenan las descripciones que se generan. Asimismo, las descripciones pueden ser recibidas por un procesador (por ejemplo, desde una memoria, desde otro sistema, o desde cualquier otra fuente de tales descripciones) con el fin de llevar a cabo diversas pruebas que usan acceso en paralelo (por ejemplo pruebas al nivel de los componentes, pruebas al nivel del sistema, y similar, así como varias combinaciones de los mismos).

El acceso en paralelo desde un sistema de prueba a un sistema en chip se puede describir de otras maneras.

Las comunicaciones entre la interfaz en paralelo 2110 y el sistema en chip 2120 pueden ser síncronas o asíncronas.

En una realización, las comunicaciones entre la interfaz en paralelo 2110 y el sistema en chip 2120 se sincronizan con la cadena de exploración. En una realización de este tipo, en el flanco ascendente de la señal de "actualización" 1149.1, se muestrea el o los valores en el puerto paralelo 2121. en esta realización, el sistema de prueba tiene meramente que presentar el o los valores al puerto (para las entradas y/o recoger el o los valores del puerto (para las salidas).

En una realización, las comunicaciones entre la interfaz en paralelo 2110 y el sistema en chip 2120 se llevan a cabo en forma de ráfaga sincronizada. En una realización de este tipo, para optimizar el ancho de banda, se envía una ráfaga de datos al puerto paralelo 2121 (en la entrada) y/o se lee a partir del puerto paralelo 2121 (en la salida). En una realización de este tipo, los datos de ráfaga se pueden enviar/leer que empiezan en el flanco ascendente de la señal de "actualización" 1149.1. La interfaz en paralelo 2110 especificará las características de la ráfaga de datos al sistema de prueba.

En una realización, las comunicaciones entre la interfaz en paralelo 2110 y el sistema en chip 2120 son síncronas. En esta realización, el puerto en paralelo 2121 funciona por su cuenta, gestionando su propio protocolo para hacer que se utilice la conexión. En esta realización, el sistema de prueba permitirá solamente un acceso de alto nivel (enviando datos y recibiendo datos). El protocolo de la transacción es gestionado por una controlador de interfaz en paralelo del sistema de prueba.

En una realización, el sistema en chip 1220 puede soportar múltiples modos de comunicación de este tipo con interfaz en paralelo 2110. En tales realizaciones, se puede usar un conjunto de funciones para conmutar entre los diferentes modos de comunicación. Por ejemplo, el conjunto de funciones puede incluir "disable_port" que deshabilita el puerto en paralelo; "set_scan_synchro", que conmuta al modo de acceso sincronizado de cadena de exploración; "set_burst", que conmuta al modo de acceso de ráfaga; y "set_asynchro" que conmuta al modo asíncrono.

La figura 22 ilustra un diagrama de bloques de nivel alto que ilustra dos esquemas ejemplares de conexión de acceso en paralelo.

Como se ha representado en la figura 22, los esquemas ejemplares de conexión de acceso en paralelo se describen dentro del contexto del entorno de prueba de la figura 1. Los esquemas ejemplares de conexión de acceso en paralelo se utilizan para conectar el sistema de prueba 120 al sistema en chip 110. Específicamente, los esquemas ejemplares de conexión de acceso en paralelo se utilizan para conectar el sistema de prueba 120 a una interfaz JTAG 2201, y una interfaz paralela 2202 de sistema en chip 110.

Como se ha representado en la figura 22, un primer esquema de conexión de acceso en paralelo 2210 utiliza un cable común para JTAG y el acceso en paralelo. El primer esquema de conexión de acceso en paralelo 2210 utiliza un único dispositivo de conexión (denominado como dispositivo de interfaz JTAG-paralela) 2111 tanto para la interfaz JTAG 2201 con la ruta de exploración del sistema en chip 110 como la interfaz paralela 2202 del sistema en chip 110.

Como se ha representado en la figura 22, un segundo esquema 2220 de conexión de acceso en paralelo se utiliza conexiones JTAG y paralela separadas. El segundo esquema 2210 de conexión de acceso en paralelo utiliza un primer dispositivo de conexión (denominado como Dispositivo de interfaz JTAG 2121) para la interfaz JTAG 2201 con la ruta de exploración del sistema en chip y un segundo dispositivo de conexión (denominado dispositivo de interfaz paralela 2122) para la interfaz paralela 2202 del sistema en chip 110.

Respecto del segundo esquema de acceso en paralelo 2210, aunque se ilustra y describe con respecto al mismo una realización en la cual el sistema de prueba 120 es la fuente/foco de prueba tanto para la interfaz JTAG 2201 como la interfaz paralela 2202, en otras realizaciones, la fuente y/o foco de prueba para la interfaz JTAG 2201 y la interfaz paralela 2201 pueden ser diferentes. Por ejemplo, la interfaz JTAG 2201 o la interfaz paralela 2202 pueden usa una fuente y/o foco de prueba distinto del sistema de prueba 120.

En general, dependiendo de la implementación de la interfaz paralela, el sistema de prueba que lleva a cabo la prueba de un sistema en chip por la interfaz paralela puede no ser capaz de acceder directamente al puerto paralelo del sistema en chip, por ejemplo debido a muchas construcciones dispuestas entre el sistema de prueba y el puerto paralelo del sistema en chip. Se ilustra y describe un ejemplo en las figuras 23A y 23B.

La figura 23A ilustra y describe un diagrama de bloques de alto nivel de un entorno de prueba ejemplar. El entorno de prueba ejemplar 2300 permite que un sistema de prueba lleve a cabo la prueba en un sistema en chip usando una interfaz de acceso en paralelo que proporciona acceso en paralelo al sistema en chip.. Específicamente, el entorno de prueba 2300 incluye un sistema de prueba (ST) 2310 y un chip/placa) (C/B= 2330, que se interconectan por un dispositivo de interfaz (ID) 2320.

El ST 2310 es un sistema de prueba adaptado para llevar a cabo la prueba de un sistema en chip por una interfaz de acceso en paralelo. El ST 2310 se puede implementar de cualquier manera para implementar un sistema para probar un sistema en chip usando acceso en paralelo al sistema en chip. En una realización, ST 2310 se puede implementar

como una versión adaptada de ST 120 ilustrado y descrito respecto de la figura 11 (por ejemplo, adaptado para soportar capacidades de prueba en paralelo).

El ST 2310 incluye software adaptado para su uso en llevar a cabo la prueba de un sistema en chip por una interfaz de acceso en paralelo. Específicamente, ST 2310 incluye un sistema operativo 2311 que controla una herramienta de prueba 2312 y un controlador de interfaz paralela 2313. El ST 2310 incluye otro hardware y software (por ejemplo, procesadores, memoria, circuitos de soporte, y similar) adaptados para llevar a cabo la prueba de un sistema en chip (que se omite por motivos de claridad).

En una realización, la herramienta de prueba 2312 puede igual a la herramienta de prueba 1121 (o al menos las funciones ilustradas y descritas respecto de la herramienta de prueba 2312 se pueden implementar como parte de la herramienta de prueba 1121.

El C/B 2330 incluye un sistema en chip 2331 y una interfaz paralela 2332. El sistema en chip 2331 puede ser cualquier sistema en chip descrito en el presente documento. La interfaz paralela 2332 es una interfaz paralela como se lustra y describe respecto de la figuras 21 y 22. La interacción entre la interfaz paralela 2332 y el sistema en chip 2331 se puede entender mejor con respecto a la figura 21. Como se ilustra en la figura 23A, el sistema en chip 2331 y la interfaz paralela 2332 se describen cada uno usando NSDL.

Las ID 2320 funciona como una interfaz entre ST 2310 y C/B 2330. La interfaz entre ST 2310 e ID 2320 se puede implementar usando cualquier tipo de interfaz soportada por ST 2310 (por ejemplo un cable USB o cualquier otro tipo de interfaz que puede ser soportado por ST 2310). La interfaz entre ID 2320 y C/B 2330 se pueden implementar como cualquier tipo de interfaz soportada por C/B 2330. En una realización, ID 2320 soporta interfaces de datos y control separadas con C/B 2330 de manera que las señales de datos y las señales de control se pueden aplicar desde ST 2310 a C/B 2330 de manera independiente.

Como se describe en el presente documento, la herramienta de prueba 2312 procesa los intercambios de datos con la interfaz paralela 2332 (y de este modo, el sistema en chip 2331) y el controlador de interfaz paralela 2313 procesa los intercambios de protocolo con la interfaz paralela 2332 (y, de este modo, el sistema en chip 2331). Dicho de otro modo, el controlador de interfaz paralela 2313 evita que la herramienta de prueba 2312 tenga que gestionar los intercambios de protocolo con la interfaz paralela 2332. Como se describe en la figura 23A, los intercambios de datos y los intercambios de protocolo entre ST 2310 y C/B 2330 son soportados por ID 2320.

En una realización, el controlador 2313 de interfaz paralela procesa los intercambios de protocolo con la interfaz paralela 2332 usando las funciones soportadas por el sistema operativa 2311 (por ejemplo, usando memorias intermedias, semáforos, buzones de correo y similar, así como varias combinaciones de los mismos). En esta realización, la herramienta de prueba 2312 no controla directamente el puerto JTAG del sistema en chip; en su lugar la herramienta de prueba 2312 interactúa con los controladores que proporcionan varias funciones a la herramienta de prueba 2312 (por ejemplo, usando las funciones declaradas por el controlador e importadas por la herramienta de prueba 2312).

La figura 23B ilustra un diagrama de bloques de alto nivel del flujo de datos en el entorno de prueba ejemplar de la figura 23A. Como se representa en la figura 23B, los datos fluyen desde ST 2310 a C/B 2330 (denominado flujo de datos 2351) y desde C/B 2330 a ST 2310 (denominado flujo de datos 2352). En el flujo de datos 2351, los datos fluyen desde la herramienta de prueba 2312 al controlador 2313 de interfaz paralela al dispositivo de interfaz 2320 a la interfaz paralela 2332 al sistema en chip 2331. En el flujo de datos 2352, los datos fluyen a lo largo de la ruta inversa desde el sistema en chip 2331 a la herramienta de prueba 2312. Como se describe en el presente documento, solamente los flujos de datos son importante en el contexto del entorno de prueba 2300.

Una descripción basada en NDL del sistema en chip 2331 incluye uno o más puertos conectados (por ejemplo, en el momento de instanciación) a la interfaz paralela correspondiente, una o más rodajas paralelas (donde se almacenan los datos para transacciones paralelas), y una o más funciones de transacciones paralelas adaptadas para iniciar transacciones en el puerto paralelo. La o las rodajas paralelas se pueden identificar usando una denominación específica (por ejemplo “parallel_xxxx”). Las funciones de transacciones paralelas se pueden identificar usando una denominación específica (por ejemplo “send_parallel_data” y “get_parallel_data”). Por ejemplo, los prototipos de las funciones de transacciones paralelas pueden incluir:

```
la función send_parallel_data (sending_slice: in string)
        return boolean;
la función get_parallel_data (receiving_slice: in string)
        return boolean;
```

Las funciones ejemplares de transacciones paralelas descritas en el presente documento revelan actividad en el puerto paralelo, e indican, además modificaciones a los flujos de bits de prueba en serie con el fin de controlar los flujos de bits de prueba en serie. La implementación de la “rodaja paralela” informa al sistema de prueba (de manera

ilustrativa la herramienta de prueba 2312 de ST 2310) de los detalles del flujo de datos entre el puerto paralelo del sistema en chip y el resto del sistema en chip (denominado en el presente documento el “núcleo” del sistema en chip). La conexión entre el puerto paralelo del sistema en chip y el núcleo del sistema en chip se puede implementar de muchas maneras (ejemplos de los cuales se ilustran y describen respecto de las figuras 24 – 27.

La figura 24 ilustra y describe un diagrama de bloques de alto nivel de una conexión ejemplar utiliza un puerto paralelo y un núcleo de un sistema en chip. Como se ha ilustrado en la figura 24, la conexión ejemplar utiliza un puerto paralelo completamente independiente (independiente de la ruta de exploración en serie). Específicamente, la conexión 2400 incluye un puerto TAP 2410 y un puerto paralelo externo 2420 que proporciona acceso en paralelo a un sistema en chip 2430. El sistema en chip 2430 incluye un núcleo 2439, al que se puede acceder por el puerto TAP 2410 o por el puerto paralelo externo 2420.

Al núcleo 2439 se accede por el puerto TAP 2410 usando registros en serie 2431 en la ruta de exploración en serie del sistema en chip 2430. Los registros en serie 2431 controlan el acceso al núcleo 2439 por una primera interfaz 2433. Al núcleo 2439 se accede por el puerto paralelo externo 2420 usando registros paralelos 2434 que están fuera de la ruta de exploración en serie del sistema en chip 2430. Los registros paralelos 2434 controlan el acceso al núcleo 2439 por una segunda interfaz 2435.

Como se ilustra en la figura 24, el acceso al núcleo 2439 por los registros paralelos 2434 es completamente independiente de los registros en serie 2431. De este modo, todas las señales de control son procesadas por una lógica paralela sin ninguna necesidad de intervenir desde la ruta de exploración en serie del sistema en chip 2430. La función “get_parallel_data” toma simplemente el nombre de la rodaja paralela (de manera ilustrativa; “parallel_reg”) como un argumento, y puesto que no se requieren modificaciones en el flujo de bits, el cuerpo de la función está vacío.

La figura 25 ilustra un diagrama de bloques de alto nivel de una conexión ejemplar entre un puerto paralelo y el núcleo de un sistema en chip. Como se ha descrito en la figura 25, la conexión ejemplar utiliza un puerto paralelo independiente con control en serie. Específicamente, la conexión 2500 incluye un puerto TAP 2510 y un puerto paralelo externo 2520 que proporciona acceso en paralelo a un sistema en chip 2530. El sistema en chip 2530 incluye un núcleo 2530, al que se puede acceder por el puerto TAP 2510 o por el puerto paralelo externo 2520.

Al núcleo 2539 se accede por el puerto TAP 2510 usando registros en serie 2531 en la ruta de exploración en serie del sistema en chip 2530. Los registros en serie 2531 controlan el acceso al núcleo 2539 por una primera interfaz 2533. Al núcleo 2539 se accede por el puerto 2520 paralelo externo usando los registros paralelos 2534 que están fuera de la ruta de exploración en serie del sistema en chip 2530. Los registros paralelos 2534 controlan el acceso al núcleo 2539 por una segunda interfaz 2535.

Como se ha descrito en la figura 25, el acceso al núcleo 2539 por los registros paralelos 2534 y la segunda interfaz asociada 2535 se controla usando un registro habilitar 2532 adicional en la ruta de exploración en serie del sistema en chip 2530. El registro habilitar 2532 controla el acceso al núcleo 2539 por los registros paralelos 2534 usando una interfaz de control 2537 desde el registro habilitar 2532 al núcleo 2539. De este modo, la ruta de exploración en serie del sistema en chip 2530 incluye: TDI → registro habilitar 2532 → registros en serie 2531 → TDO.

De este modo, como se ilustra en la figura 25, el acceso al núcleo 2539 por los registros paralelos 2534 se controla en serie a partir de la ruta de exploración en serie del sistema en chip 2530. En esta realización, la función “get_parallel_data” toma el nombre de la rodaja paralela (de manera ilustrativa, “parallel_reg”) como un argumento y, además, puesto que se requieren la modificación en el flujo de bits, el cuerpo de la función incluirá una instrucción para establecer e valor del registro habilitar 2532 al valor deseado.

La figura 26 ilustra un diagrama de bloques de alto nivel de una conexión ejemplar entre un puerto paralelo y el núcleo de un sistema en chip. Como se ilustra en la figura 26, la conexión ejemplar utiliza un puerto de acceso compartido al núcleo para datos en serie y en paralelo. Específicamente, la conexión 2600 incluye un puerto TAP 2610 y un puerto paralelo externo 2620 que proporciona acceso en paralelo a un sistema en chip 2630. El sistema en chip 2630 incluye un núcleo 2639, al que se puede acceder por el puerto TAP 2610 o por el puerto paralelo externo 2620.

Al núcleo 2639 se accede por el puerto TAP 2610 usando registros en serie 2631 en la ruta de exploración en serie del sistema en chip 2630. Los registros en serie 26310 controlan el acceso al núcleo 2639 por una primera interfaz 2633. Al núcleo 2639 se accede por el puerto paralelo externo 2620 usando registros paralelos 2634 que están fuera de la ruta de exploración en serie del sistema en chip 2630. Los registros paralelos 2634 controlan el acceso al núcleo 2639 por una segunda interfaz 2635.

Como se describe en la figura 26, el acceso al núcleo 2639 por los registros en serie 2631 y los registros en paralelo 2634 se controla usando un puerto de acceso compartido 2636. El puerto de acceso compartido 2636 toma la primera interfaz 2633 de los registros en serie 2631 como primer entrada y toma una segunda interfaz 2635 de los registros

paralelos 2634 como segunda entrada. El puerto de acceso compartido 2636 selecciona una de las entradas y proporciona la entrada seleccionada de las entradas al núcleo 2639 por una interfaz de acceso compartido 2638.

Como se ilustra en la figura 26, la selección de una de las entradas por el puerto de acceso compartido 2636 se controla usando un registro habilitar adicional 2632 en la ruta de exploración en serie del sistema en chip 2630. El registro habilitar 2632 controla el acceso al núcleo 2639 por el puerto de acceso compartido 2636 y la interfaz de acceso compartido 2638 usando una interfaz de control 2637 del registro habilitar 2632 al puerto de acceso compartido 2636. De este modo, la ruta de exploración en serie del sistema en chip 2630 incluye: TDI → registros enserie 2631 → registro habilitar 2632 → TDO.

De este modo, como se ilustra en la figura 26, el acceso al núcleo 2639 por los registros paralelos 2634 se controla en serie a partir de la ruta de exploración en serie del sistema en chip 2630. En esta realización, el sistema en chip 2539 revela la dicotomía (por ejemplo, etiquetando registros en serie 2631 y registros paralelos 2634 como “alternados"). En tales realizaciones, el sistema de prueba sabe cuando la interfaz paralela está activa, los registros en serie 2631 no tienen influencia sobre el núcleo 2639 (es decir, son efectivamente “almacenamiento muerto”). Esta regla es asociativa porque muchos registros pueden compartir el mismo puerto paralelo.

La figura 27 ilustra un diagrama de bloques de alto nivel de una conexión ejemplar entre un puerto paralelo y el núcleo de un sistema en chip. Como se describe en la figura 27, los registros en serie y los registros paralelos que proporcionan acceso en paralelo al núcleo del sistema en chip comparten los mismos cambios, minimizando de este modo los recursos necesarios para proporcionar acceso en paralelo al núcleo del sistema en chip. Específicamente, la conexión 2700 utiliza un registro habilitar 2732, una pluralidad de registros de datos 2731_1 – 2731_8 (de manera colectiva, registros de datos 2731), y una pluralidad de puertos de acceso compartido 2733_1 – 2733_8 (de manera colectiva, puertos de acceso compartido 2733).

Como se ilustra en la figura 27, la entrada de datos del registro habilitar 2732 es una entrada TDI, la salida de datos del registro habilitar 2732 es una de las entradas de un primer puerto de acceso compartido 2733_1, la salida de datos del primer puerto de acceso compartido 2733_1 es la entrada de datos de un primer registro de datos 2731_1, una primera salida de datos del primer registro de datos 2731_1 es una de las entradas de un segundo puerto de acceso compartido 2733_2, la salida de datos del segundo puerto de acceso compartido 2733_2 es la entrada de datos de un segundo registro de datos 2731_2, una primera salida de datos del segundo registro de datos 2731_2 es una de las entradas de un tercer puerto de acceso 2733_3, la salida de datos del tercer puerto de acceso compartido 2733_3 es la entrada de datos de un tercer registro de datos 2731_3, y así sucesivamente, hasta la primera salida de datos del octavo registro de datos 2731_8 es una salida TDO.

Como se ilustra asimismo en la figura 27, cada uno d los puertos de acceso compartido 2733 incluye una segunda entrada de datos (además de estar acoplados a una salida de datos del registro anterior en la ruta de exploración). La segundas entradas de datos de puertos de acceso compartido 2733_1 – 2733_8 se acoplan a las entradas de datos respectivas procedentes del puerto paralelo externo 2720 (denominado en el presente documento conexiones de entrada paralela), respectivamente. De este modo, cada puerto de acceso compartido 2733 selecciona datos de una de sus dos entradas de datos (es decir, seleccionando bien la entrada de datos del registro anterior en la ruta de exploración en serie o la entrada de datos del de las conexiones de entrada paralela del puerto paralelo externo 2720 que se conecta a ese puerto de acceso compartido 2733).

Como se ilustra en la figura 27, la salida del registro habilitar 2732 se aplica a cada puerto de acceso compartido 2733 como señal de selección de entrada para cada puerto de acceso compartido 2733, controlando de este modo la selección de datos por cada puerto de acceso compartido 2733.

Si el valor del registro habilitar 2732 indica que los datos en serie (del puerto TAP 2710) se deberían proporcionar al núcleo 2739, la señal de selección de entrada proporcionada a partir del registro habilitar 2732 a cada puerto de acceso compartido 2733 dirige cada puerto de acceso compartido 2733 para seleccionar la entrada a partir del registro anterior en la ruta de exploración (en lugar de a partir de la conexión de entrada paralela que se conecta al puerto de acceso compartido a partir del puerto paralelo externo 2720).

En este caso, el puerto de acceso compartido 2733_1 selecciona la entrada desde el registro habilitar 2732 (en lugar de a partir de la conexión de entrada paralela del puerto paralelo externo 2720), haciendo de este modo que el valor del registro habilitar 2732 sea leído en el primer registro de datos 2731_1 (y, de este modo proporcionado al núcleo 2739). Asimismo, en este caso, el puerto de acceso compartido 2733_2 selecciona la entrada a partir del primer registro de datos 2731_1 (en lugar de a partir de la conexión de entrada paralela del puerto paralelo externo 2720), haciendo de este modo que el valor del primer registro de datos 2731_1 sea leído en el segundo registro de datos 2731_2 (y de este modo, proporcionado al núcleo 2739). Dicho de otro modo, aunque se omiten las descripciones del resto de las transiciones de datos por razones de claridad, transiciones de datos similares por otros de los registros de datos 2731 permiten proporcionar datos en serie al núcleo 2739.

Si el valor del registro habilitar 2732 indica que los datos paralelos (a partir del puerto paralelo externo 2720) se

deberían proporcionar al núcleo 2739, la señal de selección de entrada proporcionada a partir del registro habilitar 2732 a cada puerto de acceso compartido 2733 dirige cada puerto de acceso compartido 2733 para seleccionar la entrada a partir de la conexión de entrada paralela que se conecta al puerto de acceso compartido a partir del puerto paralelo externo 2720 (en lugar de a partir del registro anterior en la ruta de exploración).

En este caso, el puerto de acceso compartido 2733_1 selecciona la entrada a partir de la conexión de entrada paralela del puerto paralelo externo 2720 que se conecta al puerto de acceso compartido 2733_1 (en lugar de a partir del registro habilitar 2732), haciendo de este modo que el valor procedente de esa conexión de entrada paralela del puerto paralelo externo 2720 sea leído en el primer registro de datos 2731_1 (y de este modo, proporcionado al núcleo 2739). Asimismo, en este caso, el puerto de acceso compartido 2733_2 selecciona la entrada a partir de la conexión de entrada paralela del puerto paralelo externo 2720 que se conecta al puerto de acceso compartido 2733_2 (en lugar de a partir del primer registro de datos 2731_1), haciendo de este modo que el valor procedente de esa conexión de entrada paralela del puerto paralelo externo 2720 sea leído en el segundo registro de datos 2731_2 (y de este modo, proporcionado al núcleo 2739). Dicho de otro modo, aunque se omiten las descripciones del resto de las transiciones de datos por razones de claridad, transiciones de datos similares por otros de los registros de datos 2731 permiten proporcionar datos en paralelo al núcleo 2739.

Asimismo, aunque una descripción directa NSDL de la conexión 2700 sería bastante complicada, cabe señalar que este tipos de conexión es funcionalmente equivalente a la conexión 2600 ilustrada y descrita respecto de la figura 26. En las conexiones 2600 y 2700, es el valor de la rodaja “select” que decide si los datos en serie alcanzan el núcleo o si los datos en paralelo alcanzan el núcleo. La única diferencia entre la conexión 2600 y la conexión 2700 está en los valores esperados: en la conexión 2700 los registros son compartidos, por lo tanto en caso de acceso en paralelo el calor que hay que esperar es el valor paralelo, ya que los datos en serie se sobrescriben.

Aunque se ilustra y describe principalmente en el presente documento respecto de la provisión de acceso en paralelo a un sistema en chip que tiene un puerto paralelo (por motivos de claridad), se puede proporcionar acceso en paralelo a un sistema en chip que tiene múltiples puertos paralelos. Asimismo, aunque se ilustra y describe principalmente en el presente documento respecto de la provisión de acceso en paralelo a un sistema en chip que tiene una rodaja paralela (por motivos de claridad), se puede proporciona acceso en paralelo a un sistema en chip que tiene múltiples rodajas paralelas.

En tales realizaciones, se pueden identificar múltiples puertos paralelos de cualquier manera. Por ejemplo, se pueden identificar múltiple puertos paralelos usando numeración ordinal (es decir, tal como se describe respecto de los dispositivos de cruce). Por ejemplo un sistema en chip con n entradas paralelas y m salidas paralelas tendrá los siguientes puertos: “parallel_in<i>” , i=0,1, …, n-1; “parallel_out_<k>”, k=0,1, …, m-1. Además, cada puerto paralelo tendrá su propia o propias funciones, cada una de las cuales se puede identificar de cualquier manera (por ejemplo, usando un nombre de puerto correspondiente ajuntado al final, tal como “get_parallel_data_parallel_in_0”, “set_scan_synchro_parallel_out_3” y similar).

Por el contrario, no hay restricciones respecto de la denominación de las múltiples rodajas paralelas siempre que los respectivos nombres de las rodajas paralelas indiquen que las rodajas son rodajas paralelas (por ejemplo, los nombres pueden empezar con “parallel_”). En una realización , en la cual “send_parallel_data" y get_parallel_data” toman un nombre de rodaja como parámetro, es posible conectar cualquier puerto con cualquier registro paralelo. En otra realización, en la cual “send_parallel_data” y “get_parallel_ data” no toman un nombre de rodaja como parámetro, el sistema en chip puede declarar exactamente la manera en que cada puerto se “conecta” a una o más rodajas paralelas.

La descripción NSDL de la interfaz paralela incluye una descripción de la ruta de exploración y sus funciones relacionadas. La descripción NSDL indica los puertos físicos reales (patillas paralelas) a los cuales se conecta la interfaz paralela. Esto se puede gestionar por las reglas clásicas BSDL/HSDL en archivos de nivel superior (por ejemplo, tal como de la manera en que BSDL identifica señales TAP) La realización del protocolo de comunicación en paralelo es confiado al controlador de interfaz paralela. Se comunica al sistema de prueba qué patillas paralelas son controladas por qué controlador de interfaz paralela.

Aunque se ilustra y describe principalmente en el presente documento respecto del flujo de datos de entrada para el puerto paralelo, el flujo de datos de salida para el puerto paralelo será simétrico. Dicho de otro modo, para cada uno de los tipos de conexión de entrada que se pueden implementar para flujos de datos de entrada al sistema en chip (como se ilustra y describe respecto de las figuras 24 – 27) , se puede implementar un tipo de conexión de salida simétrica correspondiente para flujo de datos de salida a partir del sistema en chip.

Aunque se ilustra y describe principalmente en el presente documento respecto de una interfaz de acceso en paralelo que soporta una única conexión interna entre el puerto paralelo interno de la interfaz de acceso en paralelo y el puerto paralelo del sistema en chip, una interfaz de acceso en paralelo puede soportar conexiones internas más complejas entre el puerto paralelo interno de la interfaz de acceso en paralelo y el puerto paralelo del sistema en chip. De esta

manera, NSDL puede describir cualquier Mecanismo de Acceso de Prueba (TAM), sin tener en cuenta su complejidad.

En una realización, se puede proporcionar la conexión interna entre el puerto paralelo interno de la interfaz de acceso en paralelo y el puerto paralelo del sistema en chip usando un uno o más dispositivos de cruce. En una realización se puede utilizar las funciones seleccionar y deseleccionar para procesar la conexión interna, bien desde la ruta de exploración en serie o a partir del puerto paralelo.

En otra realización se puede proporcionar la conexión interna entre el puerto paralelo interno de la interfaz de acceso en paralelo y el puerto paralelo del sistema en chip usando esquema de entrada en abanico/salida en abanico. En tales realizaciones, se pueden usar los bits del puerto paralelo para accionar múltiples sistema en chip (es decir, el ancho de banda del puerto paralelo externo se comparte entre los múltiples dispositivos de sistema en chip por el puerto paralelo interno.).

De este modo, aunque se ilustra y describe principalmente en el presente documento con respecto a una interfaz de acceso en paralelo que proporciona acceso en paralelo a un dispositivo de sistema en chip, en otras realizaciones una interfaz de acceso en paralelo puede proporcionar acceso en paralelo a múltiples dispositivos de sistema en chip. S ilustra y describe un esquema general de conexión de tal interfaz de acceso en paralelo respecto de la figura 28.

La figura 28 ilustra un diagrama de bloque de alto nivel de un esquema de conexión interna de una interfaz de acceso en paralelo. Específicamente, el esquema de conexión interna 2800 proporciona una interfaz de acceso en paralelo 2810 a tres sistemas en chip 2820_1 – 2820_3 (de manera colectiva, sistemas en chip 2820). La interfaz de acceso en paralelo 2810 incluye un puerto paralelo interno 2811, un puerto paralelo externo 2812, y una interfaz 2813. El puerto paralelo externo 2812 y el puerto paralelo interno 2811 soportan n conexiones de entrada a partir de un sistema de prueba a sistemas en chip 2820 y m conexiones de salida a partir de sistemas en chip 2820 a un sistema de prueba.

Como se ilustra en la figura 28, cada sistema en chip 2820 incluye un puerto paralelo que soporta conexiones de entrada en paralelo y conexiones de salida en paralelo. El puerto interno de la interfaz de acceso en paralelo 2810 soporta: (1) una salida en abanico del flujo de datos de entrada a cada uno de los sistema en chip 2120 y (2) una entrada en abanico al flujo de datos de salida a partir de cada uno de los sistemas de chip 2120. Las n conexiones de entrada del puerto paralelo externo 2812 salen en abanico dentro de i conexiones de entrada al sistema en chip 2820_1, j conexiones de entrada al sistema en chip 2820_2, y k conexiones de entrada al sistema en chip 2820_3 (es decir, n= i+j+k). Las m conexiones de salida del puerto paralelo externo 2812 entran en abanico desde p conexiones de salida a partir del sistema en chip 2820_1, q conexiones de salida desde el sistema en chip 2820_2, y r conexiones de salida desde el sistema en chip 2820_3 (es decir, m = p+q+r).

De este modo, usando el lenguaje de descripción NSDL, se pueden describir fácilmente los dispositivos de sistema en chip de cualquier complejidad. Se pueden describir cualesquiera recursos de prueba de un dispositivo de sistema en chip, incluyendo componentes (por ejemplo IPs, instrumentos, dispositivos de cruce y similar), interconexiones entre componentes y similar, así como descripciones, donde cada descripción algorítmica incluye una o más reglas de composición definidas en un formato adaptado para ser entendido por una herramienta de prueba.

La figura 29 ilustra un procedimiento para describir recursos de prueba de un sistema en chip. Aunque se ilustra y describe como llevado a cabo enserie, al menos una parte de las etapas del procedimiento 2900 de la figura 29 se puede llevar a cabo simultáneamente, o en un orden diferente del ilustrado y descrito respecto de la figura 29. El procedimiento 2900 empieza en la etapa 2902 y continúa hasta la etapa 2904.

Una etapa 2904 se genera una descripción algorítmica de cada componente del sistema en chip

La descripción algorítmica de cada componente describe una representación de al menos una función soportada por el componente a al menos un valor de registro para el componente. La descripción algorítmica de cada componente describe una ruta de exploración interna del componente.

En una realización, se genera una descripción algorítmica de un componente de sistema en chip identificando al menos una función soportada por el componente, generando una descripción algorítmica del componente que define, para cada una de la al menos una función, una representación de la función a al menos un valor de registro para al menos un registro del componente, y almacenando la descripción algorítmica del componente.

En la etapa 2906, se genera una descripción de las interconexiones entre los componentes del sistema en chip. La descripción algorítmica de las interconexiones entre componentes especifica una topología al nivel del sistema del sistema en chip.

Una etapa 2908, se genera una descripción algorítmica del sistema en chip usando las descripciones algorítmicas de los componentes y la descripción algorítmica de las interconexiones entre los componentes.

La descripción algorítmica del sistema en chip describe una topología del sistema en chip, a partir del cual se puede componer una descripción de una ruta de exploración del sistema en chip.

En la etapa 2910, se almacena la descripción algorítmica del sistema en chip. Se pueden almacenar las descripciones algorítmicas individuales de cada uno de los componentes. Se almacena la descripción algorítmica de las interconexiones entre componentes. Las descripciones algorítmicas se pueden almacenar de cualquier manera. En la etapa 2912, se termina el procedimiento 2900.

Las descripciones algorítmicas se adaptan para ser entendidas por una herramienta de prueba para su uso en la prueba del sistema en chip. Asimismo, las descripciones algorítmicas pueden ser recibidas por un procesador (por ejemplo, a partir de una memoria, a partir de otro sistema, o a partir de cualquier otra fuente de tales descripciones) con el fin de llevar a cabo varias pruebas (por ejemplo, pruebas al nivel del componente, pruebas al nivel del sistema y similar, así como varias combinaciones de los mismos).

Como se describe en el presente documento, en una realización, el lenguaje NSDL se puede implementar usando VHDL. En una realización de este tipo, las reglas gramaticales para NSDL se pueden formalizar a través de la gramática de Notación Normal de Backus (BNF). Por ejemplo, BNF describe fácilmente la generación de la estructura sintáctica, como en el siguiente ejemplo

```
<entity_declaration> ::=
        ENTITY <identifier> IS
                <entity_header>
                <entity_declarative_part>
        [BEGIN
                <entity_statement_part>]
        END [ENTITY] [<entity_simple_name>];
```

En este ejemplo, el símbolo '::=' indica que el elemento del lado izquierdo puede derivar en la construcción del lado derecho. El lado derecho se puede componer bien por más lexemas de derivación (elementos atómicos que no se puede derivar más, indicados en caracteres en MAYÚSCULAS). Un nodo es un punto de derivación, mientras que una hoja es un nodo que ya no se puede derivar (es decir el lado derecho contiene solo lexemas). Los corchetes '[' y ']' se usan para expresar derivación opcional (son útiles para definir reglas recursivas). Los símbolos '< y >' se usa para indicar una derivación adicional. Las marcas de cuotación se usan para indicar una secuencia, coherentemente con las reglas VHDL.

Este tipo de sintaxis puede generar cualquier “expresión”posible que coincida estructuralmente con el lenguaje, y se puede usar para verificar si un texto dado pertenece al lenguaje (es decir, sigue las reglas). Es meramente una descripción estructural y no puede llevar ninguna información acerca de su “significado”, en su lugar, se debe añadir atributos para encargarse de la información contextual:

$$\text{Left\_hand} \uparrow(H)\downarrow(L) ::= \text{right\_hand\_0} \uparrow(H0)\downarrow(L0)\ [\text{right\_hand\_1}\uparrow(H1)\downarrow(L1)],$$

en la que

↑ (L) indica una información que procede de las derivaciones de nivel inferior y que se transmite a derivaciones de nivel superior;

↓ (H) indica una información que procede de derivaciones de nivel superior y que transmite a derivaciones de nivel inferior;

cada nodo puede definir un conjunto de reglas para definir la manera en que H se calcula empezando por de H0... Hn, y de que manera se obtienen las diferentes L0...Ln empezando por L; y

cada nivel puede definir un conjunto de condiciones sobre H, L, Hi y Li para que la expresión tenga un “significado” en el lenguaje.

Las siguiente reglas (numeradas [1] a [14] describen declaraciones de los IPs e instrumentos

```
[1] <IP_declaration> ↑(H,n)↑(P_info)↑(Ext)↑(Cross)↑(Par)::=
        IP <identifier> IS
                <device_header> ↑(Ext)↑(Cross_decl)↑(Par)
        BEGIN
        <IP_instrument_archi_body>↑(H,n)↓(Ext)↑(P_info)
                ↑(Sel_Cross)↑(Par_dec)
```

END [IP] [<device_simple_name>];
Regla: Cross=Cross_decl ∪ Sel_cross
Si Cross_decl /= ø , verificar que hay una instrucción "select" /"deselect" (seleccionar/deseleccionar)
para cada elemento "wired" (alámbrico) y al menos un "select" (seleccionar) para cada elemento de
"transaction" (transacción).
Nota: Cross_decl = ø ↔ Sel_Cross = ø, si no error
Verifique que se han resuelto todas las dependencias inter-módulo en "architecture_body".
Si Par /= ø verificar que dentro de (H,n) hay al menos un registro paralelo,
y que dentro de (P_info) hay la declaración de
"get_parallel_data"/ "send_parallel_data" correspondiente.
Si Par_dec/= ø verificar su consistencia con la información en Par (nombres de puerto,
registros paralelos, conexiones, abaniqueo, etc...)
Nota: Par_decl = ø ↔ Par = ø, si no error

[2] <instrument_declaration>↑(H,n)↑(P_info)↑(Ext)↑(Cross)↑(Par)::=
I NSTRUMENT <identifier> IS
<device_header> ↑(Ext)↑(Cross_decl)↑(Par)
↑(Cross_decl)
BEGIN
<IP_instrument_archi_body>↑(H,n)↑(Ext)↑(P_info)
↑(Sel_Cross)↑(Par_dec)
END [instrument] [device_simple_name] ;
Regla: Cross=Cross_decl U Sel_Cross [igual que la regla 1]

[3] <IP_instrument_archi_body>↑(P_info)↓(Ext)↑(Sel_Cross)↑(Par_dec)
::= ARCHITECTURE <architecture_simple_name> OF <entity_name> IS
architecture_declarative_part
BEGIN
<IP_instr_stat_part>↑(H,n)↑(P_info)↓(Ext)↑(Sel_Cross)↑(Par_dec)
END [ARCHITECTURE] [<architecture_simple_name>];

[4] <IP_instr_stat_part>↑(H,n)↑(P_info)↓(Ext)↑(Sel_Cross)↑(Par_dec) ::=
<internal_scan_path>↑(H,n)
[<parallel_declarations>↑(Par_dec)]
<IP_instrumen_statement_part>↓(Ext)↑(P_info)↑(Sel_Cross)

[5] <device_header> ↑(Ext)↑(Cross-decl)↑(Par) ::=
GENERIC (
[<crossroad_information>↑(Cross_decl)]
[; <parallel_information>↑(Par)]
[; <external_dependencies>↑(Ext)]);
Esta regla permite la definición de los enlaces con la ruta de exploración y
la declaración de referencias externas eventuales.
Nota: <crossroad_information> se deriva en la Regla [10]

[6] <external-dependencies> ↑(Ext) ::=
<external_reference>↑(New_Ext)[;<external_dependencies>↑(Old_Ext)]
Regla: Ext= New_Ext U Old_Ext

[7] <extemal_reference> ↑(Ext)::= <string_identifier> : string
Regla: ↑(Ext)= string_identifier
Cada una de estas entradas define un nombre simbólico para un elemento
externo. Este símbolo se usará para referirse a este elementos para dependencias de
funciones externas.

[8] IP_instrument_statement_part ↓(Ext)↑(P_info)::=
<proc_func_list>↓(Ext)↑(P_use)↑(Sel_Cross)
[TEST_SET <proc_func_list>(Ext)↑(P_test) END TEST_SET;]
Esta regla describe el conjunto de procedimientos obligatorios en el uso del
instrumento y el conjunto de procedimiento de prueba opcional. P_Info = P_Test ∪ P_use

[9] <internal_scan_path> ↑(H,n) ::=
<component_instantiation_statement> [<internal_scan_path>]

En el caso de "Acceso parcial" o "Pleno o Acceso", la ruta de exploración se describe por instanciación directa de sus componentes (véase las reglas [37] – [41]para reglas contextuales de (H,n)). Obsérvese que esta regla habilita el anidamiento de instrumentos e IPs. Se identifican los dispositivo "No Access" (sin acceso) mediante la ausencia de esta derivación.

[10] <crossroad_information> ↑(Cross)::=
PRECEDENT : STRING := "TDI";
FOLLOWING : STRING:= "TDO"
[<tributaries>↑(Affls)]
[<affluents>↑(Tribs)]
Regla: Cross = Affls ∪ Tribs

[11] <tributaries>↑(Tribs) ::=
TRIBUTARY : STRING := "deselected" ;
| <numbered_tributaries>↑(num_tribs);
Regla: Tribs = "tributary" (tributario)
| Tribs = num_tribs

[12] <numbered_tributaries>↑(Tribs) ::=
TRIBUTARY_<numeral> : STRING := "deselected"
[; <numbered_tributaries>↑(Old_Tribs)]
Regla: Tribs = "tributary_<numeral>" U Old_tribs

[13] <affluents> ↑(Affls)::=
AFFLUENT : STRING := "deselected" ;
| <numbered_affluents>↑(num_affls) ;
Regla: Affls = "affluent" (afluente)
| Affls = num_affls

[14] <numbered_affluents> ↑(Affls)::=
AFFLUENT_<numeral>: STRING := "deselected"
[; <numbered_affluents>↑(Old_Affls)]
Regla: Affls = "affluent_<numeral>" U Old_Affls

Las definiciones de instrumento e IP de las Reglas [1] - [14] permiten la instanciación de un momento múltiple de IP/Instrumento (como lo que se hace con los componentes de VHDL clásico). Esto se ilustra en la Regla [15]:

[15] component_instantiation_statement ::=
instantiation_label :
instantiated unit
[generic map aspect]
[port map aspect];

Desde el punto de vista sintáctico, la regla de instanciación (Regla [15]) es exactamente la misma que en el VHDL clásico. Todas las novedades se dan en el aspecto contextual:

(1) La instanciación crea una copia del dispositivo en la tarea de exploración. El compilador puede recuperar fácilmente su información a partir de la biblioteca de compilación y usarla para completar la ruta de exploración del sistema.

(2) La ruta genérica se encarga de especificar el punto de inserción de ruta de exploración precisa (“precedent” (precedente), “following” (siguiente), afluentes y tributarios).

(3) Las otras representaciones genéricas resolverán los nombres simbólicos de elementos externos en nombres reales (es decir, la etiqueta de la instancia correspondiente). La herramienta de prueba tendrá también que comprobar que el procedimiento referido existe realmente en el elemento instanciado.

Las siguientes reglas (numeradas de [16] a [33]) describen procedimientos ejemplares:

[16] <proc_func_list>↓(Ext)↑(P_info) ::=
[<proc_func_proto_list>]<complete_proc_func_list>↓(Ext)↑(P_info)

[17] <proc_func_proto_list> ::=
<proc_func_prototype> [; <proc func_proto_list>]

[18] <proc_func_prototype> ::= <procedure_prototype> |
<function_prototype>

[19] <procedure_prototype> ::=
procedure <procedure_name> (<formal_parameter_list>)
[DEPENDENCIES (<dep_list>);]
LENGTH (<length_descriptor>);
BUSY_MODE (<mode_identifier>);
[CONNECTION <connection_type>;]
Regla: Los prototipos de procedimiento son solo un artífice sintáctico para hacer que el código sea más legible para un usuario humano. No llevan información contextual.
Nota: La derivación "connection" (conexión) se permite solo para procedimientos de selección.

[20] <function_prototype> ::=
FUNCTION <procedure_name> (<formal_parameter_list>)
[DEPENDENCIES «dep_list»;]
LENGTH (<length_descriptor>);
BUSY_MODE (<mode_identifier>);
[<optional_attributes>]
RETURN <type> ;
Regla: Los prototipos de procedimiento son solo un artífice sintáctico para hacer que el código sea más legible para un usuario humano. No llevan información contextual.
Nota La derivación "connection" (conexión) se permite solo para procedimientos de selección.

[21] <formal_parameter_list>
Regla: Los parámetros siguen la sintaxis de los parámetros de VHDL normal.
La referencia a las rodajas de ruta de exploración se hará por denominación explícita.

[22] <complete_proc_func_list>↓(Ext)↑(P_info)↑(Cross) ::=
<complete_procedure>↓(Ext)↑(new_P)↑(Cross_info)
[;<complete_proc_func_list>↓(Ext)↑(OldP)↑(Old_Cross)] |
<complete_function>↓(Ext)↑(new _P)
[;<complete_proc_func_list>↓(Ext)↑(OldP)]
Regla: P_info=Old_P U New_P
Cross=Old_Cross U Cross_info

[23] <complete_procedure>↓(Ext)↑(Proc_info)↑(Cross_info)::=
PROCEDURE <procedure_name↑(Sel_info)>
(<formal_parameter_list>↑(P))
[DEPENDENCIES (<dep_list>)↓(Ext)↑(D);]
LENGTH (<length_descriptor>↑(L));
BUSY_MODE (<mode_identifier>)↑(M);
[CONNECTION <connection_type>↑(C_type);]
IS
begin
<procedure_body>
END <procedure_name>;
Regla:
P= información de parámetros (VHDL estándar). Un parámetro al menos se debería referir a una rodaja o un flujo de bit estático.
L= información de longitud de procedimiento
D= información de dependencias
M= información de modo ocupado
<procedure_name> es un identificador literal
<procedure_body> derivación como en VHDL normal
Proc_info=P U L U D U M
Cross_info= if Sel_info/= ∅(naming rules in Sel [1] - Sel [6])
then Cross_info= Sel_info U C_type
Nota: Sel_info= ø ↔ C_type= ø, si no error

[24] <complete_function> ↓(Ext)↑(Proc info)::=
FUNCTION <function_name> (<formal_parameter_list>↑(P))
[DEPENDENCIES (<dep_list>)↓(Ext)↑(D);]
LENGTH (<length_descriptor>↑(L));
BUSY_MODE (<mode_identifier>)↑(M);
RETURN <type> IS
BEGIN
< function_body>
END < function_name>;
Regla:
P= información de parámetro (VHDL estándar). Un parámetro al menos se debería referir a una rodaja o un flujo de bits estático
L= información de longitud de procedimiento
D= información de dependencias
M= información de modo ocupado
< function_name> es una identificador literal
< function_body> derivación como en VHDL normal
Proc_info=P U L U D U M

[25] <dep_list>↓(Ext)↑(D_new) ::=
<dependence>↓(Ext)↑(D)[;dep_list↓(Ext)↑(D_old)]
Regla: D_new= D_old U D o D_new=D

[26] <dependence>↓(Ext)↑(D) ::= <string_identifier>↑(P)|
<string-identifier>↑(E).<string_identifier>↑(P)
Regla:
P= nombre de procedimiento dependiente
E= el nombre del dispositivo externo P se define en
Check ECExt, si no error
D=P U E

[27] <mode_identifier>↑(M) ::= HOLD | DONT_CARE
Regla: M= "hold" (retener) o "dont_care" (no importa)

[28] <length_descriptor>↑(L) ::= <length_exp>↑(T) [: <end_cond>↑(C)]
Regla: L=T U C
[
29] < length_exp >↑(T) ::= <time_exp>↑(T) |
<time-exp>↑(L),<time_exp>↑(A),<time_exp>↑(U)
Regla: T=T o L U A U U
L= Límite inferior para longitud de función
A= Longitud media de función
U= Límite superior para longitud de función

[30] < time_exp >↑(T) ::= <numeral> <time_type> |<numeral>
Regla: T contiene el tiempo absoluto o la cuenta de reloj (si no se especifica la unidad de tiempo)

[31] < end_cond >↑(C) ::= <boolean_expression>
Regla: Una expresión bouleana que indica la condición final del procedimiento. Se debería usar señales procedentes de las rodajas, identificada por denominación explícita.

[32] <connection_type> ::= WIRED TRANSACTION

[33] <optional_attributes> ::=
Regla: La derivación se deja abierta a propósito. Definiendo parámetros opcionales, dos actores serán capaces de intercambiar información en el formato que prefieran, en lugar de uno elegido arbitrariamente en el momento de la estandarización. Esto se podría usar, por ejemplo, para dar una estimación de la actividad de conmutación para la gestión de la potencia, o dar directamente los julios/vatios. La Herramienta de Prueba ignorará las derivaciones que no implementa, produciendo eventualmente un aviso.

Como se describe en el presente documento, los procedimientos de selección se identifican por su nombre. Las siguientes reglas (numbered Sel [1]) a Sel [6]) explican el control de la denominación de los procedimientos de

selección en la sintaxis de tipo BNF. Los dispositivos de cruce ejemplares ilustrados y descritos en el presente documento respecto de las figuras 8, 9 y 10, ilustran la aplicación ejemplar de estas reglas.

sel[1] <procedure_name> ↑(Sel_info) :: =
<radix>[<extentions>↑(Affls)↑(Tribs)]
Regla : Sel_info=(Affls) U (Tribs)

sel[2] <radix> ::= SELECT | DESELECT

sel[3] <extensions> ::=
<affluents↑(Affls)> | <tributaries↑(Tribs)>

sel[4] <affluents↑(Affls)> := AFFLUENT
| <numbered_affluents↑(Nmb_Affls)> |
ø
Regla: si (affluent) Affls="All";
si (numbered_affluents) Affls= Nmb_Affls
si (ø) Affls= ø;
Explicación: Esta regla detecta el afluente que ordena la función. Puede ser ninguno (es decir, la regla termina como un conjunto vacío), todos los afluentes (no especificado), o solo un subconjunto (uno o más affluent_<nmb>);
sel[5] <numbered_affluents↑(New_Affls)> ::=
AFFLUENT_<nmb>[;< numbered_affluents↑(Old_Affls)>]
Regla : <nmb> puede ser cualquier número natural
New_Affls= Old_Afflsu afflunent_<nmb>;

sel[6] <tributariess↑(Tribs)> ::= TRIBUTARY
| <numbered_tributaries↑(Nmb_Tribs)> |
ø
Regla : si (tributary) Tribs="All";
si (numbered_tributaries) Tribs= Nmb_Tribs
si (ø) Tribs= ø;
Explicación: esta regla detecta los tributarios que la función ordena. Puede ser
ninguno es decir, la regla termina como un conjunto vacío), todos los tributarios (no especificado),
o solo un subconjunto (uno o más tributary_<nmb>);

Para los procedimientos de selección a usar por una herramienta de generación de prueba automática, existe asimismo la necesidad de argumentos estandarizados, de manera que la herramienta de generación de prueba automática sepa como procesarla. Siguiendo el algoritmo de selección real, puede haber dos maneras de referenciar las derivaciones: (1) por denominación explícita, es decir usando una "secuencia" o un tipo equivalente; (2) por numeración ordinal de la derivación, es decir, usando un vector std_logic_vector de tamaño adecuando.

Los argumentos cambiarán según las derivaciones que el procedimiento controla (como se puede ver en las reglas de selección Sel[1] – Sel[6]). Los siguientes prototipos son relativos a la mayoría de los casos generales (Nota: aunque el ejemplo es un "select" (seleccionar), las mismas reglas son también evidentemente válidas para "deselect"· (deseleccionar).

1) select(tributary_nmb : in std_logic_vector,
affluent_nmb : in std_logic_vector) 2) select(tributary_name : in string,
affluent_name : in string)

La herramienta de generación de prueba automática solo tendrá que rellenar el nombre(número. En el caso de funciones de selección más precisas, solo se necesita el uso de algunos argumentos (por ejemplo, las funciones de deselección del dispositivo de conmutación jerárquica de la figura 9), que no tienen eventualmente ninguno si el propio nombre ya identifica de manera unívoca el objetivo (por ejemplo, la SIB de la figura 8.)

Las siguientes reglas (Reglas [34] a [41]) se basan e VHDL 93, y son compatibles con el mismo, pero son sensiblemente más sencillas. Todas las derivaciones no relacionadas directamente con NSDL se han eliminado. Cabe señalar que las Reglas [34] a [39] son reglas sintácticas de VHDL clásico, mostradas aquí solo para describir las nuevas reglas contextuales.

[34] library_unit ::=
primary_unit

| secondary_unit

[35] primary_unit ↑(H,n)↑(P_info)↑(Cross)↑(Par) ::=
entity_declaration↑(H,n)
| configuration-declaration
| package-declaration
| IP_declaration ↑(H,n)↑(P_info)↑(Ext)↑(Cross)↑(Par)
| Instrument_declaration ↑(H,n)↑(P_info)↑(Ext)↑(Cross)↑(Par)
Nota: esta es la regla donde NSDL se integra con VHDL, permitiendo la definición de IPs e Instrumento como entidades de nivel superior. Asimismo es el punto donde la herramienta de prueba completa el análisis jerárquica, cuya información se almacena en (H,n), (P_info), (Cross) y (Par).
Regla: Ext = ø, si no error (que es el módulo superior)

[36] secondary_unit ::=
architecture_body
| package_body

[37] entity_declaration ↑(H,n)::=
ENTITY <identifier> IS
entity_header
entity_declarative_part
[BEGIN
architecture_body ↑(H,n)]
END [entity] [entity_simple name];
Regla: Esta regla describe los componentes internos de la ruta de exploración de la entidad definida, a usar por el compilador en el momento de la instanciación . Se puede usar para describir entidades no conforme con P1687 que tienen una ruta de exploración pero no un conjunto de funciones/ procedimientos. Obsérvese que las entidades permanecen como en el VHDL clásico, por lo tanto no permiten dependencias externas.

[38] architecture_body ↑(H,n) ::=
ARCHITECTURE <architecture_simple_name> OF entity_name IS
architecture_declarative_part
BEGIN
architecture_statement_part ↑(H,n)↑(P,F)
END [architecture] [<architecture_simple_name>];
Regla: Verificar la integridad de la cadena de exploración definida por (P,F) (empezando en TDI, terminando en TDO, sin agujeros, lineal aparte de la jerarquía, etc..).

[39] architecture_statement_part ↑(H,n)↑(P,F)::=
[component_instantiation_statement ↑(Hi,ni)↑(P,F)i]
Regla:
H= U Hi + (H_in, H_out)i
(P,F)= U (P,F)i,
n=Σini

[40] <scan_path>↑(SP_info) ::=
<component_instantiation_statement>↑(C_info)
[<scan_path>↑(S_old)]
Regla: esta regla se puede interpretar también como una verificación contextual en la regla de VHDL en las instrucciones concurrentes que limita el desarrollo de la regla a las instancias de exploración-cadena relacionadas con las instancias de células.
SP_info = S_old U C_infor
Verificar para la integridad de ruta de exploración usando P y F dentro de C_info

[41] component_instantiation_statement ↑(C_info) ::=
instantiation_label:
instantiated_unit ↑(H,n)↑(P,F[,H_in, H_out])
[generic_map_aspect]
[port_map_aspect];
Regla: n (número de células), H(Información jerárquica), P(anterior), F(siquiente), (H,n) tomados de "instatiated_unit"

descripción en la base de datos
rutas de exploración jerárquica H_in, H_out introducidas por células de control.
C_info = (H,n) U (P,F[,H_in, H_out])

Las siguientes reglas (numeradas de [42] a [57]) incluyen reglas formales ejemplares para acceso en paralelo:

[42] <parallel_information> ↑(Par)::=
[<parallel_inputs>↑(par_in)]
[<parallel_outputs>↑(par_out)]
;
Regla: Par = par_in U par_out

[43] <parallel_declarations>↑(Par_dec) ::=
[<parallel_connection> ↑(par_connection)]
[<alternates>↑(alter_info)]
[<fanning>↑(fanning_info)]
Regla: Par_dec = par_connection•fanning_info•alter_info

[44] <parallel_inputs>↑(par_in) ::=
<single_parallel_input>↑(idf)
| <numbered_par_inputs>↑(par_ins);
Regla: par_in = idf
| par_in = par_ins

[45] <single_parallel_input>↑(idf)::=
PARALLEL_IN: STRING:= "deselected" ;
Regla: idf = "parallel_in"

[46] <numbered_par_inputs>↑(par_in) ::=
<numbered_par_input>↑(idf)
[; <numbered_par_inputs>↑(par_ins)]
Regla: par_in = idf U par_ins

[47] <numbered_par_input>↑(idf)::=
PARALLEL_IN_<numeral> : STRING := "deselected"
Regla: idf = "parallel_in_<numeral>"

[48] <parallel_outputs> ↑(par_out)::=
<single_parallel_output>↑(idf)
| <numbered_par_outputs> ↑(par_outs);
Regla: par_out = idf
| par_out = par_outs

[49] <single_parallel_output>↑(idf)::=
PARALLEL_OUT : STRING := "deselected" ;
Regla: idf = "parallel_out"

50] <numbered_par_outputs>↑(par_outs) ::=
<numbered_par_output> ↑(idf)
[; <numbered_par_outputs>↑(old_par_outs)]
Regla: par_outs = idf U old_par_outs

[51] <numbered_par_output>↑(idf) ::=
PARALLEL_OUT_<numeral> : STRING := "deselected"
Regla: idf = "parallel_out_<numeral>"

[52] <fanning>↑(fanning_info) ::=
<port_name>↑(idf) FAN <provenance>↑(provenance_info);
Regla: fanning_info = idf U provenance_info

[53] <port_name>↑(idf) ::= <single_parallel_output>↑(idf)
| <numbered_par_outputs>↑(idf)
| <single_parallel_input>↑(idf)

| <numbered_par_inputs>↑(idf)
[54] <provenance> ↑(provenance_info) ::=
<port_name>↑(idf)[&<provenance>↑(old_prov)]
Regla: provenance_info = old_prov U idf
Nota: "identifier" (identificador)

Esta regla permite la descripción de entradas en abanico y salidas en abanico (ilustradas y descritas respecto de la figura 28), sin ninguna restricción en los puertos paralelos usados para la composición. La composición se realiza según las reglas de concatenación de las señales de VHDL (símbolo '&'), usando los puertos completos.

[55] <alternates>↑(alter_info) ::=
<serial_reg>↑(idf) IS ALTERNATE OF <alter_regs>↑(idf_list);
Regla: alter_info = idf U idf_list

[56] <register_list>↑(idf_list) ::=
<parallel_reg>↑(idf) [,<register_list>↑(old_idf_list)];
Regla: idf_list = idf U old_idf_list
"parallel_reg" es un identificador de VHDL clásico

[57] <parallel_connection>↑(par_connection) ::=
<port_name>↑(idf) CONNECTS <register_list>↑(idf_list);
Regla: par_connection = idf U idf_list

Como se describe en el presente documento, de manera similar a los dispositivos de cruce, la interfaz paralela explota algunas reglas de denominación para identificar recursos clave. Los elementos que necesitan denominación incluyen:

Rodajas paralelas: Una rodaja a la cual se puede acceder por una conexión paralela tiene su nombre que empieza con “parallel_”. Según los esquemas de conexión, estas rodajas pueden ser completamente independientes de la ruta de exploración en serie o parte de la ruta de exploración en serie.

Funciones paralelas: El acceso a los recursos paralelos se obtiene a través de dos funciones específicas: “get_parallel_data” y “send_parallel_data”.

Como se describe en el presente documento, hay tres modos posibles de sincronización para una interfaz paralela: sincronizada con la cadena de exploración, ráfaga, y asíncrona. En una realización la conmutación entre estos modos se realiza por funciones específicas que, como para las funciones de selección de cruce, especifican la manera en que se debe cambiar el flujo de bits, si fuese necesario. Estas funciones incluyen:

la función set_scan_synchro return boolean;
la función set_burst(length : in burst-length-type) return boolean;
la función set_asynchro return boolean;
la función disable_port return boolean;

La herramienta de prueba puede conocer fácilmente qué modo está activo manteniendo el seguimiento de las llamadas a estas funciones. Un dispositivo solamente declarará las funciones para los modos que realmente implementa.

El tipo “burst_length_type” se debe definir dentro de la interfaz paralela, como un subtipo de número entero, de manera que el desarrollador puede indicar el intervalo de valores permitido para la ráfaga. Los ejemplos incluyen: “subtype burst_length_type es el intervalo de número enteros de 3 a 10” type burst_length_type es (6,8, 10), y similar). Esta solución significa que cada Interfaz paralela declarará su propio “burst_length_type”, que será válido solo localmente y por lo tanto no interferirá con otras eventuales interfaces.

En una realización en la cual una interfaz paralela tiene más de un puerto, el nombre del puerto al cual se refiere la función se adjuntará al nombre de función. Los ejemplos incluyen: “ser_scan_synchro_parallel_out_0”, “disable_port_parallel_in”, y similar.

Las anteriores reglas de BNF constituyen meramente ejemplos de reglas que se pueden usar para implementar BSDL. La presente invención no se destina a limitarse a o por tales reglas.

La figura 30 ilustra un diagrama de bloques de alto nivel de un ordenador universal apropiado para su uso en la realización de las funciones descritas en e presente documento. Como se representa en la figura 30, el sistema 3000 comprende un elemento procesador 3002 (por ejemplo una CPU), una memoria 3004, por ejemplo una memoria de acceso aleatorio (RAM), y/o una memoria de solo lectura (ROM), un módulo de prueba 3005, y varios dispositivos de

entrada/salida 3006 (por ejemplo, dispositivos de almacenamiento, incluyendo pero no limitándose a, una unidad de cinta magnética, unidad de disquete, una unidad de disco duro o una unidad de disco compacto, un receptor, un transmisor, un altavoz, una pantalla de visualización, un puerto de salida, y un dispositivo de entrada de usuario (tal como un teclado, un teclado numérico, un ratón, y similar)).

Cabe resaltar que la presente invención se puede implementar en software y/o en una combinación de software y hardware, por ejemplo usando circuitos integrados para aplicaciones específicas (ASIC), un ordenador universal o cualesquiera otros equivalentes de hardware. En una realización, el presente proceso de prueba 3005 se puede cargar en la memoria 3004 y ejecutar por el procesador 3002 para implementar las funciones examinadas anteriormente. Asimismo, el proceso de prueba 3005 (incluyendo estructuras de datos asociadas) de la presente invención se puede almacenar en un medio legible por ordenador, por ejemplo memoria RAM, unidad magnética u óptica o disquete, y similar.

Aunque en el presente documento se ilustra y describe principalmente respecto de aplicaciones específicas de dispositivos de sistema en chip que se pueden describir y probar usando NSDL, se pueden describir y probar otros varios dispositivos de sistema en chip usando NSDL. Aunque en el presente documento se ilustra y describe principalmente respecto del uso de NSDL para describir y probar dispositivos de sistema en chip, se pueden describir y probar otros varios circuitos electrónicos usando NSDL. La presente invención no está destinada a limitarse a describir y probar los circuitos electrónicos específicos ilustrados y descritos en el presente documento.

Aunque en el presente documento se ilustra y describe principalmente respecto de aplicaciones específicas de un sistema de prueba que se puede utilizar y probar dispositivos de sistema en chip usando NSDL, se pueden utilizar otras varias aplicaciones de sistemas de prueba para describir y probar dispositivos en chip usando NSDL La presente invención no está destinada a limitarse a aplicaciones específicas de sistemas de prueba ilustrados y descritos en el presente documento.

Se contempla que algunas de las etapas examinadas en el presente documento se pueden aplicar como procedimientos de software en el hardware, por ejemplo, como circuitos que cooperan con el procesador para llevar a cabo diversas etapas del procedimiento. Las partes de la presente invención se pueden aplicar como un programa informático en el cual las instrucciones informáticas, cuando son procesadas por un ordenador, adaptan el funcionamiento del ordenador para de este modo invocar o de lo contrario proporcionar los procedimientos y/o técnicas de la presente invención. Las instrucciones para invocar los procedimientos de la invención se pueden almacenar en soportes fijos o removibles, transmitir por un flujo de bits en un soporte de difusión u otro soporte portador de señales, y/o almacenar dentro de una memoria dentro de un dispositivo informático que funcionan según las instrucciones.

Aunque se han mostrado y descrito varias en detalle realizaciones que incorporan las enseñanzas de la presente invención en el presente documento, el experto en la técnica puede elaborar fácilmente muchas otras variadas realizaciones que siguen incorporando estas enseñanzas.

REIVINDICACIONES

1.- Procedimiento para probar un sistema en chip (110) que tiene una pluralidad de componentes (210) interconectados por una pluralidad de interconexiones y usar al menos una herramienta de prueba, comprendiendo el procedimiento:

a) seleccionar (1404, 1704) un componente (210) del sistema en chip;

b) obtener (1406, 1706) una descripción del componente seleccionado (210) con lo cual la descripción es una descripción algorítmica del componente (210) del sistema en chip (110), con lo cual la descripción algorítmica incluye una o más reglas de composición definidas en un formato adaptado para ser entendido por la al menos una herramienta de prueba; con lo cual la descripción algorítmica del componente (210) describe una representación de cada una de al menos una función soportada por el componente (210) con al menos un valor registro para el componente (210);

c) obtener (1410, 1710) una descripción de sistema del sistema en chip (110), con lo cual la descripción de sistema se usa para especificar flujos de bits de prueba para el sistema en chip (110) ; y

d) determinar (1712) la posición del componente seleccionado (210) dentro de los flujos de bits de prueba del sistema en chip (110).

2.- Procedimiento según la reivindicación 1, que comprende, además:

a) convertir (1708) cada función soportada por el componente (210) en valores de registros asociados a los registros del componente (210);

b) insertar (1714) valores de registro en la posición localizada del flujo de bits de entrada;

c) recuperar (1716) los valores resultantes desde la posición situada del flujo de bits de salida; y

d) procesamiento de los valores resultantes recuperados con el fin de determinar varios resultados de prueba.

3.- Procedimiento según la reivindicación 1 o 2, en el cual, para cada componente del sistema en chip (110), la al menos una función comprende al menos una función para probar el componente (210) y una función para ejecutar una operación en el componente (210).

4.- Procedimiento según la reivindicación 1 o 2, en el cual, para cada componente (210), la descripción algorítmica del componente (210) describe una ruta de exploración interna del componente (210).

5.- Procedimiento según la reivindicación 1 o 2, en el cual la descripción algorítmica del sistema en chip (110) comprende una descripción de una topología del sistema en chip (110).

6.- Procedimiento según la reivindicación 5, en el cual la descripción de la topología del sistema en chip (110) comprende una descripción de una ruta de exploración del sistema en chip (110)

7.- Procedimiento según la reivindicación 6, en el cual la descripción de la ruta de exploración del sistema comprende información de longitud de ruta de exploración e información jerárquica de ruta de exploración.

8.- Procedimiento según cualquier reivindicación anterior, en el cual la descripción algorítmica de la composición de los componentes comprende descripciones algorítmicas de los componentes respectivos de la composición y una descripción de interconexiones entre los componentes de la composición; comprendiendo, además, el procedimiento las etapas de:

- recibir, para cada componente de la composición, una descripción algorítmica de acceso al componente, en el cual la descripción algorítmica de acceso al componente comprende una descripción algorítmica de uno o más mecanismos de acceso configurados para su uso en el acceso al componente y una descripción de una o más interconexiones entre el o los mecanismos de acceso y el componente; y
- almacenar la descripción algorítmica de la composición y las descripciones algorítmicas de acceso a los componentes para su uso en la prueba de al menos una parte del sistema.

9.- Aparato para probar un componente de un sistema en chip (110), en el cual el procedimiento de cualquiera de las reivindicaciones 1 a 8 se puede cargar y el procedimiento se puede ejecutar por el procedimiento.

10.- Medio legible por ordenador que almacena instrucciones que, cuando se cargan en una memoria (3004) y se ejecutan por un procesador (3002), hacen que el procesador (3002) lleve a cabo el procedimiento de cualquiera de las reivindicaciones 1 a 8.

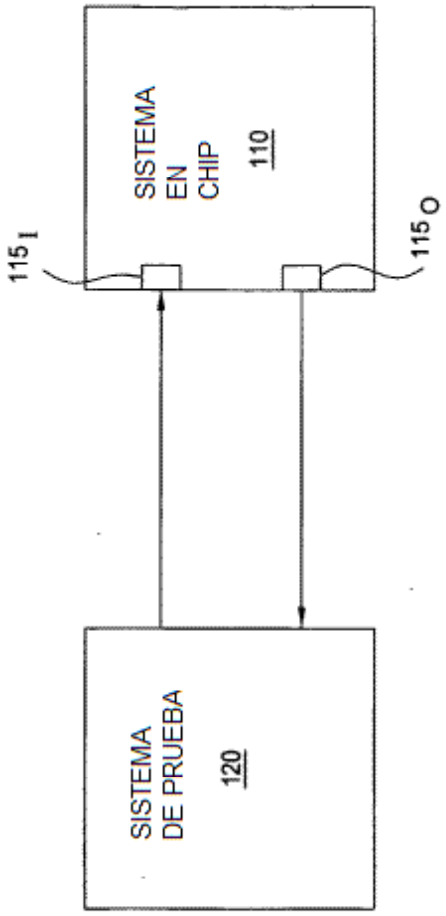


FIG. 1

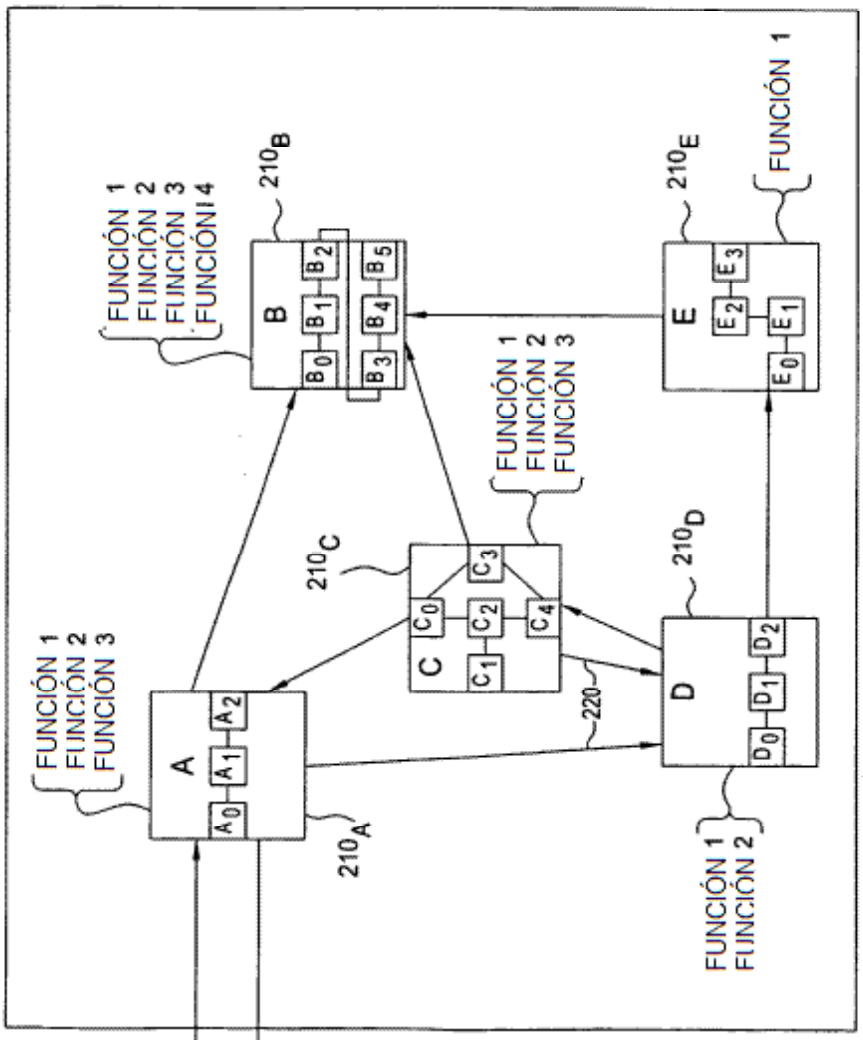

110
TDI
TDO
A
A0
A1
A2
FUNCIÓN 1
FUNCIÓN 2
FUNCIÓN 3
210A
B
B0
B1
B2
B3
B4
B5
FUNCIÓN 1
FUNCIÓN 2
FUNCIÓN 3
FUNCIÓN 4
210B
C
C0
C1
C2
C3
C4
FUNCIÓN 1
FUNCIÓN 2
FUNCIÓN 3
210C
D
D0
D1
D2
220
FUNCIÓN 1
FUNCIÓN 2
210D
E
E0
E1
E2
E3
FUNCIÓN 1
210E

FIG. 2

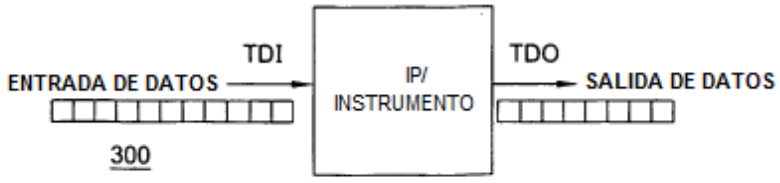


FIG. 3

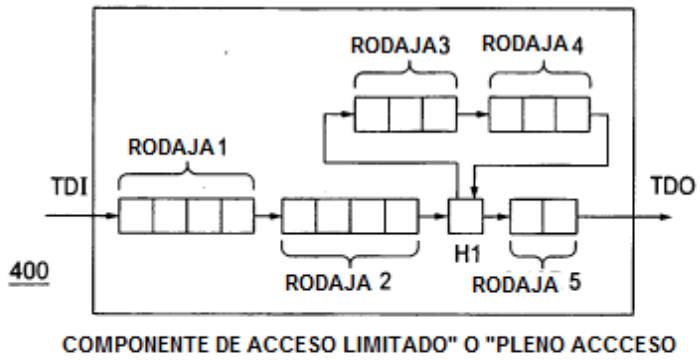


FIG. 4

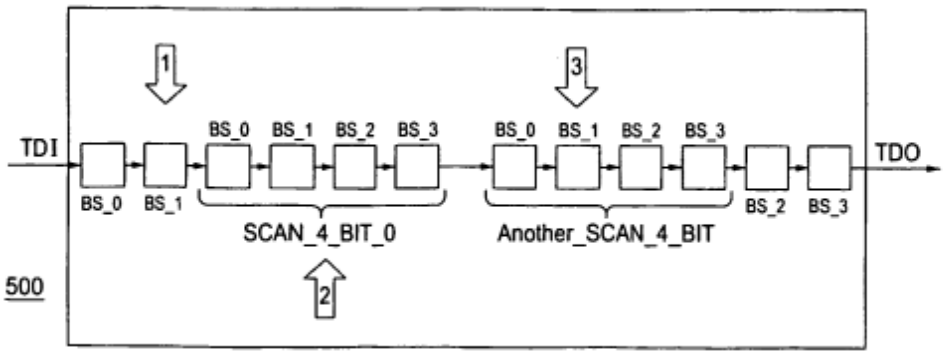
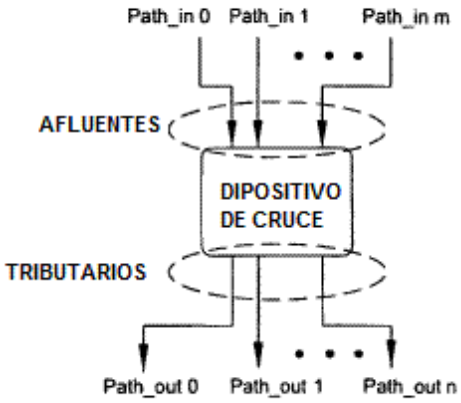
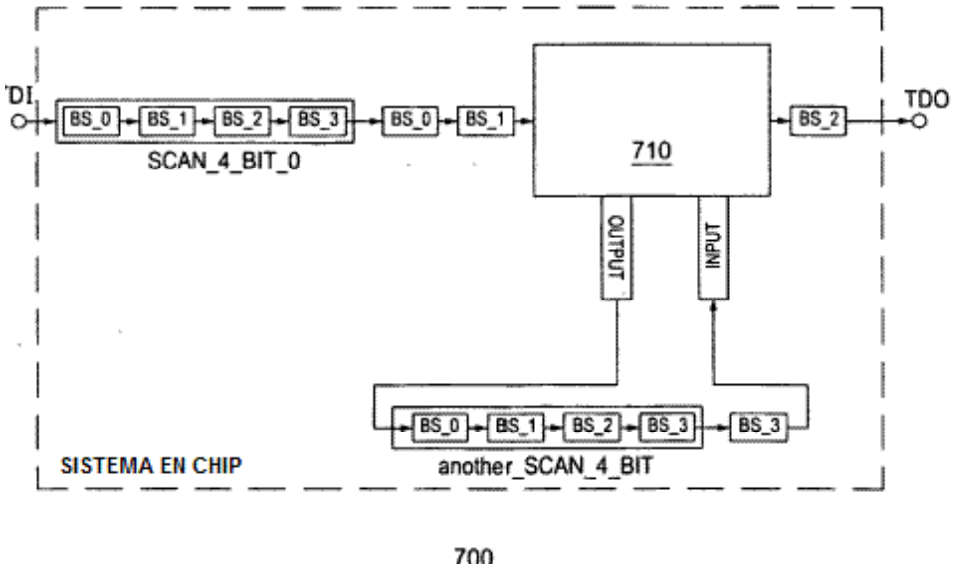


FIG. 5



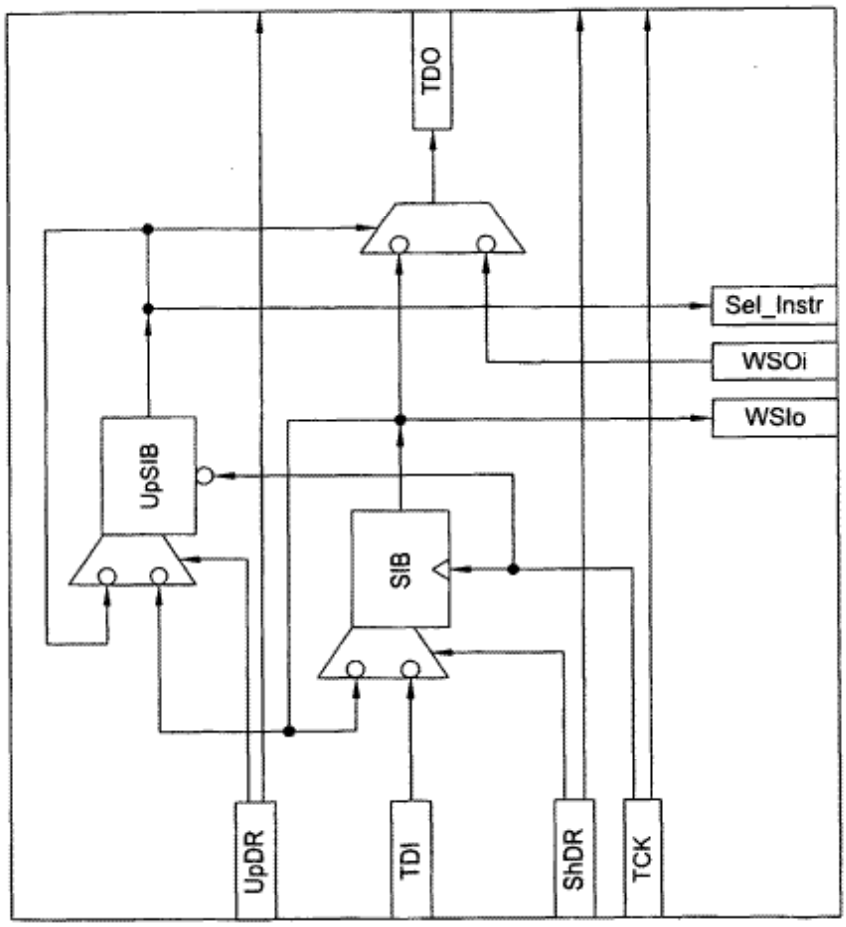
600

FIG. 6



700

FIG. 7

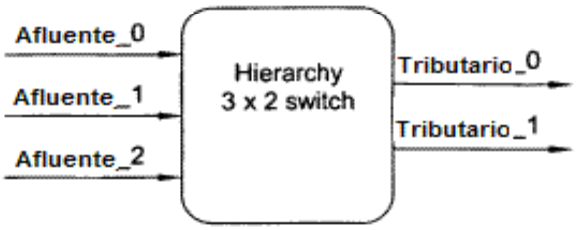


800

FIG. 8

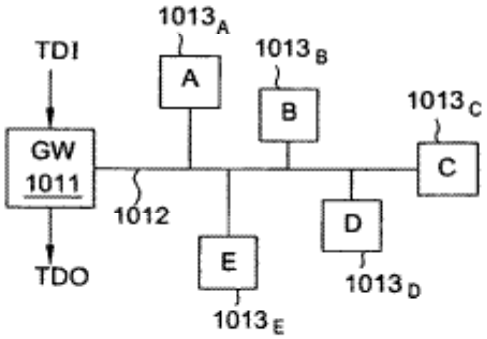
900

FIG. 9



1000

FIG. 10



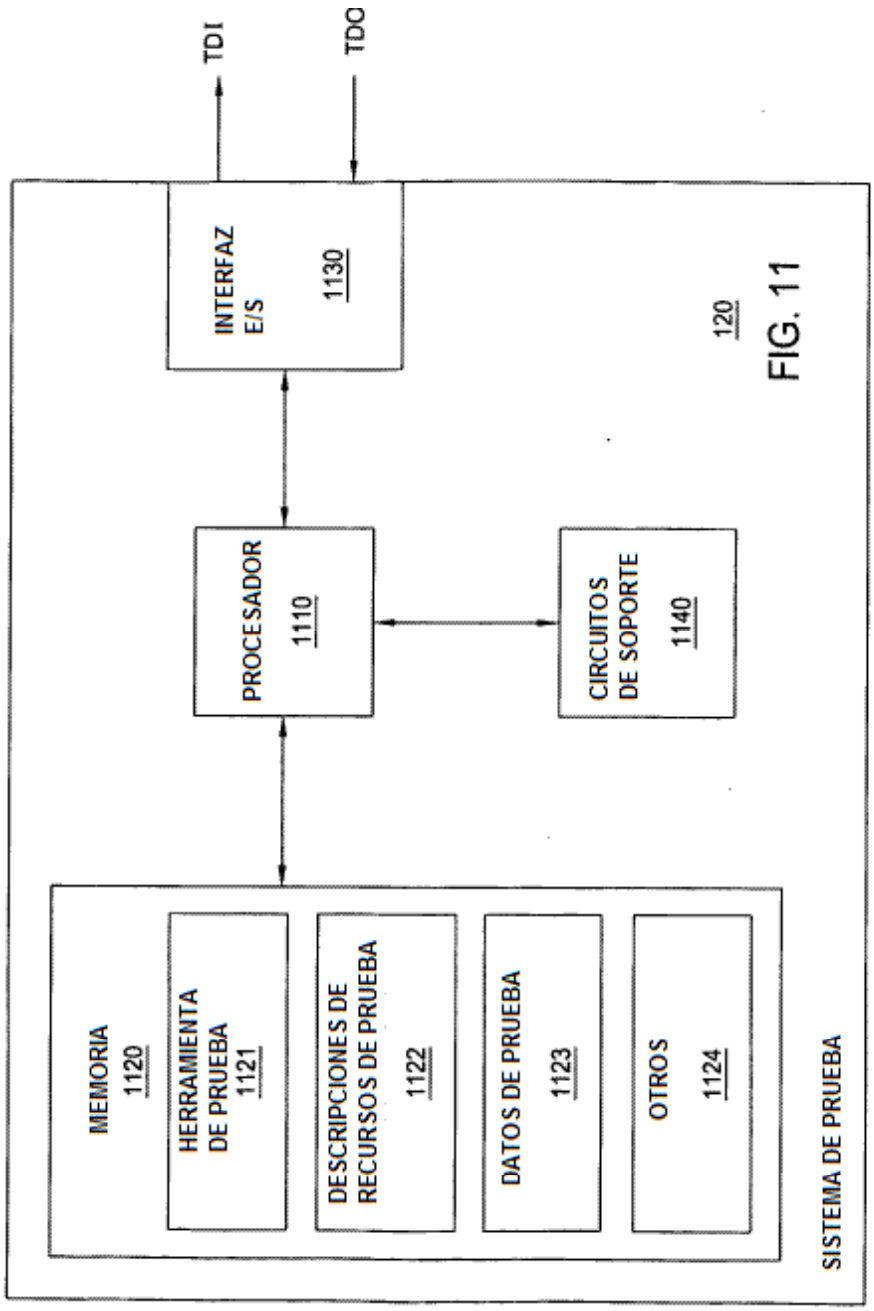
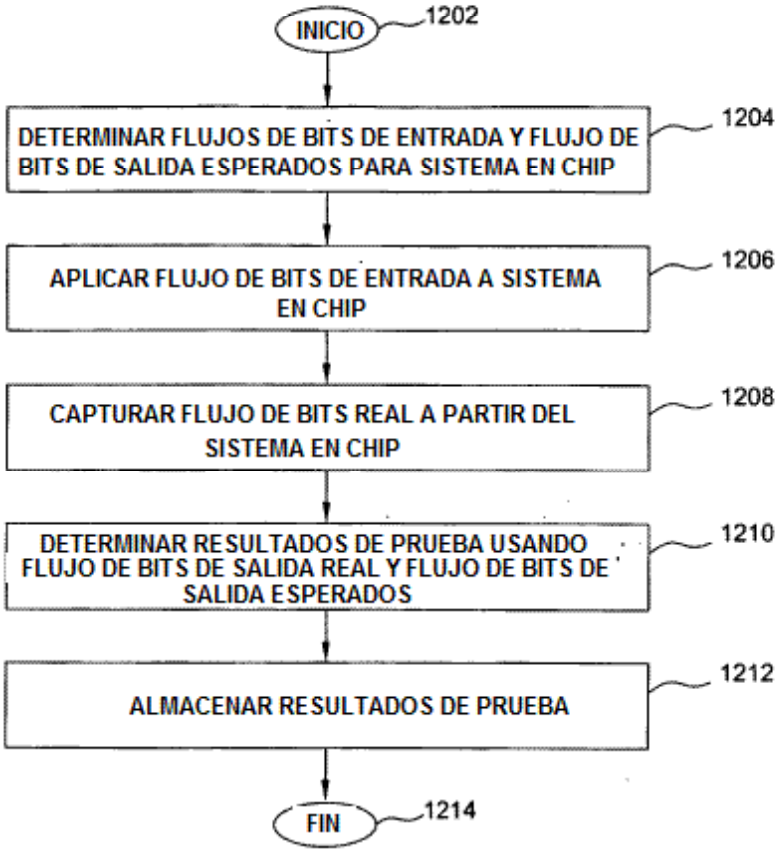

TDI
TDO
INTERFAZ E/S
1130
PROCESADOR
1110
CIRCUITOS DE SOPORTE
1140
MEMORIA
1120
HERRAMIENTA DE PRUEBA
1121
DESCRIPCIONES DE RECURSOS DE PRUEBA
1122
DATOS DE PRUEBA
1123
OTROS
1124
120
SISTEMA DE PRUEBA

FIG. 11


INICIO
1202
DETERMINAR FLUJOS DE BITS DE ENTRADA Y FLUJO DE BITS DE SALIDA ESPERADOS PARA SISTEMA EN CHIP
1204
APLICAR FLUJO DE BITS DE ENTRADA A SISTEMA EN CHIP
1206
CAPTURAR FLUJO DE BITS REAL A PARTIR DEL SISTEMA EN CHIP
1208
DETERMINAR RESULTADOS DE PRUEBA USANDO FLUJO DE BITS DE SALIDA REAL Y FLUJO DE BITS DE SALIDA ESPERADOS
1210
ALMACENAR RESULTADOS DE PRUEBA
1212
FIN
1214

1200

FIG. 12

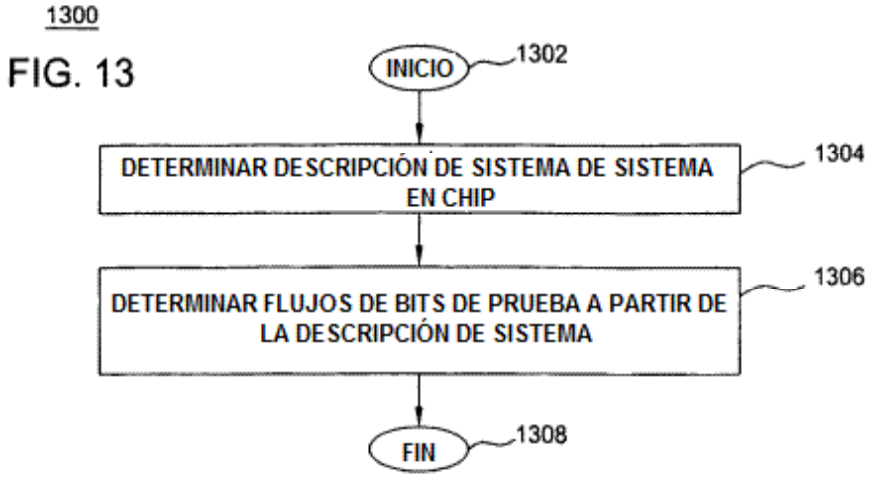

1300
INICIO
1302
DETERMINAR DESCRIPCIÓN DE SISTEMA DE SISTEMA EN CHIP
1304
DETERMINAR FLUJOS DE BITS DE PRUEBA A PARTIR DE LA DESCRIPCIÓN DE SISTEMA
1306
FIN
1308

FIG. 13

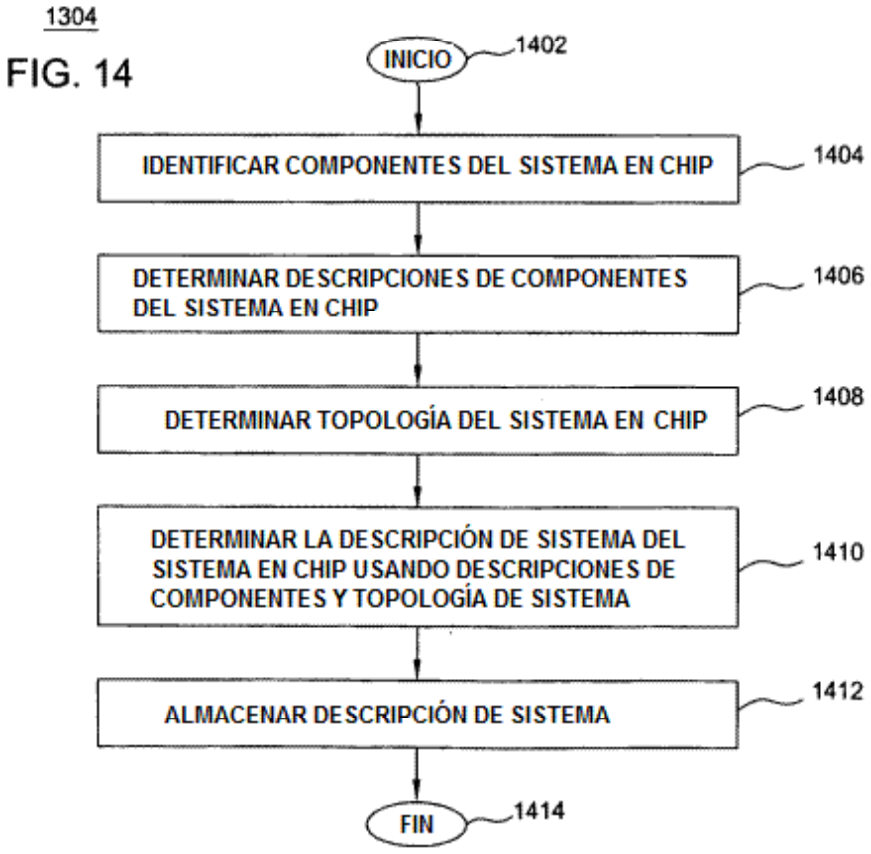
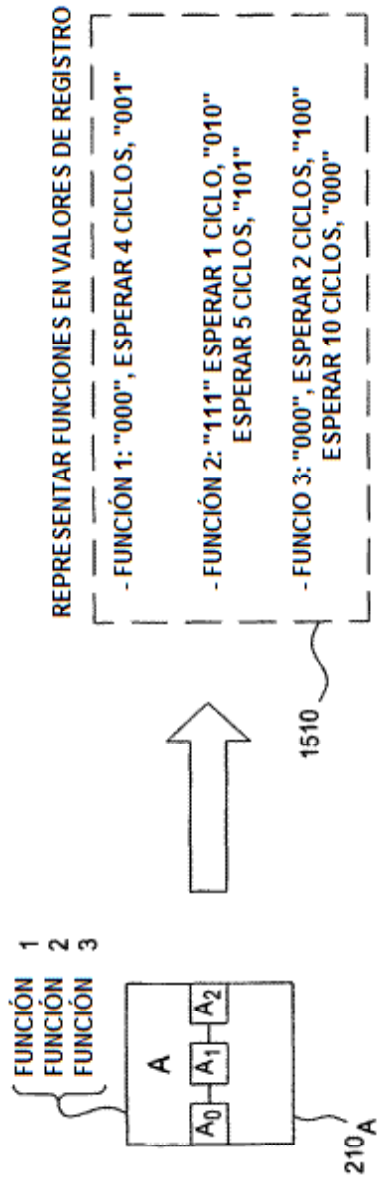
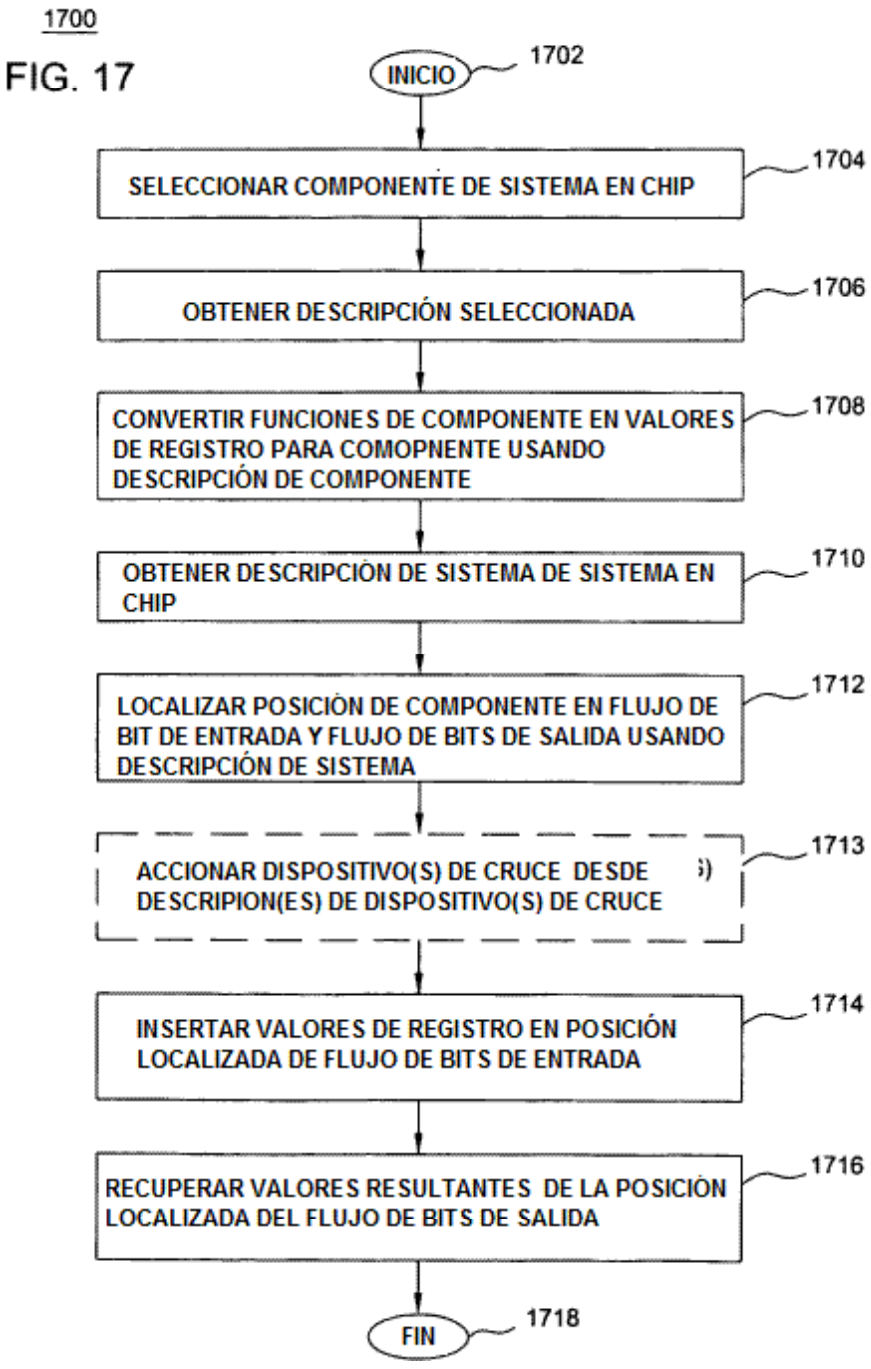

1304
INICIO
1402
IDENTIFICAR COMPONENTES DEL SISTEMA EN CHIP
1404
DETERMINAR DESCRIPCIONES DE COMPONENTES DEL SISTEMA EN CHIP
1406
DETERMINAR TOPOLOGÍA DEL SISTEMA EN CHIP
1408
DETERMINAR LA DESCRIPCIÓN DE SISTEMA DEL SISTEMA EN CHIP USANDO DESCRIPCIONES DE COMPONENTES Y TOPOLOGÍA DE SISTEMA
1410
ALMACENAR DESCRIPCIÓN DE SISTEMA
1412
FIN
1414

FIG. 14


1500
FIG. 15
FUNCIÓN 1
FUNCIÓN 2
FUNCIÓN 3
A
A_0
A_1
A_2
210_A
REPRESENTAR FUNCIONES EN VALORES DE REGISTRO
- FUNCIÓN 1: "000", ESPERAR 4 CICLOS, "001"
- FUNCIÓN 2: "111" ESPERAR 1 CICLO, "010"
ESPERAR 5 CICLOS, "101"
- FUNCIO 3: "000", ESPERAR 2 CICLOS, "100"
ESPERAR 10 CICLOS, "000"
1510

1600
FIG. 16
DESCRIBIR COMPOSICIÓN DESDE REGISTROS A FLUJOS DE BITS
FLUJO DE ENTRADA
ENTRADAS A
TAP
SALIDAS A
FLUJO DE SALIDA
A
B
C
D
E
210_A
210_B
210_C
210_D
210_E
220
110
1611_I
1611_O
1610

FIG. 17


1700
INICIO
1702
SELECCIONAR COMPONENTE DE SISTEMA EN CHIP
1704
OBTENER DESCRIPCIÓN SELECCIONADA
1706
CONVERTIR FUNCIONES DE COMPONENTE EN VALORES DE REGISTRO PARA COMOPNENTE USANDO DESCRIPCIÓN DE COMPONENTE
1708
OBTENER DESCRIPCIÓN DE SISTEMA DE SISTEMA EN CHIP
1710
LOCALIZAR POSICIÓN DE COMPONENTE EN FLUJO DE BIT DE ENTRADA Y FLUJO DE BITS DE SALIDA USANDO DESCRIPCIÓN DE SISTEMA
1712
ACCIONAR DISPOSITIVO(S) DE CRUCE DESDE DESCRIPION(ES) DE DISPOSITIVO(S) DE CRUCE
1713
INSERTAR VALORES DE REGISTRO EN POSICIÓN LOCALIZADA DE FLUJO DE BITS DE ENTRADA
1714
RECUPERAR VALORES RESULTANTES DE LA POSICIÓN LOCALIZADA DEL FLUJO DE BITS DE SALIDA
1716
FIN
1718

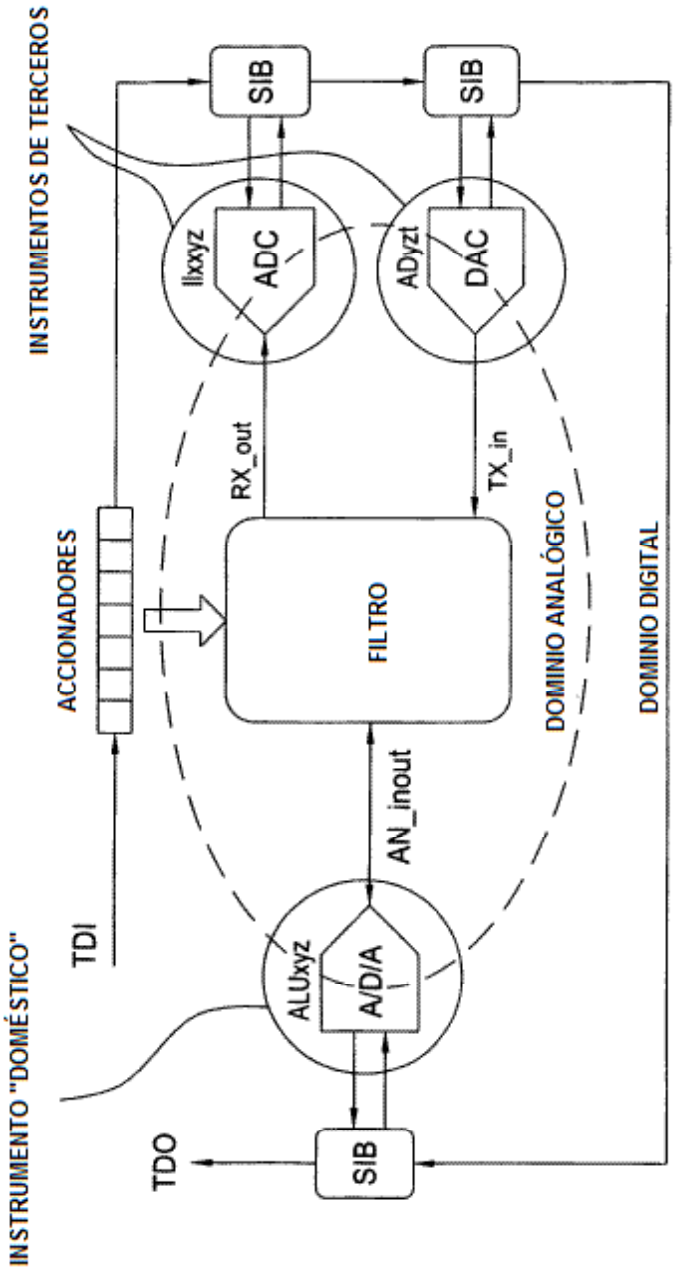

INSTRUMENTOS DE TERCEROS
SIB
SIB
IIxxyz
ADC
ADyzt
DAC
RX_out
TX_in
ACCIONADORES
FILTRO
DOMINIO ANALÓGICO
DOMINIO DIGITAL
TDI
AN_inout
ALUxyz
A/D/A
INSTRUMENTO "DOMÉSTICO"
TDO
SIB
1800

FIG. 18

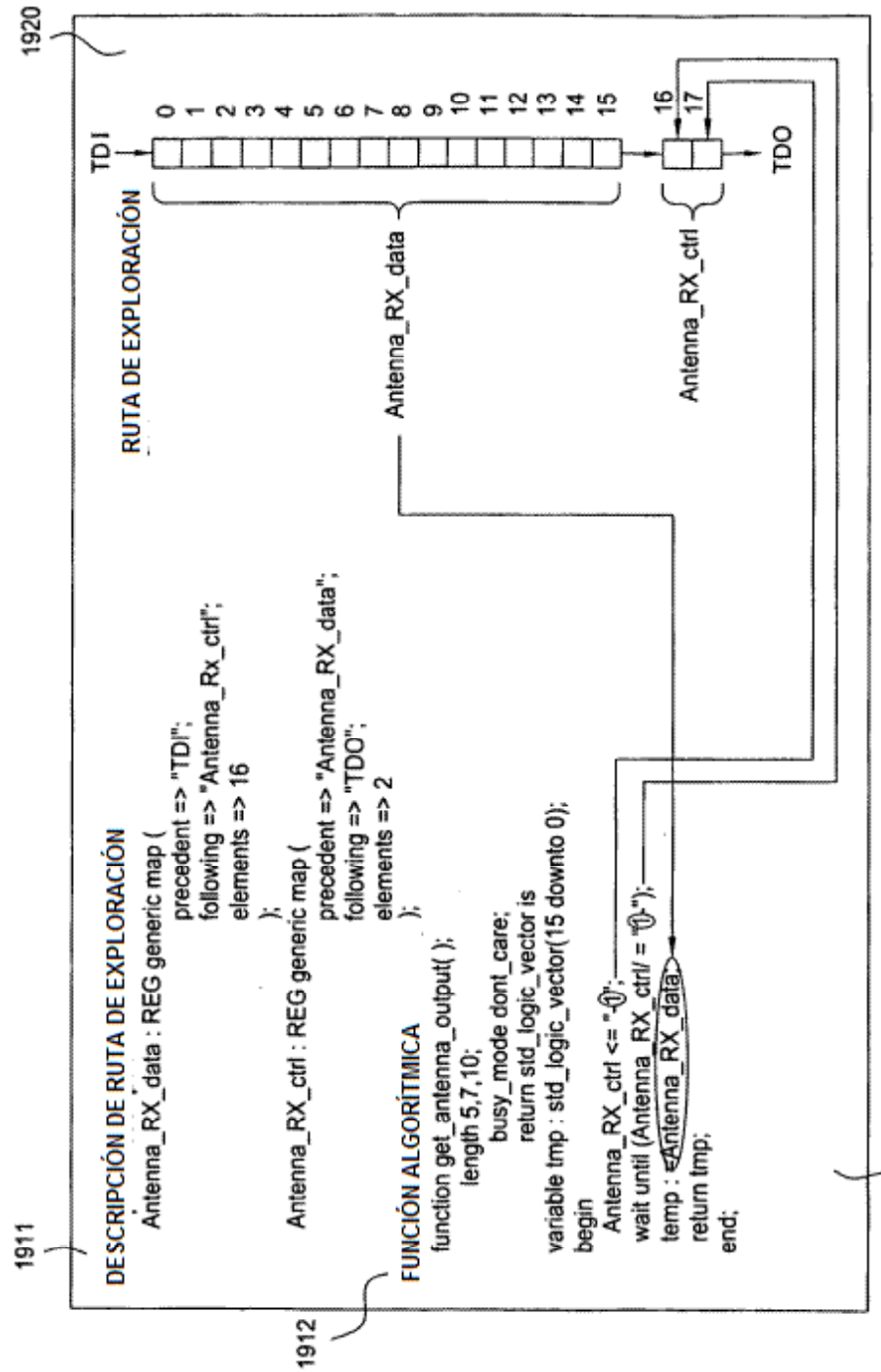

1920
TDI
RUTA DE EXPLORACIÓN
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
16 17
TDO
Antenna_RX_data
Antenna_RX_ctrl
1911
DESCRIPCIÓN DE RUTA DE EXPLORACIÓN
Antenna_RX_data : REG generic map (
precedent => "TDI";
following => "Antenna_Rx_ctrl";
elements => 16
);
Antenna_RX_ctrl : REG generic map (
precedent => "Antenna_RX_data";
following => "TDO";
elements => 2
);
1912
FUNCIÓN ALGORÍTMICA
function get_antenna_output( );
length 5,7,10;
busy_mode dont_care;
return std_logic_vector is
variable tmp : std_logic_vector(15 downto 0);
begin
Antenna_RX_ctrl <= "0";
wait until (Antenna_RX_ctrl = "0");
temp := Antenna_RX_data;
return tmp;
end;
1910
1900

FIG. 19

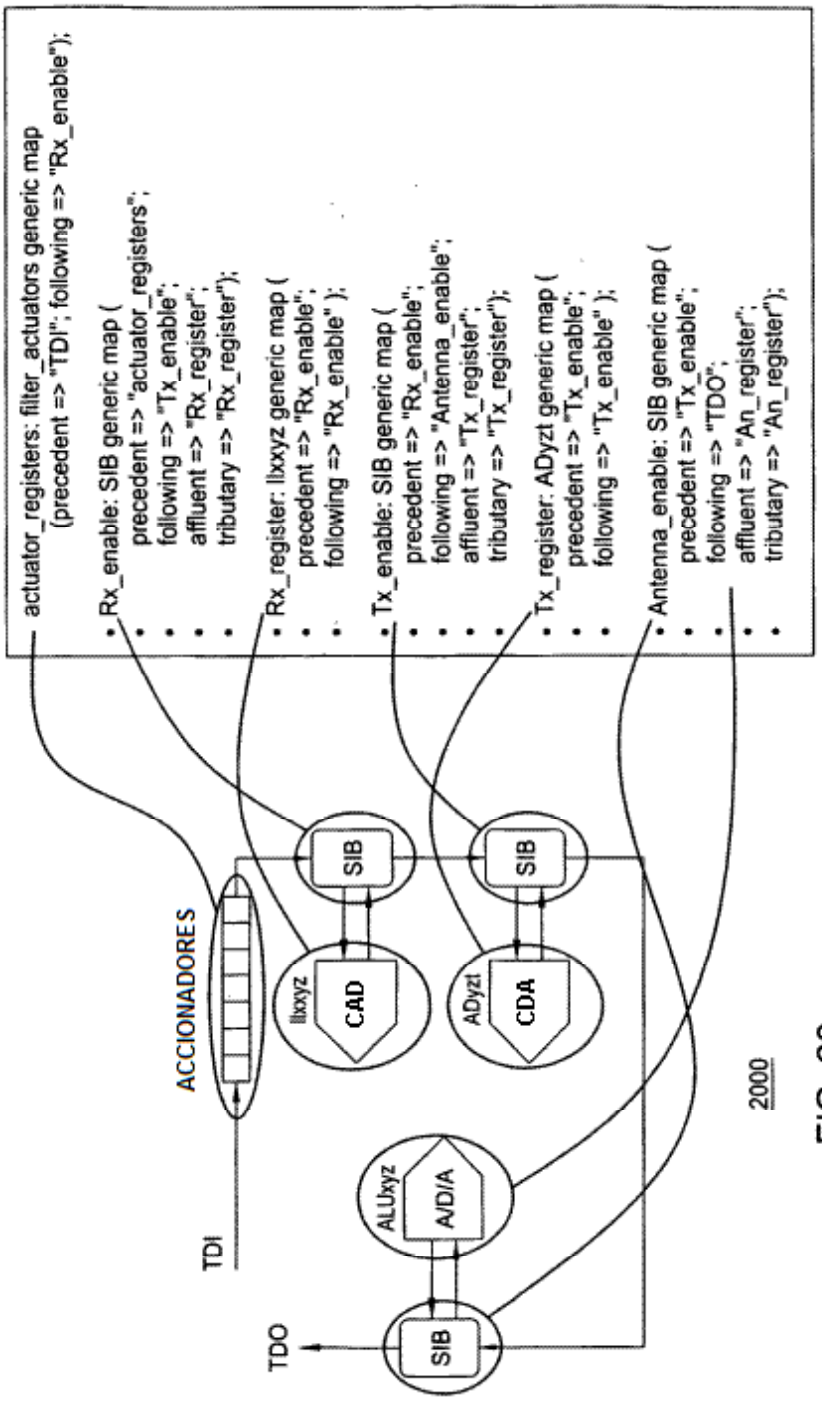

actuator_registers: filter_actuators generic map
(precedent => "TDI"; following => "Rx_enable");
• Rx_enable: SIB generic map (
• precedent => "actuator_registers";
• following => "Tx_enable";
• affluent => "Rx_register";
• tributary => "Rx_register");
• Rx_register: llxxyz generic map (
• precedent => "Rx_enable";
• following => "Rx_enable");
• Tx_enable: SIB generic map (
• precedent => "Rx_enable";
• following => "Antenna_enable";
• affluent => "Tx_register";
• tributary => "Tx_register");
• Tx_register: ADyzt generic map (
• precedent => "Tx_enable";
• following => "Tx_enable");
• Antenna_enable: SIB generic map (
• precedent => "Tx_enable";
• following => "TDO";
• affluent => "An_register";
• tributary => "An_register");
ACCIONADORES
TDI
TDO
SIB
llxxyz
CAD
SIB
ADyzt
CDA
SIB
ALUxyz
A/D/A
2000

FIG. 20

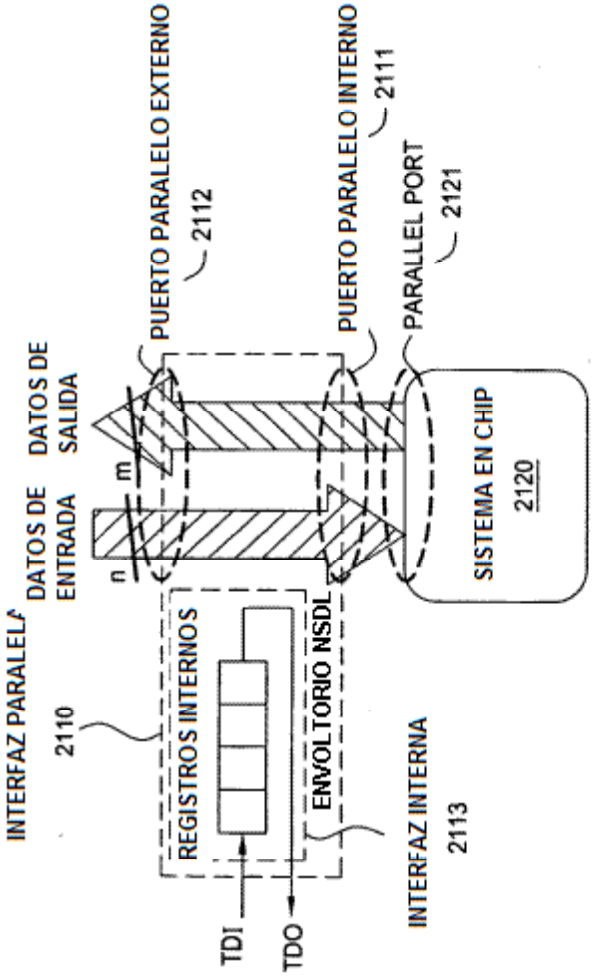

INTERFAZ PARALELA
2110
DATOS DE ENTRADA
DATOS DE SALIDA
n
m
PUERTO PARALELO EXTERNO
2112
PUERTO PARALELO INTERNO
2111
PARALLEL PORT
2121
SISTEMA EN CHIP
2120
REGISTROS INTERNOS
ENVOLTORIO NSDL
TDI
TDO
INTERFAZ INTERNA
2113
2100

FIG. 21

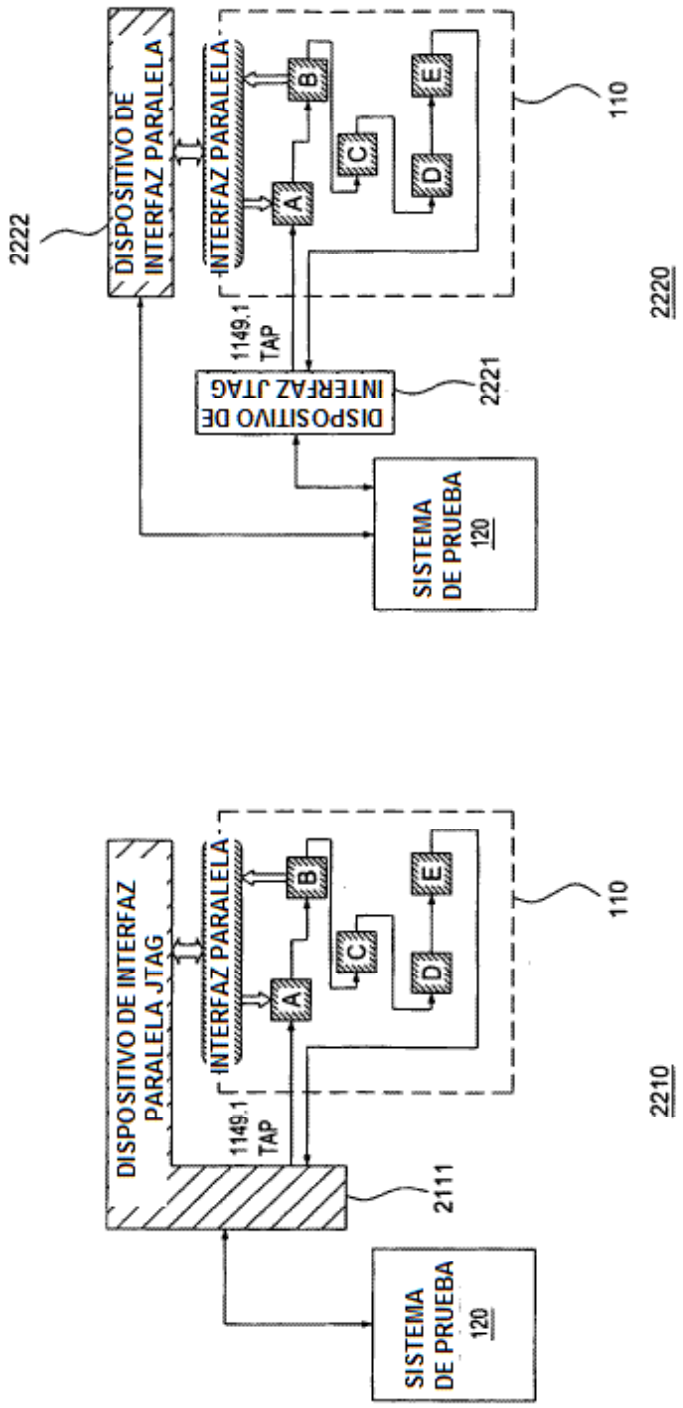

2222
DISPOSITIVO DE INTERFAZ PARALELA
INTERFAZ PARALELA
A
B
C
D
E
110
2220
1149.1 TAP
DISPOSITIVO DE INTERFAZ JTAG
2221
SISTEMA DE PRUEBA
120
DISPOSITIVO DE INTERFAZ PARALELA JTAG
INTERFAZ PARALELA
A
B
C
D
E
110
2210
1149.1 TAP
2111
SISTEMA DE PRUEBA
120

FIG. 22

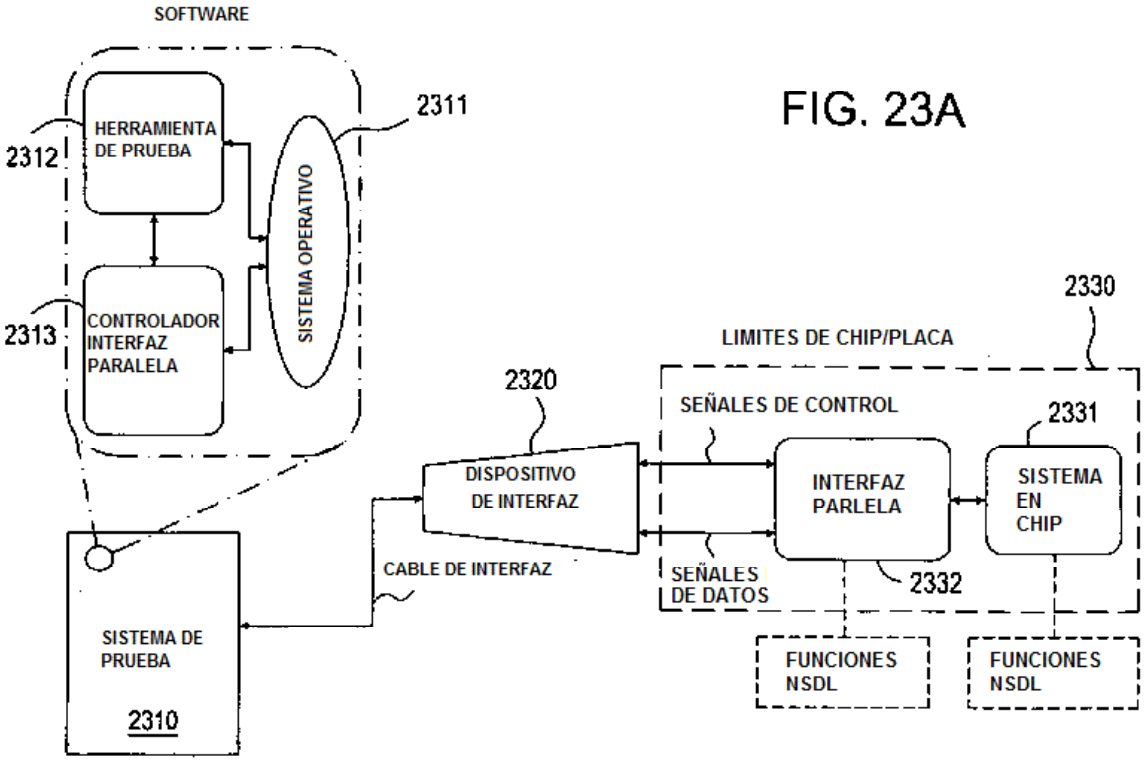


FIG. 23A

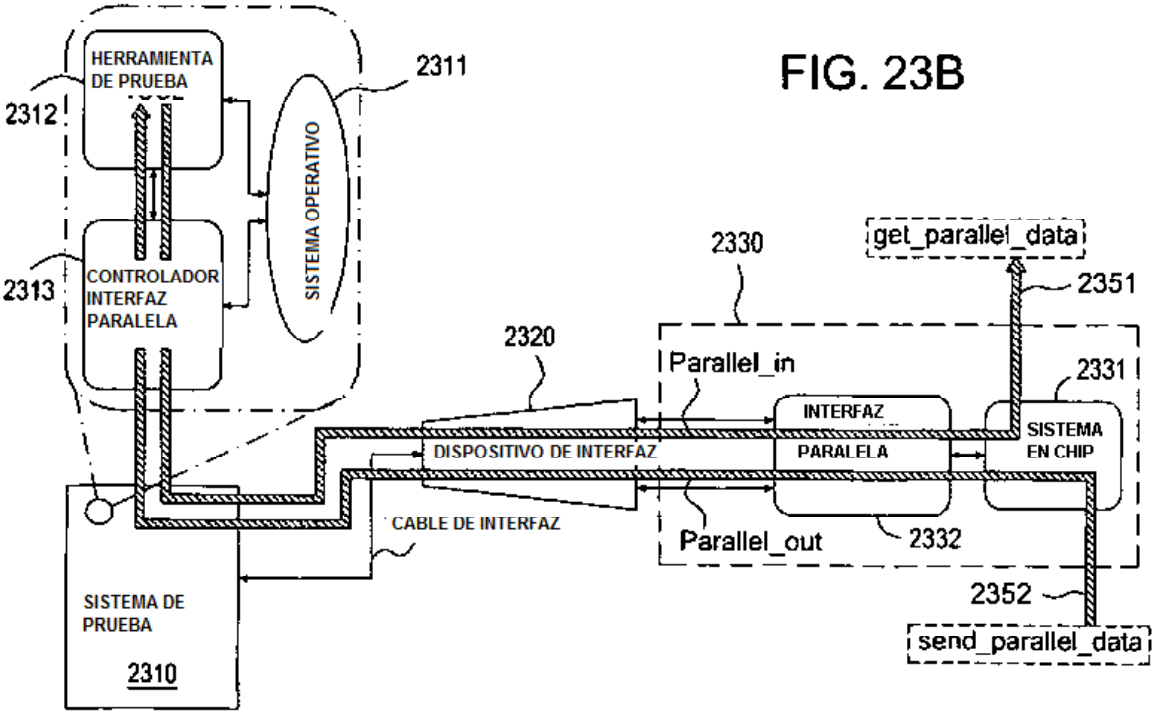


FIG. 23B

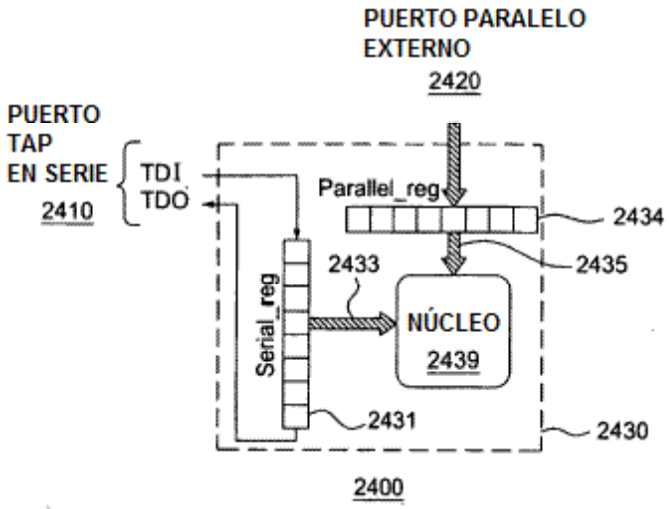

PUERTO PARALELO
EXTERNO
2420
PUERTO
TAP
EN SERIE
2410
TDI
TDO
Parallel_reg
2434
2435
2433
Serial_reg
NÚCLEO
2439
2431
2430
2400

FIG. 24

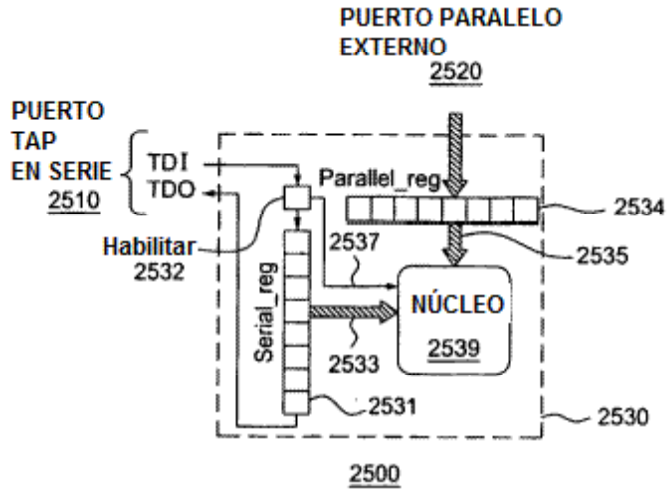

PUERTO PARALELO
EXTERNO
2520
PUERTO
TAP
EN SERIE
2510
TDI
TDO
Parallel_reg
2534
Habilitar
2532
2537
2535
Serial_reg
NÚCLEO
2539
2533
2531
2530
2500

FIG. 25

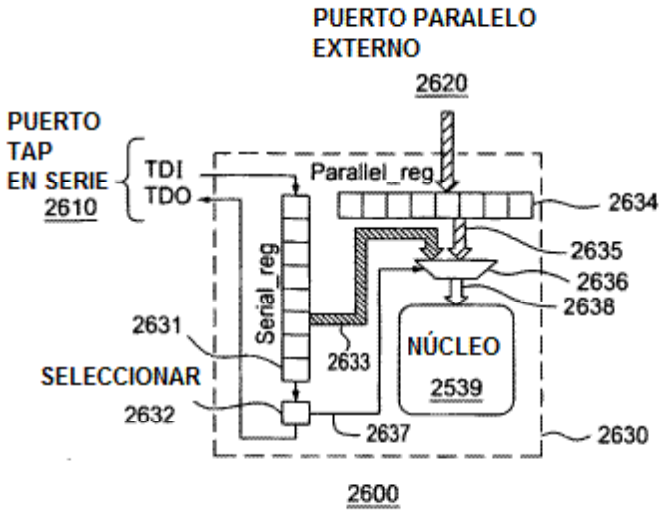

PUERTO PARALELO
EXTERNO
2620
PUERTO
TAP
EN SERIE
2610
TDI
TDO
Parallel_reg
2634
2635
2636
2638
Serial_reg
2631
2633
NÚCLEO
2539
SELECCIONAR
2632
2637
2630
2600

FIG. 26

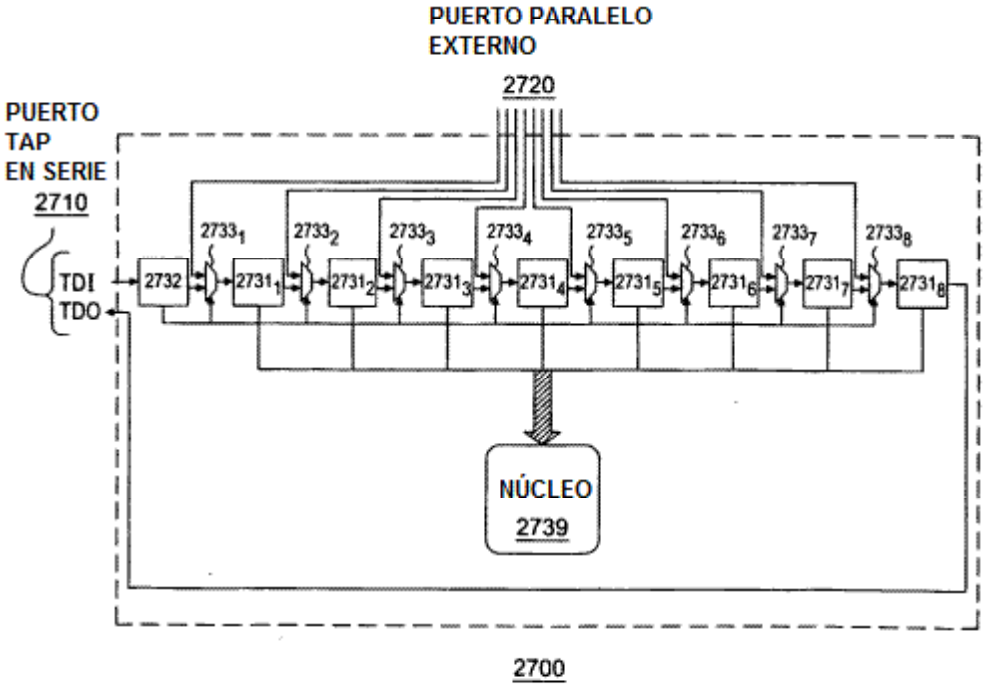

PUERTO PARALELO
EXTERNO
2720
PUERTO
TAP
EN SERIE
2710
TDI
TDO
2732
NÚCLEO
2739
2700

FIG. 27

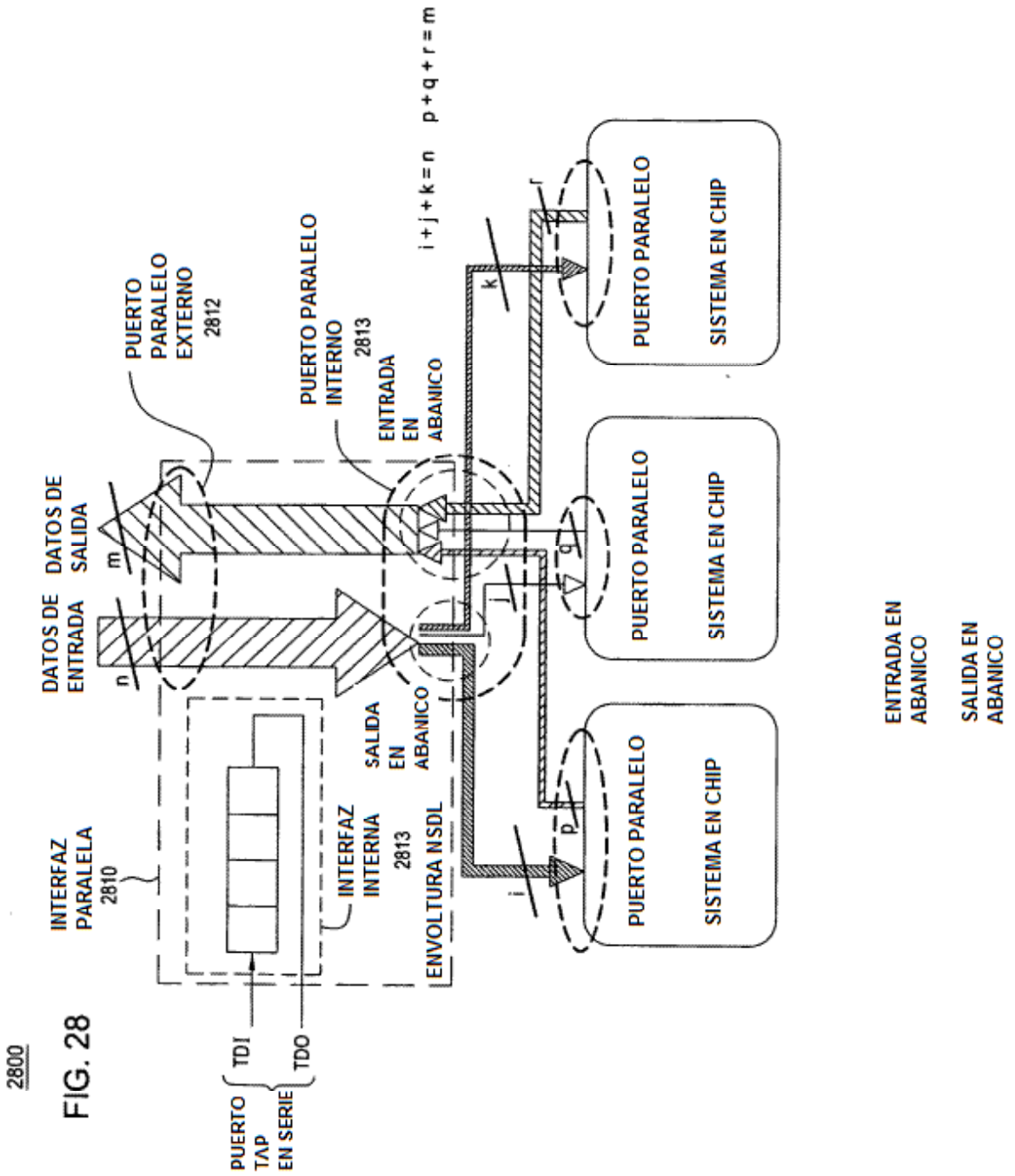

2800
INTERFAZ PARALELA 2810
DATOS DE ENTRADA
DATOS DE SALIDA
n
m
PUERTO PARALELO EXTERNO 2812
PUERTO PARALELO INTERNO 2813
ENTRADA EN ABANICO
SALIDA EN ABANICO
i + j + k = n
p + q + r = m
INTERFAZ INTERNA 2813
ENVOLTURA NSDL
PUERTO TAP EN SERIE
TDI
TDO
PUERTO PARALELO
SISTEMA EN CHIP
PUERTO PARALELO
SISTEMA EN CHIP
PUERTO PARALELO
SISTEMA EN CHIP
ENTRADA EN ABANICO
SALIDA EN ABANICO

FIG. 28

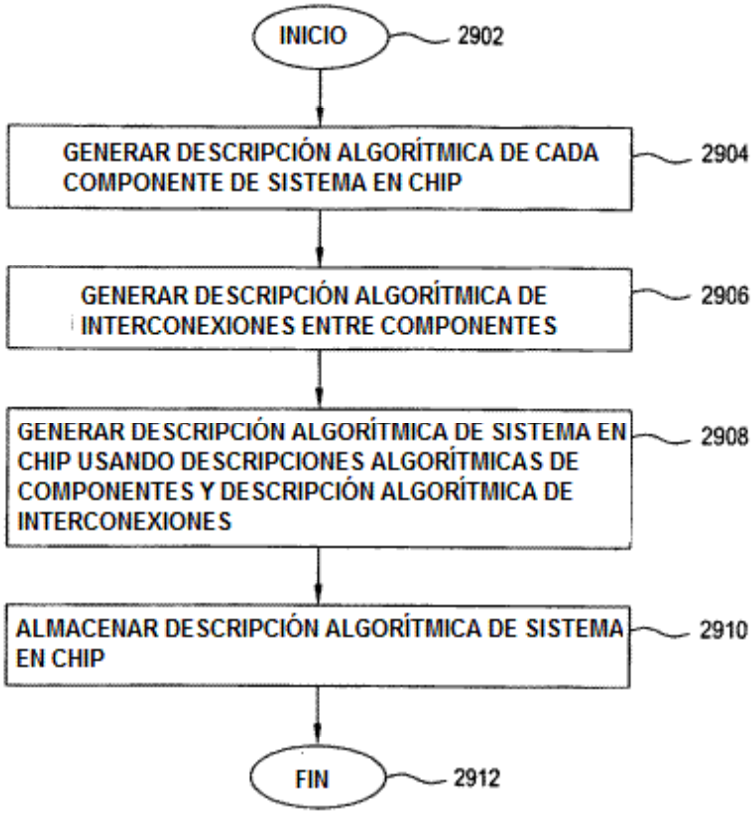

INICIO
2902
GENERAR DESCRIPCIÓN ALGORÍTMICA DE CADA COMPONENTE DE SISTEMA EN CHIP
2904
GENERAR DESCRIPCIÓN ALGORÍTMICA DE INTERCONEXIONES ENTRE COMPONENTES
2906
GENERAR DESCRIPCIÓN ALGORÍTMICA DE SISTEMA EN CHIP USANDO DESCRIPCIONES ALGORÍTMICAS DE COMPONENTES Y DESCRIPCIÓN ALGORÍTMICA DE INTERCONEXIONES
2908
ALMACENAR DESCRIPCIÓN ALGORÍTMICA DE SISTEMA EN CHIP
2910
FIN
2912
2900

FIG. 29

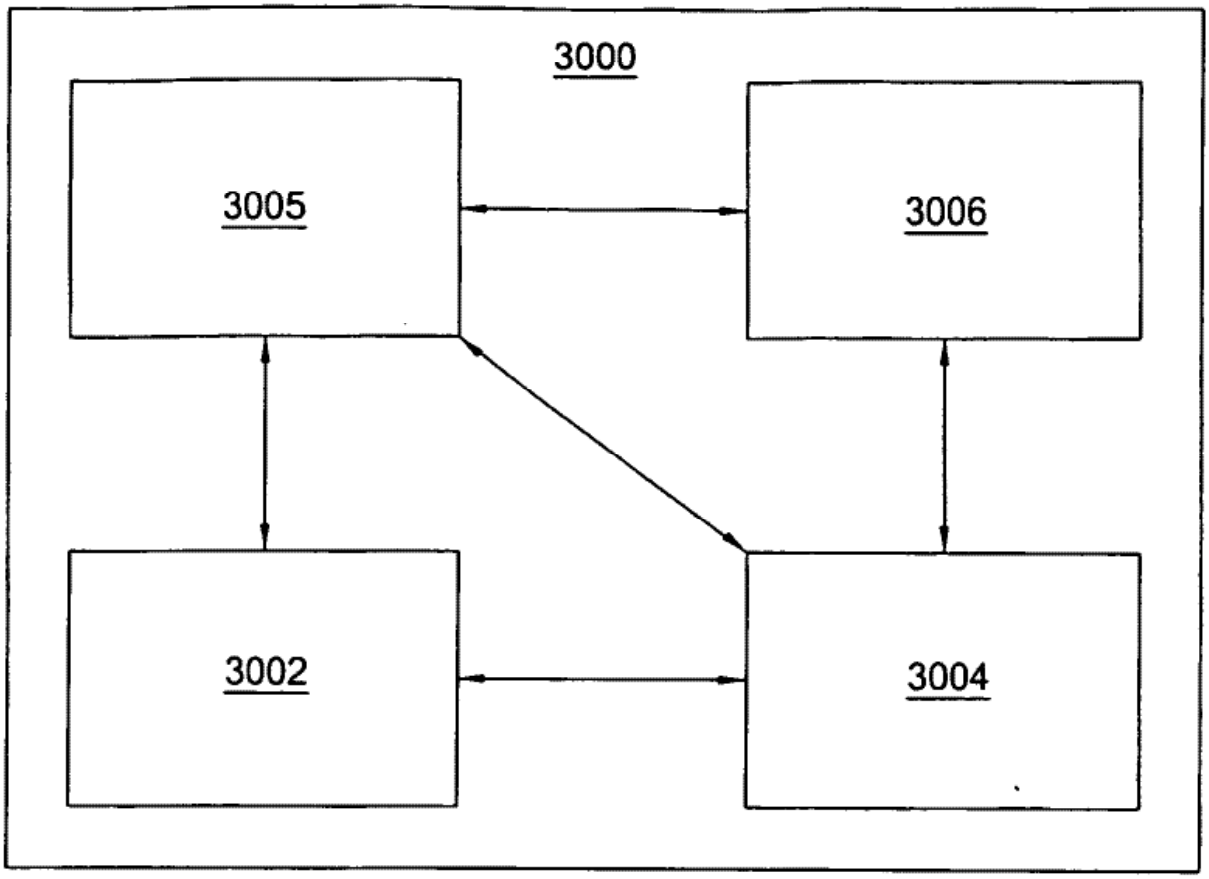


FIG. 30