(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 562 072**

(21) Número de solicitud: 201530862

(51) Int. CI.:

G06F 7/00 (2006.01)
G06F 17/00 (2006.01)

(12) PATENTE DE INVENCIÓN B1

(22) Fecha de presentación:

18.06.2015

(43) Fecha de publicación de la solicitud:

02.03.2016

Fecha de la concesión:

20.12.2016

(45) Fecha de publicación de la concesión:

28.12.2016

(73) Titular/es:

UNIVERSIDAD DE MÁLAGA (60.0%)
Avda. Cervantes, 2
29071 Málaga (Málaga) ES y
FUNDACIÓN UNIVERSITARIA SAN PABLO CEU (40.0%)

(72) Inventor/es:

HORMIGO AGUILAR, Francisco Javier;
CAFFARENA FERNÁNDEZ, Gabriel y
GARCÍA CHICO, José Manuel

(74) Agente/Representante:

CARVAJAL Y URQUIJO, Isabel

(54) Título: **Sistema y método para la optimización de anchos de palabra de circuitos digitales mediante simulaciones bit-true**

(57) Resumen:

Sistema y método para la optimización de anchos de palabra de circuitos digitales mediante simulaciones bit-true. El sistema comprende un modelo del circuito objetivo (5) con circuitos limitadores (7) que modifican el valor de las señales cuyos anchos de palabra se quiere optimizar. Cada circuito limitador (7) genera una salida limitando a nivel lógico la precisión y/o rango efectivo del valor de un dato de entrada (NUM_i) según una instrucción de control de precisión y/o rango efectivo (10_i) recibida. El modelo del circuito objetivo (5) se puede implementar en un circuito hardware de un PLD (2) o mediante software en un procesador (8). Cada circuito limitador (7) está normalmente ubicado a la entrada y/o salida de un operador aritmético-lógico (9) o de una unidad aritmético-lógica (19).

Permite reducir los tiempos de ejecución de procesos de optimización mediante simulación bit-true, evitando la reconfiguración del PLD o la emulación de anchos de palabra mediante instrucciones software.

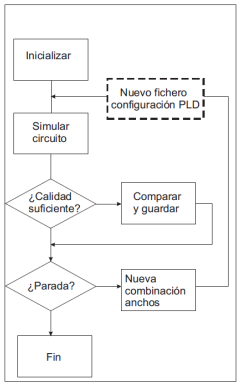

Fig. 1

ES 2 562 072 B1

Sistema y método para la optimización de anchos de palabra de circuitos digitales mediante simulaciones bit-true

DESCRIPCIÓN

Campo de la invención

La siguiente invención se refiere al diseño de sistemas digitales y más concretamente a la aceleración del proceso de la optimización de circuitos digitales mediante el uso de dispositivos hardware (o dispositivos lógicos programables).

Antecedentes de la invención

En los circuitos digitales las señales o números pueden representarse en varios formatos. Los formatos más utilizados son el formato en coma flotante (FP) y el formato de coma fija (FF). En formato de coma fija, el cual incluye los números enteros, el número de dígitos fraccionarios y dígitos enteros es fijo. En esta representación, los números negativos se representan típicamente en formato de complemento, respecto de la base. Por ejemplo para números binarios se utiliza un formato de complemento a dos. En coma flotante, el número se compone de la mantisa (Ma), la base (B) y el exponente (Ex). Por lo tanto, el valor (Va) representado sería Va = Ma * B^Ex. Entonces, solamente los números Ma y Ex necesitan almacenarse. En los números en coma flotante en bloque un grupo de números de mantisas diferentes y exponente común se procesan como un bloque de datos.

A la hora de definir una arquitectura digital es necesario determinar el formato y el número de bits (ancho de palabra) de cada una de las señales del circuito. En el caso de microprocesadores, éstos disponen de pocas unidades aritméticas que en general usan formatos dentro de los estándares más utilizados. Sin embargo, para el caso de circuitos específicos, éstos pueden tener cientos de señales con diferentes representaciones numéricas, y además diferentes anchos de palabra.

El formato y el ancho de palabra, junto con los modos de redondeo y desbordamiento, influyen en la precisión de los cálculos. Al ser imposible utilizar un números de bits infinito, siempre se producen errores de redondeo en los sistemas digitales. Cuanto mayores sean las longitudes de palabra, menores serán los errores producidos. Por lo tanto, es necesario calcular el error matemático que se produce al llevar a cabo operaciones matemáticas. El cálculo de este error se realiza tanto para arquitecturas ya existentes (microprocesador)

como para arquitecturas en fase de desarrollo (circuito específico o microprocesador). En ambos casos se intenta optimizar el uso de recursos dada una restricción de precisión.

La optimización del uso de recursos conlleva reducciones de coste asociadas al tiempo de ejecución y consumo de potencia. En el caso de circuitos en desarrollo, también añade una reducción en el área de silicio.

El proceso de optimización del ancho de palabra requiere evaluar repetidas veces el error debido al uso de longitudes de palabras reducidas. El objetivo final es encontrar un conjunto de combinaciones de anchos de palabras que minimice el coste y cumpla con las restricciones de error. La optimización de anchos de palabra es un problema NP completo por lo que se recurre al uso de métodos heurísticos iterativos. En cada iteración se van refinando los anchos de palabra hasta llegar a un punto en el que se considera que la solución es satisfactoria. Ejemplos de técnicas de optimización son las basadas en el descenso de gradiente, los algoritmos genéticos, las técnicas de recocido simulado (*simulated annealing*), etc. En general, en estos métodos es necesario realizar un gran número de iteraciones por lo que el tiempo de optimización es elevado. Esta situación se agrava especialmente si se utilizan técnicas de medida del error costosas en tiempo.

La evaluación del error puede realizarse mediante técnicas analíticas, pero estás suelen limitar el tipo de sistemas a las que se pueden aplicar o, en el caso de que se basen en aproximaciones, también limitan la calidad de la evaluación misma. La técnica más habitual es la simulación, ya que no plantea restricciones al tipo de sistemas destino. El proceso consiste en realizar inicialmente una simulación de gran precisión cuyos resultados se usarán como referencia. Acto seguido, en cada iteración del proceso de optimización se realizará una simulación de tipo *bit-true* utilizando los anchos de palabra concretos que se quieren evaluar. Las simulación *bit-true* emula el comportamiento real del hardware con unas restricciones dadas en los anchos de palabra. La medida del error se obtiene comparando con la referencia y se basa en diferentes métricas, tales como el error cuadrático medio, la relación señal a ruido de cualificación, etc. Para aplicaciones concretas la métrica puede variar. Por ejemplo, en el diseño de sistemas de comunicaciones digitales se puede comparar la tasa de error de transmisión de la referencia con la tasa de error de la simulación con anchos de palabras reducidas.

Las simulaciones *bit-true* son muy costosas en tiempo, puesto que los microprocesadores

no disponen de hardware que realice operaciones multiprecisión de grano fino y al final se realiza una emulación software de un sistema hardware. Además esta simulación se complica aún más cuando se consideran otras opciones que suelen aparecer en circuitos de aplicación específica en coma fija como la saturación o distintos tipos de redondeo, o incluso otras representaciones numéricas, tales como coma flotante, coma flotante en bloque, coma fija dual, etc.

A veces se han utilizado dispositivos lógico programables (PLDs), tales como FPGAs, para acelerar esta simulación, implementando el circuito en este dispositivo para un valor fijo de los parámetros y obtener así la salida del circuito mucho más rápidamente. Cada vez que queremos probar una nueva combinación anchos de palabra, se debe volver a implementar el circuito en el PLD. Aunque los lenguajes de descripción hardware permiten definir los circuitos en una forma parametrizables, que evita tener que rediseñar el circuito para cambiar estos parámetros, estos cambios obligan a realizar el resto de fases que requiere la implementación en el PLD. Aunque estas fases se realizan de forma automática en un computador, el tiempo requerido para generar el nuevo fichero de configuración y su carga en el PLD, puede ser bastante considerable.

Por todo ello que sería deseable disponer de algún sistema que permita acelerar las simulaciones *bit-true* mediante elementos hardware, ya sean procesadores que permitan definir con grano fino la precisión de sus operandos, o modelos de los circuitos a optimizar implementados en PLDs que también permitan definir esta precisión sin tener que reconfigurar dichos dispositivos cada vez que se cambia el valor de un parámetro del circuito.

Descripción de la invención

La presente invención se refiere a métodos y circuitos para reducir los tiempos de ejecución de procesos de optimización basados en simulación *bit-true,* mediante el uso de procesadores específicos, o modelos implementados en PLDs, los cuales incluyen circuitos específicos para permitir variar de forma precisa (o con granularidad fina) el ancho de palabra de cada variable, o señal sin que sea necesario emular dicho ancho mediante instrucciones software o reconfigurar dichos PLDs durante el proceso de optimización.

A continuación se define qué se entiende en la presente invención por limitación de la precisión y rango a nivel lógico (o efectivo). Dado un formato, la precisión de un valor en

dicho formato se suele definir como la distancia entre dicho valor y el siguiente valor representable. Por lo tanto, ésta dependerá del peso del dígito más a la derecha, o dígito menos significativo (LSD), que tenga el formato. El rango de un formato es la distancia entre el mayor y el menor número representables en dicho formato. En este caso, el rango dependerá del peso del dígito más a la izquierda, o dígito más significativo (MSD), del formato. Por tanto, dada una señal coma fija con un número determinado de bits a la derecha e izquierda de la coma decimal, su precisión y rango están determinados a nivel físico. Sin embargo, si limitamos los valores que puede tomar dicha señal a nivel lógico, eliminando la información de los bits más significativos (MSBs) o los menos significativos (LSBs) (por ejemplo, forzándolos a cero, o a uno, en ciertos casos) podemos limitar la precisión y/o rango efectivo del número, ya que estamos cambiando cual es el MSB o LSB a nivel lógico. Aunque a nivel físico, dicha señal podría tomar más valores y por tanto, podría tener más precisión y/o rango, hemos limitado estos valores a nivel lógico. Por tanto, llamaremos precisión o rango efectivos, o a nivel lógico, a la correspondiente a tener en cuenta esas limitaciones a nivel lógico de los valores que puede tomar dicha señal.

Llamaremos circuito objetivo al circuito que se pretende optimizar, para lo cual calcularemos la combinación de anchos de palabra de sus señales que minimizan una función de coste pero de forma que sigue cumpliendo con un valor mínimo del parámetro o parámetros de calidad.

Un primer aspecto de la presente invención se refiere a un sistema para la optimización de anchos de palabra de circuitos digitales mediante simulaciones bit-true, que comprende un modelo del circuito objetivo. Dicho modelo del circuito objetivo comprende al menos un circuito limitador para modificar el valor de las señales cuyos anchos de palabra se quiere optimizar, estando cada circuito limitador configurado para generar una salida con un valor correspondiente a limitar a nivel lógico la precisión y/o rango efectivo del valor de un dato de entrada de acuerdo a una instrucción de control de precisión y/o rango efectivo recibida por el circuito limitador.

En una realización preferente el modelo del circuito objetivo es un circuito hardware que se implementa en al menos un PLD. El al menos un circuito limitador está preferentemente ubicado a la entrada y/o salida de un operador aritmético-lógico.

En otra realización preferente, el modelo del circuito objetivo se implementa mediante

software en un procesador. El procesador comprende preferentemente al menos una unidad funcional con al menos una unidad aritmético-lógica, estando el al menos un circuito limitador ubicado a la entrada y/o salida de la unidad aritmético-lógica. La unidad funcional puede comprender al menos un elemento de memoria para almacenar los valores de precisión/rango efectivos asociados a las variables que representan señales cuyo ancho de palabra se quiere optimizar.

La instrucción de control de precisión y/o rango efectivo que recibe un circuito limitador incluye preferiblemente el número de bits a anular, una máscara o una instrucción indicando cómo modificar el valor o máscara utilizado en la iteración del ciclo de optimización anterior. Dicha instrucción de control de precisión y/o rango efectivo puede incluir además el modo de redondeo a utilizar.

En una posible realización, al menos un circuito limitador comprende un circuito limitador de precisión configurado para recibir un primer número en coma fija de N bits en una primera entrada y generar en una salida un segundo número en coma fija de M bits, con N≥M, siendo los K MSBs de dicho segundo número el resultado de redondear, mediante algún modo de redondeo, dicho primer número mientras los (M-K) LSBs de dicho segundo número son igualados a cero, siendo K dependiente de la instrucción de control de precisión y/o rango efectivo recibida en una segunda entrada y cumpliendo K≤M. El primer número puede corresponder a la mantisa de un número en coma flotante o coma flotante en bloque y el circuito limitador de precisión puede estar configurado además para incrementar el exponente y ajustar adecuadamente la mantisa de dicho número si el redondeo de la mantisa produce un desbordamiento.

En otra posible realización, al menos un circuito limitador comprende un circuito limitador de rango configurado para recibir un primer número en coma fija de L bits en una primera entrada y detectar si dicho primer número no se puede representar usando únicamente sus (L-R) LSBs, y en cuyo caso indicar un desbordamiento lógico, siendo R dependiente de la instrucción de control de precisión y/o rango efectivo recibida en una segunda entrada y cumpliendo R <=L. El circuito limitador de rango puede estar configurado además para generar en una salida un segundo número en coma fija de L bits, cuyo valor es igual al del primer número en coma fija si no se indica desbordamiento, o en caso contrario, al máximo valor representable con L-R bits si dicho primer número es positivo o al mínimo valor representable con L-R bits si dicho primer número es negativo. El primer número en coma

fija puede corresponder al exponente de un primer número en coma flotante o coma flotante en bloque y el circuito limitador de rango está configurado además para distinguir entre si el desbordamiento se debe a que el primer número es demasiado grande o es demasiado pequeño. En este último caso, el circuito limitador de rango está preferentemente configurado además para recibir la mantisa y el signo del número en coma flotante y generar en una salida un segundo número en coma flotante, cuyo valor es igual al número en coma flotante de entrada si no se produce desbordamiento lógico, o en caso contrario, al máximo valor representable con L-R bits de exponente en valor absoluto y manteniendo el signo del número de entrada si dicho desbordamiento es por exceso o cero si dicho desbordamiento es por defecto.

En otra realización alternativa, al menos un circuito limitador comprende un circuito limitador de rango por wrapping configurado para recibir un primer número en coma fija de T bits en una primera entrada y generar en una salida un segundo número en coma fija de T bits, cuyo valor es el mismo que tendrían un número representado por los (T-P) LSBs de dicho primer número, siendo P dependiente de la instrucción de control de precisión y/o rango efectivo recibida en una segunda entrada y cumpliendo P <=T.

En otra realización preferente, al menos un circuito limitador comprende un circuito medidor de rango configurado para recibir una serie de números consecutivos de Q bits por una entrada y generar en una salida un valor que representa el número mínimo de bits necesarios para representar todos los números de dicha secuencia. El circuito medidor de rango puede estar configurado además para, de dicha serie de números de entrada, solo tener en cuenta los números que representan valores positivos.

El sistema puede comprender un módulo generador de señales de entrada, encargado de generar una serie de valores de entrada que alimenta la entrada de datos del modelo del circuito objetivo durante la simulación bit-true.

El sistema puede comprender también un módulo calculador de parámetros de calidad, encargado de recibir una serie de valores de salida producidos por el modelo del circuito objetivo en cada iteración de la simulación bit-true y calcular al menos un parámetro de calidad. El sistema comprende preferentemente una unidad de control configurada para generar, en cada iteración de la simulación bit-true, una serie de instrucciones de control de precisión y/o rango efectivo que especifican la precisión y/o rango efectivo de las distintas

señales del circuito objetivo, recibir los parámetros de calidad para cada iteración, y determinar, al finalizar la simulación bit-true, los diferentes anchos de palabra del circuito digital óptimos.

Un segundo aspecto de la presente invención se refiere a un método para la optimización de anchos de palabra de circuitos digitales mediante simulaciones bit-true, que comprende los siguientes pasos:

- Enviar, a al menos un circuito limitador incluido en un modelo del circuito objetivo y encargado de modificar el valor de las señales cuyos anchos de palabra se quiere optimizar, una instrucción de control de precisión y/o rango efectivo.
- Generar, cada circuito limitador, una salida con un valor correspondiente a limitar a nivel lógico la precisión y/o rango efectivo del valor de un dato de entrada de acuerdo a la instrucción de control de precisión y/o rango efectivo recibida.

El método puede comprender generar una serie de valores de entrada que alimenta la entrada de datos del modelo del circuito objetivo durante la simulación bit-true.

El método también puede comprender calcular, a partir de una serie de valores de salida producidos por el modelo del circuito objetivo en cada iteración de la simulación bit-true, al menos un parámetro de calidad. El método comprende preferentemente generar, en cada iteración de la simulación bit-true, una serie de instrucciones de control de precisión y/o rango efectivo que especifican la precisión y/o rango efectivo de las distintas señales del circuito objetivo; y determinar, al finalizar la simulación bit-true, los diferentes anchos de palabra del circuito digital óptimos en base a los parámetros de calidad calculados en las distintas iteraciones.

Un tercer aspecto de la presente invención se refiere a un producto de programa que comprende medios de instrucciones de programa para llevar a cabo el método anterior cuando el programa se ejecuta en un procesador. El producto de programa está preferentemente almacenado en un medio de soporte de programas, tales como un CD, DVD, pendrive, disco duro, etc. Otro aspecto de la presente invención se refiere a una señal transmisible que comprende instrucciones de programa capaces de llevar a cabo el método anterior cuando el programa se ejecuta en un procesador.

Breve descripción de los dibujos

A continuación se pasa a describir de manera muy breve una serie de dibujos que ayudan a comprender mejor la invención y que se relacionan expresamente con una realización de dicha invención que se presenta como un ejemplo no limitativo de ésta.

Fig. 1 muestra un diagrama de flujo del proceso de optimización de circuitos digitales mediante simulación bit-true, de acuerdo al estado del arte.

Fig. 2A representa un sistema para la optimización de circuitos mediante simulación bit-true acelerada utilizando un PLD. La Fig. 2B muestra un procesador 8 para acelerar la optimización de anchos de palabra de circuitos mediante simulación bit-true.

Fig. 3 representa un circuito limitador de precisión mediante truncamiento para números en coma fija.

Figs. 4A y 4B muestran dos ejemplos distintos de generación de la máscara del circuito limitador de la Fig. 3.

Fig. 5 muestra un circuito limitador de precisión mediante redondeo al más cercano para números en coma fija.

Fig. 6 muestra un circuito para la obtención del valor binario empleado en la Fig. 5.

La Fig. 7 muestra un circuito limitador de precisión mediante redondeo al par más cercano para números en coma fija.

La Fig. 8 representa una realización del circuito calculador de sticky del circuito limitador de precisión de la Fig. 7.

La Fig. 9 ilustra la implementación del circuito de forzado de bit del circuito limitador de precisión de la Fig. 7.

Fig. 10 muestra un circuito limitador de precisión mediante jamming para números en coma fija.

Fig. 11 representa un circuito limitador de precisión mediante jamming sin sesgo para

números en coma fija.

Fig. 12 muestra un circuito detector de desbordamiento para números en coma fija.

Fig. 13 muestra un circuito detector de desbordamiento para medir cuántos bits por la izquierda hay que mantener, para números en coma fija.

Fig. 14 muestra un circuito limitador de rango con saturación para números en coma fija.

Fig. 15 muestra un circuito limitador de rango con wraping para números en coma fija.

Fig. 16 muestra un circuito limitador de rango y de precisión mediante truncamiento o jamming para números en coma fija.

Fig. 17 muestra un circuito limitador de rango y de precisión mediante redondeo al más cercano para números en coma fija.

Fig. 18 muestra un circuito limitador de precisión para números en coma flotante mediante redondeo al más cercano.

Fig. 19 muestra un circuito limitador de precisión mediante redondeo al más cercano para números en coma flotante en bloque.

Fig. 20 representa un circuito detector de desbordamiento para números en coma flotante.

Fig. 21 representa un circuito detector de desbordamiento para medir cuántos bits por la izquierda hay que mantener, para números en coma flotante.

Fig. 22 representa un circuito limitador de rango con saturación para números en coma flotante.

Descripción detallada de la invención

La presente invención realiza una optimización de circuitos digitales mediante simulación bit-true acelerada. La Fig. 1 muestra un diagrama de flujo que representa, de acuerdo al estado del arte, un algoritmo de optimización completo mediante simulación. Tras una etapa

de inicialización, se simula el circuito con una combinación de anchos de palabra determinado. A continuación se comprueba la calidad obtenida en la simulación, normalmente calculando una medida del error de la salida al compararla con una salida de referencia. Si los resultados obtenidos cumplen con los mínimos de calidad exigidos, y el valor de la función de costes para la combinación de anchos de palabras simulada es menor que la correspondiente a la última almacenada, se almacena ésta combinación en lugar de la anterior. Mientras no se haya terminado de ejecutar todas las iteraciones establecidas para la simulación, se ejecuta una nueva iteración, en la que se establece una nueva combinación de anchos de palabra y se vuelve a simular en el circuito.

La etapa de simulación del circuito con una combinación de anchos de palabra dada puede realizarse por hardware (e.g. PLDs) o por software (procesador). En el caso de realizarse por PLD, en el estado del arte actual cuando se calcula la nueva combinación de anchos de palabra se debe generar además un nuevo fichero de configuración del PLD con los nuevos anchos de palabra y se configura de esta forma el PLD antes de realizar la simulación (ver cuadro en líneas discontinuas en la Fig. 1). En el caso de realizarse por procesador, al no tener los procesadores actuales ni instrucciones ni circuitos hardware específicos para soportar multiprecisión de grano fino, incluso las operaciones aritméticas simples requieren de múltiples instrucciones máquina para emular el comportamiento del hardware. La especial configuración interna del modelo del circuito a simular implementado en el PLD según la presente invención, permite eliminar el paso de configuración del PLD en cada iteración, sustituyéndolo por una simple configuración de los anchos de palabra. Para el caso del procesador, la inclusión de circuitos específicos (y, por lo tanto, instrucciones) para permitir seleccionar, con grano fino, el rango y/o la precisión de los operandos permite reducir drásticamente el número de instrucciones máquina que es necesario ejecutar para simular el comportamiento del circuito objetivo.

La simulación del circuito, con una combinación de anchos de palabra determinada, que se ejecuta en cada iteración se puede llevar a cabo mediante PLDs (Fig. 2A) o mediante procesador (Fig. 2B).

La Fig. 2A ilustra un sistema para la optimización de circuitos mediante simulación bit-true acelerada mediante el uso de PLDs 2 de acuerdo a una posible realización. Dicho sistema comprende un computador host 1 que se conecta con un PLD 2. La etapa de simulación del circuito de la Fig. 1 se llevaría a cabo en el PLD 2, y el resto de etapas de la Fig. 1 se

ejecutaría en el host. En una realización alternativa el host 1 podría estar implementado dentro del propio PLD 2, siendo el PLD 2 el encargado de realizar todas las etapas mostradas en la Fig. 1. El host 1 se utiliza para implementar el algoritmo de optimización, salvo las simulaciones del circuito objetivo (llamamos circuito objetivo al circuito que queremos optimizar) y el cálculo del parámetro de calidad correspondiente a las diferentes combinaciones de anchos de palabra, que se realizan en el PLD 2. El host 1 también se utiliza para configurar o programar el PLD 2 y comunicarse con él, así como la implementación del interfaz con el usuario. El PLD 2 se utiliza para implementar un modelo de circuito objetivo que permite limitar a nivel lógico la precisión y/o rango de las señales del circuito objetivo y así realizar la simulación bit-true y el cálculo de los parámetros de calidad, para las distintas combinaciones de anchos de palabra. En una implementación alternativa se podrían utilizar varios PLDs 2, en lugar de uno solo, para realizar esta función.

El PLD 2 del ejemplo de la Fig. 2A implementa (o está configurado) una unidad de control 3 que realiza por un lado la comunicación con el host 1 y por otro, el control y la sincronización del resto de elementos implementados en el PLD 2. Dicha comunicación con el host comprende la recepción de la información sobre la nueva combinación de anchos de palabra y, al finalizar la simulación, el envío de los parámetros de calidad obtenidos para dicha combinación. En una implementación alternativa la unidad de control 3 podría calcular la nueva combinación en lugar de recibirla desde el host, basándose en los parámetros de calidad obtenidos. Basándose en la información recibida sobre la nueva combinación de anchos de palabra, la unidad de control 3 genera unas señales o instrucciones de control de precisión y/o rango efectivo 10 (líneas discontinuas en Fig. 2A) para especificar la precisión lógica (o efectiva) y/o rango efectivo de cada una de las señales del circuito objetivo. La unidad de control 3 genera además unas instrucciones de control 11 (líneas de puntos en Fig. 2A) que envía al resto de módulos implementados en el PLD para sincronizar y controlar la ejecución de las simulaciones bit-true. Aunque en la Fig. 2A se muestra la unidad de control 3 como un circuito hardware, parte integrante del PLD 2, en una realización alternativa la unidad de control 3 podría estar implementada en el computador host 1.

El PLD 2 implementa además un módulo generador de señales de entrada 4 cuya salida está conectada a la entrada de datos (línea continua) del modelo del circuito objetivo 5, cuya salida a su vez está conectada a un módulo calculador de parámetros de calidad 6. El módulo generador de señales de entrada 4, cuando recibe una instrucción de control 11 de

la unidad de control 3, genera una secuencia de valores de entrada VAL_{IN} para alimentar la entrada de datos del modelo del circuito objetivo 5 durante la simulación bit-true. Dichos valores están pre-calculados y almacenados en dicho módulo generador 4. En una implementación alternativa dichos valores se podrían generar mediante un algoritmo adecuado. En una posible realización, el módulo generador de señales 4 es una memoria que se llena desde el host 1, habiendo una fase previa de envío de los datos y posteriormente una fase de lectura interna de los mismos para enviar los valores de entrada VAL_{IN} al modelo del circuito objetivo 5. En una implementación alternativa, los datos o valores de entrada VAL_{IN} son generados internamente en el PLD 2, en lugar de ser recibidos procedentes del host 1, de forma que el módulo generador de señales 4 obtiene los datos mediante una serie de señales aleatorias o deterministas.

El módulo calculador de parámetros de calidad 6 recibe los valores de salida VAL_{OUT} que produce el modelo del circuito objetivo 5 durante la simulación y los compara con los valores "teóricamente" correctos para calcular el (o los) parámetro(s) de calidad (es posible que haya varios parámetros de calidad simultáneos; por ejemplo, SQNR y valor de pico). Dichos valores correctos están pre-calculados y almacenados. En una implementación alternativa los valores correctos podrían calcularse a partir de la salida del módulo generador de señales de entrada 4. El módulo calculador de parámetros de calidad 6 envía el (o los) parámetro(s) de calidad PQ calculado(s) para la simulación correspondiente a una combinación dada de anchos de palabra por una salida hasta una entrada de la unidad de control 3 para que ésta lo envíe al host 1. El host 1 utiliza esta información para validar la combinación de anchos de palabra testeada. También puede utilizar esta información para generar una nueva combinación de anchos de palabra. En una implementación alternativa esto se podría hacer directamente en la unidad de control 3, en lugar de realizarlo en el host 1. Por otro lado, el módulo calculador de parámetros de calidad 6, en lugar de estar implementado en un circuito hardware en el propio PLD tal como se muestra en la Fig. 2A, podría estar implementado en el host 1. En este caso, se podría simplemente almacenar las salidas del modelo del circuito objetivo 5 en una memoria para enviarse al host 1 tras la simulación o, no utilizar una memoria y enviar los datos de salida del modelo del circuito objetivo 5 durante la simulación al host 1, y que sea el host 1 el que calcule el parámetro de calidad.

El modelo del circuito objetivo 5 implementa el propio circuito objetivo pero incluyendo además una serie de circuitos limitadores 7 que se colocan a la entrada de unos

operadores aritméticos y/o lógicos 9 en las señales cuyo ancho de palabra queremos optimizar. Por lo tanto, el modelo del circuito objetivo 5 recibe las instrucciones de control de precisión y/o rango efectivo 10 provenientes de la unidad de control 3, y las distribuye entre dichos circuitos limitadores 7. En la Fig. 2A se muestra un detalle de una pequeña parte del modelo del circuito objetivo 5, en la que se aprecia la ubicación de varios circuitos limitadores 7, para un posible ejemplo en el que el operador aritmético-lógico 9 tiene dos entradas de datos. Un circuito limitador 7 recibe una señal de datos de n1 bits (un número NUM_i, que depende de la secuencia de valores de entrada VAL_{IN}) y genera una señal de salida de n1 bits que se conecta a una primera entrada de un operador aritmético-lógico 9. Paralelamente, otro circuito limitador 7 recibe una señal de datos de n2 bits (un número NUM_j, que depende de la secuencia de valores de entrada VAL_{IN}) y genera una señal de salida de n2 bits que se conecta a una segunda entrada del operador aritmético-lógico 9, el cual genera un valor de n3 bits. Dicho operador aritmético-lógico 9 podría tener un número diferente de entradas, las cuales podrían estar conectadas o no a circuitos limitadores 7.

Dichos circuitos limitadores 7 generan un valor de salida que aunque a nivel físico tienen el mismo número de bits que la entrada, a nivel lógico (o efectivo) su número de bits puede ser menor. Para ello anulan (o eliminan la información) de una serie de bits por la izquierda y/o por la derecha del valor de entrada. Dichos circuitos limitadores 7, para determinar cuántos bits debe anular, utilizan la información que reciben en una segunda entrada, correspondiente a una instrucción de control de precisión y/o rango efectivo (10_i, 10_j). Las instrucciones de control de precisión y/o rango efectivo 10 que recibe el modelo del circuito objetivo 5 contiene todas las instrucciones de control de precisión y/o rango (10_i, 10_j, …) para los circuitos limitadores 7; es decir, cada circuito limitador 7 recibe su propia instrucción de control de precisión y/o rango efectivo (10_i, 10_j) que puede coincidir o no con la que recibe otro circuito limitador 7. De esta forma, todas las instrucciones de control de precisión y/o rango efectivo 10 pueden llegar en paralelo y repartirse entre los distintos circuitos limitadores 7, o pueden multiplexarse en el tiempo, o una combinación. La instrucción de control de precisión y/o rango efectivo (10_i, 10_j) que recibe un circuito limitador 7 podría ser el número de bits a anular codificado en binario, una máscara, o una instrucción indicando cómo modificar el valor o máscara utilizado en la simulación anterior. En una implementación alternativa, la instrucción de control de precisión y/o rango efectivo (10_i, 10_j), además de indicar el número de bits a anular, podría indicar el modo de redondeo a utilizar en dicha anulación. En una implementación alternativa el circuito limitador 7 en lugar de anular ciertos bits por la izquierda, podría medir cuántos bits por la izquierda contienen

información significativa o indicar mediante una señal cuando se utilizan los bits anulados.

En una implementación alternativa el número de bits a la salida del circuito limitador 7 podría ser menor al número de bits de la entrada (esto sería así cuando la señal se genera con un tamaño pero la cota máxima del número de bits de la señal queremos que sea -o sabemos que tiene que ser- menor, por ejemplo para que encaje en el ancho de una memoria). En otra implementación el circuito limitador 7 podría estar a la salida del operador aritmético-lógico 9 (el operador aritmético-lógico en el ejemplo).

Según se ha explicado anteriormente, en una realización alternativa uno o varios de los módulos del PLD 2 mostrados en la Fig. 2A (la unidad de control 3 y/o el módulo generador de señales 4 y/o el módulo calculador de parámetros de calidad 6), con excepción del modelo del circuito objetivo 5, pueden estar implementados en el computador host 1, mediante módulos o funciones software. De esta forma, en una posible realización el PLD 2 tendría únicamente el modelo del circuito objetivo 5, el cual recibiría del host 1 los valores de entrada VAL_{IN}, las instrucciones de control de precisión y/o rango efectivo 10 y las instrucciones de control 11 y enviaría al host 1 los valores de salida VAL_{OUT} para su análisis y para que el host 1 inicie una nueva iteración de la simulación bit-true.

La Fig. 2B muestra un procesador 8 preparado para acelerar la ejecución de algoritmos de optimización de anchos de palabra de circuitos digitales mediante simulación bit-true de acuerdo a una posible realización. El procesador 8 además de los típicos elementos que conforman un procesador y que no se representan en la figura por simplicidad (por ejemplo, unidad de control, unidad funcional en coma fija, unidad funcional en coma flotante, controlador de cache, etc.), comprende una unidad funcional 12 configurada para especificar con grano fino el ancho de palabra de los operandos. Todas las etapas de la Fig. 1 se ejecutarían en el procesador 8, si bien al menos las instrucciones relativas a la simulación del circuito objetivo que requieran especificar el ancho de palabra se ejecutarían en la unidad funcional 12 (el modelo del circuito objetivo 5 se implementa mediante software y las instrucciones que implementan dicho modelo que requieren de limitación de ancho de palabra se ejecutan en la unidad funcional 12, que es un elemento hardware). Dicha unidad funcional 12 recibe dos entradas, A y B, correspondiente a la dirección de los registros fuente de la operación a realizar y una tercera entrada C, correspondiente a la dirección del registro destino para el resultado de dicha operación. Las entradas A y B se introducen en los dos puertos de lectura del banco de registro de precisión/rango efectivo 13 y del banco

de registro de datos 14, que devuelven los valores de precisión /rango efectivos y los valores de los registros (NUM_i, NUM_j) correspondientes en dos salidas cada uno, respectivamente, según el ejemplo mostrado en la Fig. 2B. El valor del dato de entrada (NUM_i) y el valor de precisión/rango efectivo 10_i de la entrada A, se introducen en un primer circuito limitador 7, mientras que los correspondientes a la entrada B (NUM_j, 10_j) se introducen en otro circuito limitador 7. Ambos circuitos limitadores 7 generan una salida con el mismo número de bits que su entrada de datos pero con un valor correspondiente a limitar la precisión/rango efectivos del valor del dato de entrada (NUM_i, NUM_j) de acuerdo al valor de precisión /rango efectivo (10_i, 10_j) recibido en la otra entrada. Ambas salidas de los circuitos limitadores 7 son la entrada de una unidad aritmético-lógica 19, la cual es capaz de realizar múltiples operaciones distintas en función de la instrucción 15 concreta recibida de la unidad de control del procesador (a diferencia del operador aritmético-lógico 9 de la Fig. 2A, el cual normalmente hará siempre una única operación). La unidad aritmético-lógica 19 tras realizar la operación correspondiente genera un resultado en una salida que se almacenará en el banco de registro de datos 14, en la dirección indicada por la entrada C. Con esta configuración los resultados se almacenan en el banco de registro de datos 14 sin limitar, con su precisión física; de esta forma, es cuando se van a usar los datos cuando se realiza la limitación de precisión y/o rango. Esto permite usar una misma salida en varias entradas con distinta precisión en cada una de las entradas. Para determinar la precisión de la salida, a la hora de implementar el circuito habría que mirar la máxima de todas en las que entran. Para aumentar la claridad de esta realización se han omitido en la Fig. 2B todas las señales de control y los circuitos de entrada y salida de datos entre los bancos de registros (tanto de datos 14 como de precisión/rango efectivo 13) y la memoria principal.

En una implementación alternativa los datos e información de precisión/rango efectivos se podrían almacenar en un mismo banco de registro, en lugar de utilizar dos. Otra implementación alternativa podría no tener banco de registros específico y obtener ambos valores de un banco de registro o memoria común a otras unidades funcionales. En otra implementación alternativa, la unidad aritmético-lógica 9 podría compartirse con otra unidad funcional clásica. En otra realización alternativa podría haber más de una unidad funcional 12 con capacidad para especificar el ancho de palabra, por ejemplo una para operandos en coma fija y otra para operandos en coma flotante.

En otra implementación alternativa, el circuito limitador 7 podría estar a la salida de la unidad aritmético-lógica 9 en lugar de en sus entradas, y la entrada C, en lugar de las entradas A y

B, sería la que entrara en el banco de registro de precisión/rango efectivo 13 para leer la información de precisión/rango efectivo de la variable en la que se va a almacenar el resultado de la operación. Con esta configuración los resultados se almacenan en el banco de registros de datos 14 ya limitados.

Aunque no se muestra en la Fig. 2B, se puede considerar que el procesador 8 implementa todas las funciones descritas en el diagrama de flujo de la Fig. 1 (excepto la programación del PLD) y que incluye también un modelo del circuito objetivo 5, un módulo calculador de parámetros de calidad 6, un módulo generador de señales de entrada 4 y una unidad de control 3 como los mostrados en la Fig. 2A para la implementación por PLD 2, aunque mientras que para la implementación por PLD 2 se trata de circuitos o elementos hardware, en el caso de la implementación por procesador 8 se implementa por funciones o módulos software. Por otro lado, el propio procesador 8 podría estar implementado en un circuito integrado de aplicación específica (ASIC) o en un PLD.

En las siguientes figuras a continuación se muestran, a modo de ejemplo, diferentes implementaciones de circuitos limitadores 7. Otros circuitos limitadores no contemplados en las figuras podrían utilizarse, siempre que permitan limitar la precisión y/o el rango efectivo de un valor numérico de entrada, ya sea para números en coma fija, para números en coma flotante o coma flotante en bloque, u otro formato numérico no especificado.

La Fig. 3 ilustra la implementación de un circuito limitador de precisión 200 mediante truncamiento de acuerdo a una posible realización, para números en coma fija. El circuito limitador de precisión 200 recibe un número (NUM_i) en coma fija de N bits, en una primera entrada 21, y genera en una salida 23, mediante truncamiento y en función de una instrucción de control de precisión y/o rango efectivo 10_i recibida en una segunda entrada 22, un número (NUM_K) coma fija de N bits que tiene una precisión efectiva de K bits, siendo K un parámetro configurable y cumpliendo $K \leq N$. Por “precisión efectiva de K bits” se entiende aquí el borrado de la información de los N-K bits menos significativos, de forma que aunque el número sigue teniendo físicamente N bits el valor almacenado tiene una precisión lógica o efectiva de K bits.

El circuito limitador de precisión 200 comprende un array de N puertas AND 201 que se conecta a la primera entrada 21, por donde se recibe el número de entrada (NUM_i), y a una máscara 25 de N bits para generar la salida realizando la operación lógica AND bit a bit del

número de entrada (NUM_i) con la máscara 25. La máscara 25 tiene los K bits más significativos (MSBs) iguales a uno y el resto a cero. La máscara 25 puede estar almacenada en un registro 26 o recibirse directamente por una segunda entrada, coincidiendo en ese caso con la instrucción de control de precisión y/o rango efectivo 10_i. Además, la máscara 25 puede almacenarse o recibirse tal cual o codificada con un número binario. La instrucción de control de precisión y/o rango efectivo 10_i es la que controla el valor efectivo de la máscara 25 y, por tanto, el parámetro de precisión K.

Las Fig. 4A y Fig. 4B muestran cómo se genera la máscara 25 de acuerdo a dos ejemplos. La máscara 25 del ejemplo mostrado en la Fig. 4A se almacena tal cual en un registro de desplazamiento 26. El registro de desplazamiento 26 permite poner todos sus bits a uno, y desplazamientos de un bit tanto a la izquierda como a la derecha, introduciendo cero, o uno, por la derecha, o por la izquierda, respectivamente, tal y como se indica en Fig. 4A, de acuerdo a una instrucción recibida, una instrucción de control de precisión y/o rango efectivo 10_i. En una implementación alternativa el registro de desplazamiento 26 podría realizar desplazamientos de más de un bit, de acuerdo a una instrucción recibida. En otra implementación alternativa el registro de desplazamiento 26 podría poner todos sus bits a cero de acuerdo a una instrucción, o incluso almacenar un valor recibido por una segunda entrada.

En el ejemplo mostrado en la Fig. 4B, la máscara 25 se recibe codificada (por ejemplo, en la propia instrucción de control de precisión y/o rango efectivo 10_i), por una entrada 31 de L bits de un circuito lógico combinacional 30. El circuito lógico combinacional 30 genera una salida 29 de N bits donde los K MSBs son unos y el resto cero, donde K es el valor codificado en la entrada 31. En una implementación alternativa el valor codificado K puede estar almacenado en un registro de L bits, en lugar de recibirse por una entrada 31. Este registro, podría estar configurado para incrementarse, decrementarse o poner un valor constante de acuerdo a una instrucción recibida. Dicho registro podría estar configurado además para cargar un valor recibido por otra entrada.

La Fig. 5 ilustra la implementación de un circuito limitador de precisión 300 mediante redondeo al más cercano de acuerdo a una posible realización. El circuito limitador de precisión 300 recibe un número coma fija de N bits, en una primera entrada 36, y genera mediante redondeo al más cercano, en una salida 39, un número coma fija de N bits que tiene una precisión efectiva de K bits, siendo K un parámetro configurable y cumpliendo

K≤N. El circuito limitador de precisión 300 comprende un sumador 35 de N bits configurado para sumar el número recibido en la primera entrada 36 del circuito limitador de precisión 300 con una máscara one-hot 32 de N bits (solo un bit vale 1 y el resto vale 0), recibido en una segunda entrada 37 del circuito limitador de precisión 300, la cual tiene todos sus bits a cero excepto el (K+1)-ésimo MSB que vale uno. La salida 38 del sumador 35 de N bits se conecta a la entrada 40 de un circuito limitador de precisión 200 mediante truncamiento como el del ejemplo ilustrado en la Fig. 3, el cual utiliza la máscara 25.

La máscara one-hot 32 se obtiene a partir de la máscara 25 del circuito limitador de precisión 200 mediante un circuito combinacional como el del ejemplo ilustrado en la Fig. 6. La Fig. 6 muestra un array de N-1 puertas OR-exclusiva 20 (puerta XOR) tal que la puerta que genera el i-ésimo MSB de la salida, tiene como entrada el (i-1)-ésimo MSB y el i-ésimo MSB de la entrada, con i variando de 2 hasta N. El MSB de la salida es constante e igual a cero. En una implementación alternativa la máscara 25 del circuito limitador de precisión 200 mediante truncamiento podría generarse a partir de la máscara one-hot 32 mediante un circuito combinacional. En otra implementación alternativa la máscara one-hot 32 podría almacenarse en un registro o recibirse en otra entrada.

La Fig. 7 ilustra la implementación de un circuito limitador de precisión 320 mediante redondeo al par más cercano de acuerdo a una posible realización. El circuito limitador de precisión 320 recibe un número coma fija de N bits en una primera entrada 36, y genera mediante, redondeo al par más cercano, en una salida 39, un número coma fija de N bits que tiene una precisión efectiva de K bits, siendo K un parámetro configurable y cumpliendo K≤N. Este circuito limitador de precisión 320 es como el de la Fig. 5 pero sin bias (en un circuito unbiased, o sin sesgo, la media del error de redondeo es cero; mientras que en un circuito biased, o con sesgo, la media del error de redondeo no es cero). La salida 38 del sumador 35 se conecta a un bloque 16 que comprende, además del circuito limitador de precisión 200 mediante truncamiento, un calculador de sticky 17 y un circuito de forzado de bit 49. El calculador de sticky 17 se encarga de calcular el bit de sticky (stk) de los bits menos significativos (LSBs) que van a ser descartados (en este caso, el bit de sticky es la OR de todos los bits que se descartan, y se usa para detectar el caso de empate en redondeo al más cercano sin sesgo), con una primera entrada 42 conectada a la salida 38 del sumador 35, y una segunda entrada 43 conectada a la máscara 25 del circuito limitador de precisión 200 mediante truncamiento, y produce una salida 44 de un bit de sticky (stk) que vale uno cuando alguno de los bits que se van a descartar tiene un valor uno. En una

implementación alternativa la primera entrada 42 del calculador de sticky 17 se podría conectar a la primera entrada 36 del circuito limitador de precisión 320 mediante redondeo al par más cercano y la salida 44 del calculador de sticky 17 valdría uno cuando el MSB de los bits que se van a descartar es igual a uno y el resto cero. La salida del calculador de sticky 17 se conecta al circuito de forzado de bit 49 en una primera entrada 45. El circuito de forzado de bit 49 comprende además una segunda entrada 46 que se conecta a la salida del circuito limitador de precisión 200 mediante truncamiento y una tercera entrada 47 que se conecta a la máscara one-hot 32, y genera una salida 39 de N bits, igual a la segunda entrada 46 del circuito de forzado de bit 49 salvo porque el K-ésimo MSB es forzado a cero si la primera entrada 45 del circuito de forzado de bit 49 es cero.

La Fig. 8 ilustra una posible implementación del circuito calculador de sticky 17 del circuito limitador de precisión 320 de la Fig. 7, empleando un array de puertas NOT 18 para invertir la máscara 25, un array de puertas AND 33 para forzar acero todos los bits que no se van a descartar y un árbol de puertas OR en cascada 34 que hace la OR acumulativa de las N entradas para obtener en la salida 44 el bit de sticky (stk).

La Fig. 9 ilustra, de acuerdo a un ejemplo, la implementación del circuito de forzado de bit 49 del circuito limitador de precisión 320 de la Fig. 7, que emplea un circuito de desplazamiento de un bit a la izquierda 50, un array de puertas NOT 52 para, a partir de la máscara one-hot 32 de la Fig. 7, calcular la máscara necesaria para forzar a cero el LSB de los bits válidos. Además, un array de puertas OR 54 con una entrada conectada dicha máscara y otra al bit el sticky, tal que la salida será todo unos, si el sticky es uno y por tanto no se forzará a cero ningún valor. La salida del array de puertas OR 54 se conecta a una de las entradas de un array de puertas AND 56, que tiene la otra entrada conectada a la entrada de datos, tal que, si el bit de sticky es igual a cero, forzará a cero el bit indicado por la máscara.

La Fig. 10 ilustra la implementación de un circuito limitador de precisión 400 mediante jamming de acuerdo a una posible realización. El circuito limitador de precisión 400 recibe un número coma fija de N bits, en una primera entrada 60, y genera mediante jamming (en el redondeo mediante “jamming”, además de truncar los bits que sobran, se fuerza a uno el LSB de los que permanecen), en una salida 69, un número coma fija de N bits que tiene una precisión efectiva de K bits, siendo K un parámetro configurable y cumpliendo K≤N. El circuito limitador de precisión 400 mediante jamming comprende un circuito limitador de precisión 200 mediante truncamiento como el del ejemplo ilustrado en la Fig. 3, cuya entrada

está conectada a la entrada 60 del circuito limitador de precisión 400 mediante jamming, y la salida 23 a la primera entrada 62 de N bits de un array de N puertas OR 68 cuya segunda entrada 63 se conecta a una máscara one-hot 32 de N bits la cual tiene todos sus bits a cero excepto en (K+1)-esimo MSB que vale uno, para generar la salida realizando la operación OR lógica bit a bit de la salida 23 del circuito limitador de precisión 200 mediante truncamiento con la máscara one-hot 32. Al igual que en el ejemplo de la Fig. 5, la máscara one-hot 32 se puede generar a partir de la máscara 25 utilizando el circuito combinacional de la Fig. 6, o alternativamente la máscara 25 se puede obtener a partir de la máscara one-hot 32 mediante un circuito combinacional.

La Fig. 11 ilustra la implementación de un circuito limitador de precisión 420 mediante jamming sin sesgo de acuerdo a una posible realización. Este circuito es similar al anterior mostrado en la Fig. 10, pero empleando además un calculador de sticky 17 y un circuito de forzado de bit 49 como los empleados en el ejemplo de la Fig. 7.

Es importante resaltar que todos los circuitos representados como limitadores de precisión para coma fija sirven por igual para números representados mediante complemento a dos, SM (Signo Magnitud), sin signo, y exceso a M. A partir de este momento, salvo que se indique lo contrario, vamos a considerar que los número coma fija están codificados mediante representación complemento a dos (que es la opción más extendida), siendo el MSB equivalente al bit de signo. Sin embargo, un experto en la técnica podría apreciar que otros formatos que tienen una representación diferente podrían ser utilizados con modificaciones menores en los circuitos descritos.

La Fig. 12 ilustra la implementación de un circuito detector de desbordamiento 500 de acuerdo a una posible realización. El detector de desbordamiento 500 recibe un número coma fija de N bits, en una primera entrada 70, y genera una señal que indica si se ha producido desbordamiento lógico cuando los R MSB de la entrada se consideran no utilizables, siendo R un parámetro configurable y cumpliendo R≤N. El detector de desbordamiento 500 comprende un array de N puertas XOR 73 cuya primera entrada 71 se conecta al bit de signo de la entrada, y la segunda entrada 72, a los bits correspondientes a la primera entrada. La salida 74 del array de N puertas XOR 73, que se corresponde prácticamente con el valor absoluto de la entrada, se conecta a una primera entrada 75 de un array de N puertas AND 77, y una máscara 28 de N bits se conecta a la segunda entrada 76 del array de N puertas AND 77. La salida 78 del array de N puertas AND 77, donde se

han seleccionado los R MSBs, se conecta a un árbol de puertas OR 79, que hace la OR acumulativa de las N entradas (esto es, entran N bits y sale uno igual a la OR de los N bits de entradas), para detectar si alguno de estos bits es distinto de cero. En una implementación para número en SM o sin signo, se prescindiría del array de puertas XOR 73 y se usarían directamente todos los bits de entrada excluyendo el de signo, si lo hubiese.

Fig. 13 ilustra la implementación de un circuito detector de desbordamiento 520 para medir cuántos bits por la izquierda hay que mantener (se ponen bits de sobra y al terminar la simulación se mira cuántos de esos bits no se han usado) de acuerdo a una posible realización. Además del array de N puertas XOR 73 como el empleado en el circuito detector de desbordamiento 500 de la Fig. 12, este nuevo circuito detector de desbordamiento 520 emplea un registro 80 que acumula la salida de la puerta OR 82 de forma que al terminar los bits a cero por la izquierda indican los que no se han usado para representar los valores de entrada y simplemente son una extensión del signo. En una implementación para números en SM o sin signo, se prescindiría del array de puertas XOR 73 y se usarían directamente todos los bits de entrada excluyendo el de signo, si lo hubiese.

Fig. 14 ilustra la implementación de un circuito limitador de rango 600 con saturación (cuando se produce un desbordamiento se pone el valor representable más cercano, es decir, el máximo valor representable si el número era positivo, y el mínimo si era negativo) de acuerdo a una posible realización. El limitador de rango 600 recibe un número coma fija de N bits en complemento a dos, en una primera entrada 60, y genera un número de N bits, en una salida, equivalente al de la entrada pero cuyo rango efectivo (o lógico) está limitado mediante saturación suponiendo que los R MSBs no son utilizables, siendo R un parámetro configurable y cumpliendo $R \leq N$. El limitador de rango 600 comprende un detector de desbordamiento 65, que podría ser como el descrito en la Fig. 12, que recibe el número de la entrada 60 y genera una señal para indicar desbordamiento lógico, que se conecta a la entrada de control de un multiplexor 67. El multiplexor 67 tiene la primera entrada de datos conectada directamente al número de la entrada 60, y una segunda entrada conectada a la salida de un array de N puertas OR-exclusivas 66, que permiten calcular a partir de la máscara 28 del detector de desbordamiento 65, el valor máximo o mínimo para el rango efectivo configurado, dependiendo del signo de la entrada. El array de puertas OR-exclusivas 66 se conectan en una primera entrada a dicha máscara 28 y una segunda entrada al inverso del MSB de la entrada 60, que se corresponde con el bit de signo de este valor. De esta forma si el número de entrada 60 es negativo, la máscara correspondería con

el valor mínimo y si fuese positivo, el inverso de la máscara sería el valor máximo para el rango efectivo configurado. La salida de datos del multiplexor 67 sería la salida del limitador de rango 600 con saturación, tal que si no se produce desbordamiento la salida es igual que la entrada y si se produce desbordamiento, la salida sería un valor igual a la salida del array de puertas OR-exclusivas 66, es decir, el máximo valor representable si el número era positivo, y el mínimo si era negativo. En una implementación para números en SM se prescindiría del array de puertas XOR 66 y se usarían los bits de la máscara 28 invertidos excluyendo el MSB, donde se usaría el bit de signo de la entrada.

Fig. 15 ilustra la implementación de un circuito limitador de rango 700 mediante wrapping (es decir, cuando se produce un desbordamiento simplemente se descarta la información de los MSBs lo que produce un valor erróneo) de acuerdo a una posible realización. El limitador de rango 700 recibe un número coma fija de N bits, en una primera entrada 60, y genera un número de N bits, en una salida, equivalente al de la entrada pero cuyo rango efectivo (o lógico) está limitado mediante wrapping suponiendo que los R MSBs no son utilizables, siendo R un parámetro configurable y cumpliendo R≤N. El limitador de rango 700 comprende un circuito 102 para detectar el valor del (R+1)-esimo MSB de la entrada, de tal forma que la salida del limitador de rango 700 es un valor igual al de la entrada pero forzando los R MSBs al mismo valor que tenga dicho bit. El circuito 102 comprende un array de puertas AND 104 cuya primera entrada está conectada a la entrada del limitador de rango 700 y la segunda a la salida de un circuito convertidor de máscara 108 como el descrito en la Fig.6, cuya entrada se conecta a la máscara 28, que tiene los R MSB igual a 1 y el resto a 0. La salida del array de puertas AND 104 se conecta a la entrada de un árbol de puertas OR en cascada 106 (u OR acumulativa) para obtener en la salida el valor del (R+1)-esimo MSB de la entrada. Dicho valor se conecta a la entrada de un multiplexor 107, de tal forma que su salida será la OR de la entrada del limitador de rango 700 con la máscara 28 (mediante OR 105), si dicho bit vale 1 o la AND de dicha entrada con el inverso de la máscara 1 (mediante AND 103), si vale 0.

Fig. 16 ilustra la implementación de un circuito limitador de precisión y de rango (recordemos que la precisión depende de los bits por la derecha y el rango de los bits por la izquierda; un número positivo de 8 bits con el punto en medio, cuatro bits enteros y cuatro fraccionarios, tiene 2^-4 de precisión y un rango entre 0 y 16) con redondeo al más cercano 800 de acuerdo a un ejemplo, que emplea un limitador de rango 810 como el mostrado en las Fig. 14 o 15 en serie con un limitador de precisión 820 mediante truncado o jamming (biased o

unbiased) como el mostrado en las Fig. 3, 10 o 11. Tienen que estar en ese orden, si el limitador de rango utiliza saturación, lo que podría modificar el valor de la entrada completamente, y a ese valor es al que hay que aplicarle la reducción de precisión. En una implementación alternativa, si el limitador de rango sólo va a detectar el desbordamiento lógico (como en la Fig. 12 o 13), éste no produce cambios en el valor de entrada y los dos módulos (810, 820) podrían están en paralelo, o en orden inverso, ya que el limitador de precisión con truncamiento o jamming no va a cambiar los bits más significativos.

Fig. 17 ilustra la implementación de un circuito limitador de precisión mediante redondeo al más cercano y de rango 900 de acuerdo a un ejemplo. Es un circuito análogo al representado en la Fig. 5 pero incluyendo un limitador de rango 910 (el circuito detector de desbordamiento o limitador de rango de cualquiera de las Fig. 12 a 15) a la salida 38 del sumador 35, porque podría haber desbordamiento después de la suma. Se puede hacer lo mismo con el circuito sin sesgo de la Fig. 7, cambiando el circuito limitador de precisión 200 mediante truncamiento por el bloque 16 de la Fig. 7. Mientras que en el circuito de la Fig. 16 se emplea truncamiento o jamming, en el circuito de la Fig. 17 se emplea redondeo al más cercano.

En los siguientes ejemplos, salvo que se indique lo contrario, se considera que los números en coma flotante, tanto los no procesados, como los preprocesados, son representados por un bit de signo, un exponente en representación “exceso a M”, y una mantisa normalizada positiva sin signo, de tal forma que el MSB es igual a uno y está implícitamente, es decir no está incluido en la representación de la mantisa (en el estándar de números en coma flotante IEE754-2008, este bit se conoce como “leading one”). Sin embargo, un experto en la técnica podría apreciar que otros formatos que tienen una representación diferente podrían ser utilizados con modificaciones menores en los circuitos descritos.

Para implementar el limitador de precisión para números coma flotante o coma flotante en bloque, simplemente se utilizarían, para las mantisas, cualquiera de los circuitos descritos anteriormente para número en coma fija, salvo en el caso de redondeo al más cercano que requiere añadir lógica adicional para manejar el exponente. Fig. 18 ilustra la implementación de un circuito limitador de precisión 1000 para números en coma flotante mediante redondeo al más cercano de acuerdo a un ejemplo. Los números en coma flotante se representan mediante una mantisa (mant) de N bits y un exponente (exp) de E bits. Este circuito limitador de precisión 1000 utiliza un incrementador 90 (circuito que incrementa en 1 el exp si Cout=1)

que recibe el exponente (exp) del número en coma flotante y la salida del acarreo (Cout) del sumador 35 de un bloque de redondeo 92 encargado del redondeo al más cercano de la mantisa. En una implementación alternativa, si la mantisa estuviese en complemento a dos, en lugar de Cout habría que usar una señal de desbordamiento. Si el “leading one” no está implícito habría que forzar el MSB de la mantisa (mant) a 1, cuando hay acarreo (como es 0 al poner el 1 simula el desplazamiento a la derecha). El bloque de redondeo 92 para el redondeo al más cercano de la mantisa (mant) se puede implementar como en el ejemplo de la Fig. 5, utilizando un sumador 35 y un circuito limitador de precisión 200 mediante truncamiento. Para realizar el redondeo unbiased se cambiaría el circuito limitador de precisión 200 mediante truncamiento por el bloque 16 de la Fig. 7.

La Fig. 19, muestra un circuito limitador de precisión 1100 mediante redondeo al más cercano para números en coma flotante en bloque, esto es, un grupo de números en coma flotante de mantisas diferentes (mant1, mant2, … mantL) y exponente común (exp), los cuales se procesan como un bloque de datos. En este caso por cada mantisa (mant1, mant2, …, mantL) hay un bloque de redondeo 92 con un sumador 35 y un circuito limitador de precisión 200 mediante truncamiento y las salidas de acarreo (Cout) de cada sumador 35 se conectan con una puerta OR 94 al incrementador 90 para incrementar el exponente en caso de que alguna de las mantisas se desborde. Pero en este caso el bit de leading uno no puede estar implícito, habiendo que forzar a 1 el MSB de la mantisa que se desborde, utilizando la puerta OR 95 (en la bifurcación N-1 mostrada en la figura se separa el MSB y luego se vuelve a poner). Por otro lado, si se incrementa el exponente, las mantisas que no provocaron desbordamiento deben desplazarse un bit a la derecha, mediante le desplazador 99. Para implementar redondeo al par más cercano (o sin sesgo), habría que añadir además un circuito calculador de sticky 17 y un circuito de forzado de bit 49 que se conectarían de manera similar a como se hace en el ejemplo de la Fig 7. Para implementar un circuito limitador de precisión para coma flotante en bloque mediante uno de los otros modos de redondeo, simplemente se usarían los limitadores de precisión descritos para coma fija y se aplicarían a las mantisas.

La Fig. 20 ilustra la implementación de un circuito detector de desbordamiento 1200 para números en coma flotante, que emplea un circuito detector de desbordamiento 1210 para números en coma fija, como por ejemplo el mostrado en la Fig. 12. Se supone que el exponente (exp) está representado en “exceso 2^(R-1)”, lo que hace que sea un complemento a dos con signo invertido. Si el exponente (exp) estuviese en complemento a

dos, dicho exponente entraría directamente al circuito detector de desbordamiento 1210 y las señales de desbordamiento (ovf) y de subdesbordamiento (underf) se intercambiarían.

Fig. 21 ilustra la implementación de un circuito detector de desbordamiento para medir cuántos bits por la izquierda hay que mantener de acuerdo a un ejemplo. El registro 80 acumula la salida de la puerta OR 82 de los exponentes que son positivos (solo interesan los exponentes positivos ya que los negativos no producen desbordamiento). El array de puertas XOR 73 de la Fig. 13 cambia por un array de puertas AND en esta Fig. 21 ya que en la Fig. 13 solo interesa la magnitud, calcular el valor absoluto; mientras que en la Fig. 21 se descartan los exponentes negativos forzándolos a cero con la puerta AND (hay que recordar que los exponentes se consideran en representación “exceso a M” lo que hace que el bit de signo a cero indique un número negativo y viceversa). Mirar los exponentes negativos no tiene mucho sentido porque en cuanto un número sea muy cercano a cero se van a usar todos los bits. Lo más razonable es usar primero este circuito para estimar un posible exponente y posteriormente usar un circuito como el de la Fig. 22 para corroborar que es suficiente precisión ya que también éste va a determinar el subdesbordamiento.

La Fig. 22 ilustra la implementación de un circuito detector de desbordamiento limitador de rango para flotantes con saturación de acuerdo a un ejemplo. Al igual que en anteriores ejemplos se ha supuesto que el “leading uno” de la mantisa (mant) está implícito y la representación del cero es un caso especial que se codifica con todos los bits a cero. En este ejemplo cuando se produce desbordamiento se satura al mayor valor posible de la mantisa y del exponente. De la misma forma cuando se produce un subdesbordamiento la salida se iguala al valor cero. El circuito limitador de rango para flotantes con saturación emplea un circuito detector de desbordamiento 1200 como el de la Fig. 20, que se conecta al exponente de la entrada. El valor de la mantisa de la salida será igual al de la mantisa de entrada, salvo si hay desbordamiento en cuyo caso se fuerza a todo uno mediante el array de puertas OR 97, o si hay subdesbordamiento, en cuyo caso se fuerza a todo cero mediante el array de puertas AND 98. De igual forma, el valor del exponente de la salida será igual al de la entrada, salvo si hay desbordamiento en cuyo caso se fuerza al inverso de la máscara utilizada en el detector de desbordamiento 1200, mediante el array de puertas NOT 111 y el multiplexor 112, o si hay subdesbordamiento, en cuyo caso se fuerza a todo cero mediante el array de puertas AND 113. El valor del signo de la salida será igual al de la entrada, salvo si hay subdesbordamiento, en cuyo caso se fuerza a cero mediante la puerta AND 114.

Para números en coma flotante en bloque el subdesbordamiento se haría sobre todas las mantisas y signos. En este caso, no parece muy razonable implementar saturación Para el desbordamiento porque habría que ver además qué mantisas lo han provocado, y dependiendo de cuánto se exceda el exponente habría que ver cuántas tienen los correspondientes MSBs a uno, y como éste sería además variable sería demasiado complejo implementarlo.

El limitador de rango y precisión en coma flotante sería análogo al caso de números en coma fija, tal que conectaríamos circuitos como los descritos en las Fig. 20 y 22 usando los esquemas descritos en las Fig. 16 y 17.

REIVINDICACIONES

1. Sistema para la optimización de anchos de palabra de circuitos digitales mediante simulaciones bit-true, que comprende un modelo del circuito objetivo (5), **caracterizado por que** dicho modelo del circuito objetivo (5) comprende al menos un circuito limitador (7) para modificar el valor de las señales cuyos anchos de palabra se quiere optimizar, estando cada circuito limitador (7) configurado para generar una salida con un valor correspondiente a limitar a nivel lógico la precisión y/o rango efectivo del valor de un dato de entrada (NUM_i) de acuerdo a una instrucción de control de precisión y/o rango efectivo (10_i) recibida por el circuito limitador (7).

2. Sistema según la reivindicación 1, **caracterizado por que** el modelo del circuito objetivo (5) es un circuito hardware que se implementa en al menos un PLD (2).

3. Sistema según la reivindicación 2, **caracterizado por que** el al menos un circuito limitador (7) está ubicado a la entrada y/o salida de un operador aritmético-lógico (9).

4. Sistema según la reivindicación 1, **caracterizado por que** el modelo del circuito objetivo (5) se implementa mediante software en un procesador (8).

5. Sistema según la reivindicación 4, **caracterizado por que** el procesador (8) comprende al menos una unidad funcional (12) con al menos una unidad aritmético-lógica (19), estando el al menos un circuito limitador (7) ubicado a la entrada y/o salida de la unidad aritmético-lógica (19).

6 Sistema según la reivindicación 5, **caracterizado por que** la unidad funcional (12) comprende al menos un elemento de memoria (13) para almacenar los valores de precisión/rango efectivos asociados a las variables que representan señales cuyo ancho de palabra se quiere optimizar.

7. Sistema según cualquiera de las reivindicaciones anteriores, **caracterizado por que** la instrucción de control de precisión y/o rango efectivo (10_i) que recibe un circuito limitador (7) incluye el número de bits a anular, una máscara o una instrucción indicando cómo modificar el valor o máscara utilizado en la iteración del ciclo de optimización anterior.

8. Sistema según la reivindicación 7, **caracterizado por que** la instrucción de control de precisión y/o rango efectivo (10_i) incluye además el modo de redondeo a utilizar.

9. Sistema según cualquiera de las reivindicaciones anteriores, **caracterizado por que** al menos un circuito limitador (7) comprende un circuito limitador de precisión configurado para recibir un primer número en coma fija de N bits en una primera entrada y generar en una salida un segundo número en coma fija de M bits, con N≥M, siendo los K MSBs de dicho segundo número el resultado de redondear, mediante algún modo de redondeo, dicho primer número mientras los (M-K) LSBs de dicho segundo número son igualados a cero, siendo K dependiente de la instrucción de control de precisión y/o rango efectivo (10_i) recibida en una segunda entrada y cumpliendo K≤M.

10. Sistema según la reivindicación 9, donde dicho primer número corresponde a la mantisa de un número en coma flotante o coma flotante en bloque y el circuito limitador de precisión está configurado además para incrementar el exponente y ajustar adecuadamente la mantisa de dicho número si el redondeo de la mantisa produce un desbordamiento.

11. Sistema según cualquiera de las reivindicaciones anteriores, **caracterizado por que** al menos un circuito limitador (7) comprende un circuito limitador de rango configurado para recibir un primer número en coma fija de L bits en una primera entrada y detectar si dicho primer número no se puede representar usando únicamente sus (L-R) LSBs, y en cuyo caso indicar un desbordamiento lógico, siendo R dependiente de la instrucción de control de precisión y/o rango efectivo (10_i) recibida en una segunda entrada y cumpliendo R <=L.

12. Sistema según la reivindicación 11, donde el circuito limitador de rango está configurado además para generar en una salida un segundo número en coma fija de L bits, cuyo valor es igual al del primer número en coma fija si no se indica desbordamiento, o en caso contrario, al máximo valor representable con L-R bits si dicho primer número es positivo o al mínimo valor representable con L-R bits si dicho primer número es negativo.

13. Sistema según la reivindicación 11, donde el primer número en coma fija corresponde al exponente de un primer número en coma flotante o coma flotante en bloque y el circuito limitador de rango está configurado además para distinguir entre si el desbordamiento se debe a que el primer número es demasiado grande o es demasiado pequeño.

14. Sistema según la reivindicación 13, donde el circuito limitador de rango está configurado además para recibir la mantisa y el signo del número en coma flotante y generar en una salida un segundo número en coma flotante, cuyo valor es igual al número en coma flotante de entrada si no se produce desbordamiento lógico, o en caso contrario, al máximo valor representable con L-R bits de exponente en valor absoluto y manteniendo el signo del número de entrada si dicho desbordamiento es por exceso o cero si dicho desbordamiento es por defecto.

15. Sistema según cualquiera de las reivindicaciones anteriores, **caracterizado por que** al menos un circuito limitador (7) comprende un circuito limitador de rango por wrapping configurado para recibir un primer número en coma fija de T bits en una primera entrada y generar en una salida un segundo número en coma fija de T bits, cuyo valor es el mismo que tendrían un número representado por los (T-P) LSBs de dicho primer número, siendo P dependiente de la instrucción de control de precisión y/o rango efectivo (10_i) recibida en una segunda entrada y cumpliendo P <=T.

16. Sistema según cualquiera de las reivindicaciones anteriores, **caracterizado por que** al menos un circuito limitador (7) comprende un circuito medidor de rango configurado para recibir una serie de números consecutivos de Q bits por una entrada y generar en una salida un valor que representa el número mínimo de bits necesarios para representar todos los números de dicha secuencia.

17. Sistema según la reivindicación 16, donde el circuito medidor de rango está configurado además para, de dicha serie de números de entrada, solo tener en cuenta los números que representan valores positivos.

18. Sistema según cualquiera de las reivindicaciones anteriores, **caracterizado por que** comprende un módulo generador de señales de entrada (4), encargado de generar una serie de valores de entrada (VAL_{IN}) que alimenta la entrada de datos del modelo del circuito objetivo (5) durante la simulación bit-true.

19. Sistema según cualquiera de las reivindicaciones anteriores, **caracterizado por que** comprende un módulo calculador de parámetros de calidad (6), encargado de recibir una serie de valores de salida (VAL_{OUT}) producidos por el modelo del circuito objetivo (5) en cada

iteración de la simulación bit-true y calcular al menos un parámetro de calidad (P_Q).

20. Sistema según la reivindicación 19, **caracterizado por que** comprende una unidad de control (3) configurada para:

generar, en cada iteración de la simulación bit-true, una serie de instrucciones de control de precisión y/o rango efectivo (10) que especifican la precisión y/o rango efectivo de las distintas señales del circuito objetivo,

recibir los parámetros de calidad (P_Q) para cada iteración, y

determinar, al finalizar la simulación bit-true, los diferentes anchos de palabra del circuito digital óptimos.

21. Método para la optimización de anchos de palabra de circuitos digitales mediante simulaciones bit-true, **caracterizado por que** comprende:

enviar, a al menos un circuito limitador (7) incluido en un modelo del circuito objetivo (5) y encargado de modificar el valor de las señales cuyos anchos de palabra se quiere optimizar, una instrucción de control de precisión y/o rango efectivo (10_i),

generar, cada circuito limitador (7), una salida con un valor correspondiente a limitar a nivel lógico la precisión y/o rango efectivo del valor de un dato de entrada (NUM_i) de acuerdo a la instrucción de control de precisión y/o rango efectivo (10) recibida.

22. Método según la reivindicación 21, **caracterizado por que** el modelo del circuito objetivo (5) es un circuito hardware que se implementa en al menos un PLD (2).

23. Método según la reivindicación 22, **caracterizado por que** cada circuito limitador (7) está ubicado a la entrada y/o salida de un operador aritmético-lógico (9).

24. Método según la reivindicación 21, **caracterizado por que** el modelo del circuito objetivo (5) se implementa se implementa mediante software en un procesador (8).

25. Método según la reivindicación 24, **caracterizado por que** cada circuito limitador (7) está ubicado a la entrada y/o salida de la unidad aritmético-lógica (19) incluida en al menos una unidad funcional (12) del procesador (8).

26. Método según cualquiera de las reivindicaciones 21 a 25, **caracterizado por que** comprende generar una serie de valores de entrada (VAL_{IN}) que alimenta la entrada de

datos del modelo del circuito objetivo (5) durante la simulación bit-true.

27. Método según cualquiera de las reivindicaciones 21 a 26, **caracterizado por que** comprende calcular, a partir de una serie de valores de salida (VAL_{OUT}) producidos por el modelo del circuito objetivo (5) en cada iteración de la simulación bit-true, al menos un parámetro de calidad (P_Q).

28. Método según cualquiera de la reivindicación 27, **caracterizado por que** comprende:

generar, en cada iteración de la simulación bit-true, una serie de instrucciones de control de precisión y/o rango efectivo (10) que especifican la precisión y/o rango efectivo de las distintas señales del circuito objetivo,

determinar, al finalizar la simulación bit-true, los diferentes anchos de palabra del circuito digital óptimos en base a los parámetros de calidad (P_Q) calculados en las distintas iteraciones.

29. Un producto de programa que comprende medios de instrucciones de programa para llevar a cabo el método definido en cualquiera de las reivindicaciones 24 a 25 cuando el programa se ejecuta en un procesador (8).

30. Un producto de acuerdo con la reivindicación 29, almacenado en un medio de soporte de programas.

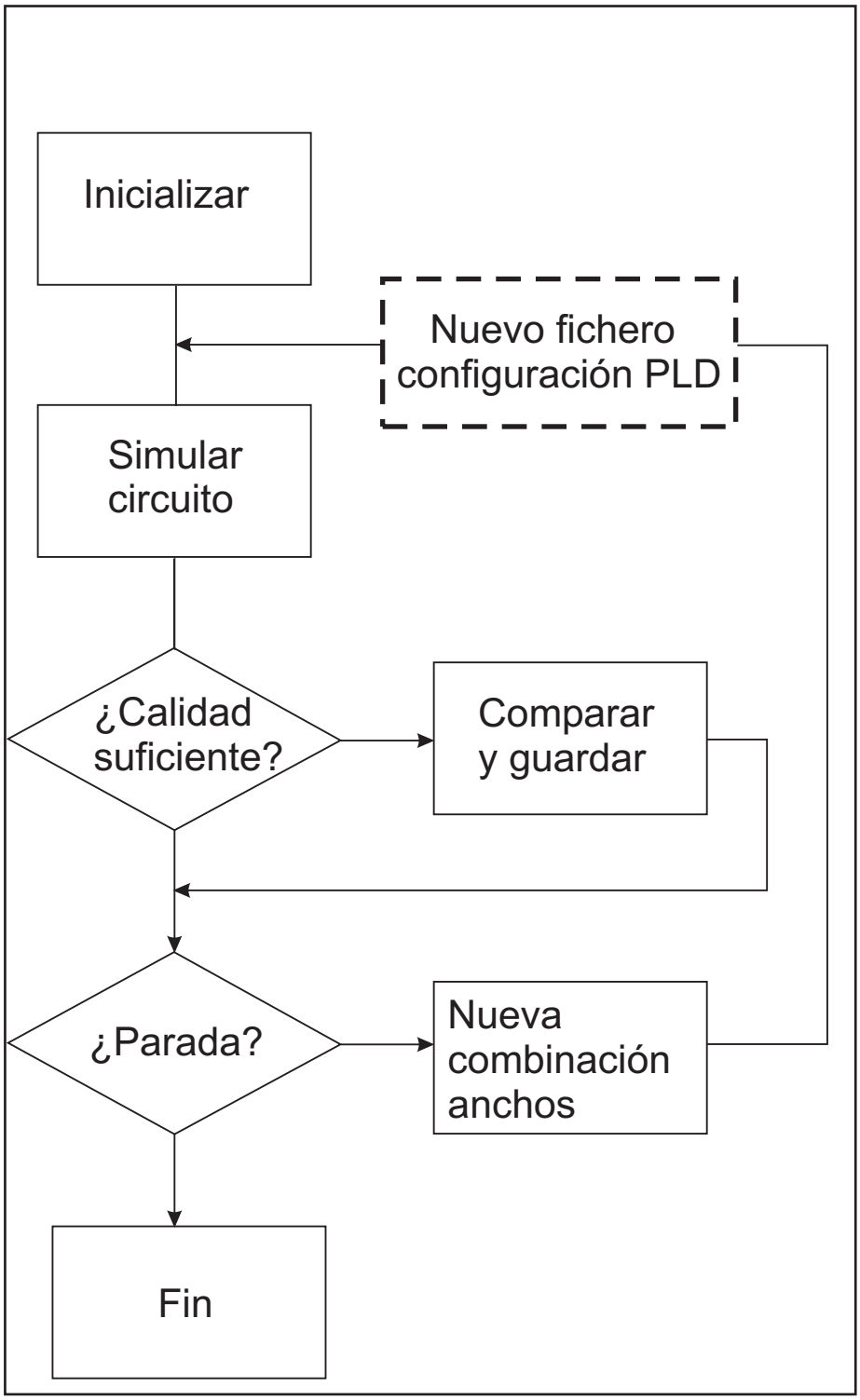
Inicializar
Nuevo fichero configuración PLD
Simular circuito
¿Calidad suficiente?
Comparar y guardar
¿Parada?
Nueva combinación anchos
Fin

Fig. 1

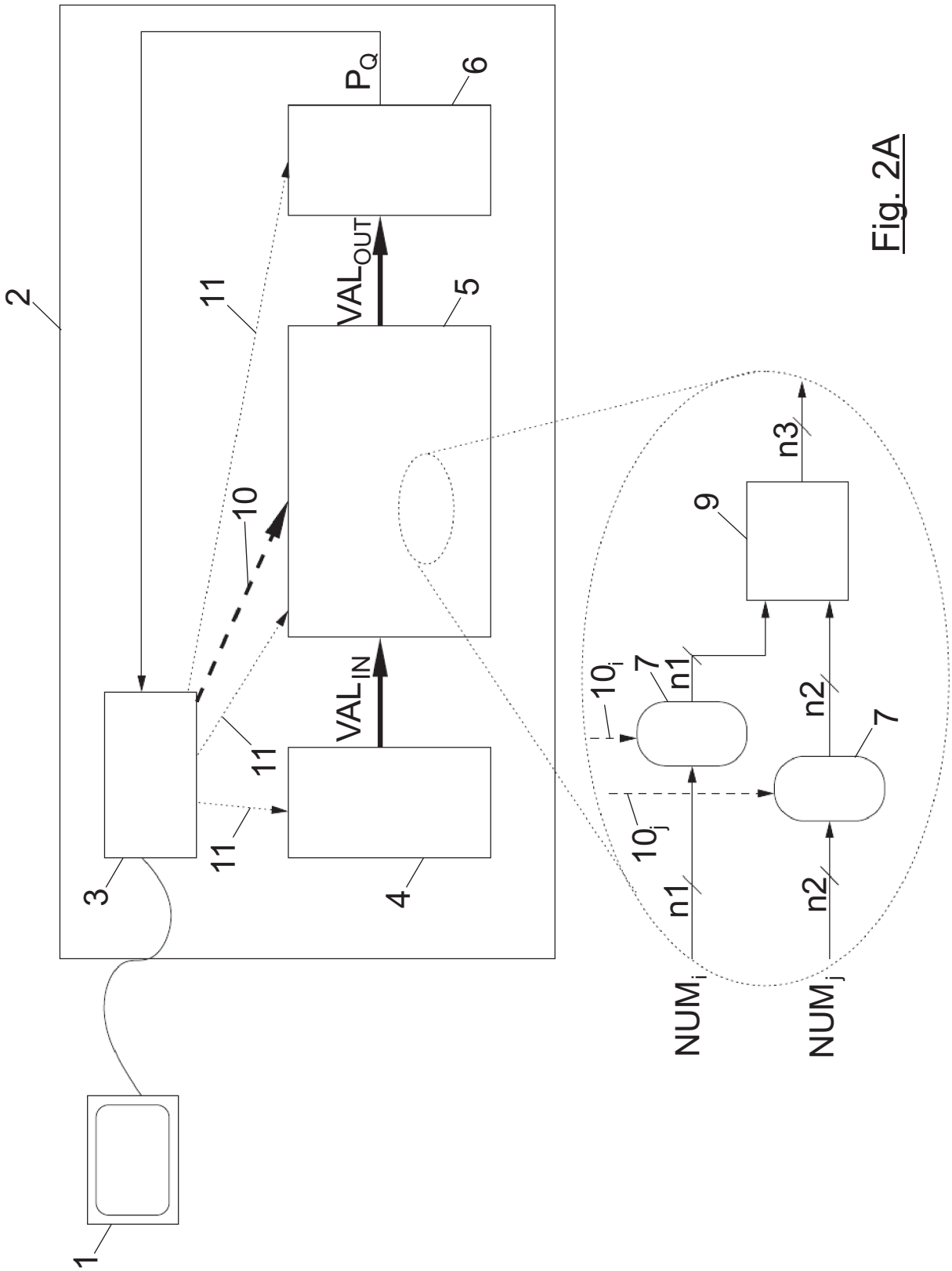
2
11
P_Q
6
VAL_{OUT}
5
10
11
VAL_{IN}
11
3
4
1
9
n3
10_i
7
n1
n2
7
10_j
n1
n2
NUM_i
NUM_j

Fig. 2A

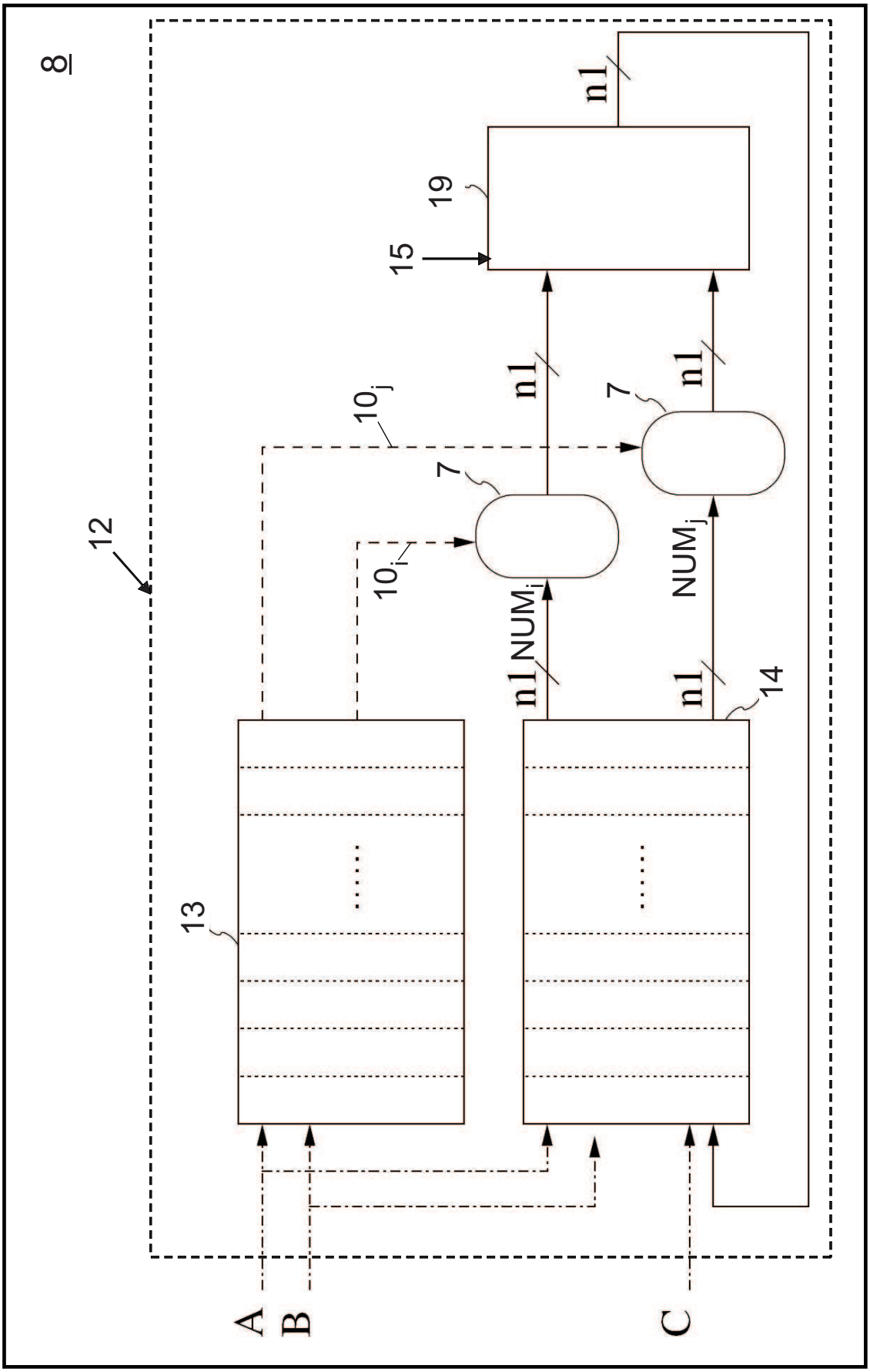

Fig. 2B

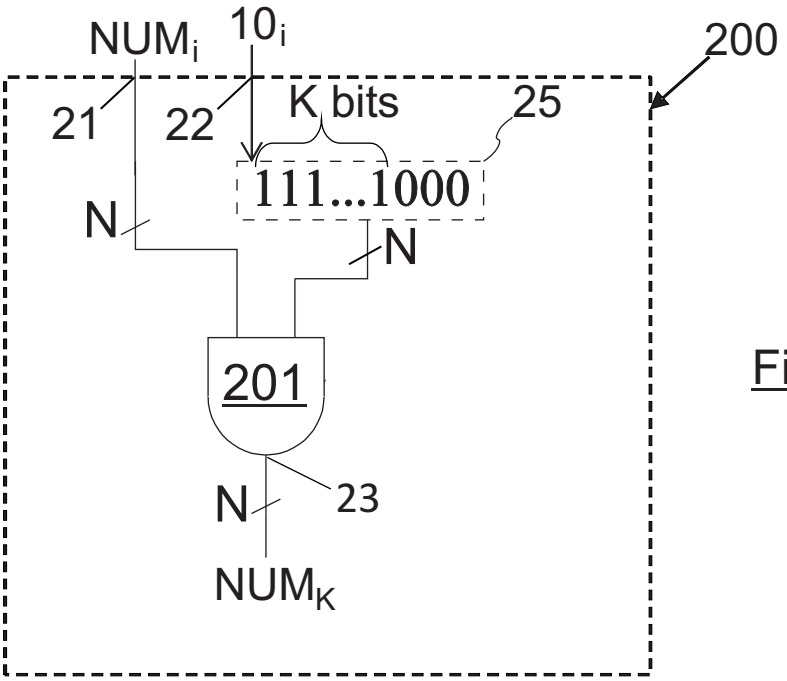

Fig. 3

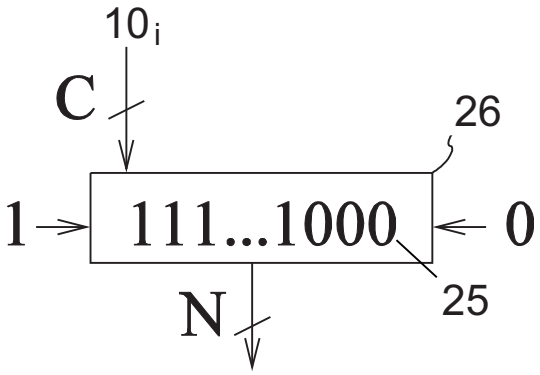

Fig. 4A

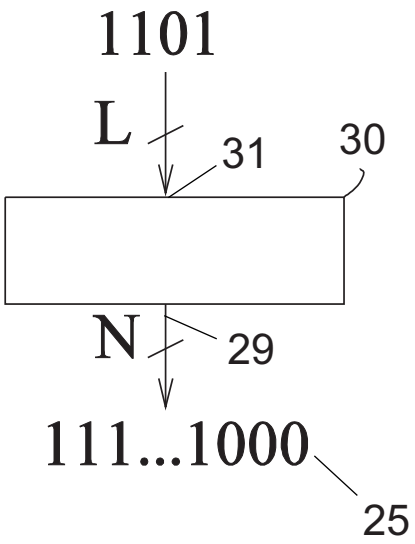

Fig. 4B

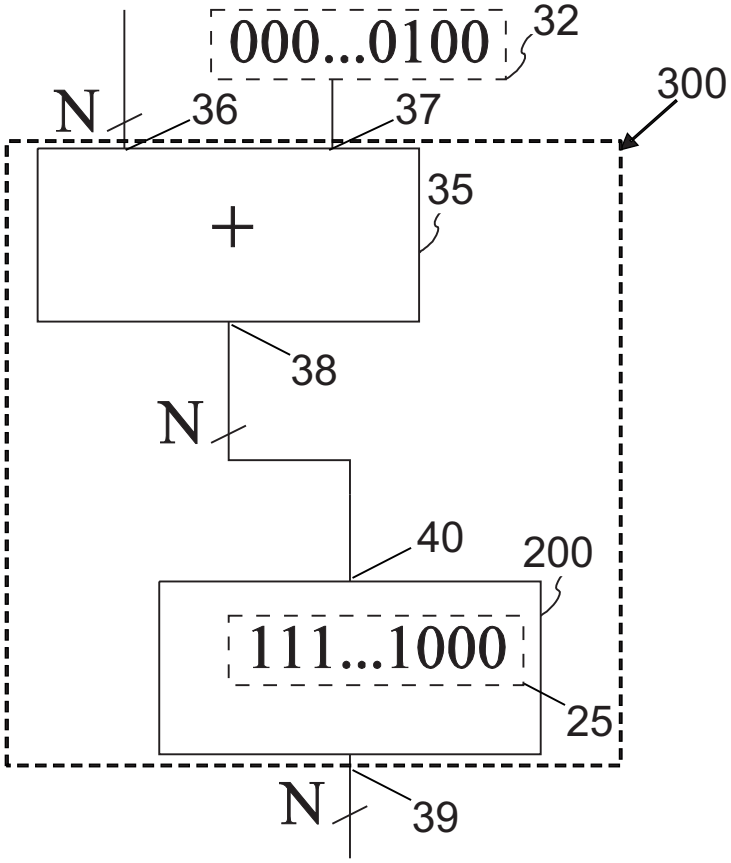
000...0100
32
300
N
36
37
35
+
38
N
40
200
111...1000
25
N
39

Fig. 5

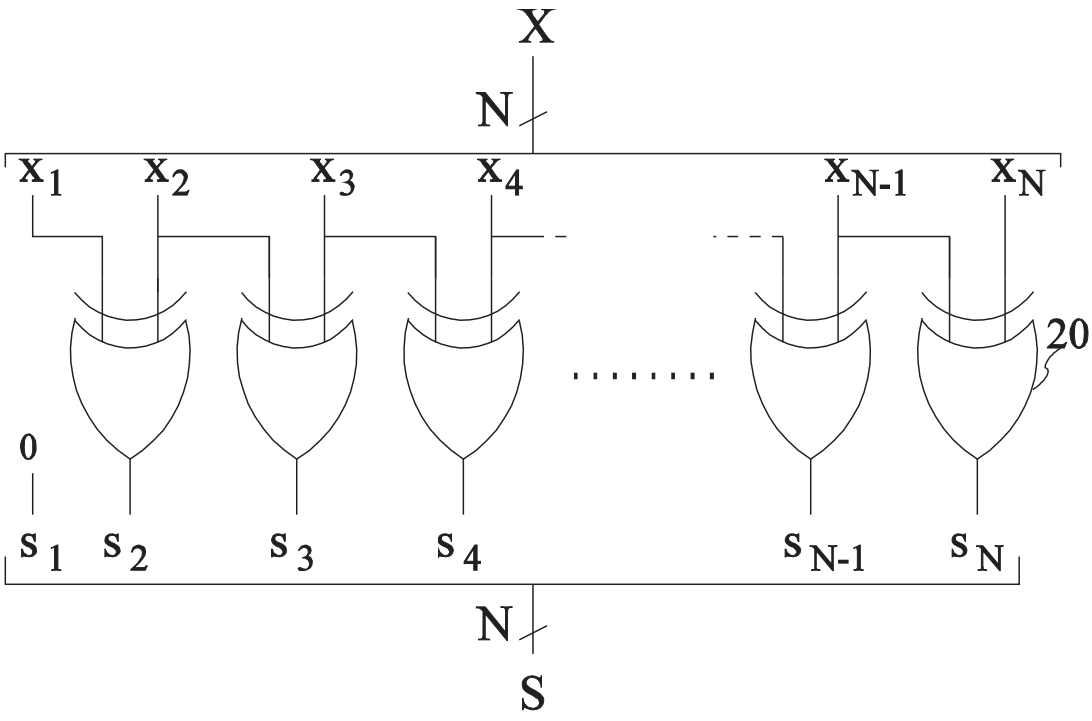
X
N
x1
x2
x3
x4
xN-1
xN
20
0
s1
s2
s3
s4
sN-1
sN
N
S

Fig. 6

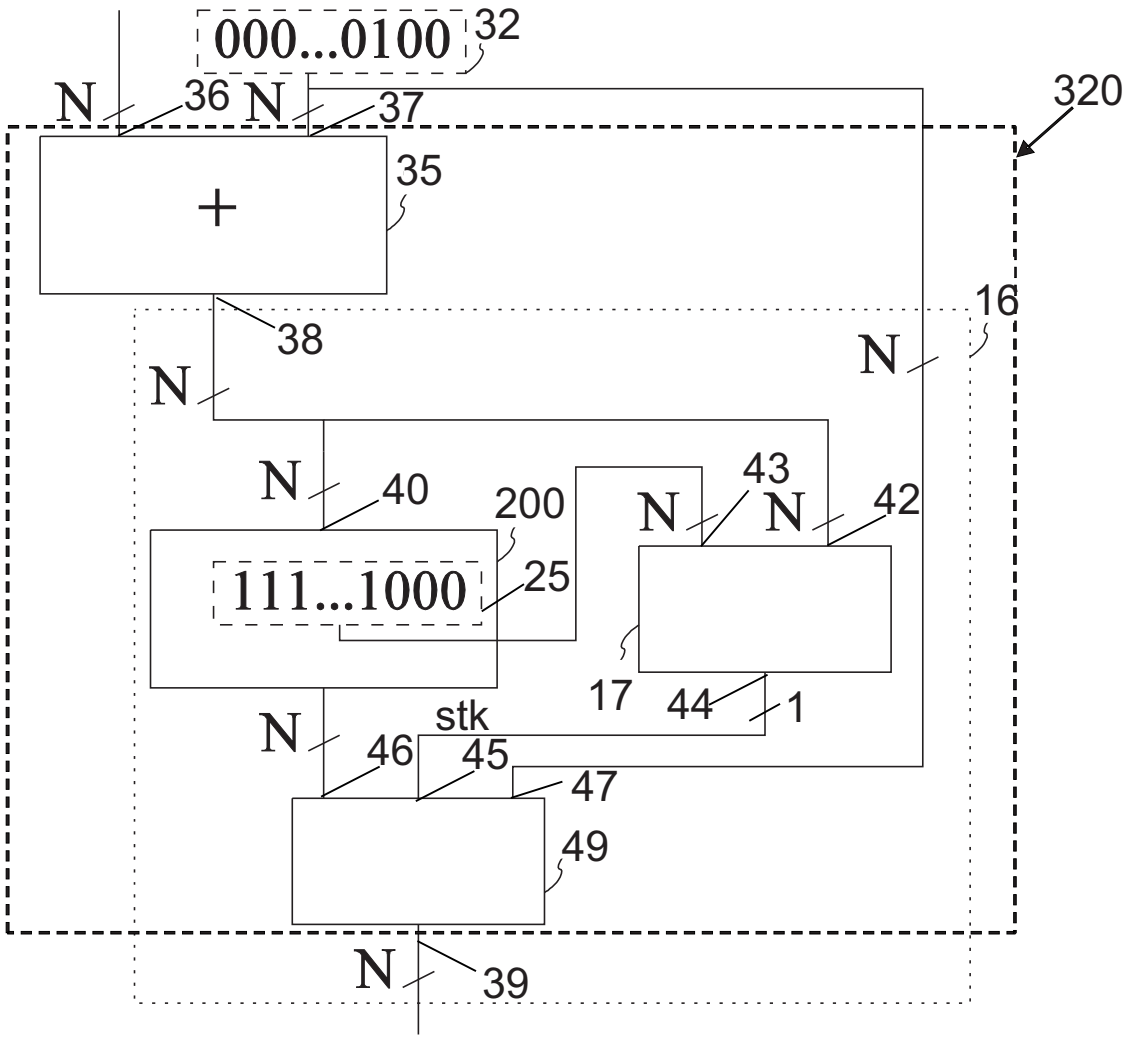
000...0100
32
320
N
36
N
37
35
+
16
38
N
N
N
40
200
43
N
N
42
111...1000
25
17
44
1
N
stk
46
45
47
49
N
39

Fig. 7

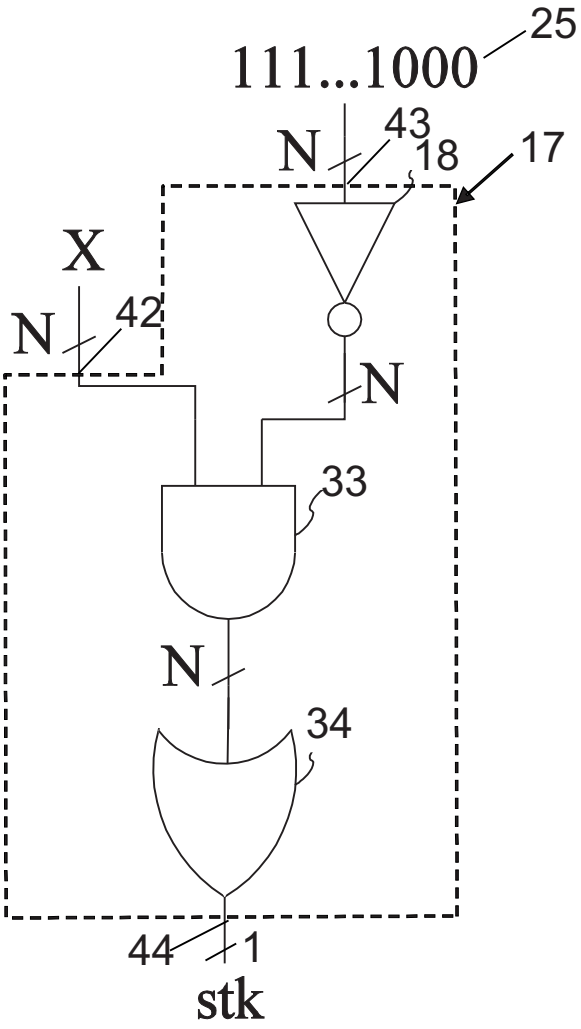
111...1000
25
N
43
18
17
X
N
42
N
33
N
34
44
1
stk

Fig. 8

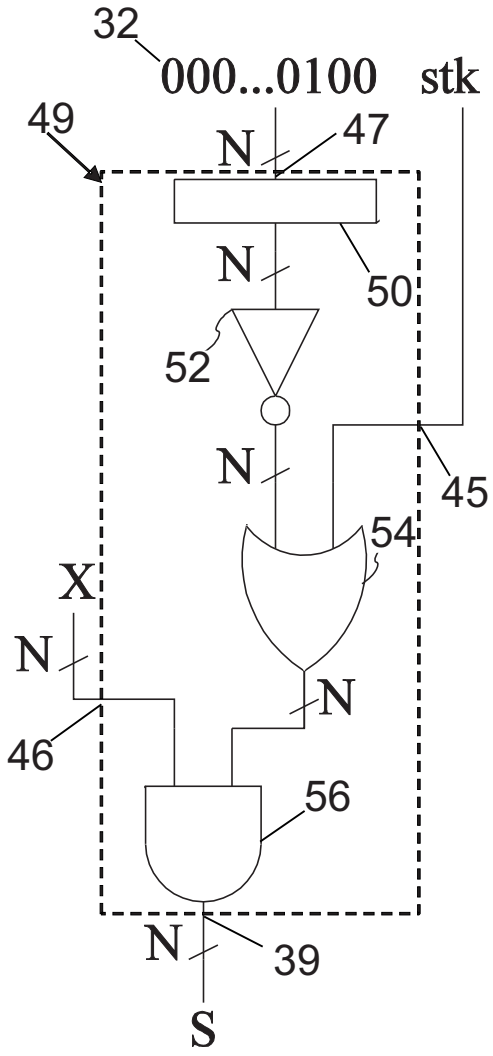
32
000...0100
stk
49
N
47
N
50
52
N
45
54
X
N
N
46
56
N
39
S

Fig. 9

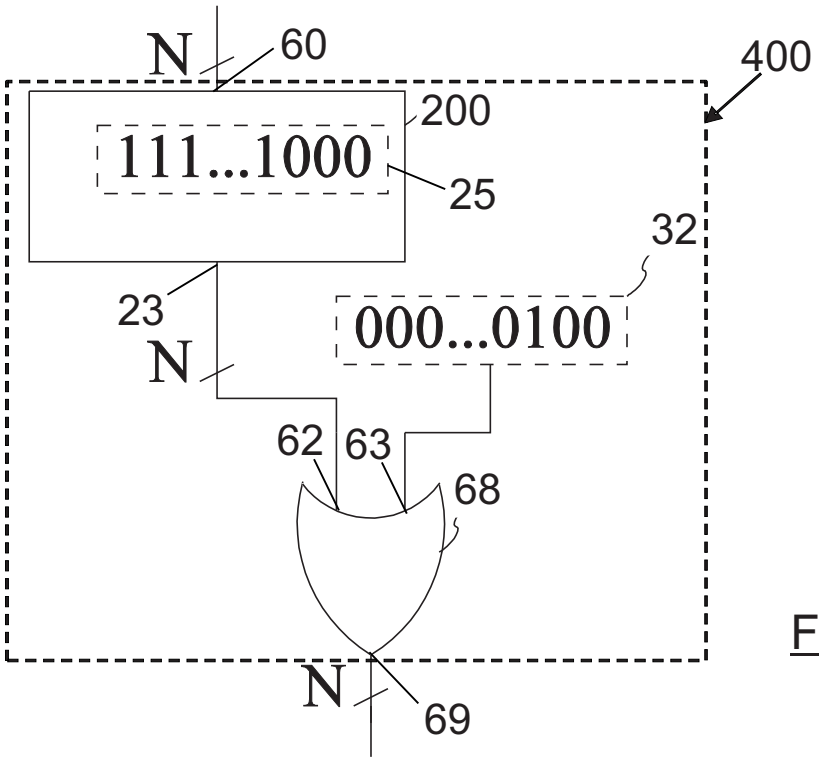
N
60
400
200
111...1000
25
32
23
000...0100
N
62
63
68
N
69

Fig. 10

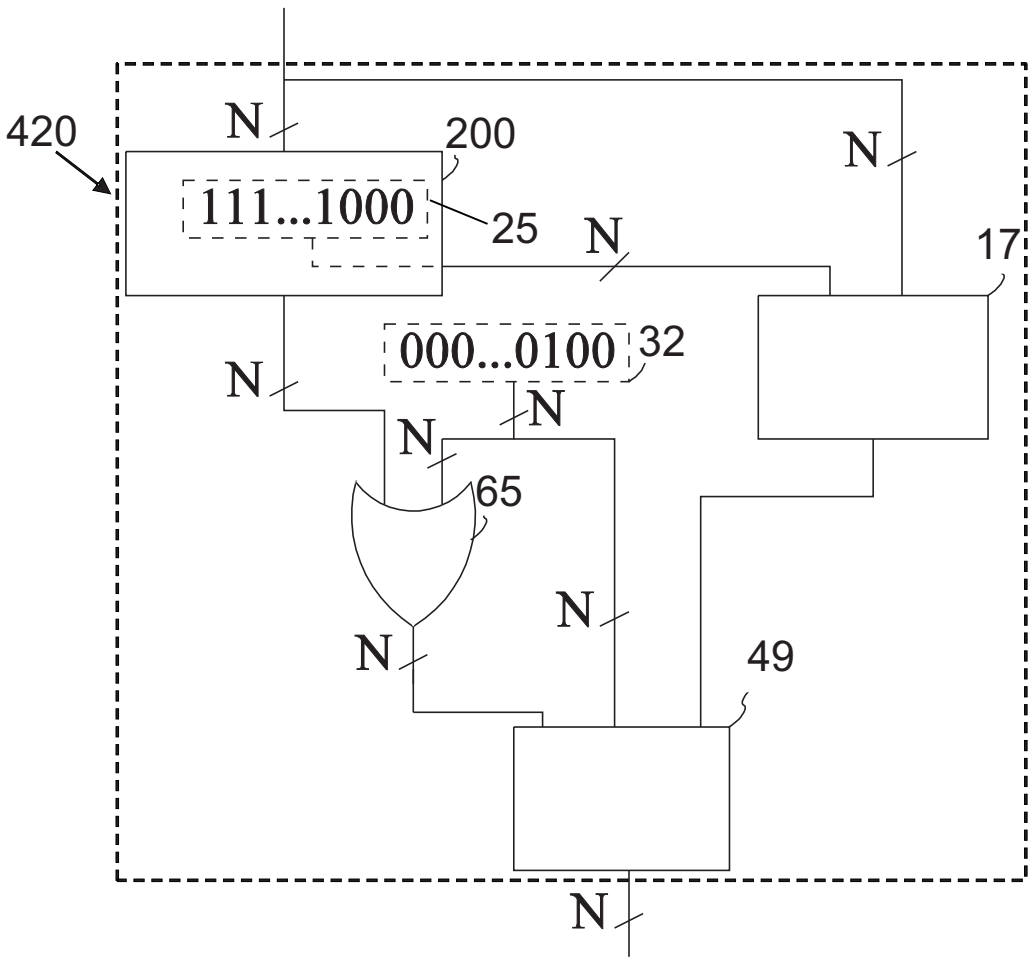
420
N
200
N
111...1000
25
N
17
000...0100
32
N
N
N
65
N
N
49
N

Fig. 11

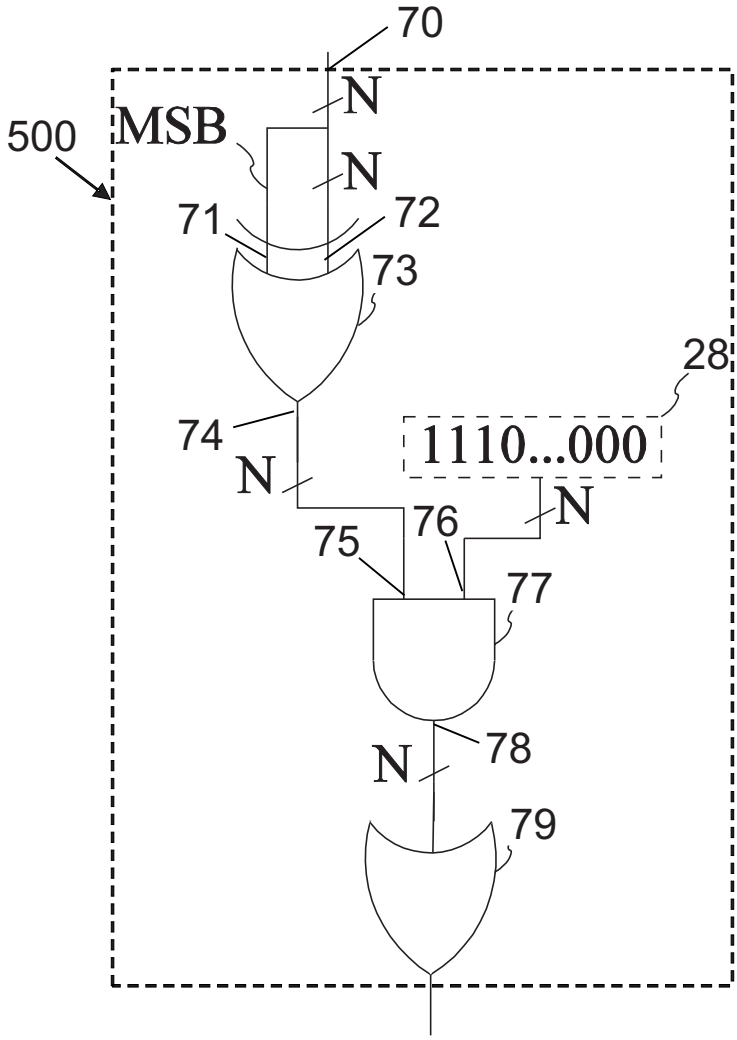

Fig. 12

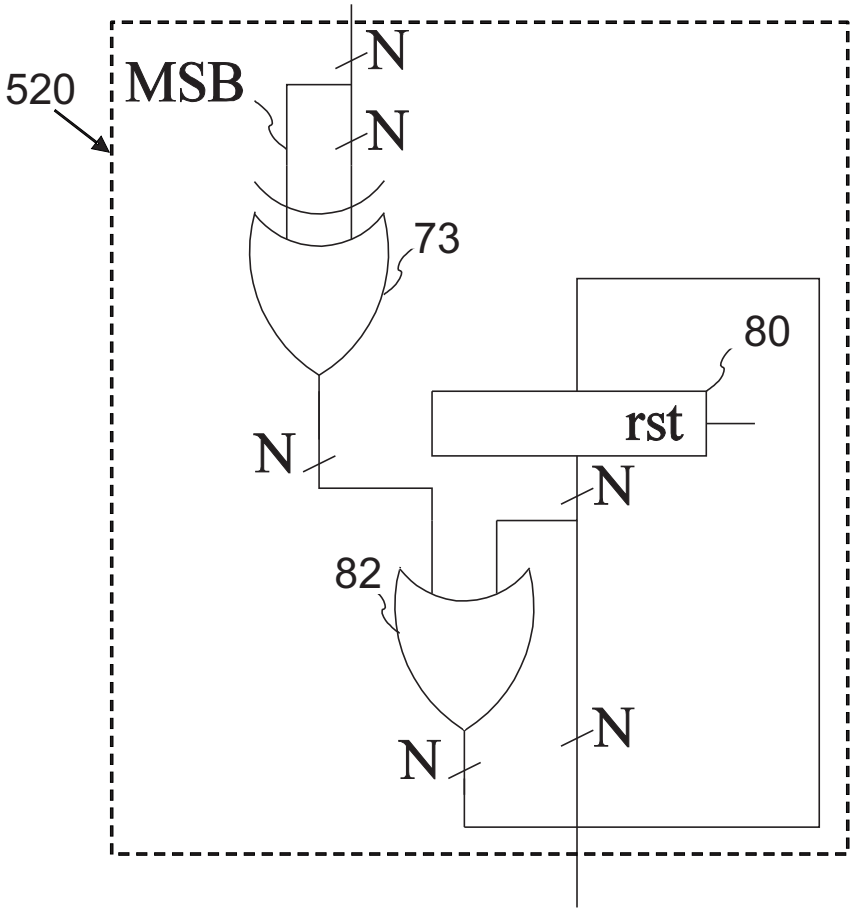
520
MSB
N
N
73
80
rst
N
N
82
N
N

Fig. 13

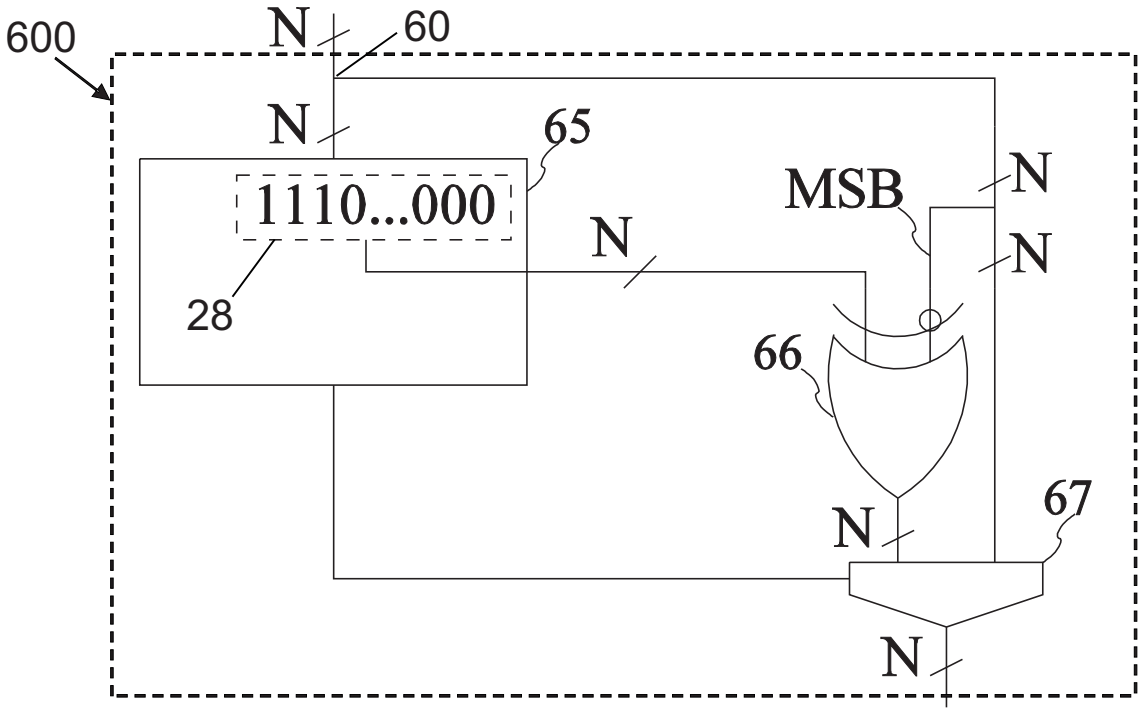
600
N
60
N
65
1110...000
28
N
MSB
N
N
66
N
67
N

Fig. 14

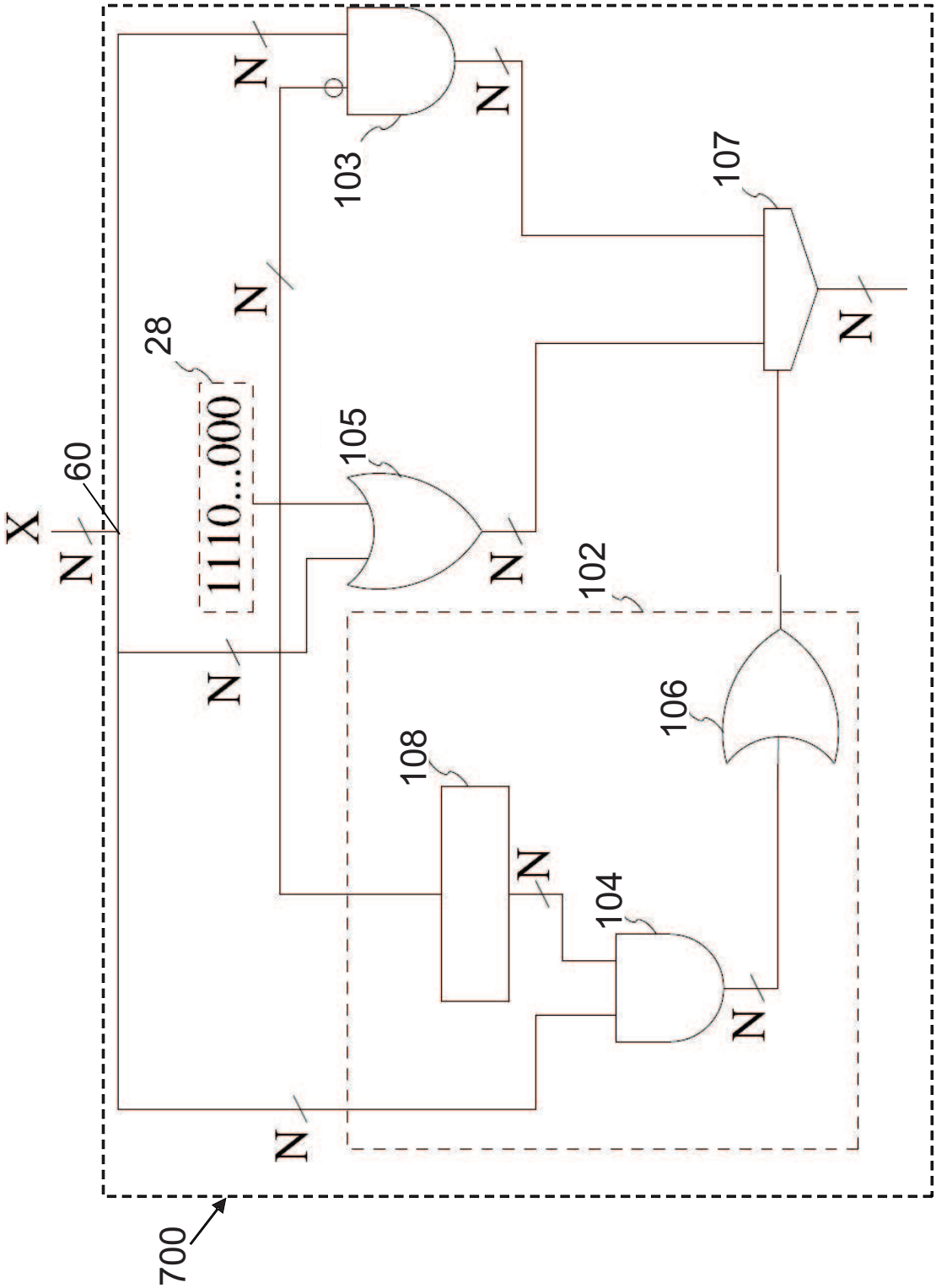

Fig. 15

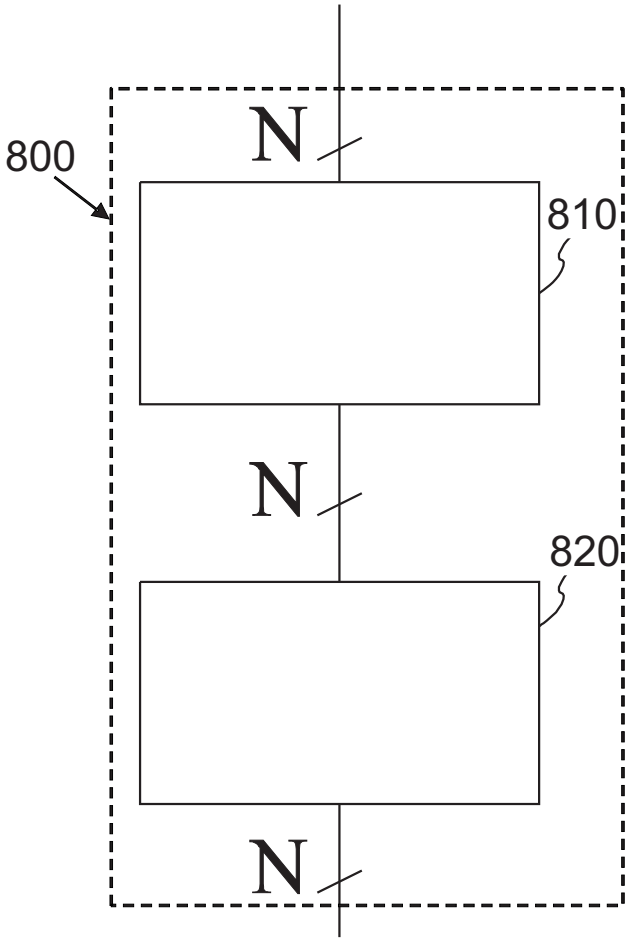

Fig. 16

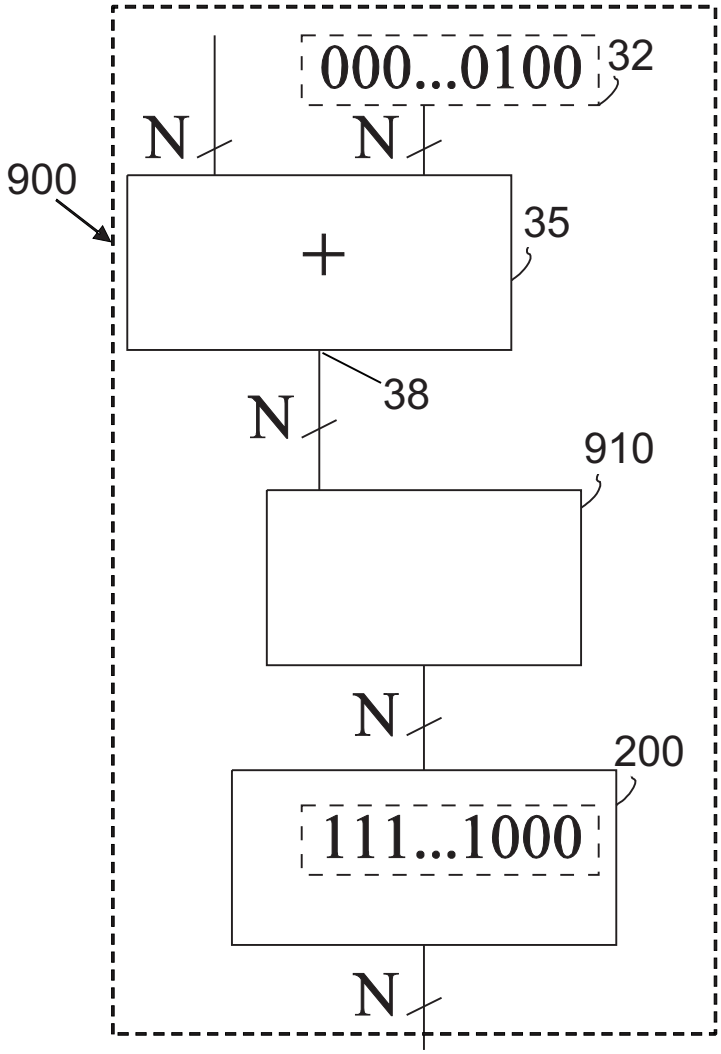

Fig. 17

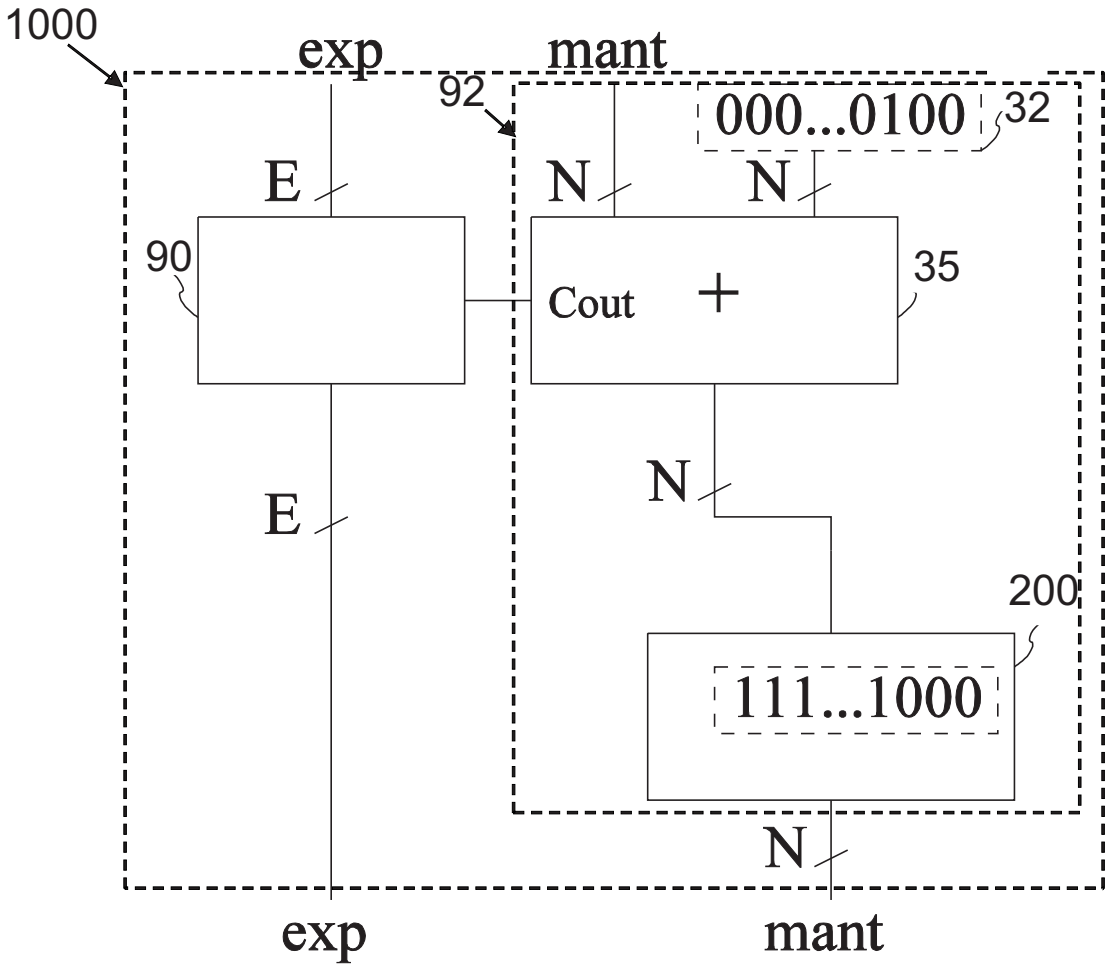
1000
exp
mant
92
000...0100
32
E
N
N
90
Cout
+
35
N
E
200
111...1000
N
exp
mant

Fig. 18

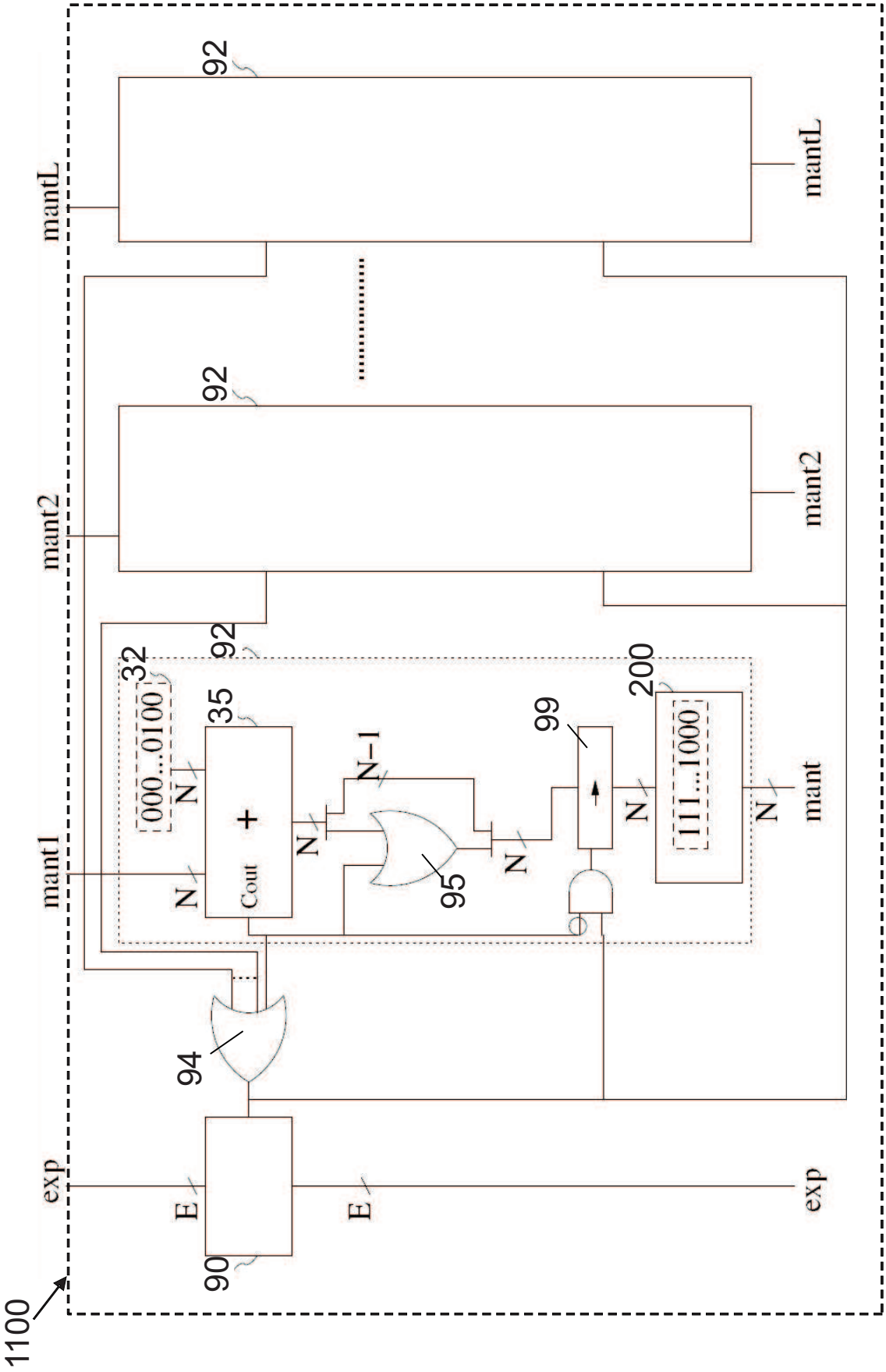
1100
mantL
92
mantL
mant2
92
mant2
mant1
92
32
000...0100
35
N
N
+
Cout
N
N–1
95
N
99
200
111...1000
N
mant
exp
E
90
94
E
exp

Fig. 19

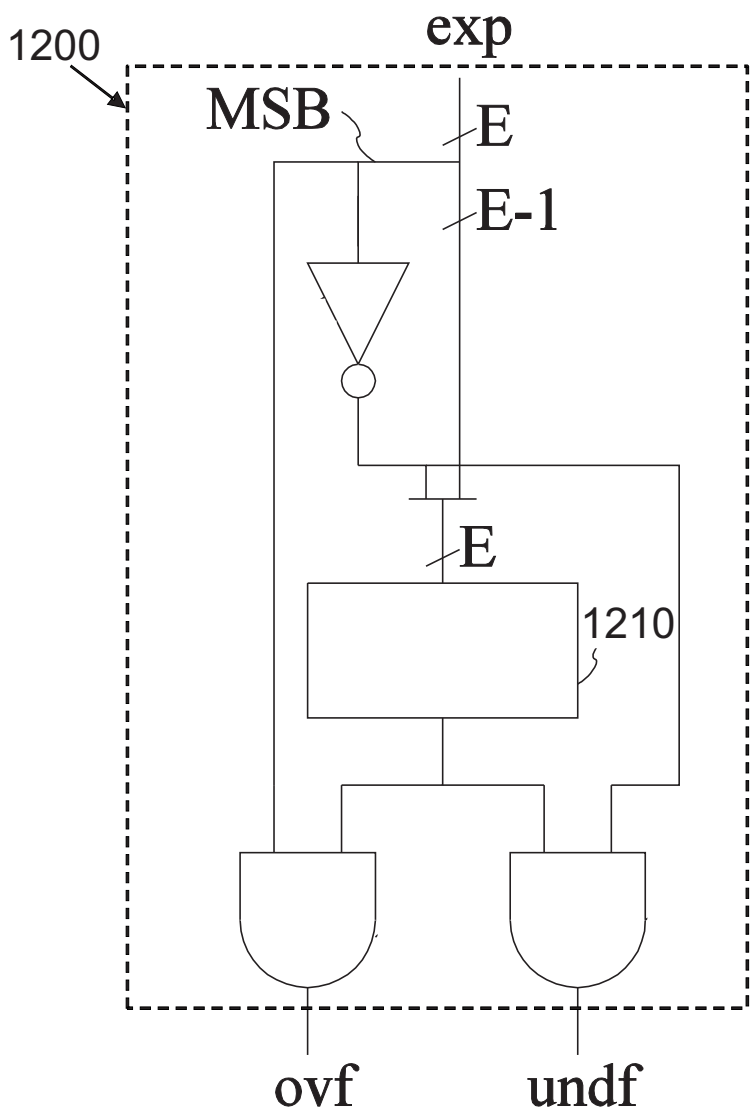

Fig. 20

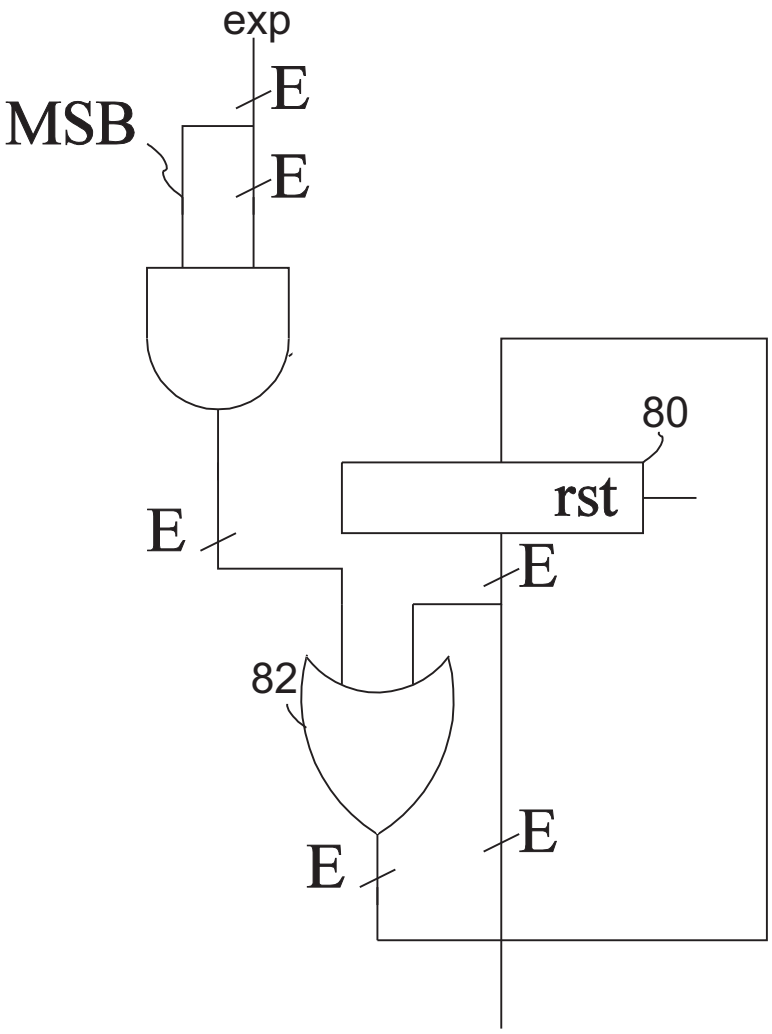

Fig. 21

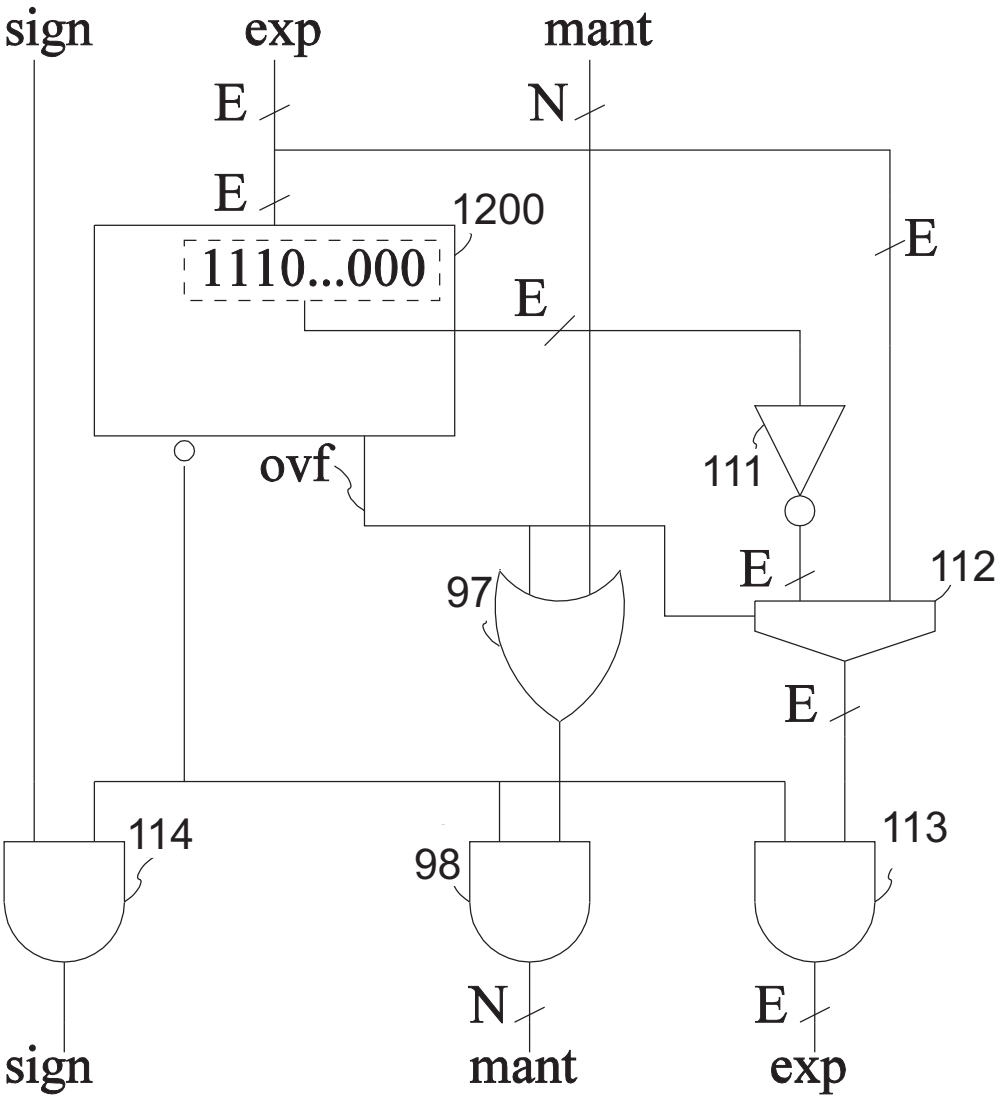
sign
exp
mant
E
N
E
1200
1110...000
E
E
ovf
111
E
112
97
E
114
98
113
N
E
sign
mant
exp

Fig. 22

OFICINA ESPAÑOLA
DE PATENTES Y MARCAS

ESPAÑA

(21) N.º solicitud: 201530862

(22) Fecha de presentación de la solicitud: 18.06.2015

(32) Fecha de prioridad:

INFORME SOBRE EL ESTADO DE LA TECNICA

(51) **Int. Cl. :** ***G06F7/00*** (2006.01)
G06F17/00 (2006.01)

DOCUMENTOS RELEVANTES

Categoría	(56) **Documentos citados**	**Reivindicaciones afectadas**
X	GAFFAR A A et al. Unifying Bit-Width Optimisation for Fixed-Point and Floating-Point Designs.Field-Programmable Custom Computing Machines, 2004. FCCM 2004. 12th An nual IEEE Symposium on Napa, CA, USA 20-23 Abril 2004, 20040420; 20040420-20040423 Piscataway, NJ, USA, IEEE 20.04.2004 VOL: Págs: 79-88 ISBN 978-0-7695-2230-2; ISBN 0-7695-2230-0 Doi: doi:10.1109/FCCM.2004.59. Todo el documento.	1-30
A	BINGYANG LIU et al. Research on Fixed-Point Simulation Approaches in the Design of FPGAs. Information Engineering and Computer Science (ICIECS), 2010 2nd International Conference on, 20101225 IEEE, Piscataway, NJ, USA 25.12.2010 VOL: Págs: 1-4 ISBN 978-1-4244-7939-9; ISBN 1-4244-7939-8. Todo el documento.	1-30
A	DE COSTER L et al. Code generation for compiled bit-true simulation of DSP applications. System Synthesis, 1998. Proceedings. 11th International Symposium on Hsinchu, Taiwan 2-4 Dic. 1998, 19981202; 19981202-19981204 Los Alamitos, CA, USA, IEEE Comput. Soc, US 02.12.1998 VOL: Págs: 9-14 ISBN 978-0-8186-8623-8; ISBN 0-8186-8623-5 Doi:10.1109/ISSS.1998.730590. Todo el documento.	1-30

Categoría de los documentos citados
X: de particular relevancia
Y: de particular relevancia combinado con otro/s de la misma categoría
A: refleja el estado de la técnica
O: referido a divulgación no escrita
P: publicado entre la fecha de prioridad y la de presentación de la solicitud
E: documento anterior, pero publicado después de la fecha de presentación de la solicitud

El presente informe ha sido realizado

☒ para todas las reivindicaciones ☐ para las reivindicaciones nº:

Fecha de realización del informe	**Examinador**	**Página**
22.02.2016	M. Muñoz Sánchez	1/4

Documentación mínima buscada (sistema de clasificación seguido de los símbolos de clasificación)

G06F

Bases de datos electrónicas consultadas durante la búsqueda (nombre de la base de datos y, si es posible, términos de búsqueda utilizados)

INVENES, EPODOC, WPI, NPL, XPIEE; XPI3E, XPESP, Internet

Fecha de Realización de la Opinión Escrita: 22.02.2016

Declaración

Novedad (Art. 6.1 LP 11/1986)	Reivindicaciones	1-30	**SI**
	Reivindicaciones		**NO**
Actividad inventiva (Art. 8.1 LP11/1986)	Reivindicaciones		**SI**
	Reivindicaciones	1-30	**NO**

Se considera que la solicitud cumple con el requisito de aplicación industrial. Este requisito fue evaluado durante la fase de examen formal y técnico de la solicitud (Artículo 31.2 Ley 11/1986).

Base de la Opinión.-

La presente opinión se ha realizado sobre la base de la solicitud de patente tal y como se publica.

1. Documentos considerados.-

A continuación se relacionan los documentos pertenecientes al estado de la técnica tomados en consideración para la realización de esta opinión.

Documento	Número Publicación o Identificación	Fecha Publicación
D01	GAFFAR A A et al. Unifying Bit-Width Optimisation for Fixed-Point and Floating-Point Designs.Field-Programmable Custom Computing Machines, 2004. FCCM 2004. 12th An nual IEEE Symposium on Napa, CA, USA 20-23 Abril 2004, 20040420; 20040420-20040423 Piscataway, NJ, USA, IEEE 20.04.2004 VOL: Págs: 79-88 ISBN 978-0-7695-2230-2; ISBN 0-7695-2230-0 Doi: 10.1109/FCCM.2004.59. Todo el documento.	20.04.2004
D02	BINGYANG LIU et al. Research on Fixed-Point Simulation Approaches in the Design of FPGAs. Information Engineering and Computer Science (ICIECS), 2010 2nd International Conference on, 20101225 IEEE, Piscataway, NJ, USA 25.12.2010 VOL: Págs: 1-4 ISBN 978-1-4244-7939-9; ISBN 1-4244-7939-8. Todo el documento.	25.12.2010
D03	DE COSTER L et al. Code generation for compiled bit-true simulation of DSP applications. System Synthesis, 1998. Proceedings. 11th International Symposium on Hsinchu, Taiwan 2-4 Dic. 1998, 19981202; 19981202-19981204 Los Alamitos, CA, USA, IEEE Comput. Soc, US 02.12.1998 VOL: Págs: 9-14 ISBN 978-0-8186-8623-8; ISBN 0-8186-8623-5 doi:10.1109/ISSS.1998.730590.	02.12.1998

2. Declaración motivada según los artículos 29.6 y 29.7 del Reglamento de ejecución de la Ley 11/1986, de 20 de marzo, de Patentes sobre la novedad y la actividad inventiva; citas y explicaciones en apoyo de esta declaración

Se considera D01 el documento más próximo del estado de la técnica al objeto de la solicitud.

Reivindicaciones independientes

Reivindicación 1: El documento D01, divulga un método de verificación de simulaciones bit-true de anchos de palabra finitos en implementaciones VLSI para aplicaciones de procesamiento de señales digitales (DSP). El documento D01 además explica los mecanismos típicos de limitación de datos (truncamiento/ redondeo para el bit menos significativo, desbordamiento para el bit más significativo). El contexto general de simulación aparece también mencionado al igual que ocurre en los documentos D02 y D03. En este sentido el documento D01 afecta a la actividad inventiva de la reivindicación 1 según el art. 8.1 de la ley 11/86 de patentes.

Reivindicación 21: El contenido de esta reivindicación y de la reivindicación 1 se corresponden estrictamente difiriendo solo en el objeto reivindicado (sistema/ método) por lo que el documento D01 también afecta a la actividad inventiva de la reivindicación 21 según el art. 8.1 de la ley 11/86 de patentes.

Reivindicación 29: En cuanto al producto de programa reivindicado lo mismo cabe decir que con respecto al procedimiento y así se concluye que el documento D01 también afecta a la actividad inventiva de la reivindicación 29 según el art. 8.1 de la ley 11/86 de patentes.

Reivindicaciones dependientes

Reivindicación 2-20: Las características contenidas en estas reivindicaciones no se consideran suficientemente definidas para apartarse de un esquema general de simulación en el que se van probando distintos anchos de palabra para distintos valores de entrada del circuito objetivo, o bien se realizan operaciones sencillas propias de las representaciones inexactas por falta de precisión o rango siendo, en la forma en la que se reivindican evidentes para el experto en la materia. Por tanto, el documento D01 afecta a la actividad inventiva de las reivindicaciones 2-20 según el art. 8.1 de la ley 11/86 de patentes.

Reivindicaciones 22-28, 30: Dado el paralelismo directo existente entre el contenido de estas reivindicaciones y las reivindicaciones 2-20 se concluye que el documento D01 afecta a la actividad inventiva de las reivindicaciones 22-28 y 30 según el art. 8.1 de la ley 11/86 de patentes.