(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 372 132**

(21) Número de solicitud: 200901646

(51) Int. Cl.:
G06F 7/49 (2006.01)
G06F 7/535 (2006.01)

(12) SOLICITUD DE PATENTE A1

(22) Fecha de presentación: **24.07.2009**

(43) Fecha de publicación de la solicitud: **16.01.2012**

(43) Fecha de publicación del folleto de la solicitud: **16.01.2012**

(71) Solicitante/s: **Universidad Autónoma de Madrid**
c/ Einstein, 3
28049 Madrid, ES

(72) Inventor/es: **Sutter, Gustavo**

(74) Agente: **Arias Sanz, Juan**

(54) Título: **Método, célula de recurrencia y circuito para realizar divisiones con operandos de coma fija de alta velocidad.**

(57) Resumen:
Método, célula de recurrencia y circuito para realizar divisiones con operandos de coma fija de alta velocidad.
La presente invención se refiere a un método para aplicar en las células de recurrencia de un circuito divisor para dividir un dividendo de coma fija por un divisor de coma fija. Además se proporcionan dichas células de recurrencia y un circuito divisor, que divide en una base $r = 2^k$, generando k bits en cada iteración. El divisor divide un dividendo X de n bits por un divisor Y de m bits mayor que cero, devolviendo un cociente Q de n bits y un resto R de m bits. La presente invención hace la división más rápida, preferentemente usando una propagación de acarreo eficaz para la suma y la resta. El circuito divisor puede modificarse fácilmente con el fin de devolver más bits fraccionales en el cociente Q. Se describen dos arquitecturas, denominadas para la célula de recurrencia de dígitos. La primera es para una implementación de hardware general y la segunda está optimizada para la lógica programable. También se proporciona un conjunto de instrucciones para implementar dicho método en un medio programable, y/o configurar dicho u otro medio programable.

ES 2 372 132 A1

DESCRIPCIÓN

Método, célula de recurrencia y circuito para realizar divisiones con operandos de coma fija de alta velocidad.

Campo de la invención

La presente invención se encuentra dentro del campo de las divisiones entre dos números de coma fija. Más en particular, se refiere a un método para aplicar en una célula de recurrencia de un circuito divisor, dicha célula de recurrencia y un circuito divisor que comprende estas células. La invención puede adaptarse para llevar a cabo la división entre un dividendo con signo y un divisor con signo, y además, obtener dígitos fraccionales adicionales en el resultado.

Antecedentes de la invención

La división digital de dos valores binarios ha sido implementada de varias maneras. Puede encontrar descripciones completas en la bibliografía (1) J.P. Deschamps, G.A. Bioul, and G. Sutter, "Synthesis of Arithmetic Circuits: FPGA, ASIC and Embedded Systems", Wiley Intescience. 2006. (2) S.F. Oberman and M.J. Flynn. "Division algorithms and implementations". IEEE Transactions on Computers. 46(8):833-854. August 1997. (3) B. Parhami. Computer Arithmetic: Algorithms and Hardware Design. Oxford University Press. 2000. (4) Milos. D. Ercegovac and Tomas Lang. Digital arithmetic San Francisco. California: Morgan Kaufmann. cop. 2004.

A modo de resumen, cuando los operandos están normalizados pueden aplicarse varios algoritmos de recurrencia como el SRT (Sweeney-Robertson-Tocher) en diferentes bases y representaciones del resto, ó bien con algoritmos de convergencia de dígitos como Newton-Rapshon, Goldschmidt (Expansión de MacLaurin), etc. Si los operandos no están normalizados se aplican normalmente los algoritmos binarios de división con restauración y sin restauración. Aunque pueden utilizarse bases superiores normalmente con el fin de hacer más rápidos estos algoritmos, las implementaciones de la división con lógica programable más comunes (Xilinx y Altera) utilizan algoritmos de división sin restauración en base 2.

Los costes de área C y de tiempo T de los circuitos divisores de base 2 para un dividendo X de n bits y un divisor Y de m bits que retorna un cociente Q de n bits y un resto R de m bits son distintos para la división con restauración y sin restauración, dependiendo de los siguientes parámetros:

- el coste de área de un restador de $m+1$ bits, $C_{restador}(m+1)$;
- el coste de área de un sumador-restador de $m+1$ bits, $C_{sumador-restador}(m+1)$;
- el coste de área de un sumador de m bits, $C_{sumador}(m)$;
- el coste de área de un multiplexador de 2 a 1, C_{mux};
- el retardo de un restador de $m+1$ bits, $T_{restador}(m+1)$;
- el retardo de un sumador-restador de $m+1$ bits, $T_{sumador-restador}(m+1)$;
- el retardo de un sumador de m bits, $T_{sumador}(m)$; y
- el retardo de un multiplexador de 2 a 1, T_{mux}.

Así para la división con restauración el coste de área y el de tiempo quedan:

- $C(n,m) = n\times(C_{restador}(m+1) + m\times C_{mux})$, y
- $T(n,m) = n\times(T_{restador}(m+1) + T_{mux})$.

Mientras que para la división sin restauración, quedan como

- $C(n,m) = n\times C_{sumador-restador}(m+1) + C_{mux} + C_{sumador}(m)$, y
- $T(n,m)=n\times T_{sumador-restador}(m+1) + T_{sumador}(m)$.

Dichos componentes constituyen una unidad capaz de realizar un conjunto de operaciones aritméticas sobre un conjunto de números expresados en una base $r=2^k$ que se denomina *célula_base_2^k*. El resultado del conjunto de operaciones de la *célula_base_2^k* comprende un cociente de k dígitos y un nuevo resto de $m+1$ dígitos.

Si se utilizan bases mayores con el fin de acelerar el proceso de división. El área y el tiempo esperados para una base $r = 2^k$ son:

$$C(n,m,k) = (n/k)\, C_{célula\_base2\^k}(m) + C_{mux} + C_{sumador}\ (m), \text{ y}$$

$$T(n,m,k) = (n/k) T_{célula\_base2\^k}(m) + T_{sumador}(m).$$

Los esfuerzos en acelerar el cómputo se basan en utilizar métodos de división y células con el menor retardo posible y con un coste en área menor e implementar mediante ellos un circuito divisor.

Resumen de la invención

El objeto de la invención es proporcionar un método más rápido y con menor costo de área para hallar los restos parciales y los cocientes durante las etapas de una división sin restauración en una célula de base 2^k según la reivindicación 1 independiente. Además, en un segundo aspecto se proporciona una célula de base 2^k según la reivindicación independiente 4. En un tercer aspecto se proporciona un circuito divisor según una primera arquitectura en la reivindicación 11 independiente, y en un cuarto aspecto inventivo se reivindica un circuito divisor según una segunda arquitectura en la reivindicación 14. Conjuntamente se proporciona en un quinto aspecto un conjunto de instrucciones que pueden ejecutarse en un medio programable que llevan a cabo las variantes de realización definidas por cualquiera de los aspectos inventivos segundo, tercero o cuarto. Además, en un sexto aspecto inventivo se proporciona un producto destinado a ejecutarse en medios programables que c comprende un conjunto de instrucciones según la reivindicación 19.

En una división sin restauración en base 2^k de un dividendo de n dígitos entre un divisor de m dígitos comprende p=0, .., [n/k] pasos en los que se halla en cada etapa:

- un resto parcial R(p) de m+1 dígitos;
- un conjunto de *k* dígitos del cociente *q(k+p), ..., q(p)*.

En un primer aspecto inventivo se describe un método para aplicar a una célula *célula_base_2^k* en que calcula en paralelo de todos los posibles restos en base y de las condiciones de selección del resto correcto.

En una primera etapa se le suministran a cada célula:

- un registro *rem(m...0)* correspondiente al resto parcial *R(p)* de los *p=0*, .., [n/k] de los restos parciales;
- 2^{k-1} registros *Reg_(2j-1)Y* de *m+j* dígitos, j= 0, .., 2^{k-1} que contienen los múltiplos impares del divisor Y (*Y, 3Y, ... (2^k-1)Y*);
- Un registro *sX(k*-1...0) de *k* dígitos del dividendo *X*. Estos dígitos son en cada ciclo *p* el registro *sX(k-1...0) ← X(n-k×p, ... n-k×(p+1))*. que se corresponde con los *k* bits más significativos desplazados en cada ciclo a la izquierda *p* veces.

En una segunda etapa se calculan los valores posibles de la suma y de la resta de los 2^{k-1} múltiplos impares del divisor con el valor del resto parcial.

En una tercera etapa se determina a su vez, de entre estos posibles valores de los restos, el valor del resto parcial siguiente *R(p+1)* a partir de los signos de estos valores posibles del resto como *c(k-1) → R(p+1)*. Utilizando la siguiente secuencia:

$$c(j) = a(j)\, R(p) + b(j) \times \text{sign}[R(p)],\quad j= 0,\ldots,k-1,$$

dónde

$a(0) = 2$;

$b(0) = 1$;

$a(j+1) = 2 \times a(j)$

$b(j+1) = 2 \times b(j) + 1$ si $c(j) < 0$

ó

$b(j+1) = 2 \times b(j) - 1$ si $c(j) > 0$

En una cuarta etapa, los signos de los valores posibles del resto son utilizados para calcular los *k* dígitos del cociente *q(k+p), ..., q(p)* asignándose el valor de cada dígito del cociente en función del signo de *c(j)*.

Preferentemente, $q(k\text{-}1\text{-}j) = 1\text{-}$ sign[c(*j*)], $j = 0, ..., k\text{-}1$.

Se dispone el resto parcial en el registro *new_rem* y los *k* dígitos del cociente en el registro new_q = *q*(n)...*q*(0).

En una variante de realización se calculan 2^k restos, seleccionando mediante 2^{k-1} condiciones.

Preferentemente, se utiliza la representación de complemento a dos para representar los números en base 2^k.

En una variante de realización el valor de los restos parciales *a(j)* se obtiene combinando los *j* bits menos significativos del registro de memoria *rem*(j-1...0) y los j bits más significativos del registro de memoria *sX*, es decir *sX*(*k*-1...*k*-1-*j*).

En otra variante de realización se utiliza un sumador condicional en donde el bit de signo de b(j+1) es utilizado para discriminar entre un resto restaurado y sin restaurar. Preferentemente se utiliza el bit de signo de b(j-1) para discriminar entre la suma (signo negativo) o resta.

En un segundo aspecto inventivo se reivindica una célula de base 2^k. Dicha célula se utiliza en un procedimiento de división sin restauración y comprende: medios adaptados para introducir registros de memoria y para proporcionar registros de memoria, medios adaptados para realizar operaciones aritméticas, medios adaptados para realizar operaciones lógicas y medios adaptados para interconectar al menos estos componentes.

(1) Los medios adaptados para introducir los registros de memoria comprenden medios para introducir:

- un registro de memoria *rem(m...0)* de *m*+1 bits que almacena el resto parcial *R(p)* correspondiente al paso *p* de la división sin restauración;
- un conjunto de registros que contienen los 2^k-1 múltiplos impares del divisor *Reg_Y(m-1...0), Reg_3Y(m-1...0), ..., Reg_(2k-1)Y(m-1, ..., 0)*;
- un registro *sX* de k bits que contenga los dígitos del divisor correspondientes a cada etapa de la división: *X(n-1-p×k...n-(p+1)×k), p* = 0, ..., [n/k].

Los medios adaptados para proporcionar registros de memoria que comprenden medios para proporcionar un registro de k bits del cociente *new_q*(k-1...0).

Preferentemente, se proporciona un registro de *m*+1 bits correspondiente al nuevo resto parcial, *new_rem*(m...0).

(2) Los medios adaptados para realizar operaciones aritméticas comprenden elementos combinacionales aritméticos que están adaptados para sumar registros de memoria. Preferentemente, se utilizan sumadores y Testadores. Una primera entrada de los sumadores está dispuesta para configurar el registro correspondiente a a(j). Preferentemente, este registro *a(j)* se forma concatenando el registro *rem* y los j bits más significativos del registro *sX*.

Otra segunda entrada del sumador está conectada a uno de los registros de memoria *rem(m...0), Reg_Y(m-1...0), Reg_3Y(m-1...0), ..., Reg_(2k-1)Y(m-1, ..., 0)*.

Preferentemente, se utiliza un sumador condicional. La entrada condicional está alimentada por el signo del registro correspondiente al resto anterior para llevar a cabo las operaciones $R(p) \pm (2j+1)Y$, $j = 1, ..., 2^{k-1}$.

Preferentemente, el signo de estos registros se obtiene utilizando el primer bit de dicho registro.

Preferentemente, el signo de estos registros se obtiene utilizando el primer bit de dicho registro utilizando la representación de complemento a dos.

(3) Los medios adaptados para realizar operaciones lógicas comprenden elementos lógicos combinaciones que seleccionan registros de memoria.

Preferentemente comprenden multiplexores. También comprende medios adaptados para obtener el signo de un número representado en dichos registros en base 2^k.

Preferentemente, el signo de estos registros se obtiene utilizando el primer bit de dichos registros.

Preferentemente, el signo de estos registros se obtiene utilizando el primer bit de dicho registro utilizando la representación de complemento a dos.

Preferentemente, estos medios para seleccionar registros de memoria están implementados como multiplexores.

Preferentemente, estos medios para seleccionar registros de memoria están implementados como sumadores condicionales.

(4) Medios adaptados para transmitir datos entre registros de memoria que comprenden.

Un conjunto de medios que comprende la unión de cada uno de los sumadores que obtiene c(j), en donde:

- un medio que une el registro con los j bits más significativos de *sX* con la primera entrada del mismo sumador
- un medio que une el registro *rem* con la primera entrada del mismo sumador.
- Un medio que une un registro *reg_(2j+1)Y* con la segunda entrada del sumador.

Un medio que une un registro de salida del sumador con una entrada de los medios adaptados para realizar operaciones lógicas.

Un medio para conectar un bit que tiene información sobre el signo de un registro de salida con una entrada de los medios para realizar operaciones lógicas.

Un medio para conectar una salida de los medios para realizar operaciones lógicas que proporciona el registro de salida *new_rem*.

Un medio para conectar una salida de los medios para realizar operaciones lógicas que proporciona el registro de salida *new_q*.

La célula adaptada para implementar el método anterior presenta una rapidez superior ya que el valor área-tiempo de la célula *célula_base_2^k* es:

$$C_{célulabase2^k}(k, m) = (2k-1)\times C_{sumador}(m+k) + (2k-1) \times C_{restador}(m+k) +$$

$$(2k-2) \times C_{signo\ de\ rest}(m+k-1) + m\times C_{mux\text{-}k\text{-}hasta\text{-}1}$$

$$T_{célulabase2^k}(k, m) = \max(T_{sumador}(m+k), T_{restador}(m+k-1))+ T_{mux\text{-}k\text{-}hasta\text{-}1}$$

Si se considera que las sumas, restas y el cálculo del signo de una resta tienen aproximadamente el mismo coste y tiempo (es decir, la generación y propagación del acarreo), descartando el tiempo necesario para ajustar el resto ($T_{sumador}$ (m)) y suponiendo que k << m. Para valores habituales, $T_{mux-(k-1)-hasta-1} < T_{sumador-rest(m+k)}$, por lo que:

$$(\text{aumento de rapidez}) > n\times T_{sumador\text{-}rest} /(n/k\times\ 2\times\ T_{sumador\text{-}rest}) = k/2$$

En un tercer aspecto inventivo, es objeto de la invención, un circuito divisor de base 2^k utiliza una arquitectura secuencial, a partir de un dividendo *X* y un divisor *Y* como entradas de división, devolviendo un cociente *Q* y un resto *R*. En el circuito, se comprende una célula de base 2k en la que se introducen los múltiplos impares del divisor, el resto parcial y k bits del dividendo y genera como resultado k bits de cociente y un nuevo resto para la siguiente etapa.

En un ejemplo de realización, la arquitectura secuencial comprende:

- una célula de base 2^k.
- un medio de control que proporciona y recibe las señales de control para la - implementación secuencial de la división con restauración.
- Un medio de cálculo aritmético para obtener los múltiplos impares del divisor.
- Un medio para capturar k dígitos del dividendo.
- Medios para concatenar registros de memoria.
- Medios para ajustar el resto.
- Medios de memoria para almacenar
  - un registro X que corresponde al dividendo X.
  - un registro sX que corresponde a k bits del dividendo X.
  - un registro Y que corresponde al divisor Y.
  - un conjunto de 2^{k-1} registros que corresponden a los múltiplos impares del divisor Y: 3Y, 5Y, ... (2^k-1)Y.
  - un registro rem que corresponde al resto parcial *R(p)*.
  - un registro new_rem que corresponde al nuevo resto parcial R(p+1).
  - un registro R que corresponde al resto final.
  - un registro new_q que corresponde a k dígitos del cociente.
  - un registro Q que corresponde al cociente.

Preferentemente, el medio de control proporciona las señales para controlar de una forma secuencial el proceso de división:

- La señal *ini* durante cada ciclo del proceso de división.
- la señal *capt* de captura del registro del resto,
- La señal *shift* para seleccionar los bits del registro dividendo y del registro del cociente.
- La señal *done* después de [n/k] +1 ciclos de división.

Preferentemente, el medio de cálculo aritmético para obtener los múltiplos impares del divisor comprende un conjunto de sumadores, multiplicadores y restadores.

Preferentemente, el medio para capturar los k dígitos del dividendo se compone de un medio para desplazar los k bits de un registro y un medio para capturar los k bits más significativos de de este registro desplazado.

Preferentemente, la señal ini generada por la máquina de estados de control está conectada

- al medio que permite desplazar k posiciones a la izquierda el contenido del registro X y captura los k bits más significativos del dividendo.
- al medio de cálculo aritmético para obtener los múltiplos impares del divisor Y.

Preferentemente, el medio de cálculo aritmético comprende elementos sumadores, restadores y multiplicadores.

Preferentemente, se utiliza como medios para ajustar el resto una “célula de ajuste” que comprende un sumador condicional, con una entrada conectada al registro del resto parcial rem, otra entrada del sumador conectada al registro Reg_Y, una entrada condicional conectada a un indicador del signo del registro rem y una salida que proporciona el registro R.

En un cuarto aspecto inventivo, es objeto de la invención, un circuito divisor de base 2k utiliza una arquitectura paralela, a partir de un dividendo X y un divisor Y como entradas de división, devolviendo un cociente Q y un resto R. El circuito comprende un conjunto de células de base 2^k en la que se introducen los múltiplos impares del divisor, el resto parcial y k bits del dividendo.

En una variante de realización, el circuito divisor con arquitectura paralela comprende:

- un conjunto [n/k]+1 células de base 2^k.
- Un medio de cálculo aritmético para obtener los múltiplos impares del divisor.
- Medios para sincronizar el funcionamiento de las células 2^k.
- Un medio para capturar k dígitos del dividendo.
- Medios para concatenar registros de memoria.
- Medios para ajustar el resto.
- Medios de memoria para almacenar
  - un registro X que corresponde al dividendo X.
  - un registro Y que corresponde al divisor Y.
  - un conjunto de 2k-1 registros que corresponden a los múltiplos impares del divisor Y: 3Y, 5Y, ... (2k-1)Y.
  - un registro R que corresponde al resto final.
  - un registro q que corresponde a k dígitos del cociente.
  - un registro Q que corresponde al cociente.

Preferentemente, en el circuito divisor en paralelo se conectan cada una de las células de base 2^k de forma escalonada.

Así, una primera célula de base 2^k adquiere un registro sX de k bits que contenga los dígitos del divisor correspondientes a los k bits más significativos del divisor: X(n-1...n-k), un conjunto de entradas conectadas a los medios aritméticos para obtener el conjunto de registros que contienen los 2^k-1 múltiplos impares del divisor Reg_Y(m-1...0), Reg_3Y(m-1...0), ..., Reg_(2^k-1)Y(m-1, ..., 0), y una entrada al registro de memoria rem(m...0) de m+1 bits que almacena correspondiente a 0.

Una segunda célula de base 2^k adquiere un registro sX de k bits que contenga los dígitos del divisor correspondientes a los k bits más significativos del divisor: X(n-k-1...n-2k), un conjunto de entradas conectadas a los medios aritméticos para obtener el conjunto de registros que contienen los 2^k-1 múltiplos impares del divisor Reg_Y(m-1...0), Reg_3Y(m-1...0), ..., Reg_(2^k-1)Y(m-1, ..., 0), y una entrada al registro de memoria del resto parcial de la segunda rem(m...0) de m+1 bits está conectada al registro de salida del resto de la primera célula de base 2^k.

De tal manera que la entrada del resto parcial de cada una de las entradas del resto parcial de una célula está conectada a la salida de la célula anterior y se adquieren los correspondientes k bits del dividendo X: X(n-1-p×k...n-(p+1)×k), p = 0, ..., [n/k].

La célula de ajuste, para efectuar la restauración, está conectada de forma que adquiere el registro correspondiente al último resto parcial y un registro que contiene al divisor, y que dispone a su salida un registro correspondiente al resto R en función del signo del último resto parcial.

Además, el registro del cociente se forma concatenando ordenadamente los registros de cada uno de los cocientes. Así, los k bits más significativos del registro de salida del cociente Q están dados por el registro de salida de la primera célula de base 2^k, los siguientes k bits más significativos por la siguiente de cada una de las células, hasta que los k bits menos significativos vienen dados por el registro de la última de estas células.

En un quinto aspecto inventivo, es posible implementar un conjunto de instrucciones que pueden ejecutarse en un medio programable. Dicho conjunto de instrucciones comprende un primer subconjunto que permite ejecutar cualquiera de las variantes de realización del método definido por el primer aspecto inventivo en un primer medio de computación. Preferentemente, dicho juego de instrucciones contiene un segundo subconjunto de instrucciones que hacen la configuración de un medio de lógica programable adopte un configuración según cualquiera de las variantes de realización definidas por los aspectos inventivos segundo, tercero o cuarto.

Breve descripción de los dibujos

Para una mejor compresión de la invención, sus objetos y ventajas se adjuntan a la memoria las siguientes figuras en las que se representa:

Fig. 1. Célula de base 4 que usa la arquitectura arch1.

Fig. 2A. Registros en una célula de base 8 que usa la arquitectura arch1.

Fig. 2B. Diagrama simplificado de una célula de base 8 que usa la arquitectura arch1.

Fig. 3. Célula de base 4 que usa la arquitectura arch2.

Fig. 4A. Registros en una célula de base 16 que usa la arquitectura arch2.

Fig. 4B. Diagrama simplificado de una célula de base 16 que usa la arquitectura arch2.

Fig. 5. Diagrama de bloques de una implementación secuencial de un circuito divisor según la invención.

Fig. 6. Diagrama de bloques de una implementación combinacional o en paralelo de un circuito divisor.

Descripción de las realizaciones preferentes de la invención

En una primera variante de realización del método para aplicar a una célula *célula_base_2^k* se utilizan registros que utilizan la representación de complemento a dos.

Inicialmente, se tiene el registro *rem(m...0)* correspondiente al resto parcial *R(p)* = 0,8 registros *Reg_Y, Reg_3Y, Reg_5Y*, ..., que ese corresponden con Y, 3Y, 5Y ... y un registro *sX*(3...0) de los 4 dígitos más significativos del dividendo *X*.

En una segunda etapa se calculan los posibles restos correspondientes a $2\times R(p) + Y$, 2×R(p) - Y, 4×R(p) + Y, 4×R(p) - Y, 4×R(p) + 3Y, 4×R(p) - 3Y, 8×R(p) + Y, 8×R(p) + 3Y, 8×R(p) + Y, 8×R(p) + 5Y, 8×R(p) + 7Y, 8×R(p) - Y, 8×R(p) - 3Y, 8×R(p) - 5Y, 8×R(p) - 7Y.

En una tercera etapa, tanto para seleccionar el resto parcial como para llevar a cabo la selección, se utiliza el criterio de selección que llevaría a escoger el resto correcto según el valor de a(j), b(j) y c(j), j = 0...k: es decir, se parte de los valores iniciales de los prefactores a y b, a(0) = 2 y b(0) = 1. A continuación se hallan los diferentes posibles valores del resto calculando los restos parciales mediante las sumas y/o restas la secuencia de signos que determina el resto parcial R(p+1) del conjunto de restos c(j). Elegir estos restos supone lleva a seguir las secuencia de signos de c(j) que determina b(j) y permite seleccionar R(p+1)= c(k-1). El resultado de este sumador corresponde a un registro.

En una cuarta etapa, esta secuencia de restos parciales da lugar a una condición lógica que determina los k nuevos dígitos del cociente q(k-1)...q(0).

Preferentemente se utiliza una representación de complemento a dos para representar a los registros de los posibles restos parciales.

Preferentemente se utiliza el bit más significativo del resto parcial introducido para discriminar entre las operaciones de suma y resta.

Preferentemente el valor de los restos parciales se obtiene combinando los j bits menos significativos del registro de memoria rem(j-1...0) y los j bits más significativos del registro de memoria sX.

En la tercera etapa, el conjunto de condiciones lógicas relacionadas con el signo de estos restos de la tercera etapa se utiliza para formar un conjunto de condiciones que se combinan para dar lugar a una condición de selección, Esta condición de selección se expresa preferentemente como una secuencia de bits.

Preferentemente, esta condición de selección se utiliza como entrada de un multiplexador que permite seleccionar utilizando esta condición de selección el resto parcial en un cuarta etapa.

Se dispone el resto parcial en el registro *new_rem*(...) y los k dígitos del cociente en el registro *q*(n)....*q*(0).

En otra variante de realización se utiliza un sumador condicional en donde el bit de signo de b(j+1) es utilizado para discriminar la suma (signo negativo) o resta.

Para la célula de base 2^k se presentan dos variantes de realización preferentes que utilizan dos arquitecturas diferentes, denominadas *arch1* y *arch2*. La primera arquitectura (*arch1*) es general para cualquier tecnología de circuitos, mientras que la *arch2* explota las mismas características que las tecnologías programables.

La figura 1 describe una célula en base 4 que usa la *arch1*; en cambio, la figura 2A, 2B se muestra una célula de base 8 que usa la misma arch1. Por otro lado, la figura 3 y la figura 4A, 4B se explican la arquitectura *arch2*, la primera muestra una célula de base 4, mientras que la segunda una célula de base 16.

En una variante de realización la célula básica sigue una arquitectura arch1, la figura 1 y la figura 2B muestran el primer tipo de célula básica desarrollada para la base-4 4 ($2^{k=2}$) y la base-8 4 ($2^{k=3}$), respectivamente.

La figura 1 es un diagrama de bloques esquemático para una célula de base 4 de *arch1* que muestra la estructura interna adecuada para su utilización en según la presente realización(Base 4, $k = 2$). En base 4 se generan los cuatro (2^k) nuevos restos 34, 35, 36, 37 posibles. Los elementos 35 y 36 calculan respectivamente la suma y la resta del circuito divisor Y, siendo solo necesario el bit más significado el cual proporciona la información del signo de la operación. Preferentemente, los operandos usan complemento a dos(1 negativo y 0 positivo) y los multiplexores 38, 39 y 40 seleccionan el resto siguiente (*new_rem*) siguiendo la semántica del pseudocódigo 1, mientras que el multiplexor 41 implementa el pseudocódigo 2.

```
if rem < 0 then
if rem*2 + Y < 0 then
new_rem := rem*4 + 3.Y;
else
new_rem := rem*4 + Y;
else -- rem >= 0
if rem*2 - Y < 0 then
new_rem := rem*4 - Y;
else
new_rem := rem*4 - 3.Y;
end if;
end if;
```

pseudocódigo 1. Calculo del resto en base 4 y arquitectura arch1.

```
q(0) := new_rem(m)
if rem < 0 then
if rem*2 + Y < 0 then
new_q(1) := 1;
else
new_q(1) := 0;
else
if rem*2 - Y < 0 then
new_q(1) := 1;
else
new_q(1) := 0;
end if;
end if;
```

Pseudocódigo2. Generación de los bits del cociente en base 4 para la arquitectura arch1.

Haciendo referencia a la figura 2B, ésta es un diagrama de bloques esquemático para una célula de base 8 de *arch1* que muestra la estructura interna adecuada para su utilización en el circuito divisor según la presente realización. (Base 8, $k = 3$). En base 8 se generan los ocho (2^k) nuevos restos posibles (agrupados en la referencia 53, designados como r8pY, r8mY, r8p3y, r8m3y, r8p5y, r8m5y, r8p7y, r8m7y en la Fig. 2A). Con el fin de expresar los ocho cálculos se usa sintaxis pseudo-VHDL donde el símbolo "&" significa concatenación. Se calcula el signo de seis sumas o restas (agrupados en la referencia 52, designados como r2_p_Y, r2_m_Y r2_p_3Y r2_m_3Y r2_p_5Y, r2_m_5Y). Sólo es necesaria la propagación del acarreo de la operación ya que los operandos usan complemento a dos. Los signos resultantes se usan en los multiplexores para determinar el nuevo resto y los bits de cociente resultantes. Los multiplexores 54, 55, 56, 57, 60, 61 y 62 seleccionan el siguiente resto (*new_rem*) siguiendo la semántica del pseudocódigo 3. Los multiplexores 58, 59 y 63 implementan el pseudocódigo 4, es decir, generan los 3 bits de cociente.

```
if rem < 0 then
if rem*2 + Y < 0 then
if rem*4 + 3.Y < 0 then
new_rem := rem*8 + 7.Y;
else new_rem := rem*8 + 5.Y;
end if;
else --rem*2 + Y >= 0
if rem*4 + Y < 0 then
new_rem := rem*8 + 3.Y;
else new_rem := rem*8 + Y;
end if;
```

```
new_q(0) := new_rem(m)
if rem < 0 then
if rem*2 + Y < 0 then
if rem*4 + 3.Y < 0 then new_q(2:1) := 11;
else new_q(1:0) := 10; end if;
else --rem*2 + Y >= 0
if rem*4 + Y < 0 then new_q(2:1) := 01;
else new_q(1:0) := 00; end if;
end if;
else -- rem >= 0
if rem*2 - Y < 0 then
```

```
end if;
else -- rem >= 0
if rem*2 - Y < 0 then
if rem*4 - 3.Y < 0 then
new_rem := rem*8 – Y;
else new_rem := rem*8 - 3.Y;
end if;
else --rem*2 - Y >= 0
if rem*4 + Y < 0 then
new_rem := rem*8 + 5.Y;
else new_rem := rem*8 + 7.Y;
end if;
end if;
end if;
```

pseudocódigo 3. Calculo del resto en base 8 y arquitectura arch1

```
if rem*4 - 3.Y < 0 then new_q(2:1) := 11;
else new_q(1:0) := 10; end if;
else --rem*2 - Y >= 0
if rem*4 - Y < 0 then new_q(2:1) := 01;
else new_q(1:0) := 00; end if;
end if;
end if;
```

pseudocódigo 4. Generación de los bits del cociente en base 8 para la arquitectura arch1.

En otra variante de realización célula de base 2^k que usa la arch2, la figura 3 muestra la arquitectura de base 4 ($2^{k=2}$), mientras que la figura 6 muestra la arquitectura de base 16 ($2^{k=4}$) de una célula.

La figura 3 es un diagrama de bloques esquemático para una célula de base 4 de *arch2* que muestra la estructura interna adecuada para su utilización en el circuito divisor según la presente invención. La *arch2* usa las características de algunas FPGA en las que los sumadores condicionales tienen un coste de área y de tiempo similar al de un sumador simple. La célula 72 calcula 2×rem+Y ó - (2×rem-Y)-1 dependiendo del signo del resto (bit de rem(m)). Obsérvese que la segunda expresión -(2×rem-Y)-1 es equivalente a: -(2×rem-Y)-1 = Y + -2×rem + 1 = Y + (not(2×rem) + 1) -1 = Y + not(2×rem). El resultado es el bit del signo (*r2_pm_y*). La célula 73 calcula 4×rem más menos tres veces el circuito divisor dependiendo de rem(m). La célula 74 calcula 4×rem + Y ó 4×rem - Y dependiendo del signo del resto (rem(m)). El multiplexor 75 selecciona el siguiente resto basándose en la señal *r2_pm*_y. El bit de cociente q(0) es el mismo que el signo del nuevo resto (rem(m) = q(0)). El bit de cociente q(1) depende de rem(m) y el bit *r2_pm_y* se calcula usando el multiplexor 76.

Haciendo referencia a la figura 4B, ésta es un diagrama de bloques esquemático para una célula de base 16 ($2^{k=4}$) de tipo *arch2*. La cantidad total de sumas 83 condicionales necesarias es 8 (2^{k-1}), calculando el resto multiplicado por 16 y sumando o restando los múltiplos del circuito divisor que se muestran en la Fig. 4A (Y, 3×Y, 5×Y, 7×Y, 9×Y, 11×Y, 13×Y, 15×Y). La decisión de sumar o restar se toma mediante el bit más significativo del resto (rem(m)). Sólo son necesarios los *m*+1 bits más significativos. Además, se calcula el signo de $7(2^{k-1}-1)$ operaciones condicionales (número de referencia 82) dependiendo del signo del resto (rem(m)). Los signos resultantes se usan en los multiplexores para determinar el nuevo resto y los bits de cociente resultantes. Obsérvese que la condición *r2_pm_y* < 0 es simplemente la propagación de acarreo de la operación condicional o, de manera equivalente, el bit más significativo de la operación. Lo mismo sucede con las señales *r4_pmy, r4_pm_3y, r8_pmy, r8_pm3y, r8_pm_5y, r8_pm_7y*. Los multiplexores 84, 85,86, 87, 90, 91 y 92 seleccionan el siguiente resto (*new_rem*) siguiendo la semántica del pseudocódigo 5. Los multiplexores 88, 89 y 93,94 implementan el pseudocódigo 6, es decir, generan los 4 bits de cociente.

```
if rem*2 +/- Y < 0 then
if rem*4 +/- 3.Y < 0 then
if rem*8 +/- 7.Y < 0 then
new_rem := rem*16 +/- 15.Y;
else new_rem := rem*16 +/- 13.Y;
end if;
else --rem*4 +/- 3.Y >= 0
if rem*8 + 5.Y < 0 then
new_rem := rem*16 +/- 11.Y;
else new_rem := rem*16 +/- 9.Y;
```

```
new_q(0) := new_rem(m)

if rem*2 +/- Y < 0 then
if rem*4 +/- 3.Y < 0 then
if rem*8 +/- 7.Y < 0 then int_q(3:1) := “111”;
else := “110”; end if;
else --rem*4 +/- 3.Y >= 0
if rem*8 + 5.Y < 0 then int_q(3:1) := “101”;
else := “100”; end if;
then
```

```
end if;
end if;
else --rem*2 + Y >= 0
if rem*4 +/- Y < 0 then
if rem*8 +/- 3.Y < 0 then
new_rem := rem*16 +/- 7.Y;
else new_rem := rem*16 +/- 5.Y;
end if;
else --rem*4 +/- Y >= 0
if rem*8 + Y < 0 then
new_rem := rem*16 +/- 3.Y;
else new_rem := rem*16 +/- Y;
end if;
end if;
end if;
```

pseudocódigo 3. Calculo del resto en base 16 y arquitectura arch2

```
if rem*4 +/- Y < 0 then
if rem*8 +/- 3.Y < 0 then int_q(3:1) := "011";
else := "010"; end if;
else --rem*4 +/- Y >= 0
if rem*8 + Y < 0 then int_q(3:1) := "001";
else := "000"; end if;
end if;
end if; --int_q(3) := r2_pm_y;

if rem < 0 then
new_q(3:1) := int_q(3:1);
else
new_q(3:1) := not (int_q(3:1));
end if;
```

pseudocódigo 4. Generación de los bits del cociente en base 16 para la arquitectura arch2.

El circuito divisor puede implementarse en cualquier aplicación en la que se necesite la funcionalidad de división entera o de coma fija.

La figura 5 es un diagrama de bloques que muestra la estructura de una implementación secuencial para un circuito divisor general de base 2^k según la presente invención. La figura 6 es un diagrama de bloques de la estructura general de una implementación combinacional y/o en paralelo de la invención. Los diagramas ilustran la naturaleza funcional del circuito divisor, el cual podría implementarse usando una amplia gama de tecnologías digitales de hardware tal como lógica programable (FPGA), aunque es posible implementar el circuito divisor mostrado a través de las figuras usando una tecnología adecuada para la aplicación específica.

Haciendo referencia a la figura 5, el circuito divisor de ejemplo es un divisor que divide un dividendo natural X de *n* bits por un divisor natural Y de *m* bits devolviendo un cociente Q de *n* bits y un resto R de *m* bits. La base 2^k de la célula de cálculo determina la cantidad de bits generados en cada ciclo (*k* bits) y la cantidad de registros 2 de dividendo necesarios 2^{k-1} registros que oscilan entre los *m* bits y los *m*+*k* bits).

El circuito divisor incluye registros 1 y 5, X y Q, de desplazamiento de *n* bits. El registro 1, X, de desplazamiento captura el dividendo cuando se fija la señal *ini* y desplaza *k* posiciones a la izquierda en cada ciclo de reloj. Los k bits más significativos (número de referencia 9) se usan por la *célula_de_base_2^k* (número de referencia 4) (Esta célula puede implementarse usando la arch1 o la arch2). El registro 5, Q, de desplazamiento captura en las posiciones menos significativas *k* bits (número de referencia 10) de la *célula_de_base_2^k* y desplaza a la derecha *k* bits por ciclo. El resto parcial de cada ciclo se almacena en un registro 6 de *m* + 1 bits de complemento a dos, fijándose en cero al principio del cálculo (número de referencia 11). Un sumador 7 condicional (célula de ajuste) se usa con el fin de ajustar el resto en caso de que sea negativo. Si no se necesita el resto, esta parte del circuito puede eliminarse. Para un circuito divisor de base 2^k deberían generarse los 2^{k-1} múltiplos impares del divisor comprendidos entre 1 y 2^k-1. Para la base 4: Y y 3×Y. Para la base 8: Y, 3×Y, 5×Y y 7×Y. Para la base 16: Y, 3×Y, 5×Y, 7×Y, 9×Y, 11×Y, 13×Y y 15×Y. Estos múltiplos pueden generarse usando sumas o restas 3 y se almacenarán en registros 2.

La máquina 8 de estados de control genera señales de control y cuenta los *n/k* ciclos necesarios para obtener el cociente Q de resultado. El resto R se obtiene un ciclo después. Con el fin de obtener más bits de resultados, es decir, un resultado *n*+*p*, la máquina de estados de control añade simplemente *p/k* ciclos y el registro de desplazamiento Q correspondiente se amplía para almacenar el resultado.

Haciendo referencia a la figura 6, el circuito divisor combinacional/en paralelo usa el mismo ancho de datos para los operandos X, Y, Q y R descritos en la figura 1. La *célula_de_base_2^k* (número de referencia 22) se usa *n/k* veces y puede implementarse usando la arch1 o la arch2. Se usa una célula 23 de ajuste (sumador adicional) para ajustar el resto en caso de que sea negativo. Si no se necesita el resto, esta parte del circuito puede eliminarse.

Para un circuito divisor de base 2^k deberían generarse los 2^{k-1} múltiplos impares del divisor comprendidos entre 1 y 2^k-1. La generación de los múltiplos no se muestra ya que es trivial y similar a la descrita en la figura 1. La interconexión entre las etapas se realiza a través de restos parciales y deberían añadirse *células_de_base_2^k* adicionales con el fin de obtener más bits de resultado. Para obtener un resultado de *n*+*p* bits, deberían añadirse *p/k* células adicionales.

Para fines de rendimiento, una implementación en paralelo añade registros de sincronización, de sesgo (*skewing*) y de corrección del sesgo (*de-skewing*). La posición de los registros se muestra como la intersección de las líneas 24 equitemporales (líneas discontinuas) y las líneas que representan buses de interconexión. Los expertos en la materia sabrán cómo implementar y reducir la latencia y la profundidad lógica dependiendo de la base seleccionada y de la cantidad de etapas en paralelo.

Aunque la invención se ha mostrado y descrito con referencia a determinadas realizaciones preferidas de la misma, los expertos en la materia apreciarán que pueden realizarse diversos cambios en la forma y en los detalles de la misma sin apartarse del espíritu y del alcance de la invención tal como definen las reivindicaciones adjuntas.

REIVINDICACIONES

1. Método para hallar los restos durante una división sin restauración en una célula de base 2^k entre un dividendo X de n bits y un divisor Y de m bits que en la etapa *p (p=0...[n/k])* de la división sin restauración comprende las siguientes etapas:

a. se introducen 2^{k-1} registros correspondientes a múltiplos impares del divisor Y, el registro sX de k dígitos correspondientes al dividendo X, y el registro de resto parcial R(p),

b. se hallan los diferentes posibles valores del resto calculando los posibles restos parciales mediante las sumas y/o restas,

c. se selecciona el resto parcial R(p+1) que coincide con la secuencia que determina el conjunto de restos c(j), j=0, ..., k-1 en función de los signos de c(j) y permite determinar el siguiente resto parcial R(p+1)= c(k-1), utilizando la siguiente secuencia:

$$c(j) = a(j)\,R(p) + b(j) \times \text{sign}[R(p)],\quad j= 0,\ldots,k\text{-}1,$$

siendo $a(0) = 2$, $b(0) = 1$, $a(j+1) = 2\times a(j)$, y $b(j+1) = 2\times b(j) +1$ si

$c(j) < 0$ ó $b(j+1) = 2\times b(j) -1$ si $c(j) > 0$; y

d. se utilizan los signos de la secuencia de restos parciales c(0), ..., c(k-1) para determinar los k nuevos dígitos del cociente q(k-1)...q(0),

de tal manera que se proporciona un registro new_q(k-1...0) que contiene k dígitos del cociente Q y un registro *new_rem* del resto parcial.

2. Método según la reivindicación 1 **caracterizado** porque la representación de los números es binaria de complemento a dos.

3. Método según la reivindicación 2 **caracterizado** porque los bits del signo de cada uno de los números del conjunto de restos c(j) se concatenan para obtener los bits del cociente.

4. Célula para hallar k dígitos de un cociente Q y un valor de un resto parcial R(p+1) entre un dividendo X de n bits y un divisor Y de m bits que comprende:

a. medios adaptados para introducir registros de memoria y para proporcionar registros de memoria,

b. medios adaptados para realizar operaciones aritméticas,

c. medios adaptados para realizar operaciones lógicas, y

d. medios adaptados para interconectar al menos estos componentes,

configurados para implementar el método según la reivindicación 1 a 3.

5. Célula según la reivindicación anterior **caracterizada** porque los medios adaptados para realizar las operaciones lógicas son multiplexores.

6. Célula según la reivindicación anterior **caracterizada** porque comprende $1+2^{k-1}$ multiplexores para implementar la selección del resto parcial en función del bit de los posibles restos parciales y un multiplexor para seleccionar el registro del cociente.

7. Célula según cualquiera de las reivindicaciones 4 a 6 **caracterizada** porque comprende sumadores condicionales que determina la operación suma o resta entre dos posibles valores de restos correspondientes a c(j), j=1, ..., k-1 en función del bit más significativo del registro del resto parcial c(j-1).

8. Célula según la reivindicación **caracterizada** porque los dígitos q(k-1-j), j=0, ..., k-1 del registro del cociente q (k-1-j) corresponden a la concatenación de los bits de signo de los restos parciales según el bit más significativo del registro del resto c(j) con una representación binaria de complemento a dos.

9. Célula según la reivindicación 2 **caracterizada** porque está implementada utilizando una arquitectura en donde los multiplexores seleccionan los registros de los restos c(j), j= 0, ..., k-1.

10. Célula según la reivindicación 2 **caracterizada** porque está implementada utilizando una arquitectura en donde los sumadores condicionales seleccionan los registros de los restos c(j), j= 0, ..., k-1.

11. Circuito para implementar un método de división sin restauración que comprende una célula de base 2^k en la que se introducen los múltiplos impares del divisor, el resto parcial y k bits del dividendo y genera como resultado k bits de cociente y un nuevo resto para la siguiente etapa en donde la arquitectura es secuencial.

12. Circuito según la reivindicación anterior **caracterizado** porque comprende:

- una célula de base 2^k según cualquiera de las reivindicaciones 3 a 7
- un medio de control que proporciona y recibe las señales de control para la implementación secuencial de la división con restauración.
- Un medio de cálculo aritmético para obtener los múltiplos impares del divisor.
- Un medio para capturar k dígitos del dividendo.
- Medios para concatenar registros de memoria.
- Medios para ajustar el resto.
- Medios de memoria para almacenar
  - un registro X que corresponde al dividendo X,
  - un registro sX que corresponde a k bits del dividendo X,
  - un registro Y que corresponde al divisor Y,
  - un conjunto de 2^{k-1} registros que corresponden a los múltiplos impares del divisor Y: 3Y, 5Y, ... (2^k-1)Y,
  - un registro rem que corresponde al resto parcial *R(p)*,
  - un registro new_rem que corresponde al nuevo resto parcial R(p+1),
  - un registro R que corresponde al resto final,
  - un registro new_q que corresponde a k dígitos del cociente,
  - un registro Q que corresponde al cociente.

13. Circuito para implementar un implementar un método de división sin restauración que comprende una célula de base 2^k en la que se introducen los múltiplos impares del divisor, el resto parcial y k bits del dividendo y genera como resultado k bits de cociente y un nuevo resto para la siguiente etapa en donde la arquitectura es paralela.

14. Circuito según la reivindicación anterior **caracterizado** porque comprende:

- un conjunto [n/k]+1 células de base 2^k
- Un medio de cálculo aritmético para obtener los múltiplos impares del divisor.
- Medios para sincronizar el funcionamiento de las células 2^k.
- Un medio para capturar k dígitos del dividendo.
- Medios para concatenar registros de memoria.
- Medios para ajustar el resto.
- Medios de memoria para almacenar
  - un registro X que corresponde al dividendo X,

- un registro Y que corresponde al divisor Y,
- un conjunto de 2^{k-1} registros que corresponden a los múltiplos impares del divisor Y: 3Y, 5Y, ... (2^k-1)Y,
- un registro R que corresponde al resto final,
- un registro q que corresponde a k dígitos del cociente,
- un registro Q que corresponde al cociente.

15. Conjunto de instrucciones que, cuando son interpretadas por un medio programable, hacen que este medio realice un método según cualquiera de las reivindicaciones 1 a 3.

16. Conjunto de instrucciones cuya interpretación por un medio programable hacen que un medio de lógica programable configure una célula de recurrencia según cualquiera de las reivindicaciones 4 a 10.

17. Conjunto de instrucciones para implementar un método en un medio programable que, cuando son interpretadas por dicho medio programable, hacen que un medio de lógica programable configure un circuito divisor según cualquiera de las reivindicaciones 11 a 14.

18. Conjunto de instrucciones para implementar un método en un medio programable que, cuando son interpretadas por dicho medio programable, hacen que este medio realice dicho método según cualquiera de las reivindicaciones 1 a 3.

19. Producto que comprende un conjunto de instrucciones registradas en un medio legible y ejecutables en un medio programable, cuya ejecución comprende que dicho medio programable realice cualquiera de las reivindicaciones 15 a 18.

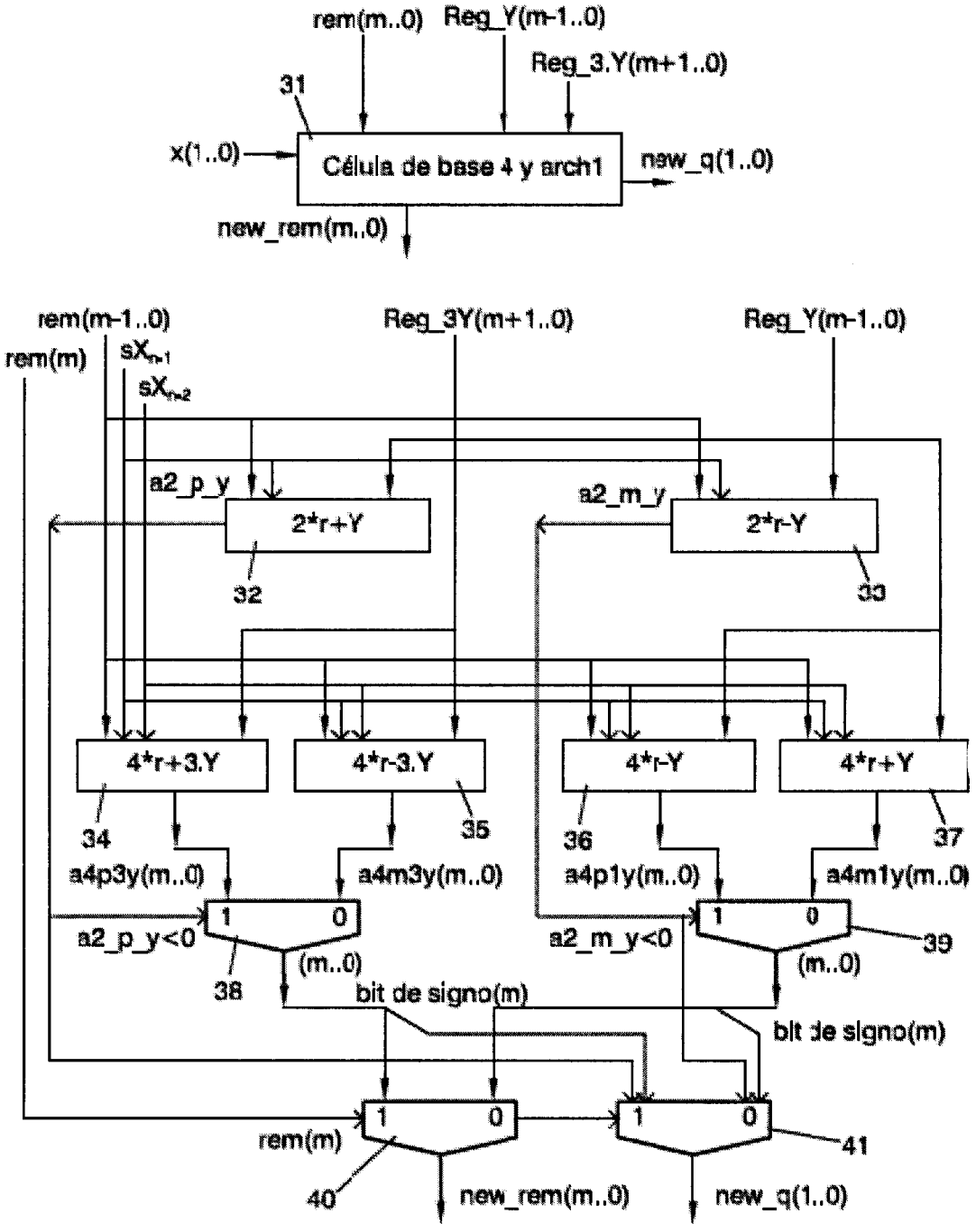
rem(m..0)
Reg_Y(m-1..0)
Reg_3.Y(m+1..0)
31
x(1..0)
Célula de base 4 y arch1
new_q(1..0)
new_rem(m..0)
rem(m-1..0)
rem(m)
Reg_3Y(m+1..0)
Reg_Y(m-1..0)
a2_p_y
2*r+Y
32
a2_m_y
2*r-Y
33
4*r+3.Y
4*r-3.Y
4*r-Y
4*r+Y
34
35
36
37
a4p3y(m..0)
a4m3y(m..0)
a4p1y(m..0)
a4m1y(m..0)
a2_p_y<0
a2_m_y<0
39
(m..0)
38
bit de signo(m)
41
40

FIG. 1

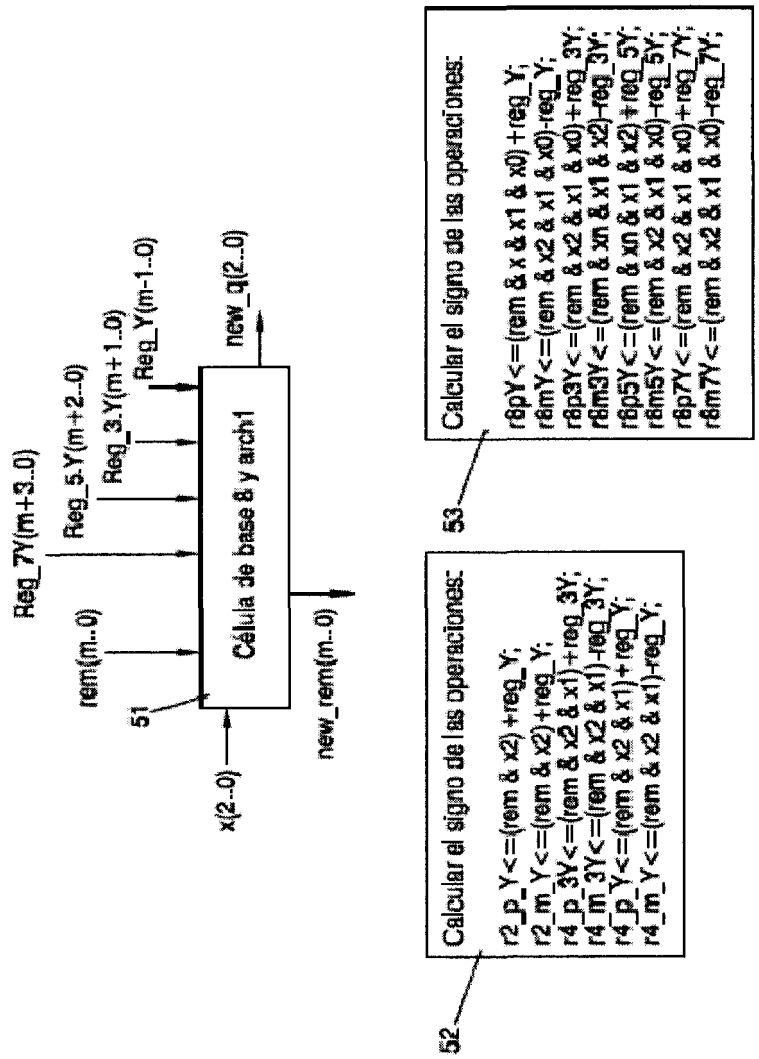
Reg_7Y(m+3..0)
Reg_5.Y(m+2..0)
Reg_3.Y(m+1..0)
Reg_Y(m-1..0)
rem(m..0)
51
Célula de base 8 y arch1
new_q(2..0)
x(2..0)
new_rem(m..0)
Calcular el signo de las operaciones:
r8pY<=(rem & x & x1 & x0)+reg_Y;
r8mY<=(rem & x2 & x1 & x0)-reg_Y;
r8p3Y<=(rem & x2 & x1 & x0)+reg_3Y;
r8m3Y<=(rem & xn & x1 & x2)-reg_3Y;
r8p5Y<=(rem & xn & x1 & x2)+reg_5Y;
r8m5Y<=(rem & x2 & x1 & x0)-reg_5Y;
r8p7Y<=(rem & x2 & x1 & x0)+reg_7Y;
r8m7Y<=(rem & x2 & x1 & x0)-reg_7Y;
53
Calcular el signo de las operaciones:
r2_p_Y<=(rem & x2)+reg_Y;
r2_m_Y<=(rem & x2)+reg_Y;
r4_p_3Y<=(rem & x2 & x1)+reg_3Y;
r4_m_3Y<=(rem & x2 & x1)-reg_3Y;
r4_p_Y<=(rem & x2 & x1)+reg_Y;
r4_m_Y<=(rem & x2 & x1)-reg_Y;
52

FIG. 2A

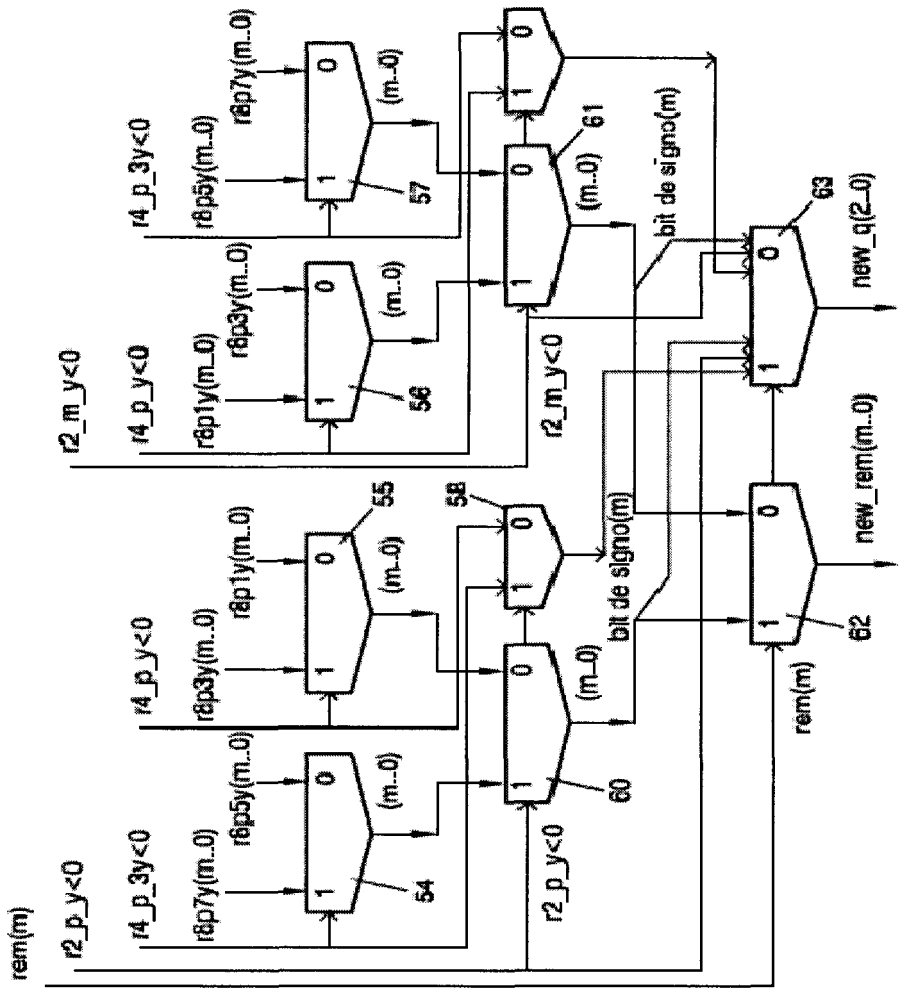
rem(m)
r2_p_y<0
r4_p_3y<0
r8p7y(m..0)
r8p5y(m..0)
r4_p_y<0
r8p3y(m..0)
r8p1y(m..0)
r2_m_y<0
(m..0)
54
55
56
57
58
60
61
62
63
bit de signo(m)
rem(m)
new_rem(m..0)
new_q(2..0)

FIG. 2B

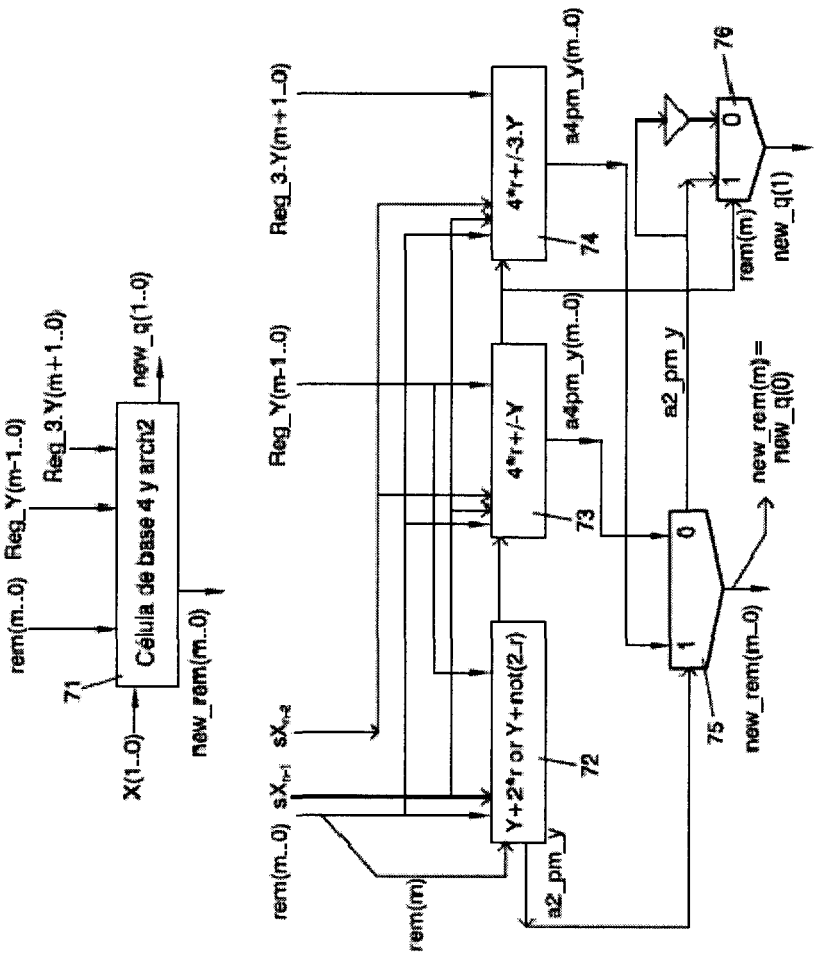
rem(m..0)
Reg_Y(m-1..0)
Reg_3.Y(m+1..0)
Célula de base 4 y arch2
new_q(1..0)
new_rem(m..0)
71
X(1..0)
Reg_Y(m-1..0)
Reg_3.Y(m+1..0)
4*r+/-3.Y
a4pm_y(m..0)
74
4*r+/-Y
a4pm_y(m..0)
73
Y+2*r or Y+not(2_r)
72
rem(m..0)
sX_i-2
sX_i-1
rem(m)
a2_pm_y
a2_pm_y
rem(m)
new_q(1)
76
new_rem(m)=
new_q(0)
new_rem(m..0)
75

FIG. 3

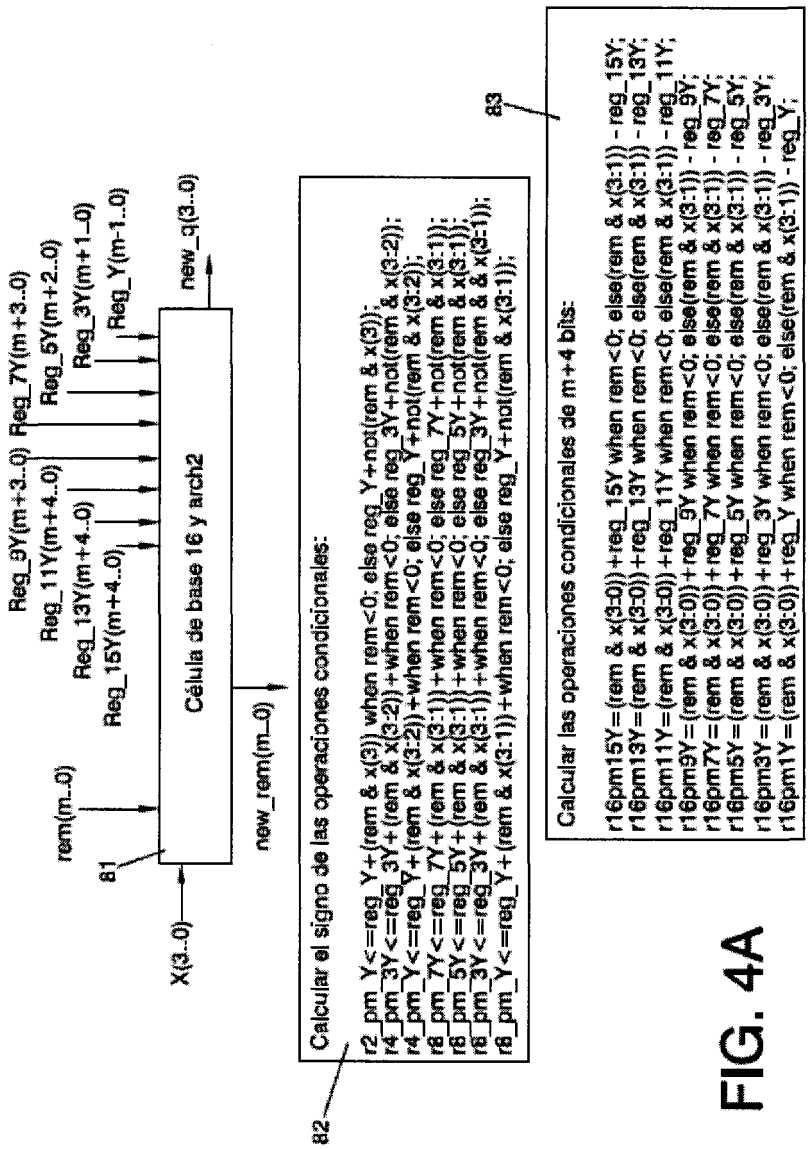
Reg_9Y(m+3..0)
Reg_11Y(m+4..0)
Reg_13Y(m+4..0)
Reg_15Y(m+4..0)
Reg_7Y(m+3..0)
Reg_5Y(m+2..0)
Reg_3Y(m+1_0)
Reg_Y(m-1..0)
rem(m_0)
X(3..0)
81
Célula de base 16 y arch2
new_q(3..0)
new_rem(m_0)
82
Calcular el signo de las operaciones condicionales:
r2_pm_Y<=reg_Y+(rem & x(3)) when rem<0; else reg_Y+not(rem & x(3));
r4_pm_3Y<=reg_3Y+(rem & x(3:2))+when rem<0; else reg_3Y+not(rem & x(3:2));
r4_pm_Y<=reg_Y+(rem & x(3:2))+when rem<0; else reg_Y+not(rem & x(3:2));
r8_pm_7Y<=reg_7Y+(rem & x(3:1))+when rem<0; else reg_7Y+not(rem & x(3:1));
r8_pm_5Y<=reg_5Y+(rem & x(3:1))+when rem<0; else reg_5Y+not(rem & x(3:1));
r8_pm_3Y<=reg_3Y+(rem & x(3:1))+when rem<0; else reg_3Y+not(rem & & x(3:1));
r8_pm_Y<=reg_Y+(rem & x(3:1))+when rem<0; else reg_Y+not(rem & x(3:1));
83
Calcular las operaciones condicionales de m+4 bits:
r16pm15Y=(rem & x(3:0))+reg_15Y when rem<0; else(rem & x(3:1)) - reg_15Y;
r16pm13Y=(rem & x(3:0))+reg_13Y when rem<0; else(rem & x(3:1)) - reg_13Y;
r16pm11Y=(rem & x(3:0))+reg_11Y when rem<0; else(rem & x(3:1)) - reg_11Y;
r16pm9Y=(rem & x(3:0))+reg_9Y when rem<0; else(rem & x(3:1)) - reg_9Y;
r16pm7Y=(rem & x(3:0))+reg_7Y when rem<0; else(rem & x(3:1)) - reg_7Y;
r16pm5Y=(rem & x(3:0))+reg_5Y when rem<0; else(rem & x(3:1)) - reg_5Y;
r16pm3Y=(rem & x(3:0))+reg_3Y when rem<0; else(rem & x(3:1)) - reg_3Y;
r16pm1Y=(rem & x(3:0))+reg_Y when rem<0; else(rem & x(3:1)) - reg_Y;

FIG. 4A

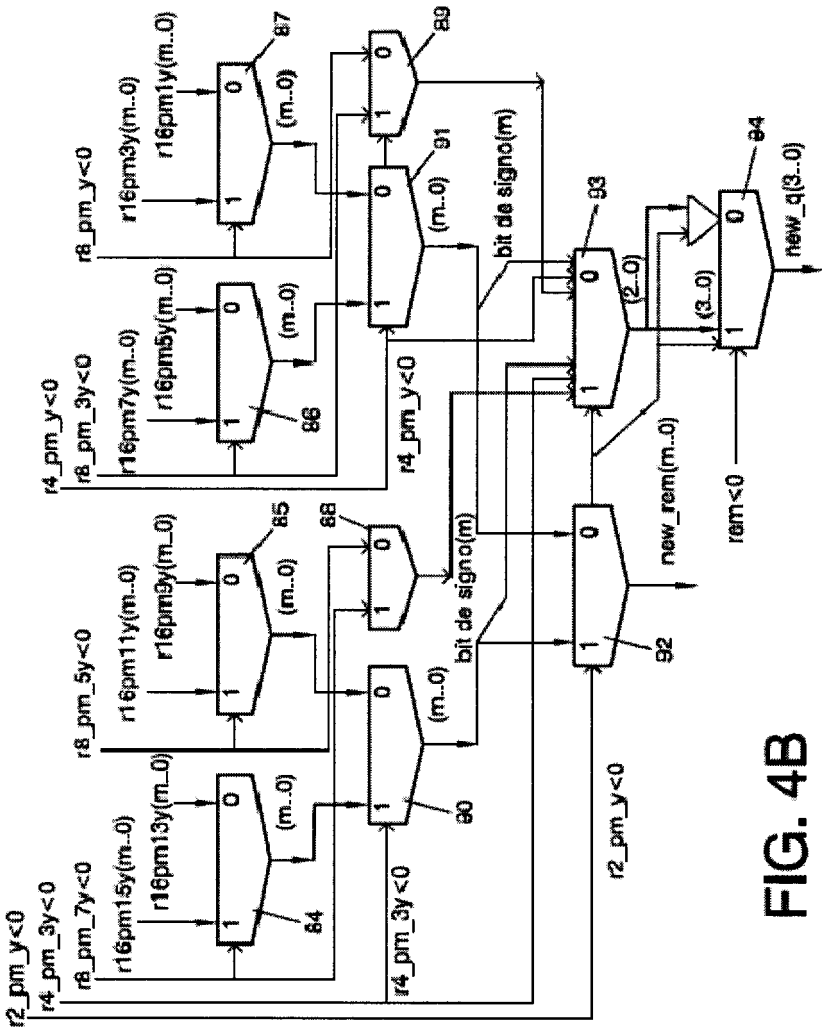
r2_pm_y<0
r4_pm_3y<0
r8_pm_7y<0
r16pm15y(m..0)
r16pm13y(m..0)
r8_pm_5y<0
r16pm11y(m..0)
r16pm9y(m..0)
r4_pm_y<0
r8_pm_3y<0
r16pm7y(m..0)
r16pm5y(m..0)
r8_pm_y<0
r16pm3y(m..0)
r16pm1y(m..0)
(m..0)
84
85
86
87
80
88
89
91
92
93
bit de signo(m)
new_rem(m..0)
rem<0
(2..0)
(3..0)
94
new_q(3..0)

FIG. 4B

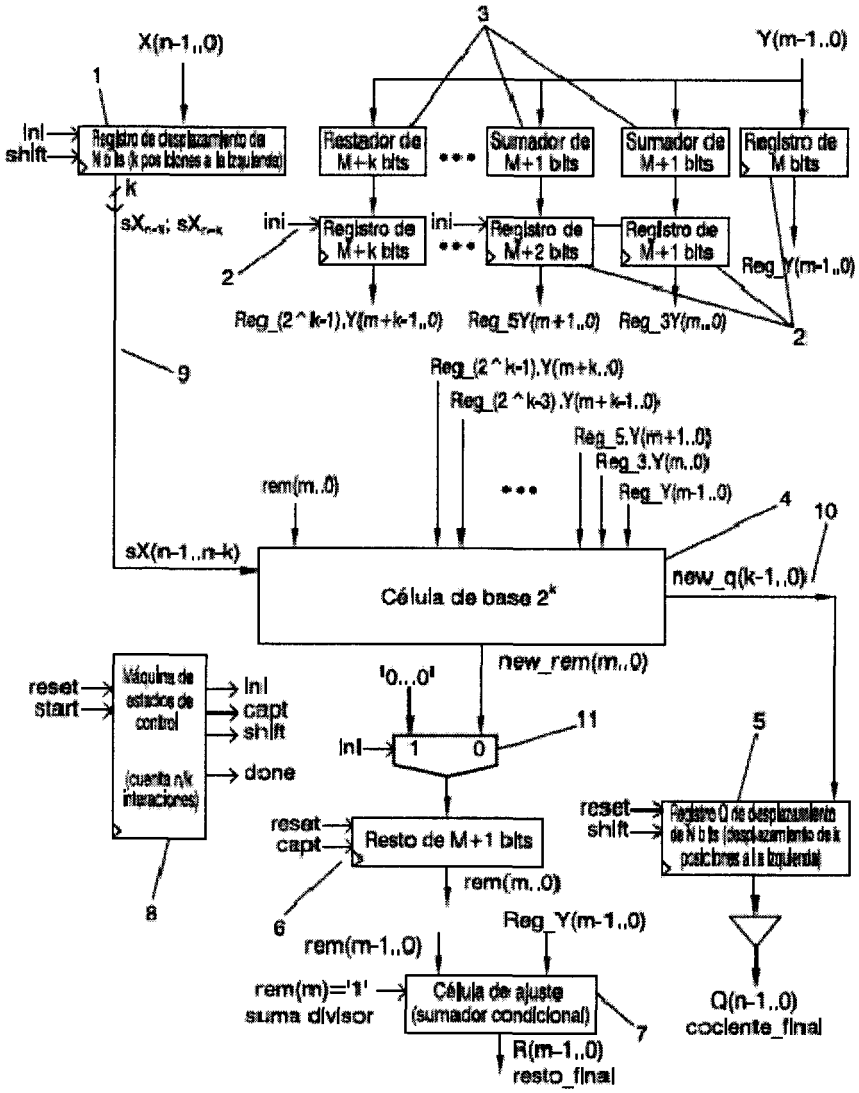
X(n-1,,0)
Y(m-1..0)
3
1
Ini
shift
Registro de desplazamiento de N bits (k posiciones a la izquierda)
k
Restador de M+k bits
Sumador de M+1 bits
Sumador de M+1 bits
Registro de M bits
ini
Registro de M+k bits
Registro de M+2 bits
Registro de M+1 bits
Reg_Y(m-1,,0)
2
Reg_(2^k-1),Y(m+k-1,,0)
Reg_5Y(m+1,,0)
Reg_3Y(m,,0)
9
Reg_(2^k-1),Y(m+k,,0)
Reg_(2^k-3),Y(m+k-1,,0)
Reg_5,Y(m+1,,0)
Reg_3,Y(m,,0)
Reg_Y(m-1,,0)
rem(m,,0)
4
10
sX(n-1,,n-k)
Célula de base 2^k
new_q(k-1,,0)
new_rem(m,,0)
reset
start
Máquina de estados de control
Ini
capt
shift
done
(cuenta n/k iteraciones)
'0...0'
11
5
1
0
reset
capt
Resto de M+1 bits
reset
shift
Registro Q de desplazamiento de N bits (desplazamiento de k posiciones a la izquierda)
rem(m,,0)
8
6
Reg_Y(m-1,,0)
rem(m-1,,0)
rem(m)='1'
suma divisor
Célula de ajuste (sumador condicional)
7
Q(n-1,,0)
cociente_final
R(m-1,,0)
resto_final

FIG. 5

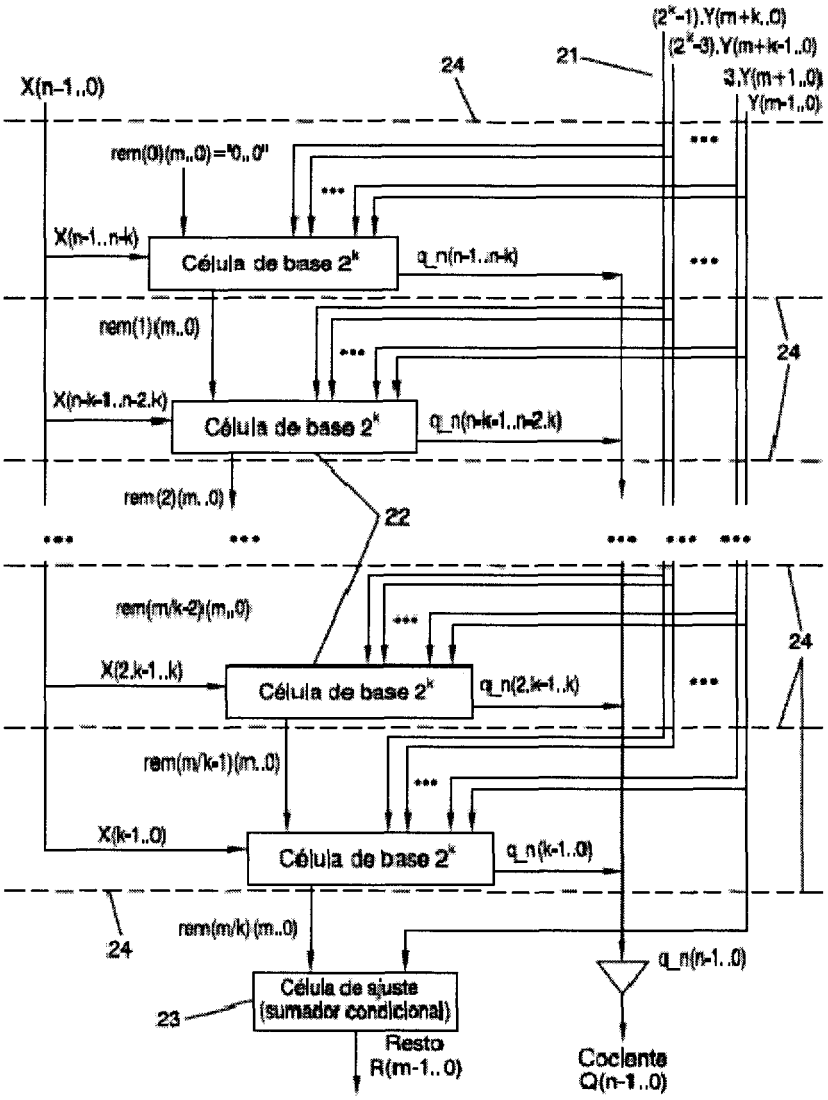
(2^k-1).Y(m+k..0)
(2^k-3).Y(m+k-1..0)
3.Y(m+1..0)
Y(m-1..0)
21
24
X(n-1..0)
rem(0)(m..0)='0..0'
X(n-1..n-k)
Célula de base 2^k
q_n(n-1..n-k)
rem(1)(m..0)
X(n-k-1..n-2.k)
Célula de base 2^k
q_n(n-k-1..n-2.k)
rem(2)(m..0)
22
rem(m/k-2)(m..0)
X(2.k-1..k)
Célula de base 2^k
q_n(2.k-1..k)
rem(m/k-1)(m..0)
X(k-1..0)
Célula de base 2^k
q_n(k-1..0)
rem(m/k)(m..0)
Célula de ajuste
(sumador condicional)
23
Resto
R(m-1..0)
q_n(n-1..0)
Cociente
Q(n-1..0)

FIG. 6

OFICINA ESPAÑOLA
DE PATENTES Y MARCAS

ESPAÑA

(21) N.º solicitud: 200901646

(22) Fecha de presentación de la solicitud: 24.07.2009

(32) Fecha de prioridad:

INFORME SOBRE EL ESTADO DE LA TECNICA

(51) **Int. Cl. :** ***G06F7/49*** (2006.01)
G06F7/535 (2006.01)

DOCUMENTOS RELEVANTES

Categoría	Documentos citados	Reivindicaciones afectadas
X	GUSTAVO SUTTER; JEAN PIERRE DESCHAMPS; “Fast radix 2K dividers for FPGAs" 5th Southern Conference on Programmable Logic, 2009. SPL. Págs.: 115-122; ISBN 978-1-4244-3847-1; ISBN 1-4244-3847-0.	1-19
A	ROY G SALTMAN. "Reducing Computing Time for Synchronous Binary Division" IEEE Transactions on Electronic Computers. Vol: EC-12; Nr: 2; Págs.: 169-174. ISSN 0367-7508.	1

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**
26.12.2011	M. Muñoz Sanchez	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, XPMISC, XPI3E, XPIETF, XPIEE, XPESP, NPL, COMPDX

Fecha de Realización de la Opinión Escrita: 26.12.2011

Declaración

Novedad (Art. 6.1 LP 11/1986)	Reivindicaciones		**SI**
	Reivindicaciones	1-19	**NO**
Actividad inventiva (Art. 8.1 LP11/1986)	Reivindicaciones		**SI**
	Reivindicaciones	1-19	**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	GUSTAVO SUTTER; JEAN PIERRE DESCHAMPS; "Fast radix 2K dividers for FPGAs" 5th Southern Conference on Programmable Logic, 2009. SPL. Págs.: 115-122; ISBN 978-1-4244-3847-1; ISBN 1-4244-3847-0.	01.04.2009
D02	ROY G SALTMAN. "Reducing Computing Time for Synchronous Binary Division" IEEE Transactions on Electronic Computers. Vol: EC-12; Nr: 2; Págs.: 169-174. ISSN 0367-7508.	01.06.1961

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 para hallar los restos de una división de dos operandos sin restauración a partir de una celda base 2^k en el que:
- se calculan los 2^{k-1} primeros múltiplos impares del divisor
- se hallan los posibles valores del resto, calculando los posibles restos parciales mediante sumas y restas
- se selecciona el resto parcial según la secuencia de la solicitud
- se utilizan los signos de la secuencia de restos parciales como en la solicitud

Por tanto el documento D01 afecta a la novedad de la reivindicación 1 según el artículo 6.1 de la Ley de Patentes.

Reivindicación 4: la célula reivindicada con los medios de operaciones aritméticas, lógicas, registros de memoria y medios de interconexión aparecen en el documento D01 específicamente conectados para implementar el método según una de las reivindicaciones 1-3. Por tanto el documento D01 afecta a la novedad de la reivindicación 4 según el artículo 6.1 de la Ley de Patentes.

Reivindicación 15: la mera implementación (directa, una transcripción) del método reivindicado no presenta novedad según el artículo 6.1 de la Ley de patentes ya que se considera que implícitamente dicho método define unívocamente las instrucciones de su implementación. Por tanto el documento D01 afecta a la novedad de la reivindicación 15 según el artículo 6.1 de la Ley de Patentes.

Reivindicación 19: la incorporación sin más del conjunto de instrucciones reivindicado al soporte reivindicado no presenta novedad según el artículo 6.1 de la Ley de patentes ya que se considera que implícitamente que el conjunto ha de residir en algún medio desde las que se leen y ejecutan. Por tanto el documento D01 afecta a la novedad de la reivindicación 19 según el artículo 6.1 de la Ley de Patentes.

Reivindicaciones dependientes

Reivindicaciones 2-3, 5-14, 16-18: el contenido de estas reivindicaciones forma explícitamente parte del documento D01 o puede considerarse implícito por ser características complementarias necesarias. Por tanto el documento D01 afecta a la novedad de las reivindicaciones 2-3, 5-14, 16-18 según el artículo 6.1 de la Ley de Patentes.