Implementación de un multiplicador de punto flotante de doble precisión basado en el estándar IEEE 754-2008

José Itzcóatl Sandoval López

CUCEI, Universidad de Guadalajara, Departamento de Electrónica, México

Blvd. Marcelino García Barragán # 1421, C.P. 44430, Guadalajara, Jalisco, México

Itzcoatl6@hotmail.com

Juan José Raygoza Panduro

CUCEI, Universidad de Guadalajara, Departamento de Electrónica, México

Blvd. Marcelino García Barragán # 1421, C.P. 44430, Guadalajara, Jalisco, México

juan.raygoza@cucei.udg.mx

Susana Ortega Cisneros

Centro de Investigación y de Estudios Avanzados del I.P.N, CINVESTAV, Unidad Guadalajara

Av. del Bosque 1145, colonia el Bajío, Zapopan, C.P. 45019, Jalisco, México

susana.ortega@gdl.cinvestav.mx

Jorge Rivera Domínguez

Centro de Investigación y de Estudios Avanzados del I.P.N, CINVESTAV, Unidad Guadalajara

Av. del Bosque 1145, colonia el Bajío, Zapopan, C.P. 45019, Jalisco, México

riveraj@gdl.cinvestav.mx

Resumen

Este artículo presenta la síntesis, a partir de la descripción en VHDL de un multiplicador de punto flotante, basado en el estándar para Aritmética de Punto Flotante de IEEE (754™-2008) para microprocesadores, del cual se utiliza el formato binario para precisión doble de 64 bits. El estándar define formatos para representar diferentes tipos

de datos los cuales son: normal, subnormal, cero positivo, cero negativo, infinito positivo, infinito negativo y un no número (NaN). Muchas aplicaciones basadas en procesadores embebidos requieren la capacidad para realizar operaciones aritméticas de punto flotante, lo cual es fundamental para una mejor precisión y desempeño del sistema en el procesamiento de los datos. Además de definir una solución de hardware, también, se implementa el diseño de un multiplicador de punto fijo, mediante el algoritmo de Booth.

Palabra(s) Clave(s): FPGAS, IEEE-754, multiplicación, punto flotante, VHDL.

1. Introducción

La multiplicación es una operación necesaria para realizar operaciones complejas, a la vez es considerada como la que mayor tiempo necesita para ejecutarse. Por este motivo se requiere de una implementación eficiente para que su desempeño sea adecuado.

En algunas ocasiones, al realizar cálculos complejos, como en ingeniería, astronomía, aeronáutica, etc., los números enteros ya no son suficientes para aplicaciones donde se utilizan números muy pequeños o demasiado grandes y es indispensable incluir una parte fraccionaria o una notación científica.

El estándar de la IEEE para aritmética en punto flotante (*IEEE Standard for Floating-Point Arithmetic* [1]), especifica formatos para su uso en los sistemas informáticos, también define la precisión simple, doble y extendida así como los métodos para el intercambio de datos; este estándar es el más utilizado para resolver operaciones de alta precisión.

En este trabajo se utiliza el formato de precisión doble de 64 bits donde un dato está compuesto de signo, exponente y mantisa debidamente ordenados; el código esta descrito, sintetizado y simulado en el software ISE™ 14.6 de Xilinx®.

2. Desarrollo

En esta sección se presentan los conceptos básicos que explican el comportamiento del multiplicador de punto flotante.

2.1. Multiplicación

Para realizar la multiplicación de punto flotante es necesario realizar una multiplicación de punto fijo. El rendimiento del sistema depende en gran medida del rendimiento de este multiplicador de punto fijo. Existen varios algoritmos para resolver la multiplicación, como El algoritmo de Booth, el algoritmo de Wallace, etc. El algoritmo de Booth es un método rápido y sencillo para implementar el producto de dos números binario.

2.2. Algoritmo de Booth

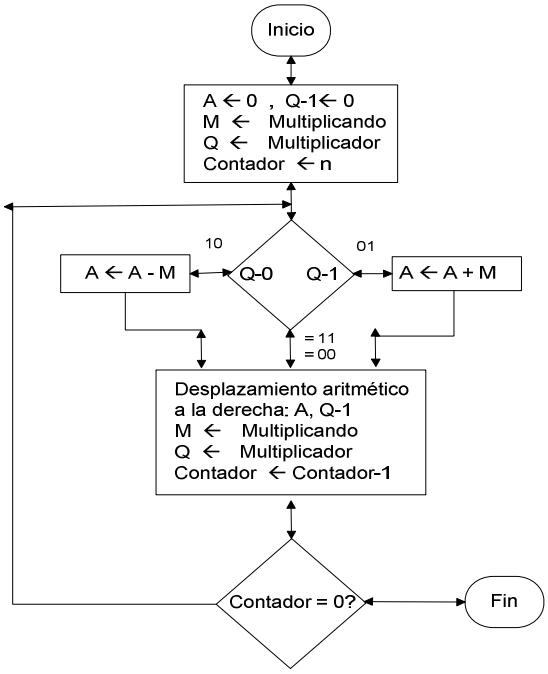

Fig. 1. Algoritmo de Booth para la multiplicación [2].

El algoritmo de Booth pretende minimizar el número de sumas que se deben realizar durante el proceso de multiplicación para reducir el tiempo en el que se realiza. En el

estado inicial se mandan a cero los bits del vector A, y al multiplicador Q se le agrega un 0 como bit menos significativo para realizar la primera comparación.

Se tiene también un contador de ciclos inicializado en n= número de bits de los multiplicandos. En cada ciclo se leen los últimos dos bits de Q y se decide si se hace la suma o resta de A y M, o simplemente se omite esa operación.

Posteriormente se hace un desplazamiento a la derecha del vector que contiene A y Q, el cual contendrá el resultado final, y se disminuye el contador. Finalmente se checa si ya se realizaron todos los ciclos, en caso de que no, se vuelve a realizar otro, y en el caso de que el contador haya llegado a cero, ya se ha terminado la multiplicación.

2.3. Punto flotante

Existen varias formas de representar números no enteros. Una de ellas es usando un punto fijo. Este tipo de representación ubica siempre el punto o coma en alguna posición a la derecha del dígito menos significativo (ver Tabla. 1). Otra alternativa común mente usada es la que se conoce como representación en punto flotante. Los números de punto flotante decimales normalmente se expresan en notación científica con un punto explícito siempre entre el primer y el segundo dígitos. El exponente o bien se escribe explícitamente incluyendo la base, o se usa una “e” para separarlo de la mantisa.

Mantisa	Exponente	Notación científica	Valor en punto fijo
1.5	4	$1.5\ x\ 10^{4}$	15000
-2.001	2	$-2.001\ \text{x}\ 10^{2}$	-200.1
5	-3	$5\ x\ 10^{-3}$	0.005
6.667	-11	$6.667\ \text{x}\ e^{-11}$	0.0000000000667

Tabla 1. Ejemplos de punto flotante decimal expresado como notación científica.

Dado que un número en punto flotante puede expresarse de distintas formas que son equivalentes, es necesario establecer una única representación. Es por ello que se trabaja con números normalizados. Decimos que un número está normalizado si el dígito a la izquierda del punto está entre 0 y la base (0< dígito a la izquierda del punto <b).

$$1.00 \times 10^{2} \text{ Normalizado} \quad (1)$$
$$0.01 \times 10^{-1} \text{ Desnormalizado} \quad (2)$$

En particular, decimos que un número binario está normalizado si el dígito a la izquierda del punto es igual a 1.

2.4. Formato de punto flotante (Estándar IEEE 754-2008)

En esta sección se describe el formato de precisión sencilla y precisión doble para punto flotante.

2.4.1. Presión sencilla

El formato con punto flotante de precisión sencilla [1] se representa en la aritmética binaria con un bit de signo (31), (Sx), 8 bits de exponente (30 al 23), (Ex), para nuestro caso b es la base=2 y 23 bits de mantisa (22 al 0), (Mx) (ver Fig. 1), (ver Ecuación 3). Para obtener una mayor precisión, se omite el bit correspondiente a la parte entera de la mantisa y se supone “1” cuando es un número normal y “0“ cuando es un subnormal, obteniendo una precisión real de 24 bits de mantisa.

$$(-1)^{Sx} \times M'_x \times b^{E_x} \quad (3)$$

Signo	Exponente	Mantisa
31	30……23	22……..0

Fig. 2. Muestra el registro del formato de presión sencilla para punto flotante.

Para representar el exponente en punto flotante se utiliza una representación en exceso N de forma que el exponente más negativo posible para precisión sencilla, quede en 0000 0001 y el más grande de los positivos en 1111 1110 .

El estándar IEEE 754 usa como exceso 127 en precisión sencilla, el exponente está situado en un rango de −126 a +127 y es desplazado mediante la suma de 127 para obtener un valor en el rango 1 a 254, recordando que su campo tiene 8 bits (0 y 255 tienen valores especiales descritos más adelante). Cuando se interpreta el valor en coma flotante, el número es desplazado de nuevo para obtener el exponente real [4].

Por ejemplo, para representar el número 0.50_{10} en punto flotante.

0.50 (4)
(0.50 - 0) * 2 = 1 d0=0 (5)
(1.00 - 1) * 2 = 0 d1=1 (6)
$0.50_{10} = 0.1_2 = 1.0_2 \times 2^{-1}$ (7)
Exponente en exceso= -1 + 127 = 126 decimal = 0111 1110 binario (8)

Signo	Exponente	Mantisa
31	30……23	22……0
0	01111110	00000000000000000000000

Fig. 3. Representación de 0.50_{10} en punto flotante representación sencilla.

2.4.2. Formato de doble precisión para punto flotante

La representación de un número en precisión doble en el formato IEEE-754 consta de las siguientes partes:

El bit de signo está situado en la posición del registro 63. Un “1” significa un número negativo, y un “0“, es un número positivo [8].

El exponente está situado entre los bits 62-52. El valor en este campo de es de 11 bits con un exceso de 1023, La mantisa es de 52 bits de longitud y ocupa los bits de 51-0. Hay un bit '1' que no está incluido en la mantisa, este bit pertenece a la parte entera del registro (ver Fig. 3).

Signo	Exponente	Mantisa
63	62…….52	51……..0

Fig. 4. Muestra el registro del formato de doble presión para punto flotante.

Por ejemplo, para representar el número 3.5 en formato de doble precisión de punto flotante El bit de signo es 0 situado en la posición 63 para representar un número positivo.

El exponente será 1024. Esto se calcula separando 3.5 como:

$$3.5=\mathbf{1.75}\ \boldsymbol{x}\ \mathbf{2^{1}} \tag{9}$$

El desplazamiento del exponente es 1023, por lo que al añadir 1023 + 1 para calcular el valor para el campo exponente. Por lo tanto, los bits 62 a 52 serán "1000000000". La mantisa corresponde a el 1,75, que se multiplica por la potencia de 2 ($\mathbf{2^{1}}$) para obtener 3,5. La parte entera del registro, '1' está implícito en la mantisa pero en realidad no incluido en el formato de 64 bits. Así 0.75 está representado por la mantisa. EL bit 51, es el bit más significativo de la mantisa, corresponde a $\mathbf{2^{(-1)}}$. El bit 50 corresponde al $\mathbf{2^{(-2)}}$ y esto continúa hasta el bit 0 lo que corresponde a $\mathbf{2^{(-52)}}$. Representar a 0,75, los bits 51 y 50 son de 1, y el resto de los bits son 0 de. Así 3.5 como un número de coma flotante de doble precisión es:

Signo	Exponente	Mantisa
63	62……52	51……..0
0	10000000000	11000000000000000000000000000000000000000000000000

Fig. 5. Numero 3.5 transformado a doble presión para punto flotante.

2.5. Casos especiales

Para valores de exponente desde 1 hasta 254 en el formato simple y desde 1 hasta 2046 en el formato doble, se representan números en punto fijo normalizados. El exponente está en exceso, siendo el rango del exponente de -126 a +127 en el formato simple y de -1022 a +1023 en el doble.

Un número normalizado debe contener un bit 1 a la izquierda del punto binario; este bit está implícito, dando una mantisa efectiva de 24 bits para precisión simple o 53 bits para precisión doble. Un exponente cero junto con una parte fraccionaria cero representa el cero positivo o negativo, dependiendo del bit de signo. Es útil tener una representación del valor 0 exacto.

Tipo De Dato	**Signo**	**Exponente**	**Mantisa**
Cero Positivo	0	E = 0	M = 0
Cero Negativo	1	E = 0	M = 0
Subnormal	S	E = 0	M > 0
Normal	S	0 < E < 255	M = X
Infinito	S	E = 255	M = 0
No un Número (NaN)	S	E = 255	M > 0

Tabla 2. Representación de tipos de datos y sus valores en precisión sencilla.

Tipo De Dato	Signo	Exponente	Mantisa
Cero Positivo	0	E. = 0	M = 0
Cero Negativo	1	E. = 0	M = 0
Subnormal	S	E. = 0	M > 0
Normal	S	0 < E. < 2047	M = X
Infinito	S	E = 2047	M = 0
No un Número (NaN)	S	E. = 2047	M > 0

Tabla 3. Representación de tipos de datos y sus valores en precisión doble.

2.6. Consideraciones

Como consideraciones podemos asumir lo siguiente:

Hay dos ceros. +0 (S es 0) y −0 (S es 1).
Hay dos infinitos +∞ (S es 0) y −∞ (S es 1).

Los NaNs pueden tener un signo y un significando, pero estos no tienen otro significado que el que puedan aportar en pruebas de diagnóstico; el primer bit del significando es a menudo utilizado para distinguir los NaNs señalizados (SnaN: *Signalling Not a Number*) de los NaNs silenciosos (QNaN: *Quiet Not a Number*). Los primeros son utilizados cuando el dato ingresado al sistema no tiene el formato de un número válido especificado por la norma de punto flotante. Los QNaN especifican que la operación realizada con las cifras ingresadas produce un valor numérico no válido. Los NaNs y los infinitos tienen todos los bits puestos a 1 en el campo exponente [5].

3. Errores de redondeo

La representación de punto flotante tiene un número de dígitos limitado, por consiguiente no es posible representar todos los reales de forma precisa: cuando hay más dígitos de los que permite el formato, los que sobran se omiten, y se redondea.

Pistas Educativas Año XXXVI - ISSN 1405-1249
Certificado de Licitud de Título 6216; Certificado de Licitud de Contenido 4777; Expediente de Reserva 6 98 92

Existen tres razones por las que esto puede ser necesario:

A) Denominadores grandes En cualquier base, cuanto mayor es el denominador de una fracción (irreducible). Un denominador lo suficientemente grande necesitará redondeo, no importa qué base o número de dígitos disponible haya.

B) Dígitos periódicos Cualquier fracción (irreducible). Por ejemplo, $\frac{1}{3}$ no es finito, ni tampoco $\frac{2}{3}$, $\frac{1}{7}$ o $\frac{5}{6}$.

C) Los números irracionales no se pueden representar como una fracción regular, requieren un número infinito de dígitos no periódicos.

D) Los tipos de redondeo más común en punto flotante son:

Hacia el cero o truncamiento: simplemente se omiten los dígitos sobrantes. Es el método más sencillo, pero introduce más error del necesario y un sesgo hacia el cero cuando se manejan sobre todo números positivos o sobre todo negativos.

Redondeo al alza: si la fracción truncada es mayor o igual que la mitad de la base, se incrementa el último dígito. Este método es el que se enseña [4].

Redondeo mitad al par: también conocido como el redondeo del banquero - si la fracción truncada es mayor que la mitad de la base, se incrementa el último dígito. Si es igual a la mitad de la base, se incrementa solamente si el resultado es par. Esto minimiza el error y el sesgo, y por eso se prefiere en contabilidad. Es el método por defecto en el estándar IEEE 754.

Número	Hacia el cero	Al alza	La mitad al par
1.4	1	1	1
1.5	1	2	2
- 1.6	-1	-2	-2
2.6	2	3	3
2.5	2	3	2
- 2.4	-2	-2	-2

Tabla 4. Ejemplo de los tipos de redondeo que existen.

4. Implementación en FPGAs

La figura 4 ilustra el diagrama esquemático principal del circuito multiplicador de punto flotante de 64 bits, el cual está formado a base de bloques diseñados para el cálculo de cada tipo de datos y sus combinaciones. Éste es implementado en FPGAs de Xilinx ® [6].

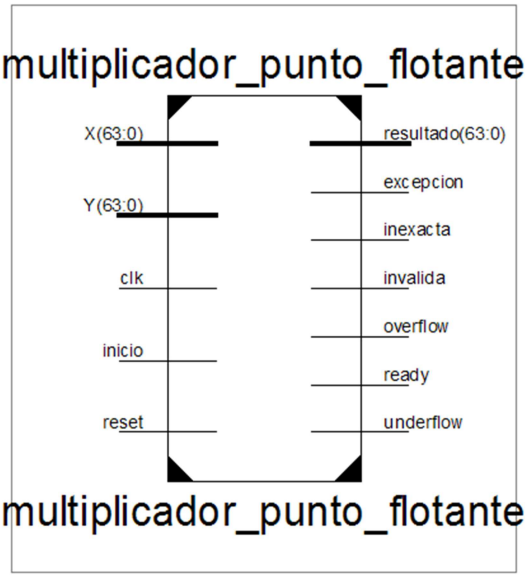

Fig. 6. Diagrama general del multiplicador de punto flotante.

Las señales de entrada al módulo son los siguientes:

1. Clk: Señal de reloj.

2. inicio: señal que indica que el dato está listo a ser procesado (en alto para iniciar la operación).

3. X: (Multiplicando de entrada, 64 bits).

4. Y: (Multiplicador de entrada, 64 bits).

Las señales de salida del módulo son los siguientes:

5. Resultado: Indica el resultado de la operación (salida de la operación, 64 bits).

6. Listo: indica que termino el cálculo (va alto cuando la salida está disponible).

7. *Underflow*: indica que el resultado no pertenece al rango mínimo de números que puede representar la unidad.

8. *Overflow*: indica que el resultado no pertenece al rango máximo de números que puede representar la unidad

9. Inexacta: si existiera más de una señal de las excepciones activas se activa.

10. Inválida: multiplicar 0 por infinito, indica que el resultado es no un número (NaN).

11. Excepción: indica alguna de las siguientes señales activas: *underflow, overflow*, inexacta o invalida.

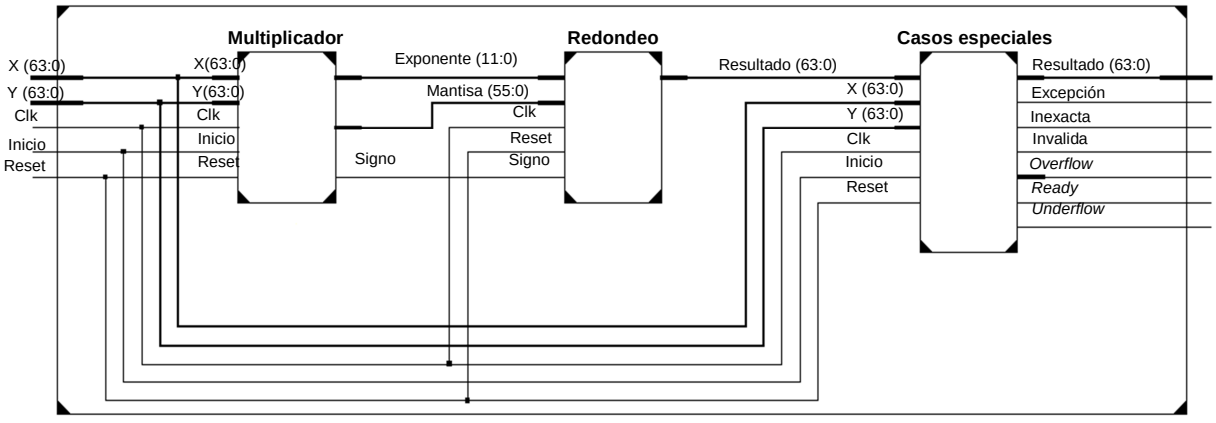

Fig. 7. Diagrama esquemático principal.

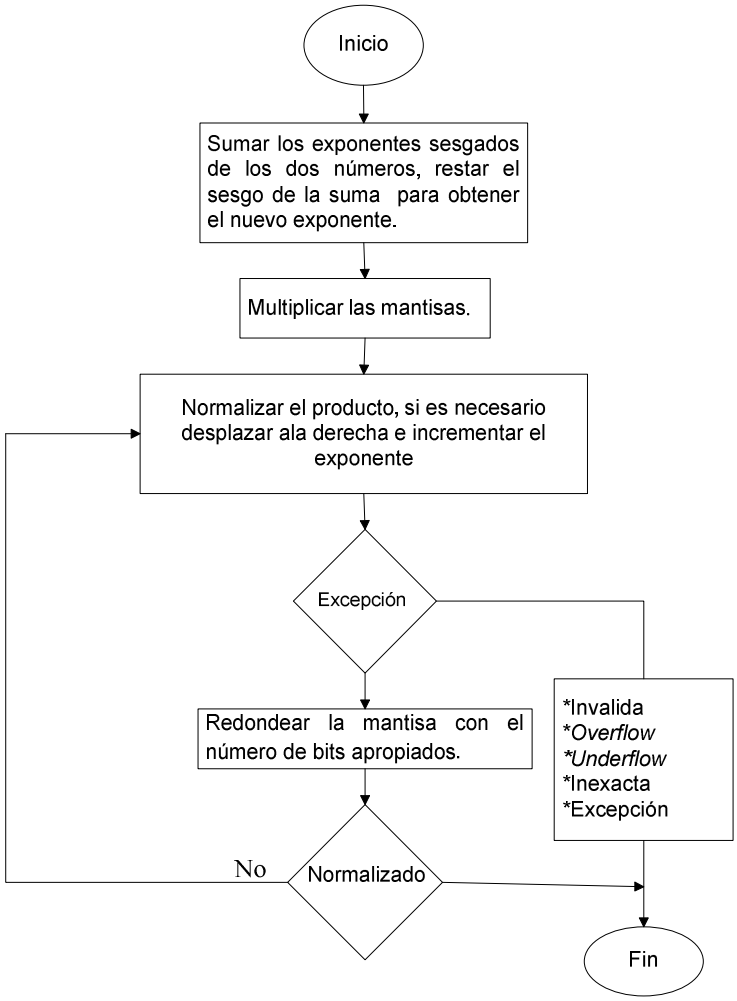

Fig. 8. Funcionamiento del multiplicador de punto flotante.

4.1. Bloque multiplicación

La multiplicación de punto flotante se realiza en el bloque “Multiplicador”. Este bloque recibe las señales exteriores del módulo general del multiplicador de punto flotante, cuenta con 5 señales de entrada y 3 de salida. Las señales de entrada son las siguientes: el multiplicando “X(63:0)” de 64 bits, que admite datos en formato de punto flotante precisión doble, normalizados, el multiplicador “Y(63:0)” que admite el mismo tipo de datos que el multiplicando, una señal de reloj “Clk”, una señal de inicio que

indica cuando el dato está listo para ser procesado (“0” para indicar alto,”1” para indicar que el dato está listo para procesar) y una señal de reinicio “*Reset*” (”1” para reiniciar).

Las señales de salida: El “signo”, que es de longitud de un bit, el “exponente (11:0)”, con una longitud 12 bits, y la “mantisa (55:0) que tiene una longitud de 56 bits. Estos 3 datos representan el producto de la multiplicación de punto flotante, se puede observar que estos datos no se encuentran aún representados de acuerdo a la norma IEEE 754-2008.

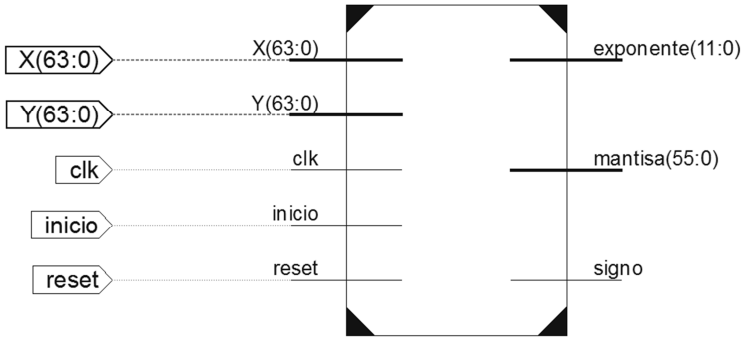

Fig. 9. Bloque multiplicación.

La salida de la mantisa tiene una longitud de 56 bits. El bit más significativo es un '0' para permitir un desbordamiento potencial en el módulo de redondeo. El primer bit '0' es seguido por '1' para los números normalizados, o "0" para los números no normalizados. Esto es la parte entera del registro. A continuación, los 52 bits de la mantisa más dos bits adicionales, que se utilizan para fines de redondeo.

4.1.1. Funcionamiento del bloque multiplicación

La mantisa del operando “X” y el bit '1' que no está incluido en la mantisa (la parte entera, para los números normalizados) se almacenan en un registro de 53 bits (mult a).

La mantisa del operando “Y” y el bit '1' que no está incluido en la mantisa (la parte entera, para los números normalizados) se almacenan en un registro de 53 bits (mult b).

Los exponentes se almacenan en registros de 12 bits. Los exponentes de X y Y se suman juntos y luego se les resta el valor de (1022) de la suma de X y Y. Si el exponente resultante es menor que 0, será necesario realizar un desplazamiento a la derecha en el producto de las mantisas. Este valor se almacena en el registro (*exponent_under*). En este caso el exponente final de la salida será 0 y el resultado será un número desnormalizado. Si *exponent_under* es mayor de 52, de la mantisa se desplaza fuera del registro de productos, y la salida será 0, y se activara una señal "*underflow*".

Al utilizar el multiplicador de Booth, todos los 53 bits de mul_a por 53 bits de mul_b resulta un producto de 106 bits, se almacena en el registro (producto). Después, se realiza un desplazamiento, de acuerdo a la cantidad de ceros a la izquierda, de este registro, si no hay un '1'en el bit más significativo del producto. El número de ceros a la izquierda en el registro (de productos) se cuentan por la señal (*product_shift*). El exponente de salida también será reducido por (*product_shift),* de acuerdo a la cantidad de ceros que se hayan encontrado.

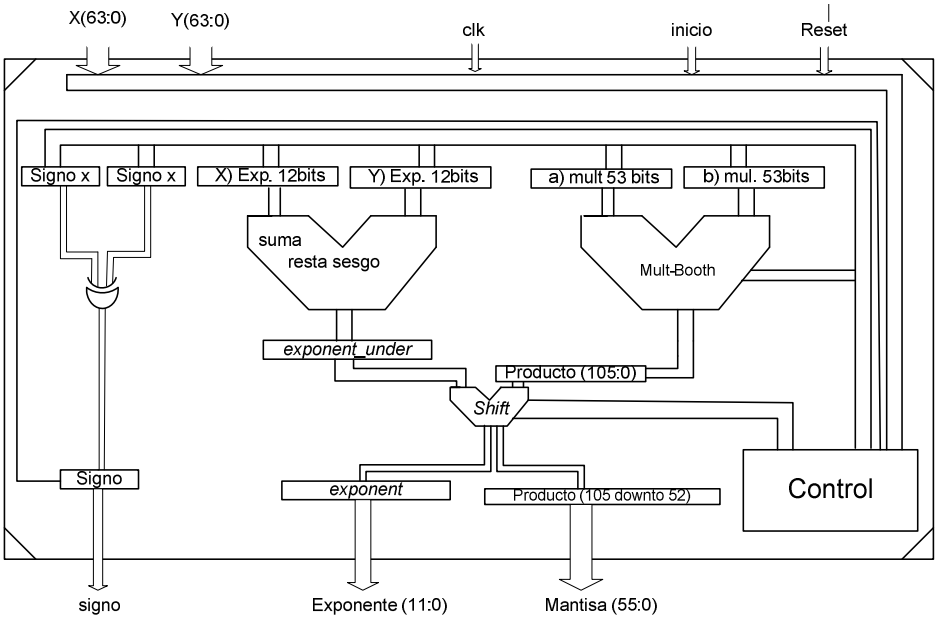

Fig. 10. Bloque de flujo de datos del multiplicador.

4.2. Bloque redondeo

La operación de redondeo se realiza en este módulo, las entradas a este bloque provienen del módulo anterior, multiplicador y son las siguientes:

Signo (1 bit), mantisa (56 bits), y exponentes (12 bits). La mantisa incluye un bit extra '0' como el bit mas significativo, seguido del bit de la parte entera, y dos bits en la parte menos significativa adicionales, y 52 bits de la mantisa. El exponente tiene un '0'en el bit mas significativo para provocar desbordamientos en caso de ser necesario. si el registro fuera de 11 bits, Un desbordamiento daría lugar a un valor de 0 en el campo de exponente, y el resultado final de la operación sería incorrecta. Cuenta con una senal de reloj (clk), un reinicio (R*eset*) y una salida para mostrar el resultado (resultado) con la norma IEEE 754-2008. (Ver Figura. 8).

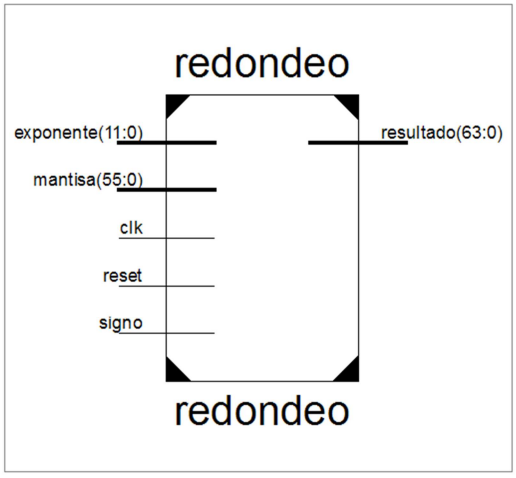

Fig. 11. Bloque redondeo.

Se redondea al modo alza de la siguiente manera: si el bit en la posición 1 y el bit en la posición 0 es un "1" de la mantisa, se redondea. Para llevar a cabo el redondeo, se suma la mantisa a la señal_redondeo. La señal_redondeo tiene un '1' en el espacio de bit que se alinea con el bit menos significativo de la mantisa de 52 bits.

Este '1' en la señal_redondeo se alinea con el 2 bit del registro mantisa (recordando que sus posiciones van de 55 a 0). Los bits 1 y 0 del registro mantisa son los bits restantes de más, y éstos son eliminados en el registro resultado.

4.3. Bloque casos

En este bloque, todos los casos especiales se comprueban, y si se encuentran, la adecuada se activa la salida: excepción, inexacta, inválida, *overflow*, etc.

Se afirma cuando se cumplen las condiciones de cada caso. Algunos casos especiales son: Multiplicar 0 por infinito: activa la señal inválida; multiplicar por un dato fuera del rango máximo o mínimo de los valores que puedan representarse será un *overflow* o *underflow*, entre otros.

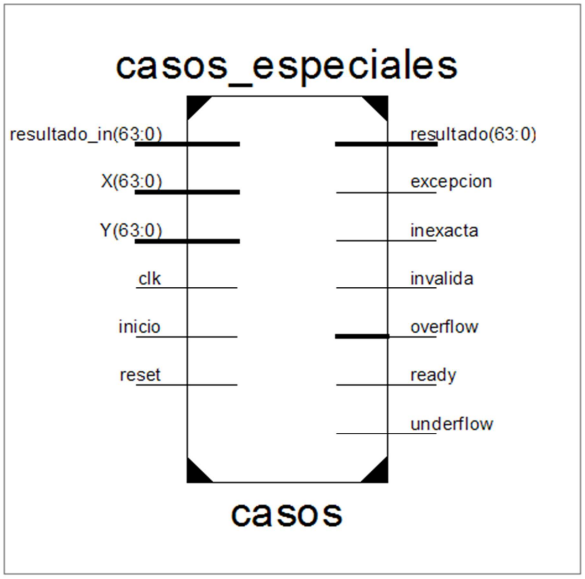

Fig. 12. Bloque casos especiales.

5. Resultados

El diseño propuesto se verificó con simulaciones del sistema completo en el software de Xilinx ISE 14.6 en donde se implementó el sistema en lenguaje VHDL. Los resultados de la simulación del circuito multiplicador de punto flotante se muestran en la figura 10

en donde el cursor en el tiempo 700ns indica el dato “X” que contiene “11111111111“ de exponente y la mantisa es cero. Revisando la tabla 3, observamos que éste es un dato inválido y, de igual manera, que el dato “Y” que contiene cero.

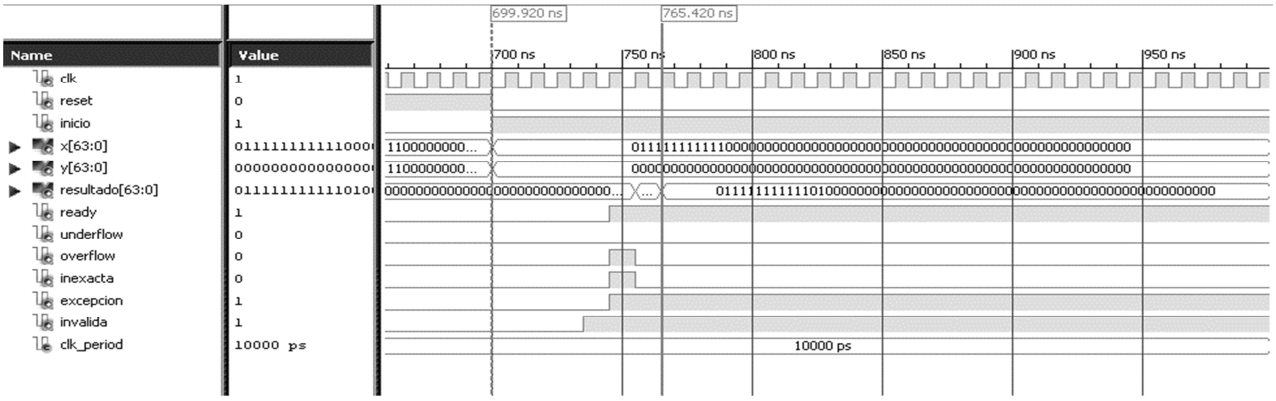

Fig. 13. Multiplicación de un número infinito x un cero. La bandera excepción se activa y la bandera inválida también se activa.

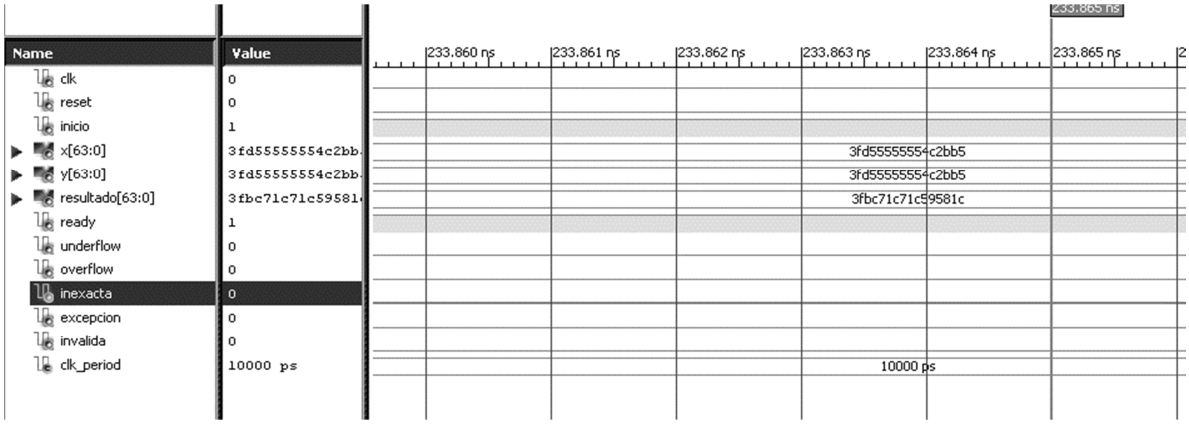

Fig. 14. Multiplicación de 0.3333333333 x 0.3333333333= 0.11111111108888888889 (formato decimal).

Los resultados de la figura 11 se muestran en formato doble precisión y base hexadecimal.

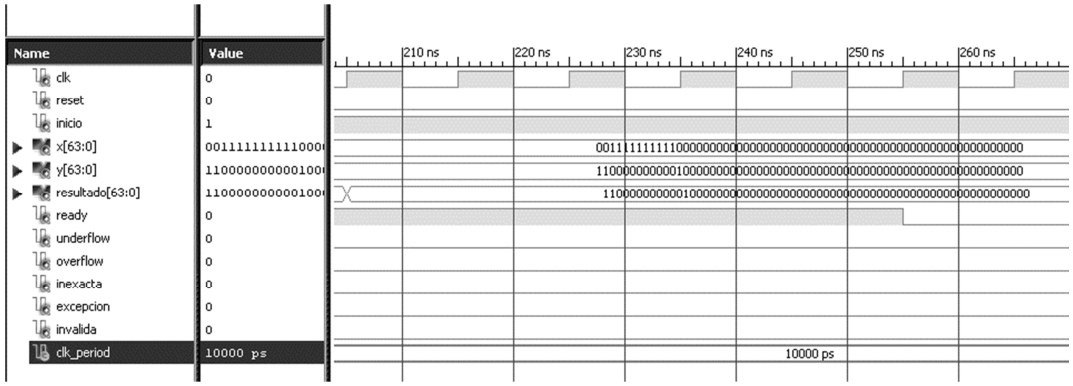

Fig. 15. Multiplicación de 1 x -3 en punto flotante doble precisión.

Se observa en la figura 12 que el multiplicador Y (63:0) es igual al producto “resultado (63:0)”. Los resultados se muestran en formato doble precisión y base binaria.

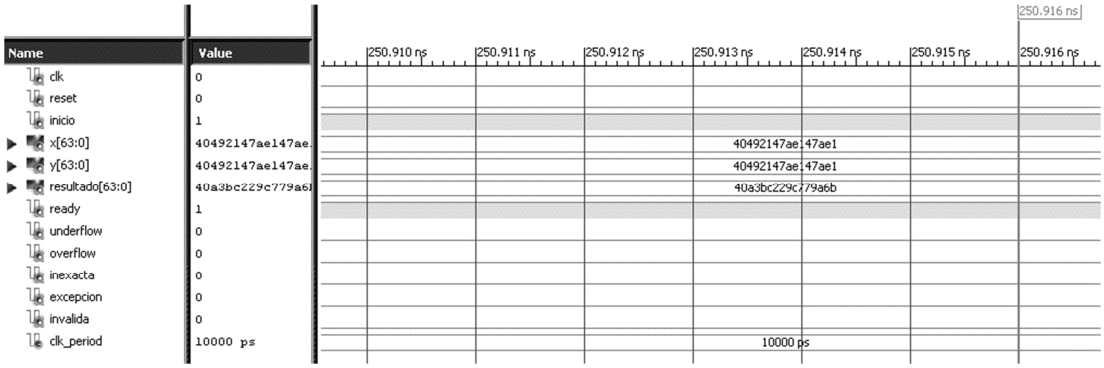

Fig. 16. Multiplicación de 50.26 x 50.26=2526.0676 (formato decimal).

Los resultados de la figura 13 se muestran en formato doble precisión y base hexadecimal.

Los resultados de ocupación de los recursos en FPGAs se muestran en la tabla 5 y tabla 6 en donde se puede observar que el circuito multiplicador puede ser implementado en las 3 FPGAs.

Dispositivo Utilizado Ocupación	**Virtex 6 (XC6VLX240T)**			**Virtex 5(xc5vtx240t)**		
	Usadas	Total	%			
Número de *Slice*	1,765	150,720	1%	968	149,760	1%
Número de *LUTs*	966	301,440	1%	3,343	149,760	2%
Número de *IOBs* utilizadas	201	400	50%	201	680	29%

Tabla 5. Tabla de ocupación para Virtex 6 y Virtex 5.

Dispositivo Utilizado Ocupación	**Virtex 7 (xc7vx330t)**		
	Usadas	Total	%
Número de *Slice*	966	408,000	1%
Número de *LUTs*	1,863	204,000	1%
Número de *IOBs* utilizadas	201	600	33%

Tabla 6. Tabla de ocupación para Virtex 7.

6. Discusión

De los tres módulos de operación, el módulo de la multiplicación es considerado como la etapa crítica y de mayor importancia para el funcionamiento correcto del sistema. Esto se debe a que el sistema incluye una multiplicación de punto fijo de 53x53 bits que en términos de hardware se traduce en una gran cantidad de componentes dependiendo del algoritmo utilizado. El sistema propuesto también es capaz de detectar condiciones de desbordamiento de datos (*underflow y overflow*) y cuando un número de entrada es igual a cero [7]. Por lo tanto, este sistema es completamente funcional.

7. Conclusiones

En este proyecto, se presentó la implementación de un multiplicador en representación de números de punto flotante. Desde la aplicación se puede concluir de la siguiente manera. Se comprobó que el circuito propuesto puede ser implementado en las 3 familias de FPGAs de Xilinx debido a su nivel de utilización de recursos de ocupación. Una de las ventajas de implementar este multiplicador en FPGAs es que tiene procesos independientes al mismo tiempo y permite agilizar los cálculos, como la suma de

exponentes y la multiplicación de mantisas. La etapa de la multiplicación es la parte más crítica. La cual conlleva más conexiones a nivel hardware y depende mucho del algoritmo utilizado. Como trabajo futuro se podría implementar en hardware un algoritmo que permita hacer más veloz la etapa de multiplicación, como por ejemplo utilizar el árbol de Wallace, o un multiplicador de Booth, para resolver las operaciones.

8. Referencias

[1] Institute of Electrical and Electronics Engineers, Inc. IEEE Standard for Floating-Point Arithmetic. IEEE Xplore®. http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?tp=&arnumber=4610935. Abril 2015.

[2] N. Goyal, K. Gupta, R. Singla. Study of Combinational and Booth Multiplier. International Journal of Scientific and Research Publications. Vol. 4. No. 5. May 2014.

[3] R. Shankar, P. Gour, B. Bihari, "Design and Implements of Booth and Robertson's multipliers algorithm on FPGA". Soni International Journal of Engineering Research and Applications (IJERA).

[4] J. Rapallini, S. Ledesma, F. Costantino, J. R. Osio, Matemática de PuntoFlotante. http://www.edudevices.com.ar/download/articulos/buceando/BC_MCU_42_ED.pdf. Abril 2014.

[5] Consideraciones acerca del Estándar IEEE 754. http://www.led.uc.edu.py/micro2/tp2/pf/pag2.htm. Abril 2015.

[6] Xilinx Inc. Xilinx ISE 14.6 Software Manuals and Help. http://www.xilinx.com/support/index.html/content/xilinx/en/supportNav/ip_docume ntation.html.

[7] E. Vilches. Números de punto flotante. http://www.erikavilches.com/Anterior/TC1004.01.200811/diapositivas/Punto%20Flotante%202.pdf. Abril 2014.

[8] S. Orley, J. Mathews. IEEE 754 Format. http://www.oxfordmathcenter.com/drupal7/node/43. Abril 2015.

[9] M. Morris, Arquitectura de Computadoras. Tercera Edición. 1994. Pearson Educación. México. 547 pp.