Síntesis VLSI de un Multiplicador de Punto Flotante de Precisión Simple

Víctor Manuel Valenzuela De La Cruz

Universidad Autónoma de Ciudad Juárez, Instituto de Ingeniería y Tecnología, Av. Del Charro #450 Norte, C.P. 32310, Ciudad Juárez, Chihuahua, Tel: (656) 688 4841 ext. 4773

victorgh10@hotmail.com

Abimael Jiménez Pérez

Universidad Autónoma de Ciudad Juárez, Instituto de Ingeniería y Tecnología, Av. Del Charro #450 Norte, C.P. 32310, Ciudad Juárez, Chihuahua, Tel: (656) 688 4841 ext. 4773

abimael.jimenez@uacj.mx

Humberto de Jesús Ochoa Domínguez

Universidad Autónoma de Ciudad Juárez, Instituto de Ingeniería y Tecnología, Av. Del Charro #450 Norte, C.P. 32310, Ciudad Juárez, Chihuahua, Tel: (656) 688 4841 ext. 5412

hochoa@uacj.mx

Marco Antonio Gurrola Navarro

Universidad de Guadalajara, Centro Universitario de Ciencias Exactas e Ingenierías, Av. Revolución 1500, C.P. 44430, Guadalajara, Jalisco, Tel: (33) 13 78 59 00 ext. 27720

marco.gurrola@cucei.udg.mx

Resumen

La multiplicación es una de las operaciones más importantes para la ejecución de instrucciones en dispositivos de procesamiento de datos. En este trabajo se presenta el diseño de un multiplicador de punto flotante, siguiendo el estándar IEEE-754. El sistema se divide en tres fases, la primera separa los datos, la segunda realiza una multiplicación en punto fijo y la tercera lleva a cabo el cálculo del nuevo exponente. La segunda fase es crítica y se desarrolla mediante un algoritmo basado en celdas unitarias para generar una matriz de multiplicación. El sistema se implementó en VHDL (*VHSIC Hardware*

Description Language) con la herramienta ISE WebPack 14.4 de Xilinx. Posteriormente, se realizó parte del proceso de síntesis lógica y física, utilizando las herramientas EDA (*Electronic Design Automation*) de Alliance y se obtuvo una versión preliminar del *layout* para su fabricación en tecnología VLSI. El *layout* presentó un gran consumo de área, sin embargo, el diseño es escalable y se podría aumentar la capacidad del multiplicador sin necesidad de un rediseño. El sistema se comportó de manera satisfactoria en respuesta a diferentes patrones de prueba diseñados en las herramientas de Xilinx y Alliance.

Palabras Claves: IEEE-754, Multiplicación, Punto Flotante, VHDL, VLSI.

1. Introducción

La multiplicación es una operación aritmética fundamental, necesaria para realizar operaciones complejas, pero 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.

La notación en punto flotante es utilizada en sistemas digitales debido a su precisión y exactitud. Asimismo, la implementación en hardware es parte de la unidad aritmética-lógica (ALU) de un procesador. Sin embargo, es necesario optimizar para lograr un buen desempeño con poca área con un bajo consumo de potencia.

Al realizar una multiplicación en punto flotante, se requiere de una operación de multiplicación en punto fijo, considerada como la parte crítica y de mayor procesamiento. Para realizar esta operación, se han propuesto diferentes algoritmos [1-3, 9-11]. En [1] se utilizaron sumadores híbridos con entradas y salidas negadas, las cuales, al conectarse entre sí se complementan, pasando el dato sin alterar (doble negación). Esto permite eliminar componentes a nivel físico y reducir el consumo de potencia en las celdas principales.

Existen propuestas que recurren a otras arquitecturas como la presentada en [2], donde se utilizó una técnica que convierte los factores a su equivalente logarítmico, suma ambos

y obtiene el antilogaritmo de la suma. Obtener de esta manera el resultado de la multiplicación tiene la desventaja de que el logaritmo de un número no es exacto, por lo que utilizan el algoritmo Mitchell's para aproximar el resultado con N iteraciones al valor exacto de la operación.

Actualmente se están desarrollando investigaciones para determinar qué algoritmos son óptimos para realizar operaciones en punto flotante, no sólo en procesadores sino en el diseño de dispositivos ASIC (*Application Specific Integrated Circuit*). En [3] se presentan los efectos de utilizar diferentes grados de *pipeline* implementados en un multiplicador de punto flotante, diseñado con tres diferentes algoritmos, Radix-4 Booth, Árbol Wallace y suma en paralelo. En este trabajo se concluye que, en cierto grado de *pipeline*, la velocidad de operación no se incrementa, sin embargo, aumenta el hardware requerido para su implementación.

Existen diferentes algoritmos para llevar a cabo una multiplicación en punto flotante. Sin embargo, estos se centran en cómo realizar la multiplicación de punto fijo. En este trabajo, se presenta el diseño y simulación de un multiplicador de punto flotante de precisión simple, como propuesta para el desarrollo de un circuito integrado con tecnología VLSI y como módulo de propiedad intelectual, que podrá ser utilizado como componente en diferentes diseños.

También, se expone el diseño de un multiplicador de punto fijo, mediante un arreglo de celdas unitarias que permite multiplicar un par de bits e ir generando el resultado de la suma final y a su vez el acarreo generado o propagado. Con este método se logra una disminución tanto de los cálculos requeridos para su implementación como del tiempo de operación, lo cual puede traducirse en un diseño escalable en hardware.

En la sección de desarrollo se presentan algunos conceptos básicos sobre el funcionamiento de un multiplicador en punto flotante, así como el estándar IEEE-754. Adicionalmente, se definen los bloques del sistema, la implementación en VHDL y el flujo de diseño lógico y físico para la futura fabricación del circuito VLSI en algún proceso de *On Semiconductor*. En la sección de resultados, se presentan las simulaciones y en el apartado de discusión se hace una breve comparación con sistemas ideales.

2. Desarrollo

En esta sección se presentan los conceptos básicos que explican el comportamiento del multiplicador de punto flotante para conocer las condiciones adoptadas en este trabajo.

2.1. Notación de Punto Flotante Binaria

En ocasiones, la representación binaria de un número fraccionario no es exacta, se puede llegar a requerir de una gran cantidad de bits, lo que incrementa la longitud del número y el procesamiento para realizar operaciones aritméticas con dicho número. El punto flotante equivale a la notación científica en decimal, por lo que se pueden representar números utilizando una potencia, en este caso, de base dos [4]. Como se observa en la ecuación (1) un número en punto flotante se divide en tres partes: un signo (s), que determina si el valor es positivo o negativo, una mantisa (m), que contiene los bits del número a representar, y un exponente (exp), que indica en dónde se coloca el punto binario en relación al inicio de la mantisa.

$$N = (-1)^s \cdot 1.m \cdot 2^{exp} \tag{1}$$

2.2. Estándar IEEE-754

Para asegurar la compatibilidad entre diversos sistemas digitales, la representación de un número en punto flotante se establece con el estándar IEEE-754, que especifica el formato de los números en notación científica, las operaciones básicas, conversiones y condiciones de excepción [5-7]. La cantidad de bits,para cada partedel número, se presentan en la Tabla 1. Bajo el estándar,un número de punto flotante se define en la ecuación (2).

$$|N = (-1)^s \cdot (1.m) \cdot 2^{\{exp - BIAS\}} \tag{2}$$

donde, s corresponde al signo del número; *m*es la mantisa y siempre se asume el valor de '1' más la fracción (bit implícito), por lo que sólo se expresa como un número binario en donde cada posición hacia la derecha se convierte en una potencia de dos negativa. Por último, *exp*(exponente) expresa la potencia como un entero positivo entre 1 y 254.Al exponente original se le suma un valor conocido como BIAS,que desplaza el valor inicial, es decir, a exceso 127 (ej. un exponente igual a 5 equivale a $5 + 127 = 132$ en el formato IEEE-754)[8-11].

Tabla 1. Cantidad de bits en el estándar IEEE-754 para precisión simple y doble.

	Signo (s)	**Exponente** (exp)	**Mantisa** (m)	$BIAS$
Precisión Simple	1[31]	8[30-23]	23[22-0]	127
Precisión Doble	1[63]	11[62-52]	52[51-0]	1023

El formato IEEE-754, de precisión simple, cuenta con las siguientes características principales [8]:

- Un número en punto flotante debe *normalizarse* poniendo el bit a la izquierda del punto binario de la mantisa a uno ('1'), éste siempre está implícito.
- Un exponente cero, con una mantisa distinta de cero, representa un número no normalizado.
- Los números de punto flotante normalizados se representan con un exponente en el rango de 1 a 254.
- Un exponente de cero, en conjunto con una mantisa cero representa el cero positivo o negativo, dependiendo del bit de signo.
- Un exponente compuesto de unos representa el infinito positivo o negativo (*over flow*) dependiendo del bit de signo.
- La condición de *under flow* se presenta al desbordarse el exponte.

- Un exponente compuesto por unos, con una mantisa distinta de cero, se le conoce como *NaN*, que significa "*Not a Number*".

2.3. Multiplicación en Punto Flotante

La multiplicación de dos números en punto flotante se define en la ecuación (3), teniendo en cuenta que los valores a multiplicar están ya normalizados [12]:

$$z = x \times y = \left(s_x \oplus s_y\right) \cdot \left(man_x \times man_y\right) \cdot \left(2^{exp_x + exp_y - BIAS}\right)$$

$$= s_z \cdot 1.man_z \cdot 2^{exp_z} \quad (3)$$

La multiplicación de punto flotante se obtiene con el siguiente algoritmo básico:

- Obtener el signo, $s_x \; XOR \; s_y$
- Multiplicar las mantisas mediante una estructura multiplicadora de 24 bits.
- Sumar los exponentes, $exp_a + exp_b - BIAS = exp_a + exp_b - 127$
- Normalizar el resultado en caso de requerirlo.Al normalizar se desplaza la mantisa, lo cual implicasumar una unidad al exponente por cada posición desplazada.

2.4. Fases de la Multiplicación en Punto Flotante

El diseño del sistema se realizó con tres bloques distintos, los cuales fueron definidos en lenguaje VHDL y posteriormente sintetizados y simulados con los softwares Xilinx ISE WebPack 14.4 y EDA de Alliance.

A. Fase 1, separación de elementos

El bloque de operación "*FP_sep*", recibe, en sus entradas, los números de 32 bitsen formato del estándar IEEE-754. En esta fase, se separa cada componente del número en un bit de signo, veintitrés bits de mantisa y ocho bits del exponente. Posteriormente,

se determina si alguno de los números corresponde con la representación de cero. Después, se calcula el bit de signo resultante, se agrega el bit implícito a las mantisas y, por último, se envían los valores de cada parte de los números A [31:0] y B [31:0] a la salida, como se muestra en la Fig. 1.

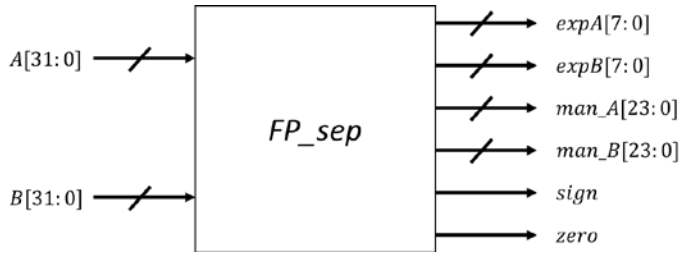

Fig. 1. Módulo "*FP_sep*" para separar componentes de un número flotante.

B. Fase 2, arreglo para multiplicación

El segundo bloque del sistema, "*Mult24*", lleva a cabo la multiplicación de las mantisas. El multiplicador sigue una estructura de matriz conformada por celdas fundamental es como se muestra en la Fig. 2. Cada celda multiplica un bit de la mantisa de A (A_{in}) y uno deB (B_{in}). Al mismo tiempo, se efectúa la suma de un bit previo (S_{in}) con el resultado de la multiplicación ($A_{in}\ AND\ B_{in}$) y un acarreo de entrada (C_{in}) para obtener el resultado de la suma (S_{out}) y el acarreo de salida (C_{out}). Este comportamiento se describe mediante las ecuaciones (4) y (5). Con este procedimiento, se puede realizar la multiplicación de dos bits y al mismo tiempo las sumas parciales de la multiplicación.

$$S_{out} = S_{in} \oplus (A_{in} AND B_{in}) \oplus C_{in} \tag{4}$$

$$C_{out} = \left(S_{in} \times (A_{in} \times B_{in})\right) + (S_{in} \times C_{in}) + ((A_{in} \times B_{in}) \times C_{in}) \tag{5}$$

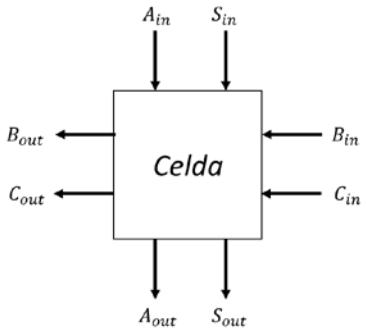

Fig. 2. Celda unitaria para multiplicación y suma.

Estas celdas se deben conectar en un arreglo lineal que permita propagar el acarreo generado por las sumas parciales y al mismo tiempo continuar multiplicando un bit del número B por cada uno de los bits del número A. Posteriormente, se requiere de la unión de varios arreglos lineales, dependiendo de la longitud de los números a multiplicar, como se muestra en la Fig. 3. No obstante, hay que considerar que cada celda debe completar su función para que la siguiente tenga un valor válido a su entrada.

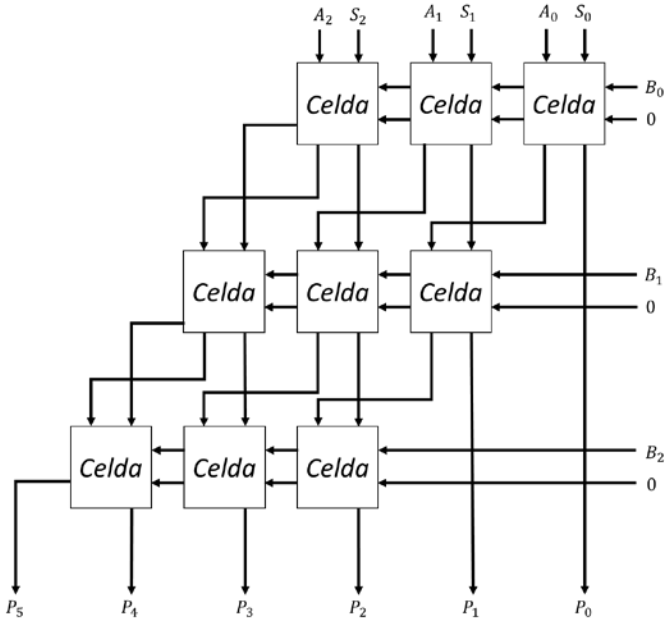

Fig. 3. Multiplicador tipo matriz de 3x3.

En la Fig. 4 se aprecia el componente utilizado en la fase dos del sistema, el bloque recibe ambas mantisas y un valor de suma inicial que comúnmente es cero. El bloque posee un arreglo como el presentado en la Fig. 3, pero con capacidad para 24 bits.

Fig. 4. Bloque correspondiente a un multiplicador de 24 bits.

C. Fase 3, suma de exponentes

El tercer bloque, presentado en la Fig. 5, determina el nuevo exponente y el resultado final, considerando los valores de los bloques anteriores. Este bloque recibe los 25 bits de mayor peso de la mantisa resultante, los bits de ambos exponentes a la entrada, el bit de signo y la indicación de cero.

Primero, se analiza la porción de mayor peso, obtenida de la multiplicación en la fase dos para determinar si la mantisa requiere un desplazamiento, lo cual implica un incremento en el nuevo exponente, también se elimina el bit implícito. Posteriormente, se toman los exponentes de los números de entrada y se suman para después restar el exceso a 127, obteniendo así el nuevo exponente.

Posteriormente, se comprueba que el exponente se encuentra dentro del rango permitido y se verifican condiciones de *overflow* y *underflow*. En seguida, se modifica el exponente, dependiendo del desplazamiento realizado a la mantisa, y se vuelven a verificar las condiciones ya mencionadas, éstas siguen las características principales del estándar IEEE-754 [8]. Finalmente, se presenta el resultado de la operación conformado por el nuevo signo, exponente y mantisa, si existe alguna excepción se presenta un valor determinado en base a la excepción.

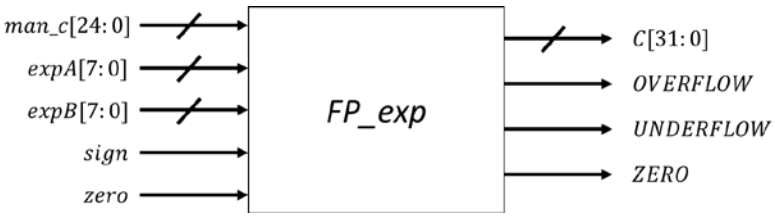

Fig. 5. Bloque para suma de exponentes y presentación de resultado final.

En la Fig. 6, se presenta el sistema completo conformado por los tres bloques ya presentados.

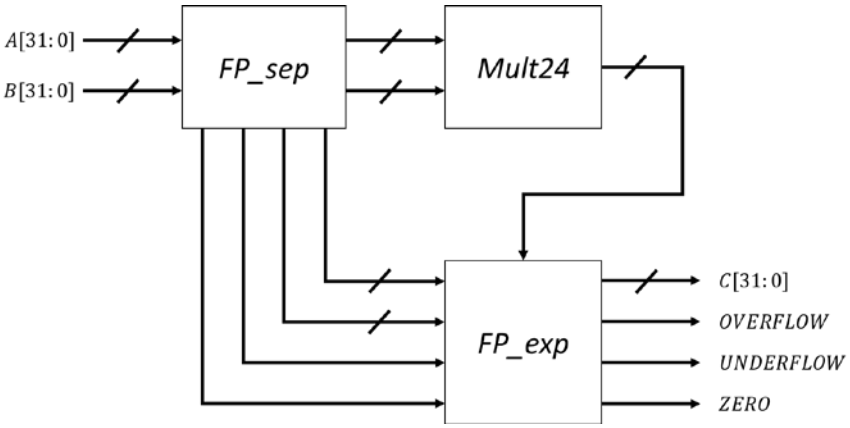

Fig. 6. Sistema completo de multiplicación para punto flotante.

2.5. Síntesis Lógica y Física utilizando herramientas de Alliance

Se sintetizó un multiplicador de 15 bits para las mantisas, en lugar de 24, esto debido a que se presentaron problemas en el proceso de síntesis para la versión de 24 bits. La síntesis lógica inicia con la traducción del comportamiento de cada bloque en lenguaje VHDL, mediante la herramienta VASY, a uno que puede ser interpretado por las herramientas de Alliance. Después, se realiza una simulación del comportamiento con la herramienta ASIMUT y un archivo de patrones de entrada. Posteriormente, la

herramienta BOOM utilizada minimiza las expresiones booleanas del sistema. La herramienta BOOG convierte la descripción de hardware a su equivalente estructural mediante celdas estándar para después reducir las capacitancias e introducir *buffers* con la herramienta LOON. En este punto, se conectan los diferentes bloques del sistema, utilizando la herramienta GENLIB, y se inicia la síntesis física con la herramienta OCP, la cual coloca las celdas estándar como se muestra en la Fig. 7y se interconectan con la herramienta de NERO. Finalmente, se obtiene el *layout* preliminar de la Fig. 8 para la fabricación VLSI en unidades genéricas conocidas como “lambdas”. Como parte del proceso de síntesis lógica y física en cada etapa se puede realizar una simulación, verificación o comparación para corroborar el proceso de síntesis y el comportamiento del sistema.

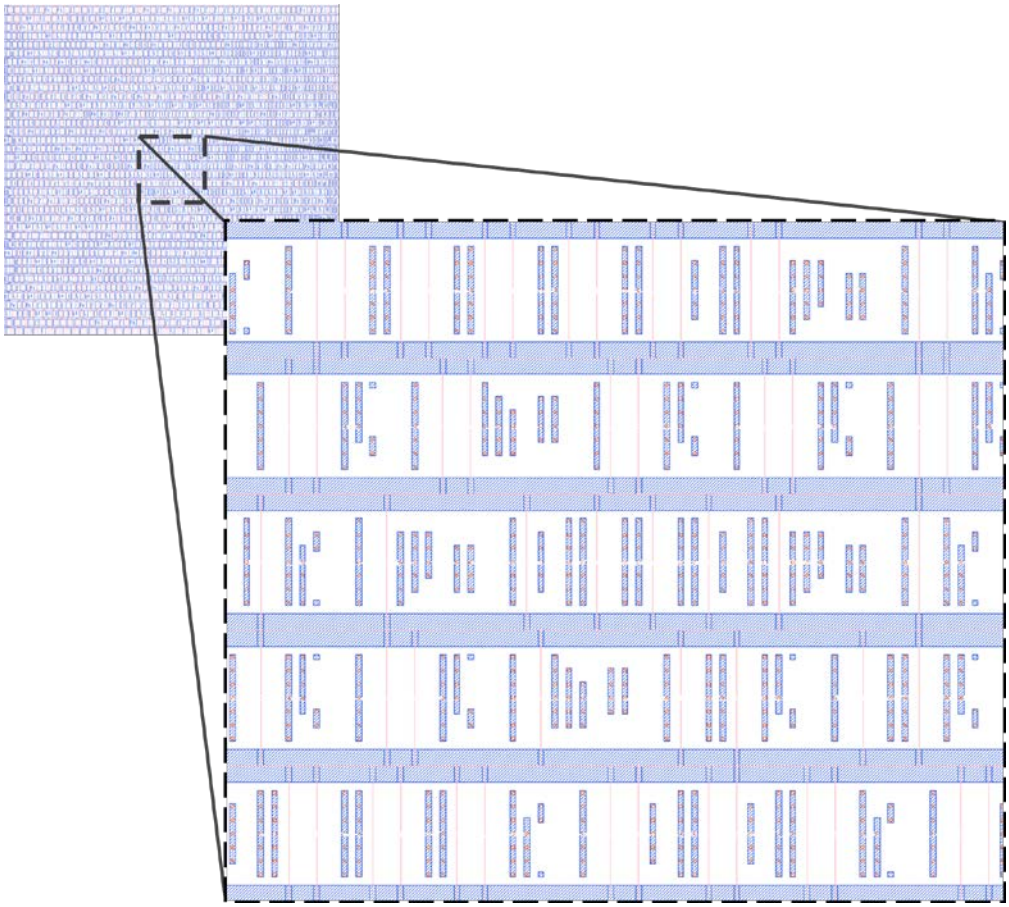

Fig. 7. *Layout* del multiplicador de punto flotante donde se presenta únicamente las celdas estándar posicionadas.

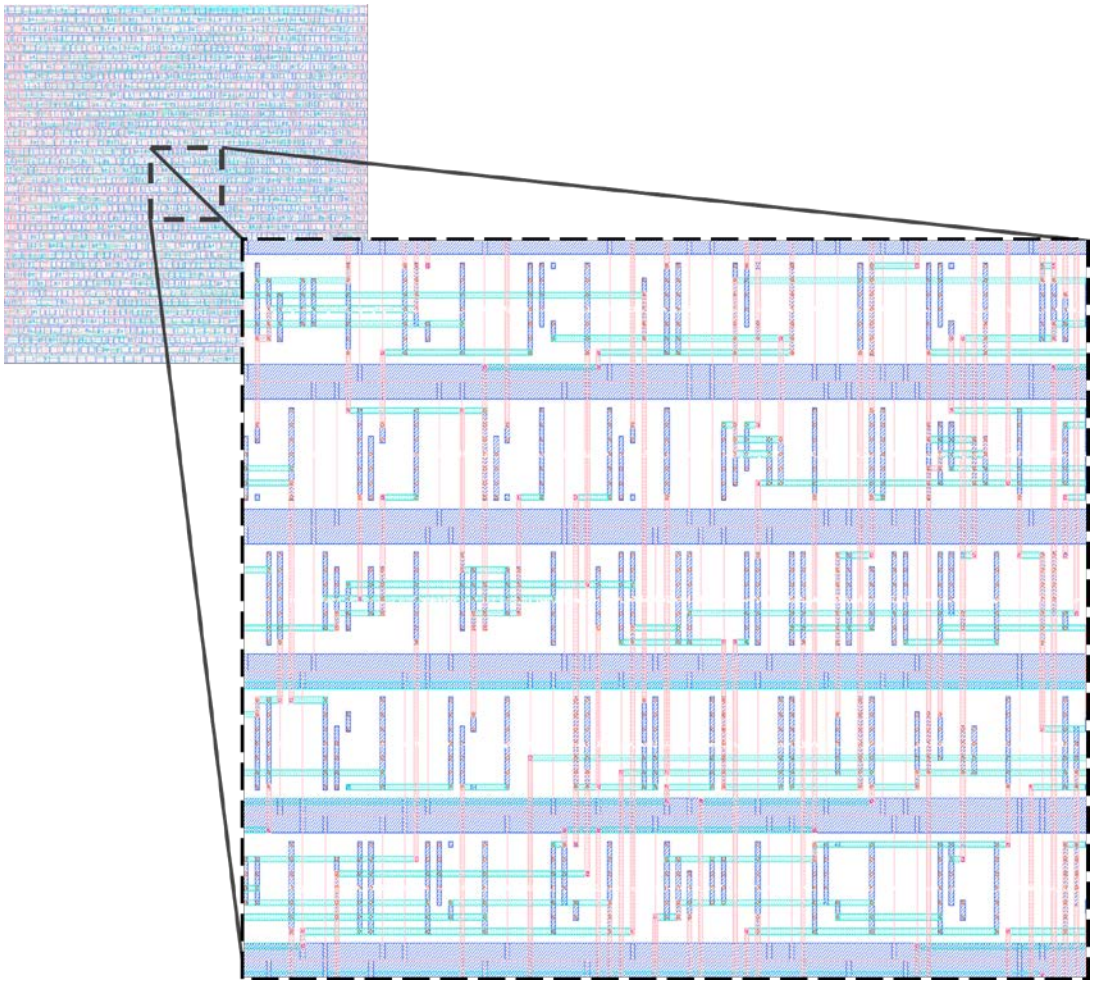

Fig. 8. *Layout* del multiplicador de punto flotante posicionado e interconectado, utilizando un multiplicador de mantisas de 15 bits.

3. Resultados

El diseño propuesto se verificó con simulaciones del sistema completo en el software Xilinx ISE WebPack 14.4 en donde se implementó el sistema en lenguaje VHDL. En la Fig. 9, se puede apreciar el caso de la multiplicación cuando ambos factores son mayores a uno y los exponentes son positivos. La conversión de estos valores a decimal se presenta en la Tabla 2.

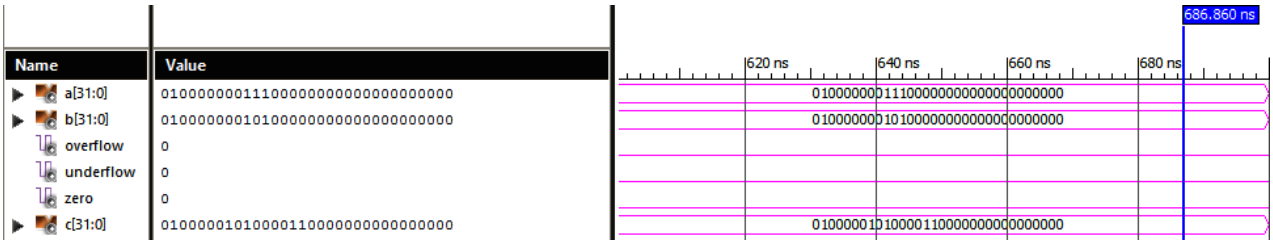

Fig. 9. Simulación con resultado dentro del rango representable.

Tabla 2. Valores del número A y B y el resultado de la multiplicación entre ellos.

Puerto	Valor Binario	Valor Decimal
$\boldsymbol{A}$	0 10000000 11100000000000000000000	$1.875 * 2^{128-127} = 3.75$
$\boldsymbol{B}$	0 10000000 10100000000000000000000	$1.625 * 2^{128-127} = 3.25$
$\boldsymbol{C}$	0 10000010 10000110000000000000000	$1.5234375 * 2^{130-127} = 12.1875$

En la Fig. 10, los valores de entrada dan una condición de *underflow* como resultado. La Tabla 3 muestra las entradas en decimal.

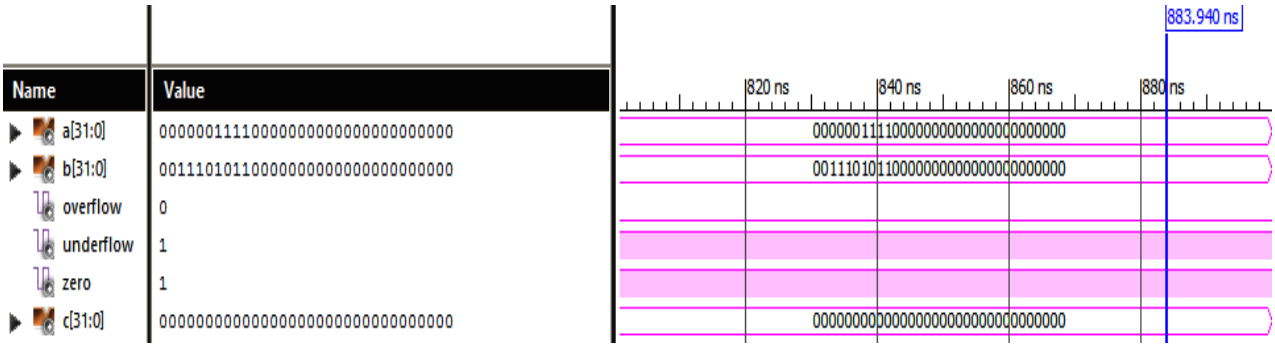

Fig. 10. Excepción por*under flow.*

Tabla 3. Valores de entrada utilizando exponentes menores a cero.

Puerto	Valor Binario	Valor Decimal
A	0 00000111 10000000000000000000000	$1.5 * 2^{7-127} = 1.12 * 10^{-36}$
B	0 01110101 10000000000000000000000	$1.5 * 2^{117-127} = 1.46 * 10^{-3}$
C	0 00000000 00000000000000000000000	*underflow*

Igualmente, en la Fig. 11, se muestra un caso donde los factores al ser multiplicados dan como resultado una excepción considerada como *overflow*. Los números dados como entradas en este caso se presentan en la Tabla 4 en sistema decimal.

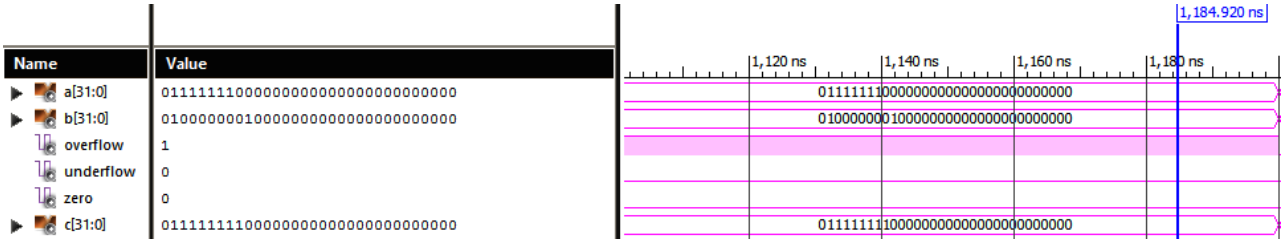

Fig. 11. Excepción por *overflow.*

Puerto	Valor Binario	Valor Decimal
A	0 11111110 00000000000000000000000	$1 * 2^{254-127} = 1.70 * 10^{38}$
B	0 10000000 10000000000000000000000	$1.5 * 2^{128-127} = 3$
C	0 11111111 00000000000000000000000	*overflow*

Tabla 4. Datos de entrada con exponentes mayores a cero.

En el caso de la síntesis del sistema, los resultados de la simulación con las herramientas EDA de Alliance son idénticos, en comparación con los conseguidos con ISE WebPack 14.4. Por motivos de espacio y resolución de gráficos los resultados obtenidos con las

herramientas de Alliance no se presentan. Sin embargo, se puede hacer mención de los parámetros o resultados obtenidos en la síntesis lógica y física de este sistema. Estos datos se obtienen al final de la ejecución de la herramienta LOON. Los parámetros de mayor impacto para el análisis son los siguientes:

- Dimensiones: $1{,}900 \times 1{,}930$ lamdas
- Peor ruta crítica: $1{,}236\ \rho s$
- Cantidad de transistores: $14{,}931$ MOSFETs
- Cantidad compuertas: Aproximadamente $3{,}733$ compuertas
- Tiempo de ejecución del sistema: Aproximadamente $37{,}827\ \rho s$

Es importante mencionar que la cantidad de celdas estándar no se considera como un dato utilizable para cuestiones de comparación de arquitecturas, si bien todas las celdas estándar son de un mismo alto, no lo son en ancho. No obstante, la cantidad de transistores puede ser utilizada como medida de comparación, y a partir de ésta se puede obtener un valor de compuertas simbólico para el mismo propósito.

4. Discusión

Los resultados obtenidos muestran un comportamiento congruente al ser comparados con multiplicadores de punto flotante considerados como ideales. De las tres fases de operación, la fase dos es considerada 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 24x24 bits que en términos de hardware se traduce en una gran cantidad de componentes dependiendo del algoritmo utilizado. Un multiplicador en matriz es un sistema con consumo de área mayor en comparación con multiplicadores que utilizan algoritmos como Radix-4 Bootho Árbol de Wallace. Sin embargo, es de fácil implementación ya que no requiere diferentes subsistemas para conformarse, únicamente necesita de una misma celda que puede reproducirse para aumentar la capacidad del sistema, evitando un rediseño, lo cual ocurriría al utilizar otro algoritmo.

En cuestión de velocidad, el principal problema de los multiplicadores es la latencia ya que el sistema es afectado por el módulo correspondiente a la multiplicación de mantisas debido a la cantidad de operaciones y rutas críticas que deben cumplirse para considerar válido el valor de salida del módulo. Este comportamiento no es exclusivo del algoritmo de multiplicación de matriz pues también se presenta al utilizar otros algoritmos de multiplicación. No obstante, este diseño de matriz implementa dos operaciones a la vez en una misma celda, por lo tanto se puede multiplicar e ir sumando cada producto parcial, a diferencia de otros algoritmos que primero efectúan la multiplicación de todos los bits involucrados para después realizar la suma en paralelo o comprimir los datos para realizar una sola suma.

Los resultados obtenidos fueron corroborados con los diferentes casos seleccionados de la multiplicación de dos números con las herramientas de simulación de Xilinx y Alliance. 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 [8]. Por lo tanto, este sistema es completamente funcional comparado con los propuestos en la literatura actual, ya sea implementado en FPGA o como componente de librerías profesionales.

El multiplicador de matriz de 24 bits no pudo ser sintetizado a causa de la cantidad de componentes que requería. No obstante, se optó por reducir el número de bits de 24 a 15, lo cual redujo la cantidad de elementos a conectar, permitiendo que la síntesis lógica y física se finalizara correctamente.

5. Conclusiones

El multiplicador de punto flotante siguiendo el estándar IEEE-754 fue diseñado mediante un algoritmo de matriz para el multiplicador de mantisas, bloques de sumadores predeterminados para la suma de exponentes y decisiones simples para las excepciones presentes en los resultados. El diseño se realizó en el software Xilinx ISE WebPack 14.4

y se realizó la síntesis lógica y física con las herramientas EDA de Alliance. El sistema simulado mostró un comportamiento estable y predecible ante diferentes estímulos de prueba, dando resultados correctos.

Como resultado de la síntesis física se observa que el multiplicador diseñado requiere de un área extensa, lo cual podría considerarse como la principal desventaja dado que la finalidad es llevar el diseño a un ASIC. Evidentemente el diseño de un multiplicador en matriz requiere de cierta cantidad de componentes que pueden reducirse si se utiliza un algoritmo adecuado para la multiplicación de mantisas. La síntesis física del multiplicador de matriz con un menor número de bits se realizó satisfactoriamente. Al utilizar herramientas no profesionales de síntesis lógica y física, es posible que el sistema propuesto no pueda ser sintetizado para su fabricación VLSI.

Como trabajo futuro se plantea agregar etapas de *pipeline* para incrementar la velocidad de procesamiento de datos por parte del multiplicador de mantisas, así como estudiar la posibilidad de utilizar las herramientas de Alliance en el proceso de síntesis para el caso de la multiplicación de mantisas de 24 bits. Esto nos permitirá una síntesis completa tanto lógica como física para fabricar el circuito integrado del multiplicador en alguno de los procesos de *ON Semiconductor*.

6. Referencias

[1] Z. Abid, H. El-Razouk, D. El-Dib, “Low power multipliers based on new hybrid full adders”.Microelectronics Journal. Vol. 39. No. 12.Junio 2008. pp. 1509-1515.

[2] Z. Babic, A. Avramovic, P. Bulic, “An iterative logarithmic multiplier”. Microprocessors and Microsystems. Vol. 35. No 1.Julio 2011. pp. 23-33.

[3] X. Jiang, P. Xiao, M. Qiu, G. Wang, “Performance effects of pipeline architecture on an FPGA-based binary32 floating point multiplier”. Microprocessors and Microsystems. Vol. 37. No. 8. Septiembre 2013. pp. 1183-1191.

[4] M. Borgwardt. Números de punto flotante. <http://puntoflotante.org/formats/fp>. Marzo 2014.

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

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

[7] S. Orley, J. Mathews. IEEE 754 Format. <http://www.oxfordmathcenter.com/drupal7/node/43>. Marzo 2014.

[8] Consideraciones acerca del Estándar IEEE 754 <http://www.led.uc.edu.py/micro2/tp2/pf/pag2.htm>. Marzo 2014.

[9] G. Ushasree, R. Dhanabal, S. K.Sahoo, “Implementation of a High Speed Single Precision Floating Point Unit using Verilog”. International Journal of Computer Applications, Abril 2013, pp. 803-808.

[10] P. R. Addanki, V. N. Tilak Alapati, M. P. Avana, “An FPGA Based High Speed IEEE - 754 Double Precision Floating Point Adder/Subtractor and Multiplier Using Verilog”. International Journal of Advanced Science and Technology. Vol. 52.No. 1,Marzo 2013. pp. 61-74.

[11] R. Saini, R. D. Daruwala, “Efficient Implementation of Pipelined Double Precision Floating Point Multiplier”. International Journal of Engineering Research and Applications. Vol. 3. No. 1, Enero - Febrero 2013. pp. 1676-1679.

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

7. Autores

Víctor Manuel Valenzuela De La Cruz. Es alumno activo de la Universidad Autónoma de Ciudad Juárez en el Instituto de Ingeniería y Tecnología, se encuentra próximo a obtener su grado como Ingeniero en Sistemas Digitales y Comunicaciones con orientación al Diseño de Sistemas Digitales Integrados. Ha participado en diferentes proyectos de diseño y síntesis de circuitos digitales utilizando lenguajes de descripción de hardware como VHDL y simulaciones SPICE.

Dr. Abimael Jiménez Pérez. Obtuvo su grado de Doctor en Ciencias en Electrónica por el Instituto Nacional de Astrofísica, Óptica y Electrónica en 2008. Se desempeñó como profesor investigador en el Centro Universitario de la Costa Sur de la Universidad de Guadalajara de 2008 a 2009. De 2010 a la fecha se desempeña como profesor Investigador en el Instituto de Ingeniería y Tecnología de la Universidad Autónoma de Ciudad Juárez. Ha participado en diferentes proyectos de investigación en el área de modelado y simulación de dispositivos semiconductores y diseño de circuitos integrados digitales y FPGAs.

Dr. Humberto de Jesús Ochoa Domínguez. Recibió su grado de Ingeniero Electrónico Industrial por el Instituto Tecnológico de Veracruz, México, el grado Maestro en Ciencias en Electrónica por el Instituto Tecnológico de Chihuahua, México y el grado Doctor en Ciencias en Ingeniería Eléctrica por la Universidad de Texas en Arlington, USA. Actualmente se desempeña en el departamento de Ingeniería Eléctrica y Computación en la Universidad Autónoma de Ciudad Juárez, Chihuahua, México.

En 1998 recibió el premio en Chihuahua por el proyecto “Classification of Digital Mammograms into Normal and Abnormal through the Texture Analysis and Microcalcifications Detection System”.En 2003 se integró al Centro de Investigación Nokia en Irvin, TeSxas. Posee dos patentes. Sus áreas de interés en docencia e investigación incluyen sistemas multirate para análisis de imágenes médicas, restauración y reconstrucción de imágenes, codificación de imagen y video, procesamiento de señales estadístico y reconocimiento de patrones.

Dr. Marco Antonio Gurrola Navarro. M.A. Es originario del estado de Jalisco, México. El Dr. Gurrola está adscrito al Dpto. de Electrónica, CUCEI, Universidad de Guadalajara, recibió el grado de Ingeniero en Comunicaciones y Electrónica (1997), y el grado de Maestro en Ciencias de la Tierra (2003) por la Universidad de Guadalajara, México, y el grado de Doctor en Ciencias, con especialidad en Diseño de Circuitos Integrados, por el Instituto Nacional de Astrofísica Óptica y Electrónica, en Tonantzintla, México en 2009.

Desde 2009 el Dr. Gurrola ha estado trabajando en la Universidad de Guadalajara y su área de interés actual son la teoría de circuitos para implementaciones analógicas de la transformada wavelet y los desarrollos tecnológicos de circuitos integrados SoC.