DISEÑO E IMPLEMENTACIÓN DE UNA UNIDAD DE FUNCIONES MATEMÁTICAS

José Alberto Díaz García, Eugenio Salazar Brenes

Escuela de Ingeniería Electrónica / Instituto Tecnológico de Costa Rica
www.ie.itcr.ac.cr

jdiaz@itcr.ac.cr; esalazar@ietec.org

SUMARIO

Este artículo presenta una unidad de funciones matemáticas: logaritmos, antilogaritmos, multiplicaciones y divisiones, potencias y raíces, se incluye su implementación en una FPGA. La unidad fue implementada usando lógica combinacional solamente, utilizando las aproximaciones de Mitchell para logaritmos y antilogaritmos. Se utilizo la FPGA XC3S1000 de la familia Spartan™ 3 de Xilinx™.

1. INTRODUCIÓN

El propósito de este artículo es describir el diseño e implementación en una FPGA de una unidad de funciones matemáticas: logaritmos, antilogaritmos, multiplicaciones, divisiones, potencias y raíces. La unidad esta basada en las aproximaciones para logaritmos y antilogaritmos desarrolladas por Mitchell [1].

Aunque, es simple y rápido generar logaritmos y antilogaritmos; su uso se restringe a los usos donde un error pequeño puede ser tolerado, por ejemplo el proceso de la señal numérica [2]. Para aumentar la exactitud se hizo necesario realizar estimaciones y ajustes a los modelos. La unidad esta diseñada para ser implementada en el procesador presentado en [3], uno de los objetivos de este último es su implementación como un ASIC, por tanto no se utilizo ninguno de los módulos dedicados de la FPGA.

2. APROXIMACIONES PARA LOGARITMOS Y ANTILOGARITMOS

El método desarrollado por Mitchell para logaritmos y antilogaritmos binarios siguió el siguiente análisis:

Siendo N un número binario en el intervalo de $[2^j : 2^{k+1}[$, k y j números enteros y $k \geq j$. Por tanto N podría ser representado por:

$$N = \sum_{i=j}^{k} 2^i z_i, \tag{1}$$

donde z_i puede ser '0' ó '1'. Asumiendo $z_k = 1$, N puede ser expresado por:

$$N = 2^k + \sum_{i=j}^{k-1} 2^i z_i \tag{2a}$$

$$N = 2^k \left(1 + \sum_{i=j}^{k-1} 2^{i-k} z_i \right) \tag{2b}$$

Definiendo el término m como:

$$m = \sum_{i=j}^{k-1} 2^{i-k} z_i \tag{3}$$

Por tanto, (2b) puede expresarse como:

$$N = 2^k (1 + m) \tag{4}$$

Aplicando logaritmo a (4):

$$\log_2 N = k + \log_2 (1 + m) \tag{5}$$

El termino $\log_2(1+m)$ estará en el intervalo [0:1[, por tanto representa la fracción de $\log_2 N$. Mitchell realizó la aproximación:

$$\log_2 N = k + m \tag{6}$$

Esto produce diferencia en la parte fraccionaria igual a:

$$dif = \log_2 (1 + m) - m \tag{7}$$

El método propuesto por Mitchell para el antilogaritmo fue:

$$2^M = anti \log_2 (M) \tag{8a}$$

$$2^M = anti \log_2 (k + m) = 2^k 2^m \tag{8b}$$

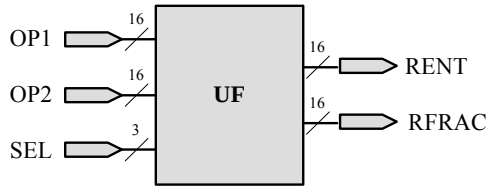

Fig. 1. Unidad de Funciones.

Tabla 1. Descripción de las funciones.

SEL	Descripción
000	Logaritmo en base 2 de OP1
001	Antilogaritmo en base 2 de OP1, OP2
010	Multiplicación de OP1 por OP 2
011	División de OP1 por OP2
100	OP1 elevado a la potencia 2
101	Raíz cuadrada de OP1

Tomando la aproximación de Mitchell 2^m a (1+m), por tanto:

$$anti\log_2(M)' = 2^k(1+m) \qquad (9)$$

Esto produce diferencia en la parte fraccionaria igual a:

$$dif = 2^k\left(2^m - (1+m)\right) \qquad (10)$$

3. UNIDAD DE FUNCIONES (UF)

Esta unidad se diseñó para realizar las siguientes funciones matemáticas: logaritmo, antilogaritmo, multiplicación, división, potencia y raíz, a partir de las aproximaciones de Mitchell para el logaritmo y antilogaritmo. En la **Fig. 1.** se muestra, en alto nivel, el UF. Recibe 2 operandos de 16 bits, 3 bits para función, tiene dos salidas, una para la parte entera y la otra para la fraccional, ambas de 16 bits, tamaños definidos para la compatibilidad en la implementación en el procesador de [3]. En la **Tabla 1.** se describen la funciones de la UF.

3.1. Logaritmo

El circuito propuesto se muestra en la **Fig. 2**, deducido a partir de (6). Para aumentar la exactitud del resultado se realiza un suma con constantes (véase **Tabla 2)** localizadas en la ROM. En la **Fig. 3** se puede observar la disminución del error con sólo agregar la ROM, en los casos más críticos se redujo a menos del 0,01%

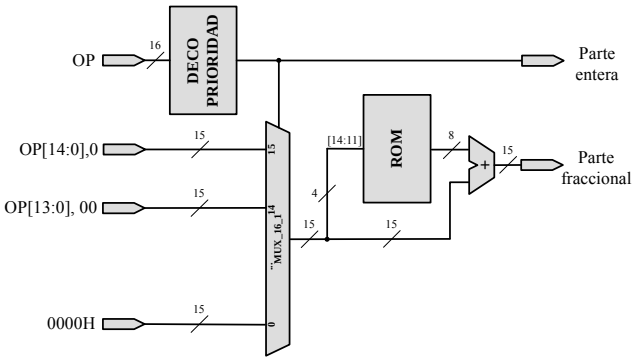

Fig. 2. Circuito propuesto para el logaritmo.

Tabla 2. Constantes a agregar para minimizar el error del logaritmo.

Rango	Constante	
	Decimal	Binario
[0.0000:0.0625[	0,00000000	0.00000000000
[0,0625:0,1250[	0,02490234	0.00000110011
[0,1250:0,1875[	0,04492188	0.00001011100
[0,1875:0,2500[	0,06005859	0.00001111011
[0,2500:0,3125[	0,07177734	0.00010010011
[0,3125:0,3750[	0,07958984	0.00010100011
[0,3750:0,4375[	0,08398438	0.00010101100
[0,4375:0,5000[	0,08593750	0.00010110000
[0,5000:0,5625[	0,08496094	0.00010101110
[0,5625:0,6250[	0,08105469	0.00010100110
[0,6250:0,6875[	0,07519531	0.00010011010
[0,6875:0,7500[	0,06738281	0.00010001010
[0,7500:0,8125[	0,05712891	0.00001110101
[0,8125:0,8725[	0,04541016	0.00001011101
[0,8750:0,9375[	0,03173828	0.00001000001
[0,9375:1,0000[	0,01660156	0.00000100010

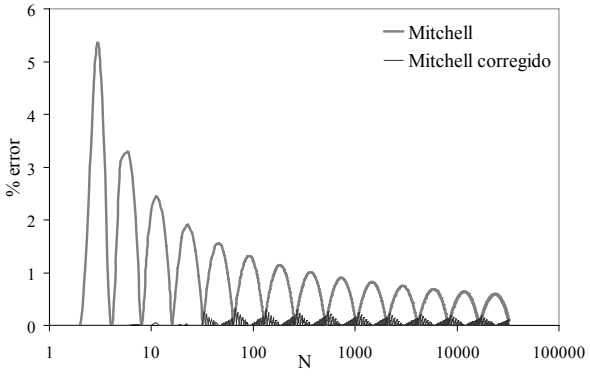

Fig. 3. Porcentaje de error para ambas aproximaciones del logaritmo.

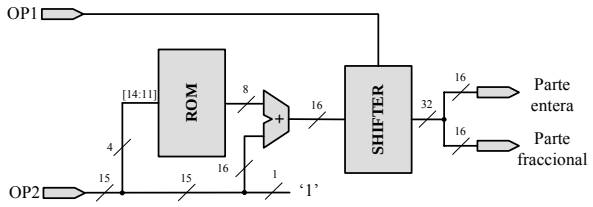

Fig. 4. Circuito propuesto para el antilogaritmo.

Tabla 3. Constantes a agregar para minimizar el error del logaritmo.

Rango	Constante
[0.0000:0.0625[	0,00927319
[0,0625:0,1250[	0,02651314
[0,1250:0,1875[	0,04177583
[0,1875:0,2500[	0,05494380
[0,2500:0,3125[	0,06591549
[0,3125:0,3750[	0,07441675
[0,3750:0,4375[	0,08076754
[0,4375:0,5000[	0,08473393
[0,5000:0,5625[	0,08595609
[0,5625:0,6250[	0,08446525
[0,6250:0,6875[	0,08014129
[0,6875:0,7500[	0,07285878
[0,7500:0,8125[	0,06248674
[0,8125:0,8725[	0,04888837
[0,8750:0,9375[	0,03192085
[0,9375:1,0000[	0,01145332

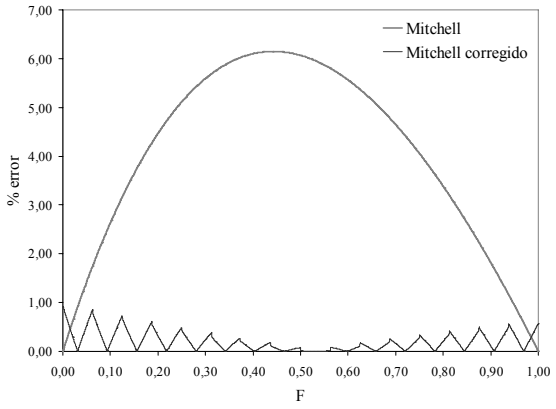

Fig. 5. Porcentaje de error para ambas aproximaciones del antilogaritmo.

3.2. Antilogaritmo

El circuito propuesto se muestra en la **Fig. 4**, obtenido a partir de (9). Para aumentar la exactitud del resultado se realiza un suma con constantes (véase **Tabla 3)** localizadas en la ROM. En la **Fig. 5** se puede observar la disminución del error al solo agregar la ROM, se redujo de un promedio del 4,07 % a un de 0,19%.

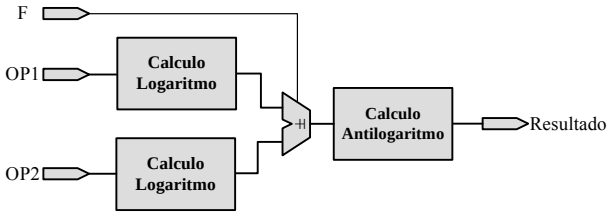

Fig. 6. Circuito propuesto para la multiplicación y división.

3.3. Multiplicación, división, potencia y raiz

Para estas operaciones, considerando A y B números enteros, se realizó el modelado a partir de las siguientes ecuaciones (para (12) y (13) se utilizó una aproximación presentada en [4]):

$$A \times B = anti\log_2\left(\log_2 A + \log_2 B\right) \quad (10)$$

$$A \div B = anti\log_2\left(\log_2 A - \log_2 B\right) \quad (11)$$

$$\sqrt{A} = \log_2 A,\ A >> 1 \quad (12)$$

$$A^2 = \log_2 A,\ A << 1 \quad (13)$$

En la **Fig. 6** se muestra el circuito propuesto para la multiplicación (F = 0) y división (F = 1), modelos similares se han desarrollado en [5], [6] y [7].

4. IMPLEMENTACIÓN

Para la implementación de la unidad se utilizó Verilog HDL, para el proceso de síntesis y simulación se utilizaron las herramientas: ISE™ V8.2 de Xilinx™ y ModelSim™ SE V6.0d de Mentor Graphics™ respectivamente.

La descarga se realizo en la FPGA XC3S1000 de la familia Spartan™ 3 de Xilinx™. La implementación utilizó 4% de los slices disponibles con un retardo máximo de 37.842ns.

5. CONCLUSIONES

Se presentó una unidad capaz de desarrollar funciones matemáticas elementales: logaritmos, antilogaritmos, multiplicaciones, divisiones, potencias y raíces. Sólo se utilizó lógica combinacional, para ello se utilizaron las aproximaciones de Mitchell para logaritmos y antilogaritmos, para aumentar la exactitud se realizaron estimaciones las cuales redujeron los porcentajes de error prácticamente por debajo del 0.2%. La unidad se implementó en la FPGA XC3S1000 de la familia Spartan™ 3 de Xilinx™ alcanzando retardos máximos de 37.842ns, utilizando 4% de los slices.

6. REFERENCIAS

[1] J. Mitchell, Jr., "Computer multiplication and division using binary logarithms," *IRE Transactions on Electronic Computers*,pp. 512-517, Aug. 1962.

[2] K. Abed, R. Siferd, “CMOS VLSI implementation of a low-power logarithmic converter,” *IEEE Transactions on Computers*,pp. 1421-1433, Nov. 2003.

[3] J. Díaz, E. Salazar, L. Quiros, “Diseño de un Filtro Digital (IIR) con Microprocesador de Arquitectura Multiciclo en FPGA” *Escuela de Ingeniería Electrónica, Instituto Tecnológico de Costa Rica*, March 2006.

[4] E.Chester, J.Coleman, "Development of a High-Speed 32b Real Arithmetic Core for DSP and Graphics Applications using Logarithmic Number System Techniques." *Proceedings IEEE/IoP PREP99 Conference*, Jan 1999.

[5] D. McLaren, “Improved Mitchell-Based Logarithmic Multiplier for Low-power DSP Applications,” *Proceedings. IEEE International , SOC Conference*, pp53-56, Sept. 2003.

[6] K. Abed, R. Siferd, “CMOS VLSI Implementation of 16-bitlogarithm and anti-logarithm converters,” *IEEE Midwest Symposium on Circuits and Systems*, Aug. 2000.

[7] S Ramaswamy, R. Siferd, “CMOS VLSI implementation of a digital logarithmic multiplier”, *Proceedings of the IEEE National Aerospace and Electronics Conference*, vol 1, pp 291-294, May 1996.