RIELAC, Vol.XXXV 2/2014 p.35-44 Enero - Abril ISSN: 1815-5928

RIELAC
Revista de Ingeniería Electrónica, Automática y Comunicaciones

Implementación de la Función Sigmoidal Logarítmica en un FPGA

José Abiel Caballero Hernández, Martha Díaz Salazar, Meyli Moradillos Paz-Lago, Sonnia Pavoni Oliver

RESUMEN / *ABSTRACT*

La función sigmoidal logarítmica es muy utilizada como función de activación de las neuronas que conforman una red neuronal artificial. En la implementación digital sobre FPGA de este tipo de redes, es importante la eficiencia en área de todos los elementos de cálculo. La síntesis *hardware* directa de la expresión matemática de la función de activación sigmoidal logarítmica no es práctica, ya que tanto las operaciones de división como la exponencial requieren lógica excesiva y convergen lentamente. En consecuencia, se han desarrollado aproximaciones matemáticas que facilitan esta implementación. En el presente trabajo se presenta la síntesis de la función sigmoidal logarítmica para un FPGA Spartan 3 de Xilinx con métodos de aproximación por tramos de primer orden, de aproximación por tramos de segundo orden y mediante el empleo de tablas de búsqueda. Para cada diseño se utilizaron las herramientas Simulink de MatLab y System Generator del programa Xilinx ISE Design Suit 12.4, a nivel de bloques específicos de Xilinx. Como resultado se obtuvo la mejor combinación entre los errores de implementación y el consumo de recursos del FPGA con una aproximación por tramos de primer orden y el empleo de datos de entrada con formato punto fijo de 12 bits de parte entera y 8 bits de resolución.

Palabras claves: Función sigmoidal logarítmica, FPGA, System Generator.

The logarithmic sigmoid function is the most commonly used activation function of the neurons forming an artificial neural network. Digital implementation of neural networks on FPGA must be efficient, especially in the elements estimation area. A direct hardware synthesis of the mathematics expression of logarithmic sigmoid activation function is not practical, because division and exponential estimation are demanding operations, which require excessive logic and convergence is slow. Therefore, some mathematics approximations have been developed to make easier this implementation. This study shows the synthesis of a logarithmic sigmoid function for a Xilinx´s FPGA Spartan-3 utilizing a piecewise first-order linear approximation method, a piecewise second-order linear approximation method, and a look-up table method. For each design was used MatLab's Simulink tool and the System Generator tool from Xilinx ISE Design Suit 12.4 program, at a Xilinx´s specifics blocks level. It is concluded that the best performance is achieved by the piecewise first-order linear approximation with input data format fixed point with 12 bits of integer bits and 8 bits of fractional bits.

Key words: *logarithmic sigmoid function, FPGA, System Generator.*

Implementation of the logarithmic sigmoid function on a FPGA.

INTRODUCCION

La implementación circuital de funciones matemáticas tiene aplicación en diversas áreas de trabajo. En particular, la función sigmoidal es muy utilizada en el desarrollo de redes neuronales artificiales, específicamente en la función de activación de las neuronas que conforman la red[1-3].

Desde el punto de vista electrónico, las redes neuronales artificiales pueden ser implementadas con tecnología analógica o digital. Aunque las soluciones analógicas tienen ventajas en cuanto a la alta densidad y rápida operación, son muy sensibles al ruido, al efecto de la temperatura y a las variaciones de suministro de potencia[4]. La tecnología digital, por su parte, ofrece mayores posibilidades en cuanto a flexibilidad, aprendizaje, tamaño expandible y precisión. Debido a la utilización de la función sigmoidal como función de activación en redes neuronales y al paralelismo de estas redes, una de las formas más utilizado es su implementación sobre *hardware* reconfigurable. Particularmente, la síntesis en FPGA (*Field Programmable Gate Arrays*, arreglos de compuertas programables por campo) es muy atractiva debido a la alta flexibilidad que se alcanza como consecuencia de la naturaleza reprogramable de estos circuitos[4].

Los recursos digitales necesarios para sintetizar la función sigmoidal sobre un FPGA dependen del método o aproximación matemática que se emplee. En este sentido se han reportado diversas alternativas aunque ninguna se ha convertido en una solución universal. Los principales reportes se basan en el uso de tablas de búsqueda y en el truncamiento de la serie de Taylor[5]. Esta última variante se puede dividir en aproximaciones lineales por tramos, aproximaciones por tramos de segundo orden y mapeo combinacional de entrada/salida[5]. Además, existe una variación considerable dentro de cada categoría. Por ejemplo, se reporta una técnica de compresión denominada A-Law[6] y una aproximación por tramos sin multiplicadores [7]. Existen también generadores de funciones elementales capaces de desarrollar múltiples funciones de activación que utilizan aproximaciones polinómicas de primer y segundo orden [8]. Algunos métodos utilizan bloques de memoria para su implementación, otros necesitan multiplicadores de punto fijo y otras aproximaciones más exigentes, demandan del uso de multiplicadores de punto flotante. Existen además métodos, donde la aproximación hecha sustituye el uso de multiplicadores por registros de desplazamiento, con lo que se logra un menor consumo de recursos. Entre los indicadores que permiten evaluar el éxito de una implementación hardware están la precisión alcanzada, la velocidad máxima de trabajo (frecuencia máxima de la señal de reloj) y la cantidad de recursos empleados en el diseño[9, 10].

Este artículo tiene como objetivo presentar tres diseños de la función sigmoidal logarítmica y su implementación en un FPGA Spartan 3 de Xilinx. Se utilizaron aproximaciones matemáticas por tramos de primer orden PLAN (*Piecewise Linear Approximation of a Nonlinear Function*), por tramos de segundo orden (Zhang), así como el método de tablas de búsqueda. Con la aproximación PLAN y el empleo de datos de entrada con formato punto fijo de 12 bits de parte entera y 8 bits de resolución, se obtuvo la mejor combinación entre los errores de implementación y el consumo de recursos del FPGA. Esta solución se recomienda para la implementación de la función de activación sigmoidal en el diseño de redes neuronales artificiales.

APROXIMACIONES MATEMÁTICAS DE LA FUNCIÓN SIGMOIDAL

La función sigmoidal puede ser logarítmica o tangente hiperbólica en dependencia de si corta o no el eje de las abscisas. A estos dos casos, cuyas características se resumen en la Tabla 1, también se les conoce en la literatura como unipolar y bipolar, respectivamente.

Algunos métodos de aproximación de la función sigmoidal reportados en la literatura, se muestran en la Tabla 2. Los valores de error promedio ($E_{promedio}$) y error máximo ($E_{máximo}$) de las aproximaciones son proporcionados por los autores de los respectivos trabajos.

La aproximación polinomial reportada por Faiedh[10] consiste en separar la función en diez polinomios de primer orden. Como se observa en la Tabla 2, esta variante presenta los menores valores de error en comparación con el resto, sin embargo, no es viable para desarrollar en *hardware* debido a la excesiva cantidad de multiplicadores que se requieren (uno por tramo).
La técnica de compresión A-Law[6] diseñada por Myers y Hutchinson, muestra los mayores valores de error. Sin embargo, constituye una solución ingeniosa puesto que propone dividir la función en siete segmentos lineales donde la pendiente de cada uno se expresa como una potencia de dos. Esto permite la sustitución de los multiplicadores por registros de desplazamiento.

Tabla 1. Características de las funciones sigmoidal.

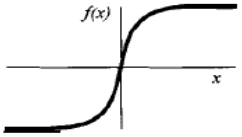
	Expresión analítica	**Rango de las ordenadas**	**Representación Gráfica**
Sigmoidal unipolar o logarítmica	$f(x)=\frac{1}{1+e^{-x}}$	(0,+1)	
Sigmoidal bipolar o hiperbólica	$f(x)=\frac{1-e^{-x}}{1+e^{-x}}$	(-1,+1)	

Tabla 2. Métodos de aproximación de la función sigmoidal logarítmica.

Aproximación	**Dominio**	**Número de tramos**	**$E_{promedio}$**	**$E_{máximo}$**
Polinomial[10]	[-5,5]	10	0,2%	1,11%
A-law[6]	[-8,8]	7	2,47%	4,90%
Alippi y Storti-Gajani[11]	[-5,5]	2	0,87%	1,89%
PLAN[7]	[-5,5]	8	0,59%	1,89%
Por tramos de segundo orden[8]	(-4,4)	2	0,77%	2,16%
Tablas de búsqueda[5]	[-5,5]	n	Depende de n	Depende de n

Por su parte, Alippi y Storti-Gajani[11] basan su aproximación en seleccionar un conjunto de puntos de interrupción y establecer los valores de "y" como potencias de dos. Al igual que A-Law, la fórmula puede implementarse mediante adiciones y operaciones de desplazamiento, lo que es muy conveniente para implementarla digitalmente[11]. Puesto que la función sigmoidal tiene un punto de simetría en (0,0.5), en esta aproximación se propone calcular sólo la mitad de los pares (*x, y*), lo que se muestra en las expresiones 1 y 2.

$$y_{x<0} = 1 - y_{x<0} \quad (1)$$

$$y_{x<0} = 1 - y_{x>0} \quad (2)$$

La denominada aproximación PLAN también se basa en dividir la función en tramos de primer orden, que se presentan en la Tabla 3. Fue propuesta por Amin, Curtis y Hayes Gill en 1997 [7]. Esta variante puede implementarse mediante adiciones y operaciones de desplazamiento al igual que la aproximación de Alippi y Storti-Gajani. Al igual que en el caso anterior se propone calcular sólo la mitad de los pares (x, y), los cálculos sólo deben realizarse para valores absolutos de la entrada x (ver la expresión 2).
Esta aproximación es más ventajosa con respecto a la A-Law en cuanto a que tiene menor error y menor cantidad de tramos, aunque el rango de entrada es más limitado.

Tabla 3. Aproximación PLAN en el intervalo [-5,5].

PLAN(x)	**Condiciones**
1	$\|x\|\geq 5$
$\|x\|/2^5+0{,}84375$	$2{,}375\leq\|x\|<5$
$\|x\|/2^3+0{,}625$	$1\leq\|x\|<2{,}375$
$\|x\|/2^2+0{,}5$	$0\leq\|x\|<1$

La función sigmoidal también puede ser representada como una aproximación por tramos de segundo orden expresados de la forma:

$$y(x) = C0 + C1 * x + C2 * x^2 \qquad (3)$$

La desventaja de este tipo de aproximación es que se necesita de varios multiplicadores en su implementación circuital. Con el objetivo de resolver este inconveniente, Zhang, Vassiliadis y Delgado–Frías[12] han presentado un esquema de aproximación de segundo orden en el rango de (-4,4) que requiere el uso de un solo multiplicador. En este intervalo la función queda como se muestra en la expresión 4:

$$y = \begin{cases} \frac{1}{2}\left(\frac{1}{4}x + 1\right)^2 & \text{para } -4 < x < 0 \\ 1 - \frac{1}{2}\left(\frac{1}{4}x - 1\right)^2 & \text{para } 0 \leq x < 4 \end{cases} \qquad (4)$$

Una tabla de búsqueda también puede emplearse para sintetizar la función sigmoidal mediante valores discretos. Este método de tabla de búsqueda consiste en una tabla de la verdad que almacena el resultado esperado para cada valor de la entrada, esta tabla de la verdad se almacena en una memoria ROM. El tamaño de la tabla dependerá de la resolución de los datos de entrada, lo que determina el número de combinaciones lógicas y del formato de los datos de salida. La implementación de este método consume gran cantidad de recursos del FPGA solo en el caso donde la memoria ROM no es implementada con bloques RAM predeterminados del FPGA.

Con la utilización de este método para realizar la función de activación, los niveles de exactitud que pueden alcanzarse dependen siempre de la cantidad de puntos que sean almacenados en la memoria. Si se aumenta la cantidad de puntos será más precisa la función que se obtenga pero aumentará el tamaño de la memoria y será mucho mayor el consumo de recursos.

REPRESENTACIÓN DE LOS DATOS Y REQUERIMIENTOS DE PRECISIÓN

En la aproximación de funciones existen dos fuentes de error, el error inherente al método de aproximación empleado y el error relacionado con la representación de los datos que resulta del uso de un número finito de bits. Para minimizar el consumo de recursos, el mínimo número de bits debe escogerse de manera que resulte en un error aceptable[13].

La representación numérica en punto flotante de precisión estándar se utiliza generalmente en la simulación de RNA en los microprocesadores de propósito general y ofrece mayor precisión y error de cuantificación mínimo con respecto a otros tipos de representación. Sin embargo, debido a los recursos limitados disponibles en un FPGA, el estándar de punto flotante no es factible en comparación con representaciones numéricas más eficientes en área, tales como 16 o 32 bits de punto fijo [14].

Otros autores han demostrado la ventaja del punto fijo por encima del punto flotante, al mostrar que las necesidades de espacio/tiempo de sumadores y multiplicadores de 32-bit de punto fijo fueron inferiores a la de sus equivalentes de 32 bits de punto flotante[15].

La cantidad de bits que representan la parte entera del número debe escogerse basado en el rango de la función de activación. Dado que el rango de salida de la función sigmoidal es diferente al de entrada, el formato de los datos de salida no tiene que coincidir con el de los datos de entrada[13].

El número de bits que se utilizan para representar la parte fraccional, dependerá de la precisión que quiera alcanzarse. Algunos autores plantean la utilización de 3, 4, 10, 12 o hasta 16 bits de precisión, aunque fue encontrado en la literatura que 10 bits son suficientes para algunas aplicaciones, como redes neuronales tipo perceptrón multicapa[6-9, 16].

MATERIALES Y MÉTODOS

Se realizó la implementación digital de la función sigmoidal logarítmica mediante tres aproximaciones: la variante PLAN que se describe en la Tabla 3; la aproximación de segundo orden propuesta por Zhang presentada en la expresión 4 y la técnica de tablas de búsqueda.

Para la implementación de la función se utilizó el programa MatLab v.7.10 (R2010a), el cual permite mediante su herramienta Simulink y el empleo de la herramienta System Generator del programa Xilinx ISE Design Suit 12.4 (ISE), el diseño *hardware* de la función, a nivel de bloques específicos de Xilinx.

Se utilizó formato de punto fijo para los datos de entrada y salida. Como el rango de entrada para las aproximaciones implementadas está entre [-5,5] y (-4,4), bastan 4 bits de parte entera para los datos de entrada. Para los datos de salida se tomó solo un bit de parte entera pues la función está acotada en el intervalo (0,1) y con esto es suficiente. En cuanto a los bits de precisión se tomaron tres casos: 4 bits, 8 bits y 16 bits para cada una de las aproximaciones. En la Tabla 4 se muestra el formato empleado tanto para los datos de entrada como para los de salida.

Tabla 4. Formato de los datos de entrada y salida utilizados en cada una de las aproximaciones.

Formatos	**Entrada**			**Salida**		
	Bits totales	**Bits enteros**	**Bits fraccionales**	**Bits totales**	**Bits enteros**	**Bits fraccionales**
Formato 1	8	4	4	5	1	4
Formato 2	12	4	8	9	1	8
Formato 3	20	4	16	17	1	16

El consumo de recursos y la frecuencia de reloj máxima se obtienen para cada aproximación mediante el programa *System Generator* y el error se calcula con el empleo de MatLab.

RESULTADOS Y DISCUSIÓN

Como muestra la Figura 1, la implementación de la función sigmoidal con la aproximación PLAN es la representación del algoritmo matemático, donde se realizan los cálculos solo con los valores positivos de la entrada. Se utilizaron cuatro multiplexores. El primero (*Mux*) se empleó para obtener el módulo del dato de entrada, para controlar la entrada de selección de este multiplexor se utilizó un bloque cuya función es extraer el signo del dato. El *Mux1* y *Mux2* se utilizaron para formar las expresiones matemáticas que representan la función sigmoidal, el bloque "*compare*" se encarga del control de la entrada de selección de ambos. Este bloque *"compare"* es un bloque tipo "*MCode*", el cual permite añadirle al diseño funciones de MatLab, las cuales pueden ser desarrolladas por el diseñador. El cuarto multiplexor (*Mux3*) se utilizó para seleccionar el valor de la salida, debido a que la función sigmoidal es simétrica, si el valor de entrada es positivo se deja pasar a la salida el valor calculado hasta el momento, sino la salida es 1-valor$_{calc}$. Para aumentar la frecuencia de trabajo del sistema, el diseño se segmentó en tres lugares: en los comparadores que seleccionan el tramo, se introdujo una latencia unitaria, además se utilizó un bloque de demora con latencia unitaria a la salida del bloque "*compare*" y por último se añadió una latencia unitaria en los multiplexores que se usan para formar la expresión matemática. La segmentación se hizo de esta forma para eliminar los caminos combinacionales más largos, señalados en los reportes del programa *System Generator*. De esta forma se introdujo una latencia total al diseño de tres períodos de reloj.

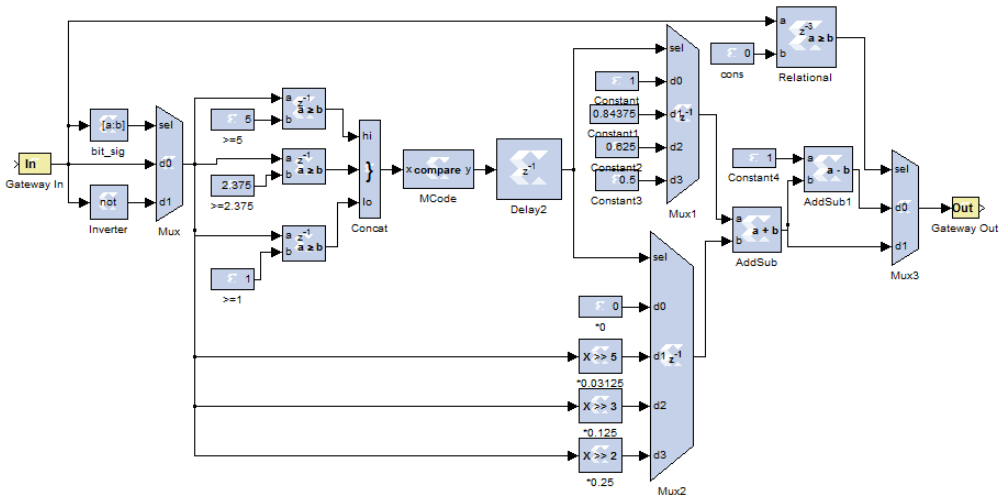


Figura 1. Implementación de la función sigmoidal logarítmica con la aproximación PLAN.

La implementación con la aproximación de segundo orden se muestra en la Figura 2. El dato de entrada se desplaza dos veces hacia la derecha, con lo que se divide entre cuatro. La entrada de selección del primer multiplexor (*Mux*) es el signo del dato de entrada, este se analiza con un bloque slice, a partir de este se selecciona que operación es la que va a propagar el circuito, si la operación de suma o la de resta (tramo positivo o negativo). Con el multiplicador se eleva al cuadrado el dato que se encuentra a la salida del multiplexor (*Mux*) y como indica la expresión 4, se divide por dos el resultado obtenido hasta el momento, es por eso que se desplaza el dato un lugar hacia la derecha. A la salida del diseño, en el último multiplexor (*Mux1*), el signo del dato a la entrada del diseño es la selección del multiplexor, de este depende si el resultado a la salida es el obtenido hasta el momento o uno menos el mismo.

Como la multiplicación es la operación más lenta a realizar en el diseño, se segmentó este bloque mediante la incorporación de una latencia unitaria, para aumentar la frecuencia de trabajo.

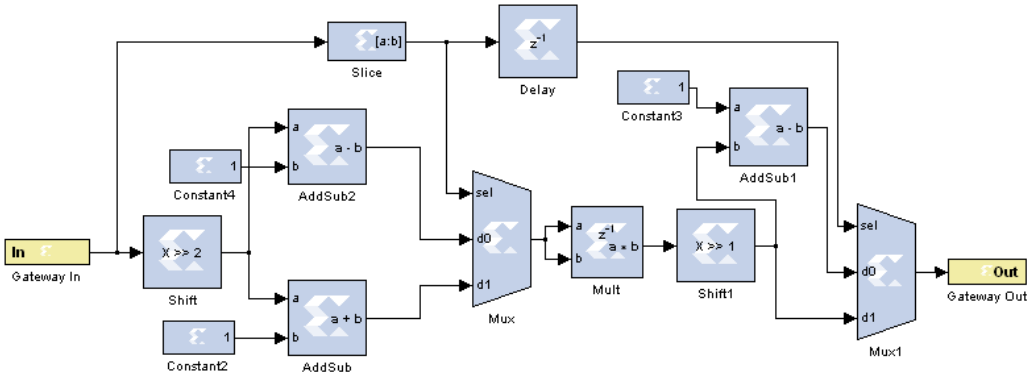


Figura 2. Implementación de la función sigmoidal logarítmica con la aproximación de segundo orden (Zhang).

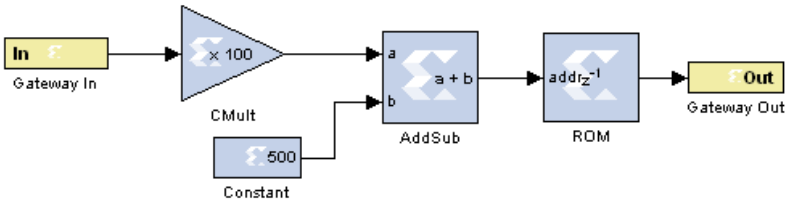


Figura 3. Implementación de la función sigmoidal logarítmica mediante una tabla de búsqueda.

En la Figura 3 se muestra el circuito que se utilizó para la solución mediante el empleo de tablas de búsqueda. Los datos de entrada a la función están en el rango de [-5,5] y en la memoria hay almacenadas 1000 muestras de la salida de la función sigmoidal logarítmica, con una resolución de 0.01 para ese tramo. Para direccionar la memoria, se necesita que los datos de entrada estén en el rango de [0,1000], para lograr esto se multiplican por cien y se les suma quinientos. El bloque de memoria ROM del *System Generator* se implementa en el FPGA con bloques de memoria RAM o con RAM distribuida, según escoja el diseñador, para este diseño se seleccionaron bloques RAM, ya que como se planteó al inicio con RAM distribuida el consumo de recursos es elevado. A este bloque se le puede definir el tamaño de la memoria y los valores con los que se quiere inicializar la misma, los cuales se obtienen de la salida de la función "*logsig*", de MatLab, para el rango de entrada en cuestión.

La Figura 4 muestra la representación gráfica de la función obtenida con las tres aproximaciones implementadas para cada uno de los formatos de los datos de entrada utilizados. Un resumen de los valores de error promedio y de error máximo cometidos con cada uno de estos formatos de entrada se presenta en la Tabla 5.

Los errores máximo y promedio se utilizan para evaluar la exactitud de una aproximación. Si se sigue la metodología descrita por Zhang[12], entonces si una función f(x) es aproximada por una función f*(x) en el intervalo x ϵ (α_0,α_1), los errores promedio ($E_{promedio}$) y máximo ($E_{máximo}$) se obtienen muestreando uniformemente x en 10^6 puntos igualmente espaciados en el dominio de (α_0,α_1) según se muestra en las ecuaciones 5 y 6.

$$Eprom = \frac{\sum_{i=0}^{10^6-1} |f^*(x_i) - f(x_i)|}{10^6} \qquad (5)$$

$$Emax = \max |f^*(x_i) - f(x_i)| \qquad (6)$$

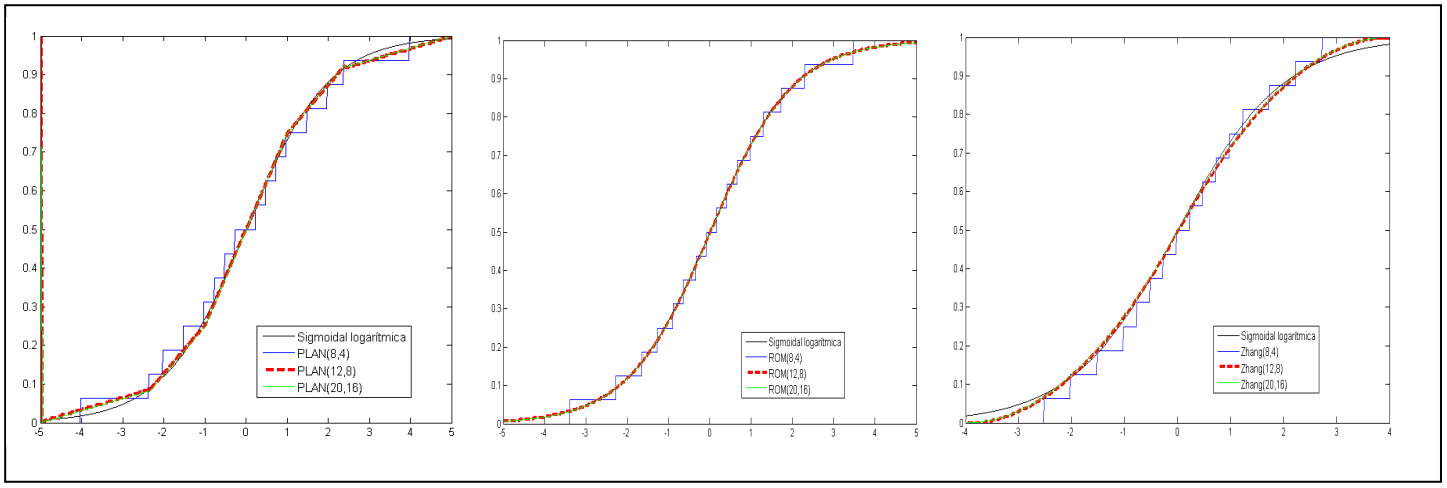


Figura 4. Representación gráfica de la función sigmoidal logarítmica.

Tabla 5. Errores de las aproximaciones implementadas

Aproximación	(20,16)*		(12,8)*		(8,4)*	
	Eprom (%)	Emáx (%)	Eprom (%)	Emáx (%)	Eprom (%)	Emáx (%)
PLAN	0,81	1,89	0,90	2,16	2,40	7,22
Zhang	1,10	2,16	1,13	2,55	2,80	7,53
Tablas de búsqueda	0,05	0,25	0,11	0,67	1,61	3,99

* Formato de los datos de entrada.

Como se puede observar en la tabla 5, los mayores errores promedio y máximo se obtuvieron con datos de entrada de 4 bits de precisión, mientras que con 8 y 16 bits de precisión, los errores difieren poco entre sí para las mismas aproximaciones. Sin embargo, al observar la Tabla 6, se ve que el consumo de recursos del FPGA sí es mucho menor cuando se usan datos con 8 bits de precisión que con el empleo de 16 bits. De este análisis se puede concluir que no es necesaria una implementación con 16 bits de precisión para alcanzar buenos niveles de exactitud.

Si se analizan los errores para las diferentes aproximaciones, con un mismo formato de entrada, se puede observar que los menores errores se obtienen con el método de tablas de búsqueda, pero según el reporte mostrado en la Tabla 6, si se utiliza este método se necesita mayor cantidad de bloques RAM, lo que no sería efectivo en un proyecto que requiera más bloques de este tipo que los que tiene el FPGA. Si fuera esta la situación, entonces es recomendable utilizar la aproximación PLAN, que tiene menor error que la de segundo orden y no utiliza multiplicadores, recurso también limitado en los FPGA.

Tabla 6. Empleo de recursos del FPGA Spartan 3 de Xilinx para las aproximaciones implementadas.

	Zhang			PLAN			Tabla de búsqueda		
Recursos	(20,16)	(12,8)	(8,4)	(20,16)	(12,8)	(8,4)	(20,16)	(12,8)	(8,4)
Slice flip-flops	34(1%)	18(1%)	6(1%)	28(1%)	22(1%)	17(1%)			
4 input LUTs	147(2%)	65(1%)	34(1%)	99(1%)	60(1%)	36(1%)	77(1%)	40(1%)	27(1%)
Occupied slices	94(2%)	47(1%)	26(1%)	57(1%)	37(1%)	24(1%)	44(1%)	23(1%)	17(1%)
Bonded IOBs	36(20%)	20(11%)	12(6%)	36(20%)	20(11%)	13(7%)	104(60%)	72(41%)	56(32%)
MULT18x18	3(18%)	1(6%)	1(6%)						
BUFGMUX	1(12%)	1(12%)	1(12%)	1(12%)	1(12%)	1(12%)	1(12%)	1(12%)	1(12%)
RAMB16s							1(6%)	1(6%)	1(6%)

En la Tabla 7 se muestra además la frecuencia máxima obtenida para cada aproximación.

Tabla 7. Frecuencia de reloj máxima para la implementación hardware.

Aproximación	Frecuencia de reloj máxima
PLAN	409.500 MHz
Zhang	732.064 MHz
Tabla de búsqueda	420.875 MHz

CONCLUSIONES

En este trabajo se presentó el diseño de la función sigmoidal logarítmica para su implementación en un FPGA Spartan 3 de Xilinx. Para ello se utilizaron tres aproximaciones matemáticas: por tramos de primer orden (PLAN), por tramos de segundo orden (Zhang) y por tablas de búsqueda, cada una con datos de entrada en formato punto fijo (20,16), (12,8) y (8,4).

Como resultado se obtuvo que el tipo de dato más eficiente en cuanto al compromiso exactitud y consumo de recursos del FPGA fue de (12, 8) independientemente de la aproximación.

Mediante las tablas de búsqueda se obtuvieron los menores valores de error, sin embargo, como se esperaba, precisó del empleo de bloques RAM, recurso escaso en los FPGA. Por esta razón se recomienda utilizarlo sólo si el número de funciones que se necesite implementar en el sistema, es menor que el número de bloques RAM disponibles en el FPGA.

Con la aproximación PLAN se obtuvo el mejor compromiso entre los errores de implementación y el consumo de recursos del FPGA, por lo que esta solución sería recomendable para aplicaciones donde se requiera de buena exactitud en la aproximación y el número de funciones que se necesite implementar en el sistema sea mayor que el número de bloques RAM disponibles en el FPGA, por ejemplo, como función de activación en el diseño de redes neuronales artificiales.

REFERENCIAS

1. **Manish Panicker, C.B.**, *Efficient FPGA Implementation of Sigmoid and Bipolar Sigmoid Activation Functions for Multilayer Perceptrons.* IOSR Journal of Engineering (IOSRJEN), Junio 2012. **2**(6): p. PP 1352-1356.

2. **Thamer M.Jamel, B.M.K.**, *IMPLEMENTATION OF A SIGMOID ACTIVATION FUNCTION FOR NEURAL NETWORK USING FPGA.* 13th Scientific Conference of Al-Ma'moon University College, Abril 2012.

3. **Djalal Eddine Khodja** , A.K., and Larbi Refoufi, *Sigmoid Function Approximation for ANN Implementation in FPGA Devices.* Recent Researches in Circuits, Systems, Electronics, Control & Signal Processing, 2009.

4. **N. lzeboudjen, A.F., S. Titri, H. Boumeridja**, *Digital Implementation of Artificial Neural Networks: From VHDL Description to FPGA Implementation.* 2004.

5. **TISAN, A., et al.**, *DIGITAL IMPLEMENTATION OF THE SIGMOID FUNCTION FOR FPGA CIRCUITS.* ACTA TECHNICA NAPOCENSIS Electronics and Telecommunications, 2009. **Vol. 50**(No. 2): p. 6.

6. **Myers, D.J. and R.A. Hutchinson**, *Efficient Implementation of Piecewise Linear Activation Function for Digital VLSI Neural Networks.* Electronics Letters, 1989. **Vol. 25**(24): p. 1662–1663.

7. **Amin, H., K. Curtis, and B.H. Gill**, *Piecewise Linear Approximation applied to Non-Linear Function of a Neural Network.* IEE Proceedings Circuits, Devices and Systems, 1997. **Vol. 144**: p. 313 – 317.

8. **Vassiliadis, S., Zhang, M., Delgado-Frias, J.**, *Elementary Function Generators for Neural-Network Emulators.* IEEE Transactions on Neural Networks, 2000. **Vol. 11** (6): p. 1438 – 1449.

9. **Tommiska, M.T.**, *Efficient digital implementation of the sigmoid function for reprogrammable logic.* Comput. Digit. Tech., 2003. **Vol. 150**(No. 6).

10. **Faiedh, H., et al.**, *Digital Hardware Implementation of a Neural System Used for Nonlinear Adaptive Prediction* Journal of Computer Science, 2006.

11. **Alippi, C. and G. Storti–Gajani**, *Simple approximation of sigmoidal functions: realistic design of digital neural networks capable of learning*, in *Proc. IEEE Int. Symp. on Circuits and Systems*. 1991: Singapore. p. 1505-1508.

12. **Zhang M., V.a.D.-F.**, *Sigmoid generators for neural computing using piecewise approximations.* 1996. **45**.

13. **Daniel Larkin, A.K., Valentin Muresan, Noel O'Connor**, *An Efficient Hardware Architecture for a Neural Network Activation Function Generator.* 2008.

14. **Omondi AR, R.J.**, *FPGA implementations of neural networks.* Springer, 2006.

15. **Ligon III WB, M.S., Monn G, Schoonover K, Stivers F, Underwood KD**, *A Re-evaluation of the Practicality of Floating Point Operations on FPGAs.* IEEE Symposium on FPGAs for Custom Computing Machines, 1998.

16. **Holt J, H.J.**, *Finite Precision Error Analysis of Neural Network Hardware Implementations.* IEEE Transactions on Computers 1993. **42**: p. 280 – 291.

AUTORES

José Abiel Caballero Hernández, Ing. en Telecomunicaciones y Electrónica, CUJAE, La Habana, Cuba, jcaballero@electrica.cujae.edu.cu.

Sonnia Pavoni Oliver, Ing. en Telecomunicaciones y Electrónica, Doctora en Ciencias, CUJAE, La Habana, Cuba, sonnia.pavoni@electrica.cujae.edu.cu.

Martha Díaz Salazar, Ing. en Automática, CUJAE, La Habana, Cuba.

Meyli Moradillos Paz-Lago, Ing. en Automática, La Habana, Cuba.