

引用格式:吴绮,金楚丰,陈刚,等. 基于 DCORDIC 算法的 Tanh 函数硬件设计与实现[J]. 微电子学与计算机,2020,37(12): 22-26. [WU Q, JIN C F, CHEN G, et al. Hardware design and implementation of Tanh based on DCORDIC algorithm [J]. Microelectronics & Computer, 2020, 37(12): 22-26.]

基于 DCORDIC 算法的 Tanh 函数硬件设计与实现

吴 绮^{1,2}, 金楚丰³, 陈 刚^{1,2}, 鲁华祥^{1,2,4,5}, 陈 旭^{1,5}

(1 中国科学院 半导体研究所, 北京 100083; 2 中国科学院大学, 北京 100089;

3 电子科学与技术大学, 四川 成都 610054; 4 中国科学院 脑科学与智能技术卓越创新中心, 上海 200031;

5 半导体神经网络智能感知与计算技术北京市重点实验室, 北京 100083)

摘 要: 双曲正切函数(Tanh)在神经网络中经常被用作激活函数, 而其非线性的特点导致难以用硬件电路直接计算实现. 为了快速有效的实现 Tanh 函数, 提出采用 DCORDIC(Differential Coordinate Rotation Digital Computer) 算法在不同模式下实现双曲正弦、双曲余弦函数和除法运算, 并在设计中增加选择模块扩大输入数据的范围, 最终设计实现的 Tanh 计算核具有超长流水线, 当迭代次数为 13 时, 计算误差在 2^{-8} 以内. 与现有的硬件实现方式相比, 本文的实现方式具有更小的误差、更高的运算速度和更大的吞吐率.

关键词: 双曲正切函数; DCORDIC 算法; 流水线; 吞吐率

中图分类号: TN492

文献标识码: A

文章编号: 1000—7180(2020)12—0022—05

Hardware design and implementation of Tanh based on DCORDIC algorithm

WU Qi^{1,2}, JIN Chu-feng³, CHEN Gang^{1,2}, Lu Hua-xiang^{1,2,4,5}, CHEN Xu^{1,5}

(1 Institute of Semiconductors, Chinese Academy of Science, Beijing 100083, China; 2 University of Chinese Academy of Sciences, Beijing 100089, China; 3 University of Electronic Science and Technology of China, Chengdu 610054, China; 4 Center for Excellence in Brain Science and Intelligence Technology, Chinese Academy of Science, Shanghai 200031, China; 5 Semiconductor Neural Network Intelligent Perception and Computing Technology Beijing Key Lab, Beijing 100083, China;)

Abstract: Hyperbolic tangent function (Tanh) is often used as an activation function in neural networks, but its non-linear character makes it difficult to be realized by hardware circuit directly. In order to implement the Tanh function quickly and efficiently, the DCORDIC(Differential Coordinate Rotation Digital Computer) algorithm is proposed to implement the hyperbolic sine/cosine function and division operation in different modes, and a selection module is added to expand the range of input data. The final design implementation of the tanh computation core has a very long pipeline with a computational error of 2^{-8} . Compared with the existing hardware implementation methods, the proposed implementation method has smaller error, higher computing speed and greater throughput.

Key words: hyperbolic tan; DCORDIC algorithm; pipeline; throughput rate

1 引言

目前,人工神经网络在模式识别、图像处理、信号处理等领域得到了广泛应用^[1]. 非线性函数是人工神经网络的重要组成部分,双曲函数(Tanh、sigmoid)经常被用作神经网络的激活函数,尤其是在循环神经网络(RNN)及其变种长短期记忆神经网络(LSTM)^[2]. 在集成电路高速发展的今天,研究如何用硬件方法快速实现双曲函数具有重要意义^[3]. 目前的硬件实现双曲正切函数的方法有分段线性逼近、查找表、分段非线性逼近以及 CORDIC 算法^[4-9]. 分段线性逼近实现比较简单,但为了获取更高的精度需要消耗较多的资源. 查找表法能简单而快速的实现双曲正切函数,但随着精度的增长其消耗的面积呈指数增加. 分段非线性逼近能实现较高的精度,然而这种方法需要大量消耗乘法器资源,运算速度比较慢. CORDIC 算法运用简单的移位和加减操作以及简单的互联就能实现非线性函数,原理和操作都非常简单. 但传统的 CORDIC 算法微旋转的方向受先前迭代结果的符号控制,使得该算法的计算时间和可实现的吞吐量受限.

本文采用 DCORDIC 算法硬件设计实现 Tanh 函数,充分利用该算法的特点,设计超长流水线获得较高的吞吐率;引入冗余数系统参与运算加快迭代速度;利用特殊编码方式简化运算以及输出结果向常规二进制数的转换;且为了解决 DCORDIC 算法的输入范围有限的问题,在本文的设计中额外增加了选择模块,扩大了输入值的范围.

2 DCORDIC 算法原理

DCORDIC 算法与普通 CORDIC 算法有着相同的精确度和收敛性,然而 DCORDIC 算法在计算中引入了一个中间变量,带来了更高的速度和更好的性能^[10]. 在此以 $\hat{p}$ 来表示这个中间变量, $\hat{p}$ 可以替代 x 、 y 、 z 中任意一个,定义:

$$\hat{p}_{i+1} = \text{sign}(p_i) \cdot p_{i+1} \quad (1)$$

此定义有两个推论:

$$|\hat{p}_{i+1}| = |p_{i+1}| \quad (2)$$

$$\text{sign}(\hat{p}_{i+1}) = \text{sign}(p_i) \cdot \text{sign}(p_{i+1}) \quad (3)$$

并进一步得到:

$$\text{sign}(p_{i+1}) = \text{sign}(p_i) \cdot \text{sign}(\hat{p}_{i+1}) \quad (4)$$

可以看出,原始迭代变量的符号可以通过简单的方式从相应新变量的符号中递归导出.

2.1 旋转模式 DCORDIC 算法

旋转模式下经合适的设置可以计算双曲正弦和双曲余弦函数值,在该模式下经过多次旋转迭代后 $z \rightarrow 0$,设置合适的输入值,最后得到的 x 和 y 就是输入角度的双曲正弦和双曲余弦值. 利用公式(1)和(2)对公式(3)进行改进得到如下公式:

$$|\hat{z}_{i+1}| = \|\hat{z}_i| - (\Delta\mu)_i|\quad (5)$$

式中,微旋转角度 $(\Delta\mu)_i = \tanh^{-1}(2^{-i})$,考虑到收敛性问题, i 的取值从 1 开始,且当迭代序数 i 等于 4, 13, 40, $\dots, k, (3k+1)$ 时需要重复迭代一次,即 $i \in \{1, 2, 3, 4, 4, \dots\}$. 该模式可计算的输入值需要满足 $z_1 < \sum_{i=1}^{\infty} \tanh^{-1}(2^{-i})$,因而输入角的范围为 $(-1.13, 1.13)$. 旋转模式输出为 n 次迭代后的值:

$$x_n = K(x_1 \cosh(z_1) + y_1 \sinh(z_1)) \quad (6)$$

$$y_n = K(y_1 \cosh(z_1) + x_1 \sinh(z_1)) \quad (7)$$

如果设置输入值 $x_1 = \cosh(\theta)$, $y_1 = \sinh(\theta)$,则最后的输出值 $x_n = K \cosh(\theta + z_1)$, $y_n = K \sinh(\theta + z_1)$,这样计算得到的结果与输入角度为 $\theta + z_1$ 一致. 尽管 z_1 的值范围有限,但通过和 θ 相加可以扩大计算角度范围. 旋转模式下的实现双曲正弦和双曲余弦函数值计算的步骤如下:

1. 输入 $x_1 = \cosh(\theta)$, $y_1 = \sinh(\theta)$, 角度初值 z_1 ;
2. 令 $\hat{z}_1 = |z_1|$ 得到 $|\hat{z}_1| = \hat{z}_1$;
3. 绝对值计算: $|\hat{z}_{i+1}| = \|\hat{z}_i| - (\Delta\mu)_i|$;
4. 符号计算: $\text{sign}(z_{i+1}) = \text{sign}(z_i) \cdot \text{sign}(\hat{z}_{i+1})$;
5. 相关数据计算:

$$x_{i+1} = x_i + \text{sign}(z_i) \cdot y_i \cdot 2^{-i},$$

$$y_{i+1} = y_i + \text{sign}(z_i) \cdot x_i \cdot 2^{-i};$$
6. 返回步骤 3 直至迭代结束.

2.2 向量模式 DCORDIC 算法

向量模式下设置合适的输入值可以实现除法运算,在该模式下经过多次旋转迭代后 $y \rightarrow 0$,设置合适的输入值, z 的最终结果是初始输入值 y 与 x 的商. 利用公式(6)和(7)对公式(1)和(2)进行改进得到如下公式:

$$|\hat{y}_{i+1}| = \|\hat{y}_i| - |x_i| \cdot 2^{-i}| \quad (8)$$

$$\hat{x}_{i+1} = |\hat{x}_i| + |\hat{y}_i| \cdot 2^{-i} \quad (9)$$

对于输入 x 可以通过适当的变换使得 $x > 0$,在该情况下可以证明 $\hat{z}_i = x_i > 0$,因此有以下公式:

$$|\hat{y}_{i+1}| = \|\hat{y}_i| - \hat{x}_i \cdot 2^{-i}| \quad (10)$$

$$\hat{x}_{i+1} = \hat{x}_i + |\hat{y}_i| \cdot 2^{-i} \quad (11)$$

线性坐标系的向量模式下微旋转角度 $(\Delta\mu)_i = 2^{-i}$,在该模式下实现除法运算法步骤如下:

- 1. 输入除数与被除数 x_0, y_0 , 设置 $z_0 = 0$;
 - 2. 令 $\hat{y}_0 = |y_0|$ 得到 $\hat{y}_0 = \hat{y}_0$, 令 $\hat{x}_0 = x_0$;
 - 3. 绝对值计算: $|\hat{y}_{i+1}| = ||\hat{y}_i| - \hat{x}_i \cdot 2^{-i}|$;
 - 4. 符号计算: $\text{sign}(y_{i+1}) = \text{sign}(\hat{y}_{i+1}) \cdot \text{sign}(y_i)$;
 - 5. 相关数据计算: $\hat{x}_{i+1} = \hat{x}_i + |\hat{y}_i| \cdot 2^{-i}$,
 $z_{i+1} = z_i + \text{sign}(y_i) \cdot (\Delta\mu)_i$;
 - 6. 返回步骤 3 直至迭代结束.
- 其中 i 的取值从 0 开始, $i \in \{0, 1, 2, 3, 4, \dots\}$.

3 DCORDIC 硬件实现 Tanh

图 1 所示为采用 DCORDIC 算法硬件实现 Tanh 函数的整体架构图. 双曲函数计算器的输入为待计算角度, 输出为该角度的 Tanh 值. 顶层模块主要包含两个子模块, sinh/cosh 计算模块和除法器模块, 输入角先进入 sinh/cosh 计算模块, 输出缩放后的 sinh 与 cosh 值作为除法器模块的输入, 除法器模块的输出即为输入角度的双曲正切值.

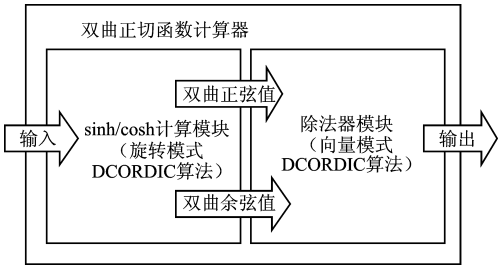


图 1 整体框架示意图

3.1 sinh/cosh 计算模块

图 2 为该 sinh/cosh 模块的设计原理框图. 该模块的输入信号为待旋转角度初值 z_0 , 输出为该旋转角度缩放后的双曲正弦值和双曲余弦值.

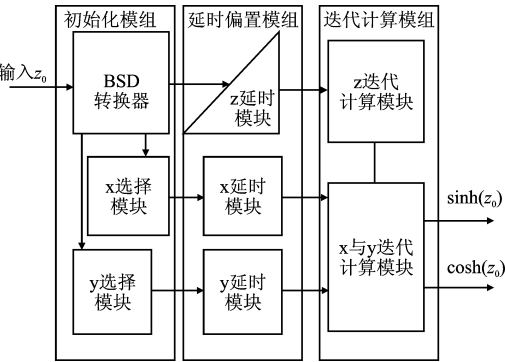


图 2 双曲函数模块原理图

将双曲函数计算模块中的子模块划分为初始化模组、延时偏置模组、迭代计算模组. 初始化模组需要对 x, y, z 三个迭代初值进行初始化. 该模块输入

是常规的二进制数, 整个模块在计算时的操作数是 (p, n) 编码的 BSD 冗余数 (Binary Signed Digit Redundant Number System)^[11], 因而需要 BSD 转换器进行编码, 该模块输入值为常规二进制值, 输出 (p, n) 编码的 BSD 冗余角度值及控制信号. x 与 y 选择模块用以扩大双曲模块可计算的输入角范围. 以输入角度 $z=2$ 为例, 选择模块选择 x 与 y 的初始值为 $\cosh(1)$ 和 $\sinh(1)$, z 迭代计算模块的输入为 1, 根据公式 (11) 和 (12), 最后输出的值为输入角度为 2 的双曲正弦值和双曲余弦值. 该模块的输出由 BSD 转换器的输出控制信号进行选择.

延时偏置模组对三个初始化变量进行延时偏置, 由于对 z 进行绝对值计算是从最高有效位开始的, 这意味着 z 的低位需要更多的延时才能进入绝对值计算, 因而设置一个 z 延时模块, 其输出从 MSD 开始经过一定延时逐位进入下一个模块. x 与 y 也需要经过一个延时模块再进入下一步的迭代计算, 该延时模块的延时时长与 z 的数据宽度有关.

迭代计算模组包括 z 迭代计算模块和 x 与 y 迭代计算, 该模组是该双曲函数计算的主体部分. z 迭代计算模块完成旋转模式 CORDIC 算法步骤 3 的操作, 主要进行了绝对值和减法运算, 同时获取 z 的符号位, 该符号位作为输出接入 x 与 y 迭代计算模块, 内部原理图如图 3 所示.

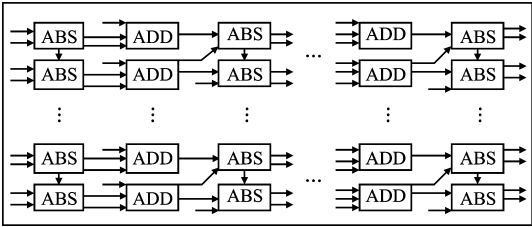


图 3 z 迭代计算模块原理图

x 与 y 迭代计算模块完成了旋转模式 CORDIC 算法步骤 4 的计算, 可以看出该模块的主要运算是加/减法运算和移位运算, 其中 x 和 y 变量都是 BSD 冗余数, 每个 digit 有两位, 因而加法器的输入是四位, 这里用两个 3-2 加法器串联构成一个 4-2 加法器, 而每次迭代的方向由 z 变量的符号控制. x 和 y 的计算同时进行最后以 BSD 冗余数的形式输出缩放后的 cosh 值和 sinh 值.

3.2 除法器模块

图 4 为除法器模块的设计原理框图. 该模块的输入信号为 x 和 y 的值, 输出为 y/x 的商. 将双曲函数计算模块输出缩放的 cosh 和 sinh 值分别接入该

模块的 x 和 y 作为输入,输出 z 即为双曲正切值。

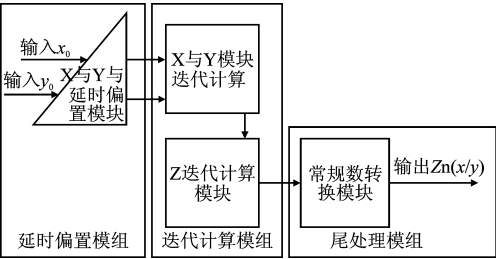


图 4 除法器模块原理图

将除法器模块中的子模块划分为延时偏置模组,迭代计算模组以及尾处理模组。延时偏置模组中输入为 x 和 y 的初值,此模块输入直接与双曲模块的输出相连,即 x 接入 $\cosh$ 的 BSD 冗余值, y 接入 $\sinh$ 的 BSD 冗余值,因而不需要再进行 BSD 冗余数编码。由于后续的运算仍然涉及以最高有效位优先的绝对值的计算,造成了倾斜的流水线,所以 x 和 y 延时偏置模块是一个三角延时模块,此处的延时偏置模块中包含两个延时偏置子模块分别对 x 和 y 进行偏置。

迭代计算模组包含 x 、 y 和 z 三个迭代变量的计算模块,是除法器的主体部分。 x 和 y 迭代计算模块完成了公式(13)和(14)的操作,其中包括了加/减法运算,移位运算和绝对值运算。对 x 进行的是加法运算和移位运算,对 y 进行的是减法运算、移位运算和取绝对值的运算, x 和 y 是两条不同的计算线路, x 通路上用两个 3-2 加法器构成加法运算, y 通路上用两个 3-2 减法器构成 4-2 减法器进行减法运算,因为下一次迭代需要的是当前迭代得到 x 、 y 移位后的值因而分别对两个通路上的变量进行移位操作,直至迭代结束。

Z 迭代计算模块完成了 $z_{i+1} = z_i + \text{sign}(y_i) \cdot (\Delta\mu)_i$ 的运算。 z 的初值为 0, $(\Delta\mu)_i = 2^{-i}$, $\text{sign}(y_i) \in \{-1, 1\}$ 。定义一种特殊的二进制表示方法: z 的每位用 $\{0, 1\}$ 表示 $\{1, -1\}$,每位的权重从左至右为 2^{-i} ,在该表示方法下,若 $z = 1010110$,则二进制实际值 $z = -2^0 + 2^{-1} - 2^{-2} + 2^{-3} - 2^{-4} - 2^{-5} + 2^{-6}$ 。从公式可以看出每轮迭代确定 z_i 中权重为 2^{-i} 的那一位,该位的具体值用 $\{0, 1\}$ 表示,决定该位取值的公式 $\text{sign}(y_{i+1}) = \text{sign}(\hat{y}_{i+1}) \cdot \text{sign}(y_i)$ 在特殊表示下改写为进行异或操作,图 5 左侧为该模块的原理图,唯一涉及的运算类型是 XOR。因为该模块的输入形式是每周期一个符号位输入,而输出是一个周期输出整个计算结果,因而设计了流水寄存器使得能在一

个周期内输出一个数据。

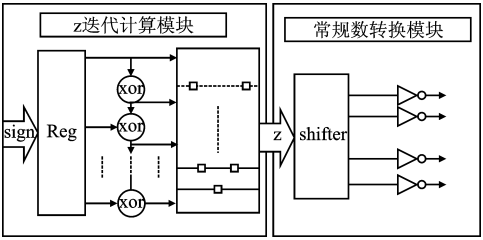


图 5 Z 迭代计算模块原理图

尾处理模组仅一个常规数转换模块,原理图如图 5 右侧框图所示,该模块将 Z 迭代计算模块特殊表示的输出数据转换成常规二进制输出。将特殊表示的二进制数转换成常规表示的二进制数所要进行的操作为左移取反,因此该模块只需要移位操作和取反操作。

4 系统验证与仿真

4.1 MATLAB 仿真结果分析

DCORDIC 算法输出数据的精度与迭代次数有关。针对不同的迭代次数执行 DCORDIC 算法的 MATLAB 代码,并绘制误差与迭代次数的关系图如图 6(a)所示,可以看出随着迭代次数的增大,误差减小;当迭代次数大于 10 时误差趋近于零。图 6(b)为数据位宽为 13、迭代 13 次时用 MATLAB 进行仿真得到的误差分布图,可以看出误差在 2^{-8} 以内。

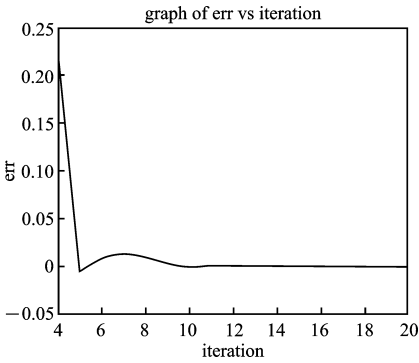
4.2 Vivado 仿真结果分析

Verilog 代码中设计迭代次数为 13,数据用 13 位定点数表示,其中最高位为符号位,两位为整数位,其余十位为小数位。考虑到函数的对称性,仅以输入值属于 $(0, 2)$ 的数作为测试激励输入计算模块,输出计算所得的 Tanh 值,处理得到最大绝对误差与平均误差与其他方法实现对比,如表 1 所示,可以看出该方法能实现比其他方法更小的误差。

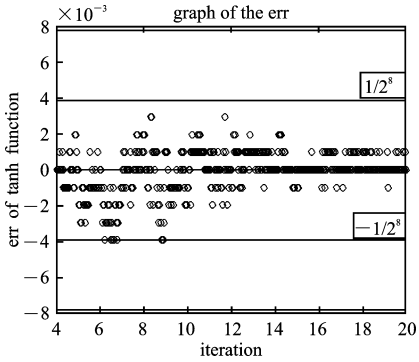
表 1 不同实现方法的误差

架构	最大误差	平均误差
分段线性逼近 ^[6]	0.043 0	0.007 8
查找表 ^[7]	0.036 5	0.004 0
分段非线性逼近 ^[8]	0.022 0	0.007 7
基本 CORDIC ^[9]	0.045 0	0.001 1
本文提出的 DCORDIC	0.005 6	0.001 0

在 Vivado2019 平台上对编写的 Verilog 代码进行仿真,当迭代次数为 13 时,结果如图 7 所示,其



(a)不同迭代次数的误差



(b)数据位宽为 13 时误差分布

图 6 误差与迭代次数关系图

中设置的仿真时钟频率为 100 MHz. 可以看出输出第一个结果需要 89 个时钟延时, 即该设计有 89 级流水, 其中 sinh/cosh 计算模块占 39 级流水, 除法器模块占 50 级流水.

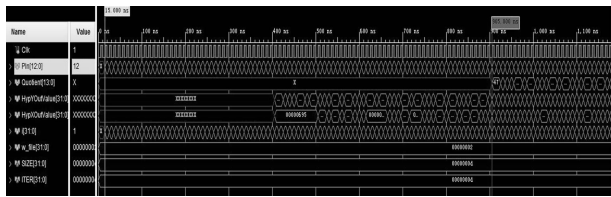


图 7 仿真波形图

迭代次数和数据位宽都会影响流水级数 sinh/cosh 计算模块的流水级数可以表示为 $\text{iteration} \times 2\text{clk} + \text{length} \times \text{clk}$, 除法器模块的流水级数表示为 $\text{length} \times \text{clk} + \text{clk} + (\text{iteration} - 1) \times 3\text{clk}$, 减少一次迭代能减少 5 级流水线, 但随之误差也会增大, 迭代次数的选择需在流水级数和误差之间折中.

用 Vivado2019 对代码进行综合, 选择的器件型号为 xc7vx485tffg1157-1, 综合报告如图 8 所示. 本文设计了超长流水线提高了并行计算能力, 电路能工作的最高频率为 550 MHz, 在首次延迟输出第一个结果后可以一个时钟周期输出一个计算结果, 具有比其他实

现方法更高的吞吐率. 整个电路的功耗为 0.707 W.

Resource	Estimation	Available	Utilization %
LUT	2773	303600	0.91
LUTRAM	241	130800	0.18
FF	3653	607200	0.60
IO	28	600	4.67
BUFG	1	32	3.13

图 8 综合报告

5 结束语

本文基于 DCORDIC 算法硬件设计实现 Tanh 函数, 在该算法的不同模式下分别硬件设计实现了双曲正弦和双曲余弦函数计算模块以及除法器模块, 两者级联最终实现了 Tanh 函数, 设计增加的选择模块能有效扩大输入数据的范围. 实验结果表明, 选择合适的迭代次数和位宽能达到较高的精度要求, 该设计采用大量并行流水结构实现了较高的运算速度以及较大的吞吐率.

参考文献:

[1] GOMPERTSA, UKILA, ZURFIUHF. Development and Implementation of Parameterized FPGA-Based General Purpose Neural Networks for Online Applications [J]. IEEE Trans. On Industrial Informatics, 2011, 7(1):78-89.

[2] CHANG A X, MARTINI B, CULURCIELLO E, et al. Recurrent Neural Networks Hardware Implementation on FPGA[J]. International journal of advanced research in electrical, electronics and instrumentation engineering, 2015, 5(1): 401-409.

[3] SONG Y K, GAO X H, ZHANG D L, et al. The piecewise non-linear approximation of the sigmoid function and its implementation in FPGA[J]. Application of Electronic Technique, 2017, 43(8):49-51.

[4] BASTERRETXEAK, TARELAJ M, CAMPOI D. Approximation of sigmoid function and the derivative for hardware implementation of artificial neurons[J]. IEE Proceedings - Circuits, Devices and Systems, 2004, 151(1):18-24.

[5] BASTERRETXEAK, TARELAJ M. Approximation of sigmoid function and the derivative for artificial neurons[C]// Zeng Z G. advances in neural networks and applications. Athens: WSES Press, 2001:397-401.

[6] LIN C, WANG J. A digital circuit design of hyperbolic tangent sigmoid function for neural networks[C]// 2008 IEEE international symposium on circuits and systems. Seattle: IEEE, 2008: 856-859.