

基于对数运算的 Sigmoid 函数的研究与实现

圣 飞, 阴亚芳, 秦晨蕊, 张 凯

(西安邮电大学 电子工程学院, 陕西 西安 710121)

摘 要: 基于对数运算的 sigmoid 函数常被用作神经网络的阈值函数, 将变量映射到 $(0, 1)$ 之间. 为了研究其高速度, 高精度, 低功耗的性能, 以对数运算为实现原理, 对 sigmoid 函数做了研究并继而实现其函数, 通过 SystemVerilog 仿真平台对设计进行简单验证. 通过比较仿真平台产生的输出与设计产生的输出来总结本次设计的 sigmoid 函数的速率, 精度, 功耗等性能指标. 结果表明: 本次设计的 sigmoid 函数在速率上得到了改善, 进行一次函数运算需七个时钟周期(主频 200 MHz)在精度上也稍有提升, 降低了硬件功耗.

关键词: 对数运算; sigmoid 函数; 精度; 速率; 功耗

中图分类号: TN492

文献标识码: A

文章编号: 1000-7180(2018)02-0011-04

Research and Implementation Based on Transcendental Function Coprocessor Sigmoid Function

SHENG Fei, YIN Ya-fang, QIN Chen-rui, ZHANG Kai

(School of Electronic Engineering, Xi'an University of Posts & Telecommunications, Xi'an 710121, China)

Abstract: Sigmoid function based on the logarithm operation is often used as the threshold function of neural network mapping variables between zero and one. In order to study its high speed, high precision, low power consumption, by logarithm operation for the realization of the principle, we do the research and then realize its function for the sigmoid function. We also do simple design verification through the SystemVerilog simulation platform, and summarize performance indicators of the sigmoid function on rate, precision, power consumption and so on by comparing the output of the simulation platform with the output of design. Results show that the designs of the sigmoid function on rate are improved, a functional operation needing seven clock cycles (frequency 200 MHz), a slight increase on the precision. Hardware power consumption has been reduced.

Key words: transcendental function; sigmoid function; precision; rate; power consumption

1 引言

对于深度神经网络, 中间的隐层的输出必须有一个激活函数. 否则多个隐层的作用和没有隐层相同. 因为 sigmoid 函数可以把实数域光滑的映射到 $[0, 1]$ 空间. 而且 sigmoid 函数单调递增, 连续可导, 导数形式非常简单, 是一个比较合适的激活函数^[1]. 根据不同需求, 超越函数的浮点操作可以为双精度浮点, 单精度浮点, 半精度浮点. 本文主要研究和实现的是单精度浮点数的基于对数运算的 sigmoid 函数.

实现超越函数的算法有很多, 经过多年的研究, 研究人员已经提出了多种的实现超越函数的算法, 这些算法要么精度不够, 如定点 CORDIC 算法 (CORDIC 算法通过多次迭加将复杂运算转换为简单运算, 是一种逼近算法); 要么实现所占资源太大^[2], 如浮点 CORDIC 算法和查表法; 要么实现速度慢, 不足以支持实时渲染的要求, 如高次多项式逼近算法. 因此, 设计精度高、速度快满足移动终端实时绘制场景要求的超越函数算法是实现超越函数运算单元的关键技术之一.

分段线性逼近算法是一种线性逼近超越函数的

实现方法,本文采用这种改进后的算法即不等分的分段线性逼近算法^[3],不等分的思想是曲线两端的分段数目比中间多,其原因是两端的曲线接近直线比较容易拟合.而一般的分段线性逼近算法都是等分的,其弊端是如果分段数目多,区间小会增加精确度,但是占用了较大的查找表资源耗费比较大,但分段数目少,区间大则会使函数逼近误差较大.同时,本次设计的 sigmoid 函数框架能支持多种函数(三角函数,指数函数的泰勒展开式有多项乘法,因此采用 booth 乘法器能提高运算效率)的设计,多种超越函数可以通过一个运算单元来完成,也优化了电路结构.

2 超越函数算法原理

对数数字系统(LNS)可以极大地简化算术运算^[4],例如,乘法、除法和平方根运算都可以简化成加法、减法和右移运算.但是,(LNS)并不能直接用于定点数格式数据的计算.因此,采用一种定点混合数字系统的设计(浮点运算需要大量硬件资源),并采用分段线性逼近的方法来实现这种数字系统.本文是将输入的单精度浮点数转化为定点数进行运算继而实现相应的函数.本文实现超越函数的算法思想就是将操作数取对数后再进行相应的对数运算最后再将结果转换成浮点输出.具体流程如图 1 所示,其中 float-fixed 模块是将输入的单精度浮点数转化为定点数;fixed-log 模块是将定点数转换为以 2 为底的对数;log-operation 模块是对数运算单元;log-fixed 模块是将以 2 为底的对数转换为定点数;fixed-operation 定点数运算单元;fixed-float 模块是将定点化为单精度浮点数.函数基本模块如下:

- (1) float-fixed $\Rightarrow$ (2) fixed-log
- (3) log-operation $\Rightarrow$ (4) log-fixed
- (5) fixed-operation
- (6) Fixed-float

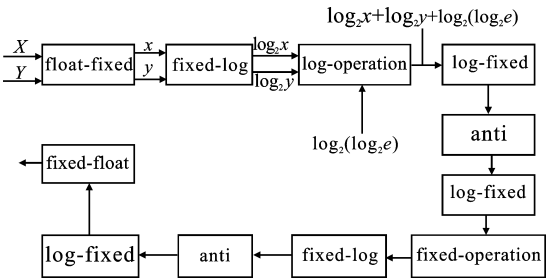


图 1 sigmoid 函数详细流程图设计

2.1 浮点数与定点数之间的转化单元

如图 2^[5]所示,IEEE 单精度浮点数包括 1 位符号位、8 位阶码位和 23 位尾数位.

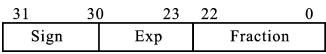


图 2 IEEE 单精度浮点数格式

32 为定点数包括 1 为零标志位 Z,1 位符号位 S,6 位整数与 24 位小数,其格式如图 3 所示:

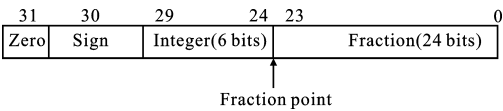


图 3 6.24 的定点数格式

浮点数与定点数 B 的关系如式(1):

$$B = (-1)^s \cdot 1.f \cdot 2^{e-127};$$

式中,s 表示符号位 (Sign),f 为浮点数的尾数 (Fraction),e 为浮点数阶码 (Exp).

浮点数 F 转化为定点数 B 的过程为:

- (1) 输入单精度浮点数 F;
 - (2) 将浮点数的尾数高位补 1 得到 24 位 $f_1(1.f)$ 形式;
 - (3) 计算 $M = e - 127$;若 $M < 0$,则 $R = f_1 >> |M|$,即将 f_1 右移 $(-M)$ 位;反之 $R = f_1 << M$,即将 f_1 左移 M 位;其中浮转定的误差主要来自于(2).
 - (4) 输出的定点数 $B = (-1)^s R$;
- 同理定点数 B 转化为浮点数 F 的过程为:
- (a) 输入定点数 B;
 - (b) 若 $Z = 1$,则此定点数表示 32 位的 0;否则(c);
 - (c) 计算 $B[29:0]$ 的前导零 L,令 $f_b = B[29:0] << (L + 1)$ 即求出首一之后的数,令 $f = f_b[29:7]$;通过截取高 23 为求出定点数的尾数部分 (Fraction). 令 $f_i = 4 - L$,则 $\text{exp} = f_i + 128 = 5 - L + 127$. 因此输出的浮点数为 $F = \{\text{exp}, f\}$.

输出的浮点数的误差主要来自于尾数部分的截取.

2.2 定点数转为对数单元

转对数算法的原理是先通过求前导零 L,然后通过与整数 5 做减法得到整数部分(首一的位置),再通过不等分分段线性逼近算法(6.24 格式的查找表)求出小数部分.任何一个定点数 x 都可以用 $\log_2 x$ 进行逼近,这样就是将任意两个数的运算可转换成对数运算,降低了计算的复杂度.

任意一个 32 位二进制定点数 x 可表示为:

$$x = z_k z_{k-1} \cdots z_2 z_1 z_0, z_{-1} z_{-2} z_{-3} \cdots z_j \text{ 即 } x =$$

$$\sum_{k=j}^k 2^i z_i, z_i = \text{“0” 或 “1”}$$

最高位 z_k 如果为 1 的话, x 就可以表示成:

$$x = 2^k (1 + d)$$

$$f = \sum_{k=j}^{k-1} 2^{i-k} z_i, 0 \leq d < 1$$

它的对数就可以表示为:

$$\log_2 x = k + \log_2 (1 + d)$$

k 是对数的整数部分(即首一的位置), $\log_2 (1 + d)$ 是它的尾数部分, 范围为 $[0, 1)$. 从而看出整数部分的求解过程简单明了, 可通过整数 5 与 L 做减法求出. 其主要工作量集中在了小数部分的求解. $\log_2 (1 + d)$ 是通过区间遵循 2^k 分段原则(分成了 64 个小区间) 进行分段线性逼近得到. 再根据最适不等分分段的思想将区间 $[0, 1)$ 划分成两端分段较多, 中间分段较少的 14 个逼近区间.

2.3 对数运算单元

基于超越函数协处理器实现的乘法、除法、平方根、三角函数、指数均用基础的模块单元就可以实现. 对于乘法、除法、平方根运算不需要用到 booth 乘法器(都是转换成对数加减形式). 对于三角函数指数函数都采用四路 booth 乘法实现(能同时进行四路乘法). 对于 Ln 函数, 操作数取对数后还需再进行简单的运算. 然后处理方法与三角函数类似. Ln 函数可以转化为:

$$\text{Ln} x = m \log_2 x, \quad m = 1/\log_2 e;$$

也就是 Ln 函数需经过取对数操作和乘法操作. sigmoid 函数具体框图如图 1.

2.4 对数转为定点数单元

对数转为定点数单元与定点数转为对数单元相似. 上述 30 位对数

$$\log_2 x = k + \log_2 (1 + d), \text{ 可以转化为:}$$

$$2^a = 2^k + 2^n;$$

$$\text{其中 } a = \log_2 x, n = \log_2 (1 + d);$$

其中 2^k 可以通过将 $a[29:24]$ 左移 k 位得到. 而 2^n 则需要进行分段线性逼近, 与定点数转对数单元类似此单元也采用不等分分段线性逼近. 不等分原则仍是两端分段多中间分段少, 将其划分为 14 个小区间. 从而得到 2^n 的逼近值.

3 基于对数运算的 sigmoid 函数的算法原理与研究设计

在信息科学中, 由于其单增以及反函数单增等

性质, Sigmoid 函数常被用作神经网络的阈值函数, 将变量映射到 $(0, 1)$ 之间其表达式为^[4]:

$$y = \frac{1}{1 + e^{-ax}} \tag{1}$$

基本的基于对数运算的超越函数采用四路流水线设计, 包含六级流水线: 浮点转定点级、定点转对数级、对数运算级、对数转定点级、定点数运算级以及定点转浮点级. Sigmoid 函数因为实现原理的不同, 故不能实现流水化. 但其基本思路却都是基于对数运算的思想. 本文将讲述基于对数运算思想的 sigmoid 函数的研究与实现.

公式(1) 中, a 表示权重, 其关系到 y 的曲线斜率问题. a 越大, 曲线越接近阶梯状, 类似于一个阈值; a 越小, 曲线越线性化, 甚至在一定的区间之内可以逼近直线

该函数也可转换为下面公式进行计算:

$$f(x, y) = 2^{-\log_2 (1 + 2^{-2(\log_2 x + \log_2 y + \log_2 (\log_2 e))})} \tag{2}$$

式中, x 为原式中的权重 a , y 为原变量 x . 因此, 根据公式(2) 的形式, 其计算流程为:

首先将 X 和 Y 输入浮转定单元得到定点数 x 和 y , 然后将定点数 x 和 y 输入定点数转对数单元, 计算出 $\log_2 x, \log_2 y$, 并将定点转对数单元产生的结果输入到下一级对数运算模块, 计算 $\log_2 x + \log_2 y + \log_2 (\log_2 e)$ 的值; 然后将结果送入下一级对数转定点数单元, 并且将计算完的结果取反, 同时再次输入到对数转定点数单元; 从对数转定点数单元第二次出来的数据被送入定点数运算模块计算 $1 + 2^{-2 \log_2 x + \log_2 y + \log_2 (\log_2 e)}$; 之后再送入定点数转对数单元, 然后将定点数转对数单元输出的数据取反, 再次将取反的结果输入到对数转定点数单元; 最后再将对数转定点数单元结果输入到定点数转浮点数单元, 从而输出浮点数形式的运算结果. 本文研究与实现的 sigmoid 函数因为没有涉及到乘法, 所以未用到 booth 乘法运算.

4 验证平台的搭建

验证结果的准确性可以通过定点测试和随机测试验证^[6], 介于定点测试需耗费大量时间和人力且测试结果不具代表性. 本文采用随机测试验证结果的准确性, 本文采用 SystemVerilog 语言来搭建验证平台, 平台框图如力图 4 所示.

本文实现的基于对数运算的 sigmoid 函数的 system verilog 验证平台结构主要包含一个驱动器 Driver, 负责和协处理器进行通信并发送运算数据

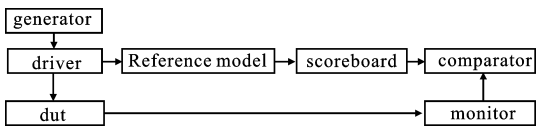


图 4 System Verilog 验证平台结构

和操作码；一个监测器 Monitor，负责和协处理器通信得到其产生的运算结果，并且可以监测运算中出现的异常；数据产生器 Generator，负责生成各种不同的运算请求和数据；参考模型 Reference Model 为 C 语言编写的一个超越函数计算模块，当输入和待测设计 dut 同样的运算请求和数据时，产生一个精度足够高的运算结果用来检验 DUT (Design Under Test，待测设计) 的正确与否；记分板 Score Board 分别记录待测设计和参考模型的输入和输出，并送去比较器 Comparator 进行对比，如果满足预设的误差要求，则说明验证正确，否则产生错误报告，返回重新修改设计

通过图 5 波形仿真可以得出：当 vld 为高电平时，浮点数据 a, b 在时钟上升沿发出(本文显示的数据均为处理过的数据)，下个时钟周期 vld 变成低电平，数据发出后经过 7 个时钟周期，输出浮点数据结果 result(显示数据也是处理过的)，结果输出的下个时钟 vld 再次变成高电平发送第二组数据 a, b ，经过 7 个时钟周期输出第二个结果。当 a, b 在 0 到 1 之间时，运算结果趋近于 0.5，当 a, b 均大于 1 时，增大 a, b 时函数值也增大，并逐步趋近于 1，经大量测试分析得知当数据在区间 $(0, 1)$ 之间误差最小，经计算其平均相对误差 1.067×10^{-5} (平均相对误差即为相对误差的平均值，这里我们是通过平台算出

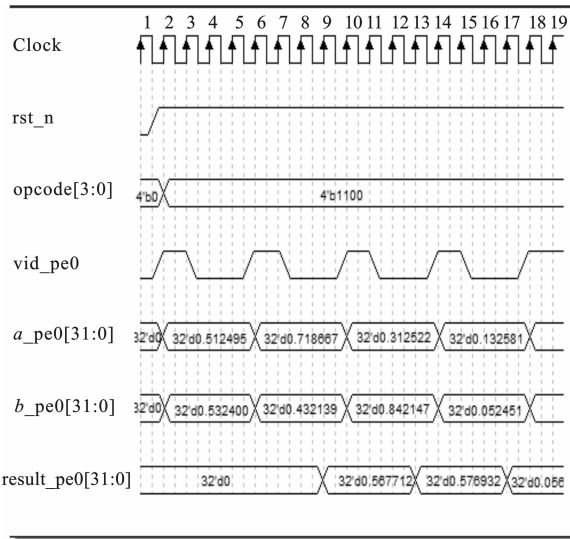


图 5 sigmoid 函数仿真的时序图

的)，经过分析，浮点数转定点数单元、定点数转对数单元、对数运算单元、对数转定点数单元和定点数转浮点数单元等模块都会不同程度的产生误差，其误差都是由截取产生的，而定点数转对数单元产生的误差最大。

5 结束语

为了研究与实现基于对数运算 sigmoid 函数，采用了基础的超越函数的六级流水线设计电路以提高电路的运算速度，经过仿真确定了 sigmoid 函数在区间 $(0, 1)$ 的精度 (1.067×10^{-5}) 满足了实际中的要求，经过对比以往的设计，本次设计速度上 7 个时钟就能完成 sigmoid 函数的运算，在运算速率上得到了很大的提升。精度上也满足了要求。且本次设计输入采用单精度浮点型数据，数据范围更大，运算处理时采用定点数数输入，节约了硬件资源，降低了功耗。

参考文献：

[1] 唐世庆. 协处理器设计技术的研究 [D]. 合肥: 合肥工业大学, 2003.

[2] 吴庆达, 何书专, 潘红兵, 等. 32 位浮点数正余弦函数 FPGA 实现方法[J]. 微电子学与计算机, 2012, 29(1): 113-116.

[3] 王少军, 张启荣, 彭宇, 等. 超越函数 FPGA 计算的最佳等距分段线性逼近方法[J]. 仪器仪表学报, 2014, 35(6): 1209-1215.

[4] 黄小康, 杜慧敏, 李涛, 等. 多核处理器中的超越函数协处理器设计[J]. 微电子学与计算机, 2016, 33(5): 42-46.

[5] 符茂松. 32 位 IEEE 和 IBM 浮点数结构及其转换方法[J]. 工程地球物理学报, 2011, 8(6): 759-766.

[6] 克里斯·斯皮尔. SystemVerilog 验证[M]. 张春, 麦宋平, 赵益新, 译. 北京: 科学出版社, 2009.

作者简介：

圣 飞 男, (1990-), 硕士研究生. 研究方向为电子与通信工程. E-mail: 1534528215@qq.com .

阴亚芳 女, (1966-), 博士, 教授, 硕士生导师. 研究方向为光信息传输技术.

秦晨蕊 女, (1993-), 硕士研究生. 研究方向为电路与系统.

张 凯 男, (1990-), 硕士研究生. 研究方向为电路与系统.