二进制数开任意正整数次方运算的硬件方法

赖大彧 刘荣科

(北京航空航天大学 电子信息工程学院, 北京 100191)

摘 要: 一种适合于硬件的、普适的、开任意次方的方法,从左至右进行计算,首先得到开方结果的高位,最后得到低位. 应用到 FPGA (Field Programmable Gate Array)中,与查表结合,对不同的开方次数,模块修改非常方便,普适性较高. 该方法消耗的时间与开方的次数呈线性关系. 在对一个 12 bit数开三次方时,消耗的时间不足传统循环搜索法的 50%,所消耗的存储空间不足传统的查表法的 2%,并且存储空间上的优势在被开方位数越大的时候越明显. 更重要的是,该方法所用的模块只需修改数据位宽和循环次数两处地方就可以提高到任意的精度.

关 键 词: 开方;硬件; Verilog HDL

中图分类号: TP 312

文献标识码: A **文 章 编 号**: 1001-5965 (2008) 08-0940-04

Novel method of extraction for binary based on hardware

Lai Dayu Liu Rongke

(School of Electronics and Information Engineering, Beijing University of Aeronautics and Astronautics, Beijing 100191, China)

Abstract: A method of any order extraction, works from left to right, which is based on hardware and calculates out the most significant bit(MSB) of the result first and the least significant bit(LSB) last. While applied in field programmable gate array(FPGA), the method can combine with looking up table(LUT). Using this method, the occupying time is linear to the extraction order. While doing a 3 order extraction for a 12 bit data, the occupying time of this method is 50% less than the traditional searching method, and the occupying memory is only 2% that of traditional method of LUT. As the order of extraction grows, the superiority will be more obvious. More important, the module only need to be modified in 2 point, the precision can be improved as wanted.

Key words: extraction; hardware; Verilog HDL

开方运算是非常常见的一种运算,硬件实现比较困难. 目前国内外主要有以下几种方法: ①查表法,即把被开方数及结果做成一张表,耗费大量存储空间,被开方数越大,表也越大,查表速度也越慢. 更重要的是,开平方的表只能运用于开平方,要做开立方或开高次方又需要建立新表,可重用性差. ②循环搜索法,即从 0开始搜索寻找,逐步逼近,精度不高. 被开方数越大,循环次数越多. ③Newton-Raphson 算法,要选择好迭代的初始值[1-2],其具体做法类似于解高次非线性方程的牛顿迭代[3]. 上述方法在推广到开高次方的时候都有一定的困难.

本文介绍一种新的开任意次方的方法,可以较快的速度达到较高的精度,且对不同的开方次数,模块修改方便.

1 开方基本方法

1.1 开任意次方

本文提到的方法可以用来开任意 n 次方 (n = 2, 3, 4…),具体步骤如下:

1) 将被开方数从低位开始 n位 n位点断,最高位不足 n位时,在前面添零补齐.

收稿日期: 2007-07-04

作者简介: 赖大彧 (1983 -),男,四川自贡人,硕士生, nirvanno@gmail.com.

2) 对当前最高 n位进行开方,即试商找出最高的 n位的 n次方根的整数部分.

3) 将被开方数的最高 n位减去试商结果的 n次方,得到第一位开方结果的余数.

4) 在前面开方余数后紧接着接被开方数的后续的 n位,构成计算第二位结果时开方需要试商的数 s_{try},设当前开方结果为 $x_{forward}$.

5) 循环试商,具体做法是找到一个 x_{lsb},使

$$C_n^{n-1}\cdot 2^{n-1}\cdot x_{forward}^{n-1}\cdot x_{lsb}^{1}+\cdots+C_n^{n-i}\cdot 2^{n-i}\cdot x_{forward}^{n-i}\cdot x_{lsb}^{i}+\cdots+C_n^{0}\cdot x_{lsb}^{n}=s_{try}-r\le s_{try}\quad(1)$$

成立. 其中, x_{lsb}是指现阶段需要试商位的值. 二进制中 x_{lsb}只可能是 0或 1,试商变得很简单.

6) 更新被试商数 s_{try},用 s_{try}减去式 (1)的运算结果,后面再接被开方数的后续的 n位,得到下一次需要试商的数 s_{try},同样更新当前的开方结果 $x_{forward}$.

7) 重复 5)、6)的过程,直到开方得到最后一位.

图 1用传统的竖式的方式,以 $n=3$为例说明该方法.

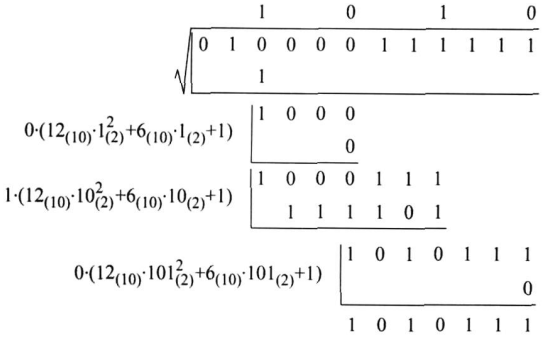


图 1　开立方的竖式形式

图中左边表示了试商的过程,根号最上面是开方结果. 可以看到,对二进制数 $10000111111_{(2)}$,即十进制的 $1\,084_{(10)}$,进行开立方运算之后,得到的结果是 $1010_{(2)}$,即十进制的 $10_{(10)}$,余数为 $87_{(10)}$. 其中,下标 (2)表示该数为二进制数, (10)表示为十进制数.

1.2　开方方法的理论证明

一个 m位的二进制数开 m次方,其结果整数部分只可能有 0或 1两种情况. 从高位到低位对一个数开 n次方,总可用 1.1节中第 2)步所述的方法得到第一位,后面位数,则可以用数学归纳法得到. 设

$$x=\overline{b_n b_{n-1}\cdots b_1 b_0}\quad(2)$$

$$s=x^n+r_0=s_0+r_0\quad(3)$$

其中 r_0表示开方结束之后的余数.

假设已经开方得到了前面的若干位,现在构造开方试商的方法计算下一位数. 设

$$s_{now}=(x_{forward}\cdot 2_{(10)}+x_{lsb})^n+r\quad(4)$$

二项式展开[4]:

$$s_{now}=C_n^n\cdot 2^n\cdot x_{forward}^n+C_n^{n-1}\cdot 2^{n-1}\cdot x_{forward}^{n-1}\cdot x_{lsb}^1+\cdots+C_n^{n-i}\cdot 2^{n-i}\cdot x_{forward}^{n-i}\cdot x_{lsb}^i+\cdots+C_n^0\cdot x_{lsb}^n+r\quad(5)$$

整理可得式 (1),即构造出试商时 x_{lsb}需要满足的条件.

而被试商数更新的过程实际上是用

$$s_{try}-(C_n^{n-1}\cdot 2^{n-1}\cdot x_{forward}^{n-1}\cdot x_{lsb}^1+C_n^{n-i}\cdot 2^{n-i}\cdot x_{forward}^{n-i}\cdot x_{lsb}^i+\cdots+C_n^0\cdot x_{lsb}^n)\quad(6)$$

得到余数,再接被开方数后面的位数.

以上推导即证明了该方法的正确性.

2　实现案例

以开三次方为例说明硬件构成. 为方便说明,定义:

$$r_{reg}=s_{0now}-8_{(10)}\cdot x_{forward}^2\quad(7)$$

$$q_{result}=x_{forward}\quad(8)$$

$$m=12_{(10)}\cdot r_{reg}^2+6_{(10)}\cdot r_{reg}+1\quad(9)$$

2.1　实现结构

根据前文叙述,给出程序流程如图 2所示.

用 FPGA (Field Programmable Gate Array)实现系统结构如图 3所示.

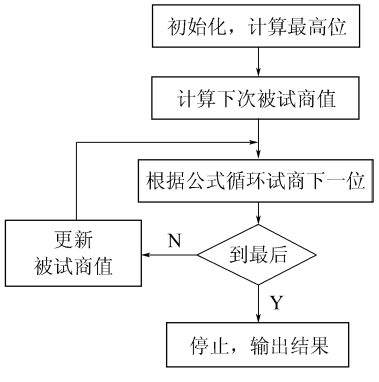


图 2　开三次方的程序流程图

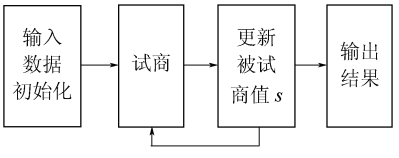


图 3　系统结构

每一部分功能如下: ①输入数据初始化,将数据输入到寄存器,并做好相应的"点断"准备工作,用试探的方法得到第一位开方数值; ②试商,不同的开方次数,根据不同的公式,得到试商的结

果,在硬件上就是比较 r_{reg}和 m 两个寄存器中数值的大小; ③构造下次试商所需要的被试商 r_{reg} 值,即用当前开方数减去当前开方结果的 n次方,再接上紧跟的被开方数后续的 n位; ④开方结束,输出开方结果.

本方法的关键在于循环试商的过程,该过程中需要计算式 (9). 该式较复杂,直接计算势必产生较大的时间延迟. 因此,可用查找表代替运算电路. 构造查找表的方法是将 r_{reg}作为输入地址线, m 作为输出. 特别地,在速度允许的条件下,可直接用运算电路计算式 (9),以减小资源消耗. 此处体现了速度与资源的矛盾,可根据实际情况选择.

2.2 综合及仿真结果

用 Synplify8.1进行综合,选用 Altera低端的 Cyclone芯片 EP1C3,约束频率为 170MHz,综合结果如图 4所示. 综合预计可以达到的频率为 177.8MHz,其中输入包括时钟、复位信号、被开方数据,输出包括开方结果和开方的余数.

根据设计,进行一个 12 bit二进制数的开方运算,所需要的时间为 8个时钟周期. 图 5以被开方数为 1080例表示仿真结果:开方结果为 10,余数为 80.

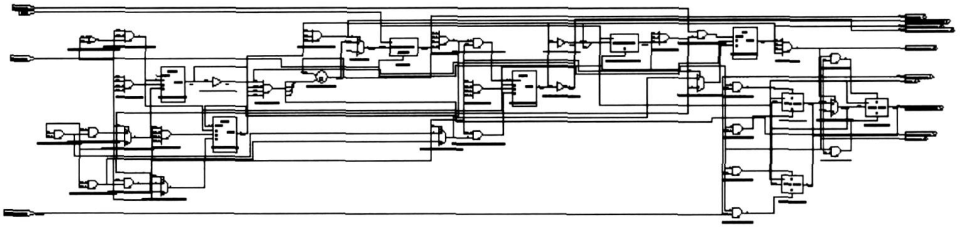


图 4 开三次方综合结果

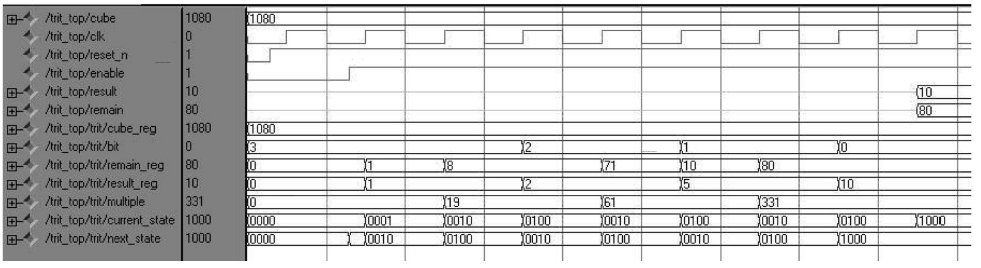


图 5 开立方波形图

3 方法性能分析

3.1 精度与开方次数的提高

该方法一个很大的优势在于可以方便地提高精度,开方到小数点后若干位,具体做法就是在被开方数的最后一位添上 3的倍数个零,开到整数部分的最后一位的时候,继续向后面运算,即可以得到更高的精度. 体现在模块上,只需要修改数据位宽和循环次数的计数器两处. 类似的,开 n次方就是添 n的倍数个 0.

该方法另一个很大的优势是模块可以很方便地修改以适应不同的开方次数,具体就是修改试商值 m 的计算.

$$m = C_n^{n-1} \cdot 2^{n-1} \cdot x_{\text{forward}}^{n-1} + C_n^{n-i} \cdot 2^{n-i} \cdot x_{\text{forward}}^{n-i} + \cdots + 1 \tag{10}$$

其中 n表示开方的次数. 硬件结构上只需要做很小的调整,可见该方法普适性较高.

3.2 与循环搜索法性能比较

对位数更多的二进制数开立方,同样根据该运算方法,对一个 k(k为 3的倍数)位的二进制数进行开立方运算,所耗的时间数量级约为 $2k/3$个时钟周期. 而如果应用循环搜索法,假定一次循环硬件实现的时候循环要消耗 2个时钟周期,即一个周期计算,一个周期比较,则对一个 k(k为 3的倍数)位的二进制数,所消耗的平均时间是

$$\frac{\sum_{i=0}^{2^{k/3}-1} (3i^2 + 3i + 1) \cdot 2 \cdot (i+1)}{2^k} \tag{11}$$

相应的也可以进行开四次方的比较.

表 1比较了开三次方和四次方时 k取不同值的时候,这两种方法所消耗的时钟周期.

表 1 循环计算和本文方法所耗时钟周期比较

k	开三次方		k	开四次方	
	循环方法	本文方法		循环方法	本文方法
3	3.75	2	4	3.875	2
6	6.875	4	8	7.0625	3
9	12.9375	6	16	14.469	4
12	24.9688	8	20	21.648	5

可见,该方法在位数越大时,优势越明显.

3.3 与查找表法性能比较

下面讨论查找表法用到的查找表所消耗的空

间. r_{reg}的宽度其实就是要构造的这张表的地址线的宽度,最后 m 为查表所需要输出的结果. 对于实际的一个 12 bi数的开方运算,式 (9)中 r_{reg}最多可能是 4 bit,也就是地址线是 4 bit,而一个 m 需要分配 12 bit的空间,总共所需要消耗的空间是 192 bit 可见,该表也很大.

考虑传统查表方法,同样构造一个用于 12 bit 数开三次方的表,地址线为 12 bit,开方结果整数部分为一个 4 bit的数,存储这些数据所需要的存储空间为

$$2^{12} \times 4 = 16\,384\ \text{bit} \qquad (12)$$

远大于上面方法的 192 bit,且被开方数位数增加后,资源消耗的增加幅度,远大于本文提到的方法.

4 结　论

本文提到的方法,可进行任意正整数次方的开方运算,且相比较于传统的循环搜索法,更容易提高精度. 相比于查表法,资源消耗要小很多. 最重要的是可以根据实际需求,非常方便地提高精度,且有非常强的普适性.

参考文献 (References)

[1] Li Yamin, Chu Wanminn. Parallel-array implementations of a non-restoring square root algorithm [C]// Computer Design: VLSI in Computers and Processors. Austin, TX: IEEE, 1997: 690 - 695

[2] 林志谋,卢贵主. 一种适合 FPGA 实现的开平方算法 [J]. 厦门大学学报, 2006, 45(2): 199 - 201
Lin Zhimou, Lu Guizhu. A square root algorithm based on FPGA [J]. Journal of Xiamen University, 2006, 45(2): 199 - 201 (in Chinese)

[3] Kincaid David, Cheney Ward. 数值分析 [M]. 王国荣译. 北京:机械工业出版社, 2005: 63 - 64
David Kincaid, Ward Cheney. Numeric analysis[M]. Translated by Wang Guorong. Beijing: China Machine Press, 2005: 63 - 64 (in Chinese)

[4] Hungerfor T W. 代数学 [M]. 冯克勤译. 湖南:湖南教育出版社, 1985: 177
Hungerfor T W. Algebra [M]. Translated by Feng Keqin. Hunan: Hunan Education Press, 1985: 177 (in Chinese)

(责任编辑:娄　嘉)

(上接第 929页)

[6] Wies R. Using a classification of management policies for policy specification and policy transformation[C]// Integrated Network Management. London: Chapman Hill, 1995: 44 - 56

[7] Katri Ylitalo. Policy core information model[EB/OL]. 2000. http://www.cs.helsinki.fi/u/kraatika/Courses/QoS00a/ylitalo.pdf

[8] Kim S Y, Kim M E, Kim K, et al. Information model for policy-based network security management[C]// Wired Communications and Management. Berlin: Springer, 2002, 2343: 662 - 672

[9] Tang Chenghua, Yao Shuping, Cui Zhongjie, et al. A network security policy model and its realization mechanism[C]// Information Security and Cryptology. Berlin: Springer, 2006, 4318: 168 - 181

[10] Bartal Y, Mayer A J, Nissim K, et al. Firmato: a novel firewall management toolkit[J]. ACM Transactions on Computer Systems, 2004, 22(4): 381 - 420

[11] Luck I, Schafer C, Krumm H. Model-based tool assistance for packet-filter design[C]// Policies for Distributed Systems and Networks. Berlin: Springer, 2001: 120 - 136

[12] Kalam A Abou El, Baida R El, Balbiani P, et al. Organization based access control[C]// Policies for Distributed Systems and Networks (POLICY03). Lake Como: IEEE CS Press, 2003: 120 - 131

[13] Cuppens F, Cuppens-Boulahia N, Sans T, et al. A formal approach to specify and deploy a network security policy[J]. Second Workshop on Formal Aspects in Security and Trust, 2004

[14] Debar H, Thomas Y, Boulahia-Cuppens N, et al. Using contextual security policies for threat response[C]//Detection of Intrusions and Malware & Vulnerability Assessment. Berlin: Springer, 2006, 4064: 109 - 128

[15] 许国志. 系统科学 [M]. 上海:上海科技教育出版社, 2000
Xu Guozhi. System science[M]. Shanghai: Shanghai Scientific and Technological Education Publishing House, 2000 (in Chinese)

[16] Nstissc. National information systems security (infosec) glossary[S]. Nstissi No 4009: Nstissc, 2000

[17] Sandhu R, Conyne E J, Lfeinstein H, et al. Role based access control models[J]. IEEE Computer, 1996, 29(2): 38 - 47

(责任编辑:赵海容)