(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 109521994 B
(45)授权公告日 2020.11.10

(21)申请号 201710852544.4
(22)申请日 2017.09.19
(65)同一申请的已公布的文献号
申请公布号 CN 109521994 A
(43)申请公布日 2019.03.26
(73)专利权人 华为技术有限公司
地址 518129 广东省深圳市龙岗区坂田华为总部办公楼
(72)发明人 徐斌 王开兴 田清霖
(74)专利代理机构 北京龙双利达知识产权代理有限公司 11329
代理人 魏雪娇 毛威
(51)Int.Cl.
G06F 7/523(2006.01)

(56)对比文件
CN 104011706 A,2014.08.27
CN 103369326 A,2013.10.23
CN 101133389 A,2008.02.27
US 2014188963 A1,2014.07.03

审查员 崔哲

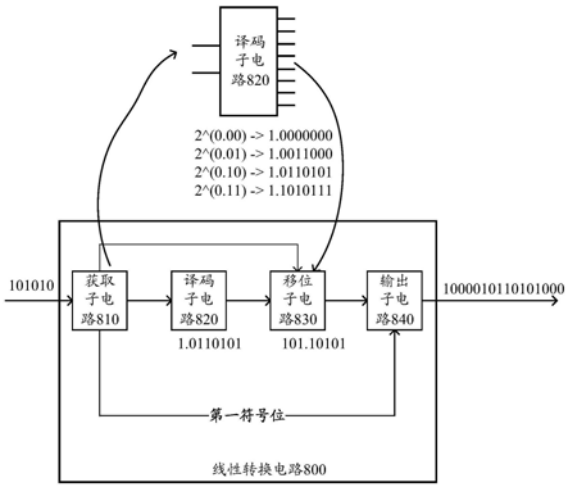
权利要求书3页 说明书20页 附图9页

(54)发明名称

乘法硬件电路、片上系统及电子设备

(57)摘要

本申请提供了一种乘法硬件电路、片上系统和电子设备，该硬件电路包括对数域加法器和线性域转换电路，其获取子电路获取对数域表示的数据，对数域表示是以数据的绝对值取以2为底的对数值表示数据的值的格式，数据在对数域表示格式下为1+m+n比特的二进制数；译码子电路译码得到第一小数位对应的线性表示格式，线性表示格式是以数据本身的值进行表示的格式；移位子电路根据对数值的整数部分的值M，对第一小数位对应的线性表示格式进行移位，得到数据的绝对值在线性表示格式下的值；输出子电路将数据表示为线性表示格式下的1+j+k比特的二进制数。本申请的硬件电路能够减小数据在线性域和对数域间转换时的开销，提高乘累加计算的速度。



CN 109521994 B

1.一种用于乘法运算的乘法硬件电路,包括:对数域加法器和线性域转换电路,所述线性域转换电路包括指数运算子电路、移位子电路和符号位确定子电路,

所述对数域加法器用于对第一对数域数据a1和第二对数域数据a2进行加法运算,得到第三对数域数据c1,其中,对数域数据是指对线性域数据进行对数域转换后得到的数据,所述线性域数据是指在进行所述对数域转换之前的数据,所述对数域转换是指将所述线性域数据的绝对值取以2为底的对数值并结合所述线性域数据正负符号位后得到的数据;所述a1和所述a2通过分别对两个用于乘法运算的第一线性域数据A1和第二线性域数据A2进行所述对数域转换得到,所述a1、所述a2和所述c1都为由1+m+n比特构成的二进制数,m和n均为正整数,1比特为第一符号位,用于指示正负符号,m比特用于指示整数部分的值,n比特用于指示小数部分的值;

所述指数运算子电路,用于根据所述c1的小数部分的值N1得到N',其中,所述N'的值是2的N1次方;

所述移位子电路,用于根据所述c1的整数部的值M1对所述指数运算子电路得到的所述N'进行移位,得到C1的绝对值,其中所述C1是所述A1乘以所述A2的乘积;

所述符号位确定子电路,用于根据所述a1的符号位的值以及所述a2的符号位的值确定所述C1的正负符号,并根据所述移位子电路得到的所述C1的绝对值以及所述C1的正负符号得到所述C1;

所述N'为一个由1+w比特构成二进制数,用于表示一个大于等于1小于2的小数,其中,1比特表示的数为小数的整数部分的值,w比特表示数为小数的小数部分的值;

所述移位子电路具体用于对所述N'左移M1-(w-k)位,得到最终移位结果,所述最终移位结果的最右k比特用于指示所述C1的绝对值的小数部分的值,所述最终移位结果的最右k比特左边的j比特用于指示所述C1的绝对值的整数部的值;或者,

所述移位子电路包括第一移位子电路以及第二移位子电路;所述第一移位子电路用于对所述N'左移M1位;所述第二移位子电路用于对所述第一移位子电路移位后的结果再左移-(w-k)位,得到最终移位结果,所述最终移位结果的最右k比特用于指示所述C1的绝对值的小数部分的值,所述最终移位结果的最右k比特左边的j比特用于指示所述C1的绝对值的整数部的值。

2.如权利要求1所述的乘法硬件电路,其特征在于,

所述指数运算子电路为译码子电路,所述译码子电路用于根据所述c1的所述N1译码得到所述N';或者

所述指数运算子电路为查表子电路,所述查表子电路用于根据所述c1的所述N1查表得到所述N'。

3.如权利要求1或2所述的乘法硬件电路,其特征在于,

所述乘法硬件电路还包括累加器,用于将所述C1与来自所述乘法硬件电路的另一个线性域数据C2进行累加运算;或者

所述累加器用于将所述C1与来自另一个乘法硬件电路的线性域数据C3进行累加运算。

4.一种片上系统SoC,包括处理器核,由一个或多个如权利要求1-3任一所述的乘法硬件电路构成的乘法硬件电路阵列、数据输入缓存、数据输出缓存以及控制电路;

所述控制电路与所述处理器核、所述乘法硬件电路阵列、所述数据输入缓存以及所述

数据输出缓存连接；

所述数据输入缓存用于通过所述控制电路获取来自所述处理器核的数据；

所述乘法硬件电路阵列用于获取所述数据输入缓存中的数据进行处理，得到处理后的结果，并通过所述控制电路将所述结果输出给所述数据输出缓存。

5.如权利要求4所述的SoC，其特征在于，还包括对数域转换电路，用于对所述乘法硬件电路阵列的输出进行所述对数域转换，并将转换后的结果输入给所述数据输入缓存。

6.如权利要求5所述的SoC，其特征在于，所述对数域转换电路包括：整数计算子电路、小数计算子电路和第二符号位确定子电路，其中，所述乘法硬件电路阵列的输出数据为由1+j+k比特构成的二进制数，其中，j和k均为正整数，1比特为第二符号位，用于指示正负符号S，j比特用于指示线性域数据的绝对值的整数部分的值J，k比特用于指示所述线性域数据的绝对值的小数部分的值K；

所述整数计算子电路，用于根据所述乘法硬件电路阵列的输出数据的j+k比特的二进制数的非零最高位所在的位数的值h1，计算h1与k的差值，所述差值用于表示所述A1的绝对值取以2为底的对数值的整数部分的值，其中，所述乘法硬件电路阵列的输出数据A1的j+k比特的二进制数的最低位记为第0位；

所述小数计算子电路，用于根据所述乘法硬件电路阵列的输出数据从高位到低位的非零最高位后的预定个数s比特，得到所述乘法硬件电路阵列的输出数据的绝对值取以2为底的对数值的小数部分的值；

第二符号位确定子电路，用于分别根据所述乘法硬件电路阵列的输出数据的符号确定所述对数域转换电路的输出数据的符号，从而得到所述对数域转换电路的输出数据。

7.如权利要求6所述的SoC，其特征在于，所述小数计算子电路具体用于：

通过查表或者译码得到所述A1从高位到低位的非零最高位后的s比特对应的值N1，通过查表或者译码得到所述A2从高位到低位的非零最高位后的s比特对应的值N2，其中，所述表中存储了所述s比特的所有可能值对应的值N。

8.如权利要求6所述的SoC，其特征在于，所述小数计算子电路具体用于：

将所述A1从高位到低位的非零最高位后的s比特对应的值与预设的2^n个比较值比较，其中，第i比较值比第i+1比较值小，且所述第i比较值对应一个值N_i；当所述A1从高位到低位的非零最高位后的s比特对应的值大于或等于第T1比较值，且小于第T1+1比较值时，确定N1为N_{T1}，所述N1为所述A1从高位到低位的非零最高位后的s比特对应的值；

将所述A2从高位到低位的非零最高位后的s比特对应的值与预设的2^n个比较值比较，其中，第i比较值比第i+1比较值小，且所述第i比较值对应一个值N_i；当所述A2从高位到低位的非零最高位后的s比特对应的值大于或等于第T2比较值，且小于第T2+1比较值时，确定N2为N_{T2}，所述N2为所述A2从高位到低位的非零最高位后的s比特对应的值。

9.如权利要求6所述的SoC，其特征在于，所述小数计算子电路具体用于：

将所述A1从高位到低位的非零最高位后的s比特的高x比特对应的值与预设的2^n个区间比较，其中，第i区间对应一对值αi和βi，x大于0并且小于s，当所述A1从高位到低位的非零最高位后的s比特的高x比特对应的值落入第一区间时，找到所述第一区间对应的一对值α1和β1，计算x*α1+β1的结果，根据所述x*α1+β1的结果得到N1，所述N1为所述A1从高位到低位的非零最高位后的s比特对应的值；

将所述A2从高位到低位的非零最高位后的s比特的高x比特对应的值与预设的2^n个区间比较，其中，当所述A2从高位到低位的非零最高位后的s比特的高x比特对应的值落入第二区间时，找到所述第二区间对应的一对值α2和β2，计算x*α2+β2的结果，根据所述x*α2+β2的结果得到N2，所述N2为所述A2从高位到低位的非零最高位后的s比特对应的值。

10.一种电子设备，其特征在于，包括如权利要求4-9中任一所述的SoC和存储器；

所述存储器用于存储程序运行所需的指令；

所述SoC中的处理器核用于执行指令运行程序，并将需要处理的数据发送给所述乘法硬件电路阵列；

所述乘法硬件电路阵列用于对所述数据进行处理后，将处理后得到的结果输出至所述数据输出缓存，并最终让所述处理器核获取所述结果。

乘法硬件电路、片上系统及电子设备

技术领域

[0001] 本申请涉及数据处理领域，并且更具体地，涉及乘法硬件电路、片上系统及电子设备。

背景技术

[0002] 在当今的信息时代，互联网和物联网应用每天都在产生大量的数据，对这些数据进行挖掘和处理往往能够获得有价值的信息。随着无人驾驶汽车、无人机、智能终端的普及，人工智能得到了广泛重视，它采用神经网络技术，对各种传感器输入的数据进行实时处理，实现对外界环境的感知。在这些数据处理的算法中，矩阵运算是一种核心计算模式，例如矩阵乘法，而矩阵乘法的基础是乘累加运算。利用一个典型的VGG16神经网络处理一张规模为224*224图片的计算量就是290亿次乘累加运算，这对当前的计算硬件与计算模式带来了严峻的考验。在乘累加运算中，乘法的代价最大，制约着矩阵运算的速度，也会影响设备的功耗。

发明内容

[0003] 本申请提供一种乘法硬件电路、片上系统及电子设备，能够减小数据在线性域和对数域之间转换时的开销，可以提高基于乘法的各种运算的速度。

[0004] 第一方面，本发明实施例提供了一种用于乘法运算的乘法硬件电路，具体的，这里的硬件电路是指基于ASIC、FPGA等实现的电路，而不是基于通用处理器(如基于x86、ARM架构的，需要读取指令来执行特定操作的处理器)来实现。当然，理论上也可以使用通用处理器来实现，但效率并不高，因此，为了更高效地处理数据，需要使用基于ASIC、FPGA实现的硬件电路。本实施例乘法硬件电路是指能够实现乘法的硬件电路，也不限定在乘法的基础上再进行一些其他的运算，例如，累加运算等。

[0005] 本实施例中的乘法硬件电路包括：对数域加法器和线性域转换电路，线性域转换电路包括指数运算子电路、移位子电路和符号位确定子电路；

[0006] 其中，对数域加法器用于对第一对数域数据a1和第二对数域数据a2进行加法运算，得到对数域数据c1，其中，对数域数据是指对将线性域数据的绝对值取以2为底的对数值并结合线性域数据正负符号位后得到的数据；a1和a2通过分别对两个用于乘法运算的第一数据A1和第二数据A2进行对数域变换得到，a1、a2和c1都为由1+m+n比特构成的二进制数，m和n均为正整数，1比特为第一符号位，用于指示正负符号(下文也简称“符号”，本申请并不进行区分)，m比特用于指示整数部分的值，n比特用于指示小数部分的值；

[0007] 本申请中，对数域数据与线性域是相对的关系，如果一个数据进行了对数域转换(即这个数据的绝对值取以2为底的对数值并结合该数据正负符号位后来进行表示)得到另一个数据，则称对数域转换前的数据为线性域数据，称转换后的数据为对数域数据。例如，对于一个数据-8，将-8的绝对值取以2为底的对数值为$\log_2|-8|=3$，再结合-8的正负符号(-)，从而得到-3，则称-8为线性域数据，-3为对数域数据。相对地，数据除了可以进行对数

域转换外，也可以反过来进行线性域转化(即取2的该数据绝对值次方，并结合该数据的符号位来表示)。例如，对于-3，先取2的|-3|次方等于8，然后结合符号位(-)得到-8。

[0008] 本申请中，前缀“第一”、“第二”等仅用于对修饰的同一类名词的不同个体进行区分，并不代表其他特殊的含义。例如，第一对数域数据以及第二对数域数据都是对修饰的名词“对数域数据”的不同个体进行区分，而不表示“第一对数域数据”是一种特定格式的对数域数据而“第二对数域数域”是另一种格式的对数域数据。本申请在提到“对数域数据”时，对数域数据具有的属性同时适用于“第一对数域数据”以及“第二对数域数据”。

[0009] 乘法硬件电路还包括：指数运算子电路，用于根据c1的小数部分的值N1得到N’，其中，N’的值是2的N1次方；即指数运算子电路用于通过计算2^{N1}来得到N’，需要说明的是，本申请中的计算过程很多时候都是具有误差的，这是因为在基于ASIC、FPGA的数字电路实现时，很多数(例如，有很多位的小数，甚至无理数)的表示会受到硬件的限制(不同位宽能表示的数据范围不同)。因此，本申请中的“N’的值是2的N1次方”需要理解成并非在所有情况下都是完全等于，而是基于特定硬件限制(如特定的位宽)下的等于，也即最后的结果可能是刚好等于，但也有可能是约等于。本申请下文中的其他各种运算也基于同样的原则，后续不再赘述。

[0010] 乘法硬件电路还包括：移位子电路，用于根据c1的整数部的值M1对指数运算子电路得到的N’进行移位，得到C1的绝对值，其中C1是A1乘以B1的乘积；需要说明的是，本申请中，左移一个负数位(如左移负3位)，则表示右移这个负数的绝对值位(如右移|-3|，即右移3位)；反之，右移一个负数位，则表示左移这个负数的绝对值位。

[0011] 乘法硬件电路还包括：符号位确定子电路，用于根据a1的符号位的值以及a2的符号位的值确定C1的正负符号，并根据移位子电路得到的C1的绝对值以及C1的正负符号得到C1。C1的符号本质上是由A1以及B1的符号决定，而A1以及B1的符号又分别决定了a1以及a2的符号，因此，C1的符号可由a1以及a2这两个对数域加法器获取的数据的符号决定，确定的原则是本领域技术人员公知的技术，即如果一个是正，一个是负，则相乘后的结果是负；如果两个都为正或者都为负，则相乘结果为正。

[0012] 在现有技术中，如果有两个数$V=(-1)^{S}\times Fraction\times 2^{Exp}$相乘，方法是将两个Fraction(分数)部分相乘，同时，将两个Exp(指数)相加，然后对Fraction乘法结果移位指数相加后得到的和。两个Fraction可以认为在线性域相乘，如果两个Fraction都用11比特来表示，则需要有一个11比特*11比特的乘法器，实现这个乘法器的代价比较大，例如，占用逻辑资源多，面积及功耗都会增大。

[0013] 而权1中，而是通过指数运算子电路、移位子电路以及符号位确定子电路的方式来实现相乘，此时，并不需要复杂的指数运算子电路，这些子电路的实现比乘法电路更节省资源，占用逻辑资源少，从而可以减少器件占用的面积以及功耗。

[0014] 针对加法：FP16(浮点16)位加法代价很大，例如有两个数$V1=(-1)^{S}\times Fraction1\times 2^{Exp1}$ $V2=(-1)^{S}\times Fraction2\times 2^{Exp2}$相加，则：

[0015] 由于exp不一样，不能直接加

[0016] 1)进行小数点对齐，exp1和exp2大的值，然后把小的值变成大的值；

[0017] 例如exp1＝5，exp2＝3；

[0018] 则将V2变成$V2=(-1)^{S}\times (Fraction2\times 2^{Exp2-Exp1})\times 2^{Exp1}$

[0019] 2)相加后得到$(-1)^{S}(Fraction1+(Fraction2\times 2^{Exp2-Exp1}))\times 2^{Exp1}$

[0020] 3)对$(Fraction1+(Fraction2\times 2^{Exp2-Exp1}))$进行归一化处理,1.xxx,会产生移位的偏移量x,加到2^{Exp1+x}

[0021] 4)表示成1.5.10标准格式,5比特$^{Exp1+x}$;10比特xxx,1比特符号位。

[0022] 结合第一方面,在第一方面第一种可能的实现方式中,所述N’为一个由1+w比特构成二进制数,用于表示一个大于等于1小于2的小数,其中,1比特表示的数为小数的整数部分的值,w比特表示数为小数的小数部分的值;

[0023] 所述移位电路具体用于对所述N’左移M1-(w-k)位,得到最终移位结果,所述最终移位结果的最右k比特用于指示所述所述C1的绝对值的小数部分的值,所述最终位移结果的最右k比特左边的j比特用于指示所述C1的绝对值的整数部的值。

[0024] 结合第一方面,在第一方面第二种可能的实现方式中,所述N’为一个由1+w比特构成二进制数,用于表示一个大于等于1小于2的小数,其中,1比特表示的数为小数的整数部分的值,w比特表示数为小数的小数部分的值,

[0025] 所述移位子电路包括第一移位子电路以及第二移位子电路;

[0026] 所述第一移位子电路用于对所述N’左移M1位;注:左移负数位其实就是右移正数位。

[0027] 所述第二移位子电路用于对所述第一移位子电路移位后的结果再左移-(w-k)位,得到最终移位结果,所述最终移位结果的最右k比特用于指示所述所述C1的绝对值的小数部分的值,所述最终位移结果的最右k比特左边的j比特用于指示所述C1的绝对值的整数部的值。

[0028] 结合第一方面及第一方面各种可能的实现方式,在第三种可能的实现方式中,所述指数运算子电路为译码子电路,所述译码子电路用于根据所述c1的所述N1译码得到所述N’;或者

[0029] 所述指数运算子电路为查表子电路,所述查表子电路用于根据所述c1的所述N1查表得到所述N’。

[0030] 结合第一方面及第一方面各种可能的实现方式,在第四种可能的实现方式中,

[0031] 所述乘法硬件电路还包括累加器,用于将所述C1与来自所述乘法硬件电路的另一个线性域数据C2进行累加运算;或者

[0032] 所述累加器用于将所述C1与来自另一个乘法硬件电路的线性域数据C3进行累加运算。

[0033] 第二方面,本申请公开了一种片上系统SoC,包括处理器核,由一个或多个如第一方面及第一方面各种实现方式中的乘法硬件电路构成的乘法硬件电路阵列、数据输入缓存、数据输出缓存以及控制电路;

[0034] 所述控制电路与所述处理器核、所述乘法硬件电路阵列、所述数据输入电路以及所述数据输出电路连接;

[0035] 所述数据输入电路用于通过所述控制电路获取来自所述处理器核的数据;

[0036] 所述乘法硬件电路阵列用于获取所述数据输入缓存中的数据进行处理,得到处理后的结果,并通过所述控制电路将所述结果输出给所述数据输出缓存。

[0037] 其中,乘法硬件电路阵列的构成(即选用多少个乘法硬件电路、以什么样的方式构

成等)是现有技术,本申请并不重点关注,本申请重点在于构成这个阵列的乘法硬件电路的具体实现。本申请中,输入缓存、输出缓存可以是由SRAM,eDRAM等存储介质实现。

[0038] 结合第二方面,在第二方面第一种可能的实现方式中,SoC还包括对数转换电路;

[0039] 对数转换电路,用于对所述乘法硬件电路阵列的输出进行所述对数域转换,并将转换后的结果输入给所述数据输入缓存。具体的,从输出缓存中获取数据,然后进行转化后输出给输入缓存,这样,后续乘法硬件电路阵列就可以从输入缓存中获取数据进行运算。每个PE中可以有一个对数转换电路,或者输入缓存中可以有一个对数转换电路,将数据转换为对数域数据后再发送给P,即数据从对数转换电路输出后再送至下一级,作为下一级的输入。PE可以从输入缓存获取数据也可以从其他PE和其他部件中获取数据。

[0040] 结合第二方面第一种可能的实现方式中,在第二方面第二种可能的实现方式中,所述对数域转换电路包括:整数计算子电路、小数计算子电路和第二符号位确定子电路,其中,所述线性域阵列输出数据为由1+j+k比特构成的二进制数,其中,j和k均为正整数,1比特为第二符号位,用于指示正负符号S,j比特用于指示所述线性域数据的绝对值的整数部分的值J,k比特用于指示所述线性域数据的绝对值的小数部分的值K;

[0041] 所述整数计算子电路,用于根据所述线性域阵列输出数据的j+k比特的二进制数的非零最高位所在的位数的值h1,计算h1与k的差值,所述差值用于表示所述线性域阵列输出数据的绝对值取以2为底的对数值的整数部分的值,其中,线性域阵列输出数据的j+k比特的二进制数的最低位记为第0位;

[0042] 所述小数计算子电路,用于根据所述线性域阵列输出数据从高位到低位的非零最高位后的预定个数s(大于等于k,如果不够s,补0)比特,得到所述线性域阵列输出数据的绝对值取以2为底的对数值的小数部分的值;具体的,可以通过查表或者译码的方法得到;

[0043] 第二符号位确定子电路,用于分别根据所述线性域阵列输出数据的符号确定所述对数域阵列输出数据的符号,从而得到所述对数域阵列输出数据。

[0044] 结合第二方面第二种可能的实现方式,在第二方面第三种可能的实现方式中,假设上面各个方面提到的A1,A2为所述线性域阵列输出数据,所述小数计算子电路具体用于:

[0045] 将所述A1从高位到低位的非零最高位后的s比特对应的值与预设的2^n个比较值比较,其中,第i比较值比第i+1比较值小,且所述第i比较值对应一个值N_i;当所述A1从高位到低位的非零最高位后的s比特对应的值大于或等于第T1比较值,且小于第T1+1比较值时,确定所述N1为N_{T1};

[0046] 将所述A2从高位到低位的非零最高位后的s比特对应的值与预设的2^n个比较值比较,其中,第i比较值比第i+1比较值小,且所述第i比较值对应一个值N_i;当所述A2从高位到低位的非零最高位后的s比特对应的值大于或等于第T2比较值,且小于第T2+1比较值时,确定所述N2为N_{T2}。

[0047] 结合第二方面第二种可能的实现方式,在第二方面第四种可能的实现方式中,所述小数计算子电路还可以有另一种实现方式,即小数计算子电路具体用于:

[0048] 将所述A1从高位到低位的非零最高位后的s比特的高x比特对应的值与预设的2^n个区间比较,其中,第i区间对应一对值αi和βi,x大于0并且小于s,当所述A1从高位到低位的非零最高位后的s比特的高x比特对应的值落入第一区间时,找到所述第一区间对应的一对值α1和β1,计算x*α1+β1的结果,根据所述x*α1+β1的结果得到所述N1;

[0049] 将所述A2从高位到低位的非零最高位后的s比特的高x比特对应的值与预设的2^n个区间比较，其中，当所述A2从高位到低位的非零最高位后的s比特的高x比特对应的值落入第二区间时，找到所述第二区间对应的一对值α2和β2，计算x*α2+β2的结果，根据所述x*α2+β2的结果得到所述N2。

[0050] 第三方面，本申请公开了一种电子设备(可以是手机、平板、智能手表、智能电视等各种电子设备)，其特征在于，包括第二方面及第二方面各种实现方式所述的SoC，存储器；

[0051] 所述存储器用于存储程序运行所需的指令；

[0052] 所述SoC中的所述处理器核用于执行所述指令运行程序，并将需要处理的数据发送给所述乘法硬件电路阵列；

[0053] 所述乘法硬件电路用于对所述数据进行处理后，将处理后得到的结果输出所述数据输出电路，并最终让所述处理器核获取所述结果。

附图说明

[0054] 图1是乘累加运算的示意图。

[0055] 图2是本申请实施例的ARM SoC架构的示意性框图。

[0056] 图3是计算引擎的结构的示意性框图。

[0057] 图4是本申请实施例的乘累加运算的示意性流程图。

[0058] 图5是本申请实施例的对数域表示格式的示意图。

[0059] 图6是本申请实施例的线性表示格式的示意图。

[0060] 图7是本申请实施例的转换数据格式的方法的示意性流程图。

[0061] 图8是本申请实施例的线性转换电路进行线性域转换的示意图。

[0062] 图9是本申请实施例的转换数据格式的方法的示意性流程图。

[0063] 图10是本申请实施例的对数转换电路进行对数域转换的示意图。

[0064] 图11是一种小数计算子电路确定小数部分的值的示意图。

[0065] 图12是本申请实施例的分段拟合的示意图。

[0066] 图13是另一种小数计算子电路确定小数部分的值的示意图。

[0067] 图14是本申请实施例的电子设备的示意性框图。

具体实施方式

[0068] 基于发明内容中的各方面及各方面相关的实现方式，下面将结合附图，对本申请中的技术方案进行具体描述。

[0069] 1.实施例一

[0070] 矩阵运算抽象成数学模式可以是乘累加运算，或者称为乘加运算。其中，P和Q均可以表示矩阵或向量。P·Q可以表示广义上的矩阵运算，包括矩阵*向量、矩阵*矩阵和向量*向量的卷积运算中的至少一种。P·Q的值可以是P中的元素p_i与Q中相应的元素q_i进行乘法运算得到乘积p_iq_i，再将这些乘积进行累加运算，这个过程即为乘累加运算。乘累加运算得到的是结果矩阵的一个元素。

[0071] 由于乘累加运算中乘法的运算量大，因而现有的一种方案是采用对数运算系统，即将数据从线性域(linear field)转化到对数域(log field)下进行表示并运算。其中，本

申请中，对数域以及线性域是指两种不同的数据表示格式，对数域是相对线性域来说的，即用对数域表示的数据(以下简称“对数域数据”、或者“对数数据”、或者“对数值”、或者“对数域值”等)是指将用线性域表示的数据(以下简称“线性域数据”、或者“线性数据”、或者“线性值”、或者“线性域值”等)的绝对值通过对数运算后转化成一个对数值(通常为了计算机计算方便会进行以2为底的对数运算，即进行$\log_2(|x|)$运算，其中，x是指线性域数据)并结合符号位来表示的数据。换而言之，对数域是对线性域数据进行对数域转换后得到的数据的表示方式，对数域转换是以线性域数据的绝对值取以2为底的对数值并结合正负符号位来表示的变换，线性域是以数据本身的值结合正负符号位来表示数据的表示方式。

[0072] 例如，要计算数据F和数据G的乘积F*G(以F和G均为正数为例)，可以先计算数据F和数据G的对数域表示格式，$\log_2(F)=f$，$\log_2(G)=g$则$F*G=2^f\times2^g=2^{f+g}$，此时，f、g即为对数域数据，相对的，F、G即为线性域数据。通过上述转化，可以将线性域数据F和G的乘法转换为对数域数据f和g的加法的运算，即f+g是乘积F*G的对数域表示格式，基于对数域结果f+g再转化成线性域数据(即通过例如移位、译码电路等方式计算2^{f+g})即可得到F*G的值。通过上述方法，可以将数据的乘法变为以数据的绝对值取以2为底的对数值的加法，从而避免乘法运算。虽然进行对数运算需要一定的开销，但是在矩阵运算中一个数据可能会参与多次乘法，因此计算一次对数域表示，之后多次使用，对于整个矩阵运算而言，计算开销也是降低的。

[0073] 现有的一种矩阵运算流程中，采用电气和电子工程师协会(Institute of Electrical and Electronics Engineers，IEEE)IEEE-754标准浮点数据表示格式，例如采用半精度的16比特(bit)浮点数据表示格式(或者采用单精度的32比特浮点数据表示格式)。

[0074] 浮点数据表示格式V通常包括符号位，指数位Exp和有效数位Fraction，其中，$V=(-1)^S\times Fraction\times2^{Exp}$。16比特浮点数据表示格式为1.5.10，1比特符号位指示数据的正负S，5比特指示指数位Exp，10比特指示有效数位Fraction。

[0075] 线性域数据F表示为$V_F=(-1)^{SF}\times FractionF\times2^{ExpF}$，线性域数据G表示为$V_G=(-1)^{SG}\times FractionG\times2^{ExpG}$。计算F和G相乘，需要将ExpF和ExpG相加得到初步的指数和，将FractionF和FractionG相乘，乘完后的结果再进行移位得到最终的指数和，最后通过F和G的符号位确定最终结果。其中，FractionF和FractionG是在线性域相乘，因此需要一个11比特*11比特的乘法器，该乘法器占用的计算资源多，导致计算引擎或计算芯片的面积大、功耗大。

[0076] 计算F和G相加代价也很大。首先需要将指数部分归一化，如果ExpF和ExpG大小不相等，则将其中较小的归一化为较大的来表示。例如，ExpF大于ExpG，则将V_G表示为$V_G=(-1)^{SF}\times[(-1)^{SG-SF}\times(FractionG\times2^{ExpG-ExpF})]\times2^{ExpF}$。为了简化说明，假设F和G的符号相同，正负均为S，则$V_F+V_G=(-1)^S\times[FractionF+(FractionG\times2^{ExpG-ExpF})]\times2^{ExpF}$。对有效部分$[FractionF+(FractionG\times2^{ExpG-ExpF})]$在进行归一化处理得到结果的值为1.o，会产生指数位的偏移量L。F和G的和最终表示为1.5.10格式，1比特为符号位，5比特的值为ExpF+L；10比特的值为o。同样，该加法运算复杂，也会导致计算引擎或计算芯片的面积大、功耗大。

[0077] 由此，在浮点数据表示格式下进行乘累加运算，会导致计算引擎或计算芯片的面积大、功耗大。

[0078] 图1示出了乘加运算的示意图,其步骤如下。

[0079] S110,输入两个对数域表示格式的数据,例如,线性域数据F和G的转成对数域数据后分别为f和g,f和g可以是16比特浮点数据表示格式下的数据,则线性域乘法运算F*G转换为对数域加法运算f+g。

[0080] S120,根据对数域下的加法结果(即f+g),采用标准的浮点指数运算电路计算2^{f+g},计算结果仍采用16比特浮点数据表示格式。

[0081] S130,将采用16比特浮点数据表示格式的S120中的数据与同为16比特浮点数据表示格式的其他数据做加法运算,得到累加和(SUM),仍采用16比特浮点数据表示格式。

[0082] 在很多场景中,一次乘加结果并不是矩阵运算的最终结果,例如在神经网络运算中,乘累加运算的结果(SUM)可能将会成为下一层计算的一个数据,需要将其转换为对数域下的表示格式,例如,可以采用标准的浮点对数运算电路计算$\log_2$(SUM),将结果保存成16比特浮点数据表示格式。

[0083] 在现有的这种矩阵运算流程中,采用标准的浮点数据表示格式,数据表达位宽较大,例如为16比特或者为32比特,浮点数据表示格式V通常包括符号位S,指数位Exp和有效数位Fraction。其中,$V=(-1)^S\times Fraction\times 2^{Exp}$,这种浮点数据表示格式较为复杂,不利于数据在线性域和对数域之间的快速转换。

[0084] 此外,标准的浮点指数运算单元和标准的浮点对数运算单元硬件资源消耗大,在浮点数据表达下,计算累加和的累加器(Accumulator)资源消耗也很大。

[0085] 基于以上问题,本申请实施例提供了一种转换数据格式的方法、电路、计算引擎和卷积计算芯片,能够减小数据在线性域和对数域之间转换时的开销,提高卷积计算的速度。

[0086] 换句话说,图1示出的步骤如下。

[0087] S110,输入两个对数域数据f和g,例如,线性域数据F和线性域数据G转成对数域数据后分别为f和g。则线性域乘法运算F*G转换为对数域加法运算f+g,其中,例如f和g可以是16比特浮点数据表示格式下的数据。

[0088] S120,根据对数域下的加法结果(即f+g),采用标准的浮点指数运算电路计算2^{f+g},计算结果仍采用16比特浮点数据表示格式。浮点指数运算电路的具体结构将在下文中说明。

[0089] S130,将采用16比特浮点数据表示格式的S120中的数据与同为16比特浮点数据表示格式的其他数据做加法运算,得到累加和(SUM),仍采用16比特浮点数据表示格式。

[0090] 在很多场景中,一次乘累加运算结果并不是矩阵运算的最终结果,例如在神经网络运算中,乘累加运算的结果(SUM)可能将会成为下一层计算的一个数据,需要将其转换为对数域下的表示格式,例如,如S140所示,可以采用标准的浮点对数运算电路计算$\log_2$(SUM),将结果保存成16比特浮点数据表示格式。

[0091] 指数运算电路和对数运算电路是运算中常用的电路,其利用PN结(Positive Negative junction,PN junction)的伏安特性所具有的指数规律,将二极管、三极管或晶体管分别接入集成运算放大器的反馈回路和输入回路,从而实现对数运算和指数运算。其中,对数运算电路和指数运算电路通常由二极管、三极管或晶体管以及集成运算放大器构成。由于二极管、三极管或晶体管的相关参数是温度函数,其运算精度受温度影响较大,所以通常要为对数运算电路和指数运算电路加上温度补偿电路,这使得指数运算电路和对数

运算电路变得更为复杂。

[0092] 在现有的这种矩阵运算流程中，采用标准的浮点数据表示格式，数据表达位宽较大，例如为16比特或者为32比特，因此，标准的浮点指数运算电路和标准的浮点对数运算电路相对于普通的指数运算电路和对数运算电路硬件资源消耗更大，实现时也更为复杂。此外，根据前文的描述，在浮点数据表示格式下，计算累加和的累加器(Accumulator)资源消耗也很大。

[0093] 基于以上问题，本申请实施例提供了一种转换数据格式的方法、电路、用于乘累加运算的方法、计算引擎和计算芯片，能够减小数据在线性域和对数域之间转换时的开销，提高乘累加运算的速度。

[0094] 下面先介绍应用本申请实施例的硬件架构。本申请实施例以卷积神经网络(Convolutional Neural Network，CNN)计算用于移动手机芯片的场景为例进行介绍。图2是本申请实施例的进阶精简指令集机器(Advanced RISC Machine，ARM)系统单芯片(System-on-a-chip，SoC)架构200的示意性框图。

[0095] 如图2所示，ARM SoC架构200例如包括主控中央处理器(Central Processing Unit，CPU)210、双倍速率(Double Data Rate，DDR)内存控制器220、先进可扩展接口(Advanced eXtensible Interface，AXI)总线230和硬件计算模块240。

[0096] 其中，硬件计算模块用于进行一些专用数据处理，即硬件计算模块用于进行例如针对图像、音频等数据的一些“专用的”的处理(如基于神经网络的机器学习)。硬件计算模块与通用的处理器(如CPU)相比，最大特点是基于各种逻辑电路(如与门、或门、非门等等)实现，而不像CPU那样具有一定的指令集(如x86指令集、ARM指令集)以执行指令的方式来完成数据处理。典型的硬件计算模块可以基于FPGA、ASIC等实现。

[0097] CPU通常有自己专用的指令集，用于通过执行指令的方式来执行除硬件计算模块执行的专用数据处理外的其他数据处理(当然，理论上也不限定使用CPU来执行硬件计算模块所执行的专利数据处理，但受限于CPU硬件架构，效率比较低)。

[0098] DDR内存又称为DDR SDRAM，即DDR同步动态随机存储器(Synchronous Dynamic Random Access Memory，SDRAM)。如图2所示，卷积计算芯片240包括输入缓存242、计算引擎244和输出控制模块246。CPU 210通过AXI总线230控制计算启动，卷积计算芯片240通过AXI总线230从DDR内存220获取需要处理的数据(如针对图像处理器时，获取图像数据以及和训练参数)，然后将数据送至计算引擎244，计算引擎244根据输入的数据内容计算并将计算结果写回DDR内存220，并通知CPU 210计算完成。

[0099] 本申请实施例的改进在于计算引擎244(可以以IP core的形式存在，另外，IP core也可以包括更多的电路，例如，包括输入缓存242等)。图3示出了计算引擎的结构的示意性框图。如图3所示，计算引擎244包括直接内存存取(Direct Memory Access，DMA)控制单元、数据缓存、参数缓存、多个处理元件(Processing Element，PE)(形成PE阵列)、输出缓存和对数转换电路。数据缓存和参数缓存可以认为是输入缓存，用于缓存图像数据和训练参数，图像数据和训练参数可以认为是数据。

[0100] 其中，PE可视为用于实现特定功能的电路，例如，本申请中，PE为乘法硬件电路，可进行基于乘法的各种运算(比如乘法、或者乘加运算)；乘法硬件电路(PE)中包括线性转换电路(也叫做线性转换单元，图3中用线性转换单元表示)。PE阵列(即乘法硬件电路)的构成

是现有技术，例如，可以像图3所示的方式进行构成，此时，阵列中第一级的PE(第一列以及第一行)为后续的PE传递数据，也可以让每个PE都连接输入缓存(参数缓存及数据缓存)。

[0101] 计算引擎244的计算过程中，DMA控制单元将需要的图像数据和训练参数从外部的DDR内存220读取到数据缓存和参数缓存；图像数据和训练参数通过PE阵列进行乘加运算；运算结果输出至输出缓存，然后通过对数转换电路将运算结果从线性域转换为对数域；最终结果可以输回到数据缓存，或者直接输出至DDR内存220进行存储。

[0102] 图4是乘加运算的示意性流程图。如图4所示，以实现F1*G1+F2*G2+F3*G3……为例，乘加运算包括以下步骤。

[0103] S410，在对数域表示格式下进行加法运算。

[0104] 数据F1和数据G1是线性表示格式的数据，在对数域表示格式下则分别为f1和g1，其中，$f1=\log_2(F1)$，$g1=\log_2(G1)$。输入两个对数域表示格式下的数据f1和g1，二者相加得到c1，即c1＝f1+g1，这样，将基于线性表示格式的乘法运算F1*G1转化成了基于对数域表示格式的加法运算f1+g1。应理解，所输入的对数域表示格式下的数据f1和g1可以是通过软件(即通过CPU等通用处理器来完成)或硬件(例如基于现场可编程门阵列(Field Programmable Gate Array，FPGA)或专用集成电路(Application Specific Integrated Circuit，ASIC)等硬件器件来完成)预处理好的。

[0105] 具体地，S410的加法运算可以由对数域加法器实现。对数域加法器用于对第一对数域数据f1和第二对数域数据g1进行加法运算，得到对数域和c1，f1和g1通过分别对两个用于乘法运算的第一乘法数据F1和第二乘法数据G1进行对数域变换得到，f1、g1和c1都为由1+m+n比特构成的二进制数，m和n均为正整数，1比特为第一符号位，用于指示正负S，m比特用于指示整数部分的值M，n比特用于指示小数部分的值N。其中，第一乘法数据F1和第二乘法数据G1的第一符号位分别表示F1和G1的符号S(正或者负)，其不参与加法运算。对数域数据的结构将在下文中详细说明。

[0106] S420，线性域转换。

[0107] 将对数域表示格式的c1＝f1+g1转换为线性表示格式C1，即计算2^{c1}，其中，$C1=2^{c1}$＝F1*G1。应理解，线性域转换是在每一个PE内部由线性转换电路执行的。现有的浮点数据表示格式下，线性转换电路是浮点指数运算单元，即计算2^{c1}是基于浮点指数运算单元进行的。由于浮点数据表示格式V是基于上文所描述的意义的，所以浮点指数运算单元计算2^{c1}的运算量巨大，运算速度慢。

[0108] S430，在线性表示格式下进行加法运算。

[0109] 将S420中得到的线性表示格式C1累加到已有的累加结果上。这样，反复执行S410和S420，得到Ci，其中，$Ci=2^{ci}$＝Fi*Gi，ci＝fi+gi，i取值1，2，3…，i的最大值由乘加运算的规模决定。累加结果SUM＝C1+C2+C3+…。图4中S430右侧指向自己的回路，表示将本次得到的Ci与上一次的累加结果进行累加。在得到累加结果SUM之后，如果下一步需要将SUM作为一个乘数继续进行乘法运算，则执行S440；如果下一步不再需要针对SUM进行运算，则可以通过输出缓存将SUM直接输出至DDR内存220进行存储。

[0110] S440，对数域转换。

[0111] 在得到累加结果SUM之后，如果下一步需要将SUM作为一个乘数继续进行乘法运算，则将SUM转换为对数域表示格式，即计算$\log_2(SUM)$，将其作为输入数据输回到数据缓

存,以便于再次执行S410。应理解,对数域转换是在PE外部由对数转换电路执行的。

[0112] 2.实施例二(对数域表示格式)

[0113] 本申请实施例主要涉及线性域转换和对数域转换的改进,亦即涉及线性转换电路和对数转换电路。

[0114] 在本申请实施例中,对数域表示格式是以数据的绝对值取以2为底的对数值(即$\log_2()$)表示数据的值的数据表示格式。图5是本申请实施例的对数域表示格式的示意图。如图5所示,数据在对数域表示格式下表示为1+m+n比特(bit)的二进制数据,以下记为1.m.n。其中,m和n均为正整数,1比特为第一符号位S,用于指示数据的值的正负,m比特为第一整数位,用于指示数据的绝对值取以2为底的对数值的整数部分的值M,值M可以认为是第一表示值,n比特为第一小数位,用于指示数据的绝对值取以2为底的对数值的小数部分的值N。

[0115] 线性表示格式的数据F与对数域表示格式1.m.n之间的关系如下述式2所示。

[0116] $F=(-1)^S 2^{M+N}=(-1)^S 2^N<<M$ (式2)

[0117] 其中,"<<"为左移符号。相应的,<<M表示向左移M位,具体的,左移M位是指:当M大于0时,表示向左移M位;当M小于0时,表示向右移M的绝对值位。第一符号位S表示F的符号(正或者负),其不参与数据在对数域表示格式下的运算。对F是负数的情况,由于在实数域下直接计算$\log_2$(负数)是不存在的,因此本申请实施例中1.m.n格式(对数域表示格式)表达的是$-\log_2(|F|)$。

[0118] 应理解,本申请实施例中,如果数据的绝对值取以2为底的对数值为正数,例如为2.35,则数据的绝对值取以2为底的对数值的整数部分的值M为2,数据的绝对值取以2为底的对数值的小数部分的值N为0.35;如果数据的绝对值取以2为底的对数值为负数,例如为-2.35,则数据的绝对值取以2为底的对数值的整数部分的值M为-3,数据的绝对值取以2为底的对数值的小数部分的值N为0.65。

[0119] 本实施例中,M又可包括以下两种表示方式。需要说明的是,为了方便阅读,本申请中的一些数字中间会有一些空格将数字进行分隔,例如,10010101010通过空格进行分隔后被表示成1 0010 101010,这些空格仅用于表示格式,并不表示真实的数据。

[0120] M表示方式一

[0121] 2.1对数域表示格式细化1(第一整数位的值为非负数)

[0122] 第一整数位的值可选地可以为非负数,第一整数位的值等于数据(例如,F或G)的绝对值取以2为底的对数值的整数部分的值M与基准值(BASE)的差。

[0123] 具体地,可以理解,数据的绝对值取以2为底的对数值的整数部分的值M可以是正数也可以是负数。为了数据表达上的简便,第一整数位可以设置为没有符号位,而是用第一整数位的值M'=M-BASE表示真实的整数部分的原始数据M,值M'可以认为是第二表示值。相应地,要得到数据的绝对值取以2为底的对数值的整数部分的值M,就需要通过M=M'+BASE来计算。换句话说,由于对于不同的数据而言,其对应的M有可能是负的,为了防止在数据表达上出现该负值,在M的基础上减去BASE,使得对数域表示格式的第一整数位的值M'始终保持为非负数。

[0124] 在本申请实施例中,BASE的选择是可配置的,在不同批次的乘加运算中BASE的取值可以不相等。该基准值BASE的取值原则是使得其适用的所有的数据(例如某一批次的数

据)对应的M'均为非负数。例如,对于一批特定属性的数据,例如商品买卖数据或医疗数据等等,其取值只可能在一定的范围内,那么这批数据所对应的M也在一定的范围内。例如,当真实的整数部分的原始数据M的范围为-7～0时,m为3bit,则BASE可以设置为-7,由此就可以保证第一整数位的值(M'＝M-(-7))的范围是0～7。在实际运算时,可以采用M'对应回到线性表示格式进行加法运算,在完成加法运算后再考虑BASE;也可以直接采用M对应回到线性表示格式进行加法运算,本申请实施例对此不作限定。在本申请实施例中,可以通过线性转换电路的外部元件配置BASE,即通过外部元件将BASE传入线性转换电路,并在线性转换电路中软件的编译过程中确定该BASE值可用。

[0125] 例如,十进制下的数据0.25取以2为底的对数的结果$\log_2(0.25)$＝-2,即M＝-2。可以取BASE的值为-2,则M'＝M-(BASE)＝-2-(-2)＝0。因此十进制下的数据0.25在对数域表示格式1.3.2下表示为0 000 00。

[0126] 由此,防止了数据在对数域表示格式的表达上在整数位出现负值,整数位不用单独设置一位符号位,可以使数据表达更简洁。

[0127] M表示方式二

[0128] 2.2对数域表示格式细化2(第三符号位)

[0129] 第一整数位可选地可以包括1比特第三符号位,第三符号位用于指示数据的绝对值取以2为底的对数值的整数部分的正负,第一整数位的剩余的m-1比特的值等于数据的绝对值取以2为底的对数值的整数部分的绝对值。

[0130] 例如,十进制下的数据0.25取以2为底的对数值的结果$\log_2(0.25)$＝-2,即M＝-2。将对数域表示格式1.3.2中的3比特整数位的最高位作为第三符号位,因此十进制下的数据0.25在对数域表示格式1.3.2下表示为0 110 00。

[0131] 由此,对数域表示格式中的第一整数位包括1比特第三符号位,可以精确表示数据的整数部分的正负和大小。

[0132] 特别地,当数据F＝0时,$\log_2(F)$是负无穷,采用1.m.n格式不便于表示和后续计算,所以可以采用特殊形式表示。例如,F＝0的对数域表示格式可以为100000…,即第一符号位S是1,第一整数位的值和第一小数位的值全部为0。

[0133] 3.线性表示格式

[0134] 在本申请实施例中,线性表示格式是以数据本身的值进行表示的数据表示格式。图6是本申请实施例的线性表示格式的示意图。如图6所示,线性表示格式下的1+j+k比特的二进制数据,以下记为1.j.k。其中,j和k均为正整数,1比特为第二符号位S,用于指示所述数据的正负,j比特为第二整数位,用于指示所述数据的绝对值的整数部分的值J,k比特为第二小数位,用于指示所述数据的绝对值的小数部分的值K。

[0135] 应理解,在本申请实施例中,第二整数位的值可以等于数据的绝对值的整数部分的值J,第二小数位的值可以等于数据的绝对值的小数部分的值K。

[0136] 例如,十进制下的数据0.25在线性表示格式1.3.3下表示为0 000 010。

[0137] 4.对数+线性表示格式示例

[0138] 下面举几个具体的例子来说明本申请实施例的数据的对数域表示格式和线性表示格式。为了简单起见,假设数据的绝对值取以2为底的对数值的整数部分的值M为非负的,则BASE值可以取0。当然,BASE也可以取-1或小于-1的其他值,本申请实施例对此不作限定。

[0139] 以对数域表示格式为1.3.2为例，数据在对数域的二进制表示格式为0 010 10，其中，用1比特(0)表示符号位，其中0表示正数，1表示负数，用3比特(010)表示整数部分，用2比特(10)表示小数部分。该数据在二进制下的值为(+)2^(0.10)<<(2-0)＝(+)101.10101，在线性表示格式1.3.5下为0 101 10101，其中，在线性域表示的格式中，用1比特(0)表示符号位，用3比特(101)表示整数部分以及用5比特(10101)表示小数部分。

[0140] 数据在对数域表示格式1.3.2下为1 011 10，该数据在二进制下的值为(-)2^(0.10)<<(3-0)＝(-)1011.0101，在线性表示格式1.4.4下为1 1011 0101。

[0141] …

[0142] 对数域表示格式1.3.2下正数的最大值为011111，该数据在二进制下的值为(+)2^(0.11)<<(7-0)＝(+)11010111，在线性表示格式1.8.0下为0 11010111。

[0143] 对数域表示格式1.3.2下正数的最小值为000000，该数据在二进制下的值为(+)2^(0.00)<<(0-0)＝(+)1.0000000，在线性表示格式1.1.7下为0 1 0000000。

[0144] 对数域表示格式1.3.2下负数的最大值为100001，该数据在二进制下的值为(-)2^(0.01)<<(0-0)＝(-)1.0011000，在线性表示格式1.1.7下为1 1 0011000。

[0145] 对数域表示格式1.3.2下负数的最小值为111111，该数据在二进制下的值为(-)2^(0.11)<<(7-0)＝(-)11010111，在线性表示格式1.8.0下为1 11010111。

[0146] 因此，本申请实施例的数据的对数域表示格式可以用较小的位数表示很大的数值范围。在实际应用中，根据图像数据的特性、训练参数的特性和中间结果所需的精度，估计所需表达的数据可能取的负数最小值～正数最大值的范围。从而确定对数域表示格式1.m.n中m的取值和n的取值。例如，图像数据或训练参数的绝对值取以2为底的对数值越大，m的取值应越大；精度要求越高，n的取值应越大。应理解，该m的取值和n的取值的确定过程可以由软件(例如，可以通过CPU等通用处理器完成)来实现，本申请实施例对此不作限定。

[0147] 5.线性域转化流程

[0148] 下面详细说明本申请实施例的转换数据格式的方法700，即图4中的S420所示的线性域转换的流程。图7是本申请实施例的转换数据格式的方法700的示意性流程图。方法700可以由线性转换电路800执行。图8是本申请实施例的线性转换电路800进行线性域转换的示意图。如图8所示，线性转换电路800可以包括获取子电路810、译码子电路820、移位子电路830和输出子电路840。各子电路可以基于FPGA或ASIC实现。其中，子电路810～840分别用于执行方法700的S710～S740。

[0149] 本实施例转换数据格式的方法700包括：

[0150] S710，获取子电路810获取对数域表示格式1.m.n的数据。其中，1比特为第一符号位，m比特为第一整数位，n比特为第一小数位。这里的数据可以包括图像数据和/或训练参数。具体地，PE中的线性转换电路可以从数据缓存获取对数域表示格式的图像数据；和/或从参数缓存获取对数域表示格式的训练参数。

[0151] 例如，获取子电路810获取对数域表示格式1.3.2的数据1 010 10，其中，第一符号位为1即数据符号为负，第一整数位为010，第一小数位为01。

[0152] S720，译码子电路820查表得到第一小数位对应的线性表示格式。具体地，译码子电路820从获取子电路810得到n比特第一小数位，进行译码操作。其中，译码操作通过硬件组合逻辑直接得到2^N的结果，即通过查表直接得到2^N的结果。应理解，由于n比特第一小数位

的取值为有限个，第一小数位对应的线性表示格式(在一定的精度要求下)是可以列举的。仍以对数域表示格式1.3.2为例进行说明，$2^{0.00}$对应于二进制的1.0000000，$2^{0.01}$对应于二进制的1.0011000，$2^{0.10}$对应于二进制的1.0110101，$2^{0.11}$对应于二进制的1.1010111，这里N是用二进制来表示的，以上对应关系是预先计算好存储在表中的。

[0153] 译码子电路820查表得到第一小数位10对应的线性表示格式1.0110101。在一个具体的例子中，译码子电路820可以是图8所示的，两位有效位输入八位有效位输出的器件，其中存储有两位有效位输入与八位有效位输出的对应关系表。应理解，八位有效位输出是本例中的情况，可根据实际需要增加或减少比特。

[0154] 应理解，当n＝2时，第一小数位共有4组不同情况(00,01,10,11)，通过事先计算2^N的结果，将结果四舍五入至8比特记录保存成表如图8以及如下：

[0155] 00--10000000(代表线性表示格式的数值1.0000000)

[0156] 01--10011000(代表线性表示格式的数值1.0011000)

[0157] 10--10110101(代表线性表示格式的数值1.0110101)

[0158] 11--11010111(代表线性表示格式的数值1.1010111)

[0159] S730，移位子电路830根据对数值的整数部分的值M，对第一小数位对应的线性表示格式进行移位，得到数据的绝对值在线性表示格式下的值。具体地，移位子电路830从获取子电路810得到m比特第一整数位，从译码子电路820得到2^N的结果，对译码子电路820译码的结果进行移位操作。m比特第一整数位的值为M'，m比特第一整数位所指示数值M，M是数据的绝对值取以2为底的对数值的整数部分的值，M＝M'+BASE。如果想得到真实的数据，则应该根据M对译码子电路820译码的结果进行移位。应理解，由于对数域表示格式1.m.n是应用于计算引擎中或卷积计算芯片中的，因此可以预先设置移位子电路830从获取子电路810获取m比特第一整数位，或者设置用户接口以方便用户设置m和/或n的值。

[0160] 如前文描述的，当采用M运算时，M是正数表示左移M位，M是负数则表示右移M的绝对值位。即，S730根据对数值的整数部分的值M，对第一小数位对应的线性表示格式进行移位，得到数据的绝对值在线性表示格式下的值，可以包括：当M大于0时，对第一小数位对应的线性表示格式左移M位，得到数据的绝对值在线性表示格式下的值；当M小于0时，对第一小数位对应的线性表示格式右移M的绝对值位，得到数据的绝对值在线性表示格式下的值。

[0161] 当然，本申请实施例中，也可以不采用M而是采用M'运算，M'为非负值，则只能进行左移。

[0162] 移位子电路830根据第一整数位所指示的值010，对第一小数位10对应的线性表示格式1.0110101左移两位，得到101.10101。

[0163] S740，输出子电路840将数据表示为线性表示格式下的1+j+k比特的二进制数据。具体地，输出子电路840从获取子电路810得到1比特第一符号位，将第一符号位的值赋予第二符号位。换句话说，输出子电路840根据第一符号位设置第二符号位。例如，如果数据是正数，第二符号位设置为0；如果数据是负数，第二符号位设置为1，但本申请实施例对此不作限定。输出子电路840从移位子电路830得到移位后的结果，对移位后的结果进行补零或者删除无效位，使之符合1.j.k的格式。可选地，输出子电路840可以将得到的1.j.k格式的结果转换为补码表示得到最终的结果，本申请实施例对此不作限定。

[0164] 输出子电路840进行补零和根据第一符号位确定第二符号位，将数据表示为线性

表示格式下的1+j+k比特的二进制数据，例如，表示为线性表示格式1.7.8下的1000010110101000。

[0165] 本申请实施例的转换数据格式的方法，获取对数域表示格式的数据，对本申请实施例的对数域表示格式的数据进行简单的查表移位的方式得到其在线性表示格式下的表示，无需进行复杂的幂运算，能够减小数据在对数域和线性域之间转换时的开销，提高卷积计算的速度。

[0166] 6.线性域转化细化

[0167] 在图8的例子中，获取子电路810获取对数域表示格式1.3.2的数据101010，其中，第一符号位为1即数据符号为负，第一整数位为010，第一小数位为01。

[0168] 译码子电路820查表得到第一小数位10对应的线性表示格式1.0110101。在一个具体的例子中，译码子电路820可以是图8所示的，两位有效位输入八位有效位输出的器件，其中存储有两位有效位输入与八位有效位输出的对应关系表。应理解，八位有效位输出是本例中的情况，可根据实际需要增加或减少bit。

[0169] 应理解，当n＝2时，第一小数位共有4组不同情况(00,01,10,11)，通过事先计算2^N的结果，将结果四舍五入至8比特记录保存成表如图8以及如下：

[0170] 00--10000000(代表线性表示格式的数值1.0000000)

[0171] 01--10011000(代表线性表示格式的数值1.0011000)

[0172] 10--10110101(代表线性表示格式的数值1.0110101)

[0173] 11--11010111(代表线性表示格式的数值1.1010111)

[0174] 移位子电路830根据第一整数位所指示的值010，对第一小数位10对应的线性表示格式1.0110101左移两位，得到101.10101。

[0175] 输出子电路840进行补零和根据第一符号位确定第二符号位，将数据表示为线性表示格式下的1+j+k比特的二进制数据，例如，表示为线性表示格式1.7.8下的1000010110101000。

[0176] 下面详细说明本申请实施例的转换数据格式的方法900，即对数域转换的流程。图9是本申请实施例的转换数据格式的方法900的示意性流程图。方法900可以由对数转换电路100执行。图10是本申请实施例的对数转换电路1000进行对数域转换的示意图。如图10所示，对数转换电路1000可以包括获取子电路1010、整数计算子电路1020、小数计算子电路1030和输出子电路1040。各子电路可以基于FPGA或ASIC实现。其中，子电路1010～1040分别用于执行方法900的S910～S940。

[0177] 7.对数转化流程

[0178] S910，获取子电路1010获取线性表示格式1.j.k的数据。其中，1个比特用于表示第二符号位，j个比特用于表示第二整数位，k个比特用于表示第二小数位。这里的数据可以是从输出缓存输出的经过累加运算后的数据，也可以是其他方式得到的数据，本申请实施例对此不作限定。

[0179] 例如，如图10所示，获取子电路1010获取线性表示格式1.7.8的数据0001100111000000(十进制表示为25.75)，其中数据符号为正，第二整数位为0011001，第二小数位为11000000。

[0180] S920，整数计算子电路1020确定j+k比特的二进制数据的非零最高位所在的位置

为第h位(j+k比特的二进制数据的最低位的位数记为第0位)，确定数据在对数域表示格式下的数据的绝对值取以2为底的对数值的整数部分的值M，其中，M等于h与k的差值。

[0181] 例如，仍以线性表示格式1.7.8的数据0 0011001 11000000为例，整数计算子电路1020对0 0011001 11000000首先找到最高位1的位置，从右往左从0开始编码，最高位1是在第h＝12位，因此数据的绝对值取以2为底的对数值的整数部分的值M＝h-k＝12-8＝4(二进制表示为0100)。

[0182] 为避免1.m.n格式下m比特的值可能出现负数，可以设置BASE，由M’＝M-BASE得到m比特第一整数位。

[0183] S930，小数计算子电路1030由高位向低位截取非零最高位后的s比特，得到s比特对应的对数域表示格式下的数据的绝对值，将该绝对值取以2为底的对数值，则得到小数部分的值N。

[0184] 例如，仍以线性表示格式1.7.8的数据0 0011001 11000000为例，小数计算子电路1030截取非零最高位后的s比特(例如为8比特)为10011100，计算$\log_2(1.10011100)$得到对应的对数域表示格式下的数据的绝对值取以2为底的对数值的小数部分的值N为0.11，数据的绝对值取以2为底的对数值的小数部分的值N在二进制下表示为0.1011。

[0185] 由于表示一个数的比特数(如8位或者16位等)是有限的，s比特的取值也只可能是有限个，所以，在一定的精度要求下，s比特对应的对数域表示格式是可以穷举的。小数计算子电路1030截取非零最高位后的s比特是根据获取子电路1010得到线性表示格式1.j.k的数据的非零最高位后的s比特的值y，然后计算$\log_2(1.y)$，得到数据的绝对值取以2为底的对数值的小数部分的值N。得到N的方式可以有多种，将在下文中举例说明。在具体地实现中，可以根据精度需要确定s的大小。可选地，s大于n。

[0186] 在一个具体的例子中，小数计算子电路1030可以是图10所示的，八位有效位输入、五位有效位输出的器件，应理解，五位有效位输出是本例中的情况，可根据实际需要增加或减少比特。

[0187] S940，输出子电路1040将数据表示为对数域表示格式下的1+m+n比特的二进制数据。

[0188] 具体地，输出子电路1040从获取子电路1010得到1比特第二符号位，将第二符号位的值赋予第一符号位，即根据第二符号位设置对数域表示格式下的第一符号位。例如，如果数据是正数，第一符号位设置为0；如果数据是负数，第一符号位设置为1，但本申请实施例对此不作限定。另外，输出子电路1040将整数计算子电路1020得到的数据的绝对值取以2为底的对数值的整数部分和小数计算子电路1030得到的数据的绝对值取以2为底的对数值的小数部分相加，分别得到第一整数位和第一小数位，并进行补零或者删除无效位，使之符合1.m.n格式。可选地，输出子电路1040可以将得到的1.m.n格式的结果转换为补码表示，本申请实施例对此不作限定。

[0189] 输出子电路1040将数据的绝对值取以2为底的对数值的整数部分的值M和数据的绝对值取以2为底的对数值的小数部分的值N相加，对相加的结果进行补零或者删除无效位。根据第二符号位确定第一符号位，将数据表示为对数域表示格式下的1+m+n比特的二进制数据，例如，表示为对数域表示格式1.3.4下的010010(第一符号位为0，第一整数位取有效的100三位，第一小数位取有效的1011四位)。

[0190] 本申请实施例的转换数据格式的方法，获取线性表示格式的数据，通过简单的截取并比对的方式得到其在对数域表示格式下的表示，无需进行复杂的对数运算，能够减小数据在线性域和对数域之间转换时的开销，提高卷积计算的速度。

[0191] 8.实施例三

[0192] 基于以上各实施例，本实施例对实施例二中的S930中的步骤进行详细说明。

[0193] 8.1方法一

[0194] 可选地，作为一个实施例，上述步骤S930中的得到s比特对应的对数域表示格式下的数据的绝对值取以2为底的对数值的小数部分的值N，可以包括：查表得到s比特对应的对数域表示格式下的数据的绝对值取以2为底的对数值的小数部分的值N，其中，表中存储了s比特的所有可能值对应的N。该确定数据的绝对值取以2为底的对数值的小数部分的值N的方法称之为查表法。

[0195] 具体地，仍以图10的例子来进行说明，小数计算子电路1030截取非零最高位后的8比特为10011100，查表或译码得到$\log_2(1.10011100)$的结果。其中，小数计算子电路1030中存储有八位有效位输入与五位有效位输出的对应关系表。表中记录了$\log_2(1.y)$的结果，结果保留小数位后4bit。

[0196] $\log_2(1.00000000)=0.0000$，对应的输出结果为0.0000

[0197] $\log_2(1.00000001)=0.0000$，对应的输出结果为0.0000

[0198] …

[0199] $\log_2(1.11111111)=1.0000$，对应的输出结果为1.0000(四舍五入后值为1.0000)

[0200] 由于1.y中的整数位1是固定的，输出结果的整数位0也是固定的，因此可以保存成一个8比特输入，4比特输出的表，本例子中查表$\log_2(1.10011100)$得到结果是数据的绝对值取以2为底的对数值的小数部分的值N用二进制表示为0.1011。

[0201] 8.2方法二

[0202] 可选地，作为另一个实施例，上述步骤S930中的得到s比特对应的对数域表示格式下的数据的绝对值取以2为底的对数值的小数部分的值N，可以包括：将s比特对应的值与预设的2^n个比较值比较，其中，第i比较值比第i+1比较值小，且第i比较值对应一个值N_i；当s比特对应的值大于或等于第T比较值，且小于第T+1比较值时，确定s比特对应的对数域表示格式下的数据的绝对值取以2为底的对数值的小数部分的值N为N_T。该确定数据的绝对值取以2为底的对数值的小数部分的值N的方法称之为逐级比较法。

[0203] 具体地，仍以图10的例子来进行说明，小数计算子电路1030截取非零最高位后的8比特为10011100，通过比较器组比较得到数据的绝对值取以2为底的对数值的小数部分的值N。图11是一种小数计算子电路1030确定小数部分的值N的示意图。其中，比较器组中可以包括$2^n=16$个比较器，例如，比较器0，比较器1，…，比较器15。每个比较器中预先设置有比较值。预先设置的比较值是从小到大排列的，即比较值0<比较器1<…<比较器15。比较值的设置也根据下表设置。

[0204] $\log_2(1.00000000)=0.0000$，对应的输出结果为0.0000

[0205] $\log_2(1.00000001)=0.0000$，对应的输出结果为0.0000

[0206] …

[0207] $\log_2(1.11111111)=1.0000$，对应的输出结果为1.0000

[0208] 将输出结果4比特有跳变的那一项的查表值，即$\log_2()$中的真数设置为比较值。例如，比较值0设置为1.00000000。可选地，直接将比较值0设置为00000000。换而言之，任意两个相邻的比较值之间的值对应的输出结果是一致的。因此，逐级比较的结果通过一个选择器可以直接得出数据的绝对值取以2为底的对数值的小数部分的值N。对于输入的8比特(10011100)，选择器的输出结果是0.1011。

[0209] 大于或等于比较器0的比较值0，小于比较器1的比较值1，选择器结果是0.0000

[0210] 大于或等于比较器1的比较值1，小于比较器2的比较值2，选择器结果是0.0001

[0211] …

[0212] 大于或等于比较器14的比较值14，小于比较器15的比较值15，选择器结果是0.1111

[0213] 大于或等于比较器15的比较值15，选择器结果是1.0000

[0214] 8.3方法三

[0215] 可选地，作为又一个实施例，上述步骤S930中的得到s比特对应的对数域表示格式下的数据的绝对值取以2为底的对数值的小数部分的值N，可以包括：将s比特的高x比特对应的值与预设的2^n个比较值比较，其中，x大于0并且小于s，第i比较值比第i+1比较值小，且第i比较值对应一对值A和B；计算x*A+B的结果，根据x*A+B的结果得到数据的绝对值取以2为底的对数值的小数部分的值N。

[0216] 具体地，可以计算x*A+B时，可以将高x比特与A相乘得到的结果右移k位，将右移k位后的结果与B相加，将相加后的结果再左移k-n位，得到的高n位即为s比特对应的对数域表示格式下的数据的绝对值取以2为底的对数值的小数部分的值N。

[0217] 仍以图10的例子来进行说明，小数计算子电路1030截取非零最高位后的8比特为10011100，其值的大小为S。先查表，再计算分段拟合结果得到数据的绝对值取以2为底的对数值的小数部分的值N。图12是本申请实施例的分段拟合的示意图；图13是本申请实施例的小数计算子电路1030确定小数部分的值N的示意图。如图12所示，因为8比特所代表的第一小数位的值的范围是[0,1)；提前把[0,1)范围分成$2^n=16$段，在log(1.x)的曲线上则分别对应16个线段，每一线段都可以计算出一个$y=A_ix+B_i$的直线表达式，记录到如图13所示的表格，即记录下A_i和B_i，$i=0,1,\cdots,15$，由此可以得到16组A_i和B_i，记录成表。最高位后的8比特为10011100，去其中的高$x=4$比特，查表得到A和B。计算A*S，得到的结果右移$k=8$(>>8)位；移位后的结果加B，再右移$k-n=4$(>>4)位，得到剩余4比特即为第一小数位1011，即数据的绝对值取以2为底的对数值的小数部分的值N为0.1011。

[0218] 9.实施例四

[0219] 基于上述各实施例，本实施例详细说明本申请实施例的乘加运算。乘加运算可以由图3所示的计算引擎执行。计算引擎包括DMA控制单元、输入缓存、多个处理元件PE形成的PE阵列和输出缓存，DMA控制单元用于将至少两个数据读取到输入缓存中；每个PE用于对输入缓存中的至少两个数据在对数域表示格式下进行加法运算，得到中间结果；PE将中间结果转换(例如，可以通过线性转换电路转换)为线性表示格式，多个线性表示格式的中间结果在PE阵列中进行加法运算得到乘加结果。乘加结果可以直接缓存到输出缓存中以待输出。或者乘加结果可以被转换到对数域表示格式，例如可以作为下一轮乘加运算的输入数据使用。可选地，可以通过对数转换电路将乘加结果转换到对数域表示格式。

[0220] 9.1.完整的一个示例

[0221] 下面以一个具体的例子说明本申请实施例的乘加运算的过程。

[0222] 输入两个数据F和G。

[0223] 对数域表示格式1.3.2下,f＝001010,表示二进制数据F:(+)2^(0.10)<<(2-0)＝101.10101。

[0224] 对数域表示格式1.3.2下,g＝101110,表示二进制数据G:(-)2^(0.10)<<(3-0)＝-1011.0101。

[0225] 例如,一个乘加运算需要计算C1＝F*G,C2＝F*G,SUM＝C1+C2,再将SUM表示成对数域表示格式1.4.4下的sum(即计算$\log_2$(SUM)),以供后续乘法使用。其中,对数域表示格式1.4.4下C1表示为c1,对数域表示格式1.4.4下C2表示为c2,BASE＝3。

[0226] (1)先计算f+g＝-(01010+01110)＝-(11000)＝111000,即计算对数乘法。

[0227] (2)线性域转换:C1＝C2＝(-)2^(0.00)<<(6-3)＝(-)1000.0000,表示为1.7.8格式为:C1＝C2＝1000 1000.0000 0000。

[0228] (3)计算SUM＝C1+C2＝1001 0000.0000 0000(十进制数为-16)。

[0229] (4)将SUM表示成对数域表示格式1.4.4下的sum:找到SUM的绝对值的最高位1的所在的位h＝12,减去线性表示格式的小数部分的位数k＝8,得到对数域表示格式的整数位所指示的值＝12-8＝4(0100),因为BASE＝-3,M'＝M-BASE＝4-(-3)＝7,所以表示成0111;取最高位1之后的连续s比特(例如8比特),记为S,查表计算log(1.S),查表$\log_2$(1.00000000),得到对数域表示格式的小数位0000;将对数域表示格式的小数位和整数位拼接,设置符号位与原始符号位一致,得到对数域表示格式1.4.4下的sum为101110000。

[0230] 应理解,本申请实施例的方法可以基于ASIC、FPGA或者其他可编程逻辑器件、分立门或者晶体管逻辑器件、分立硬件组件等来执行。需要说明的是,ASIC、FPGA或者其他可编程逻辑器件、分立门或者晶体管逻辑器件、分立硬件组件可以为独立的器件,或者可以与存储器(存储模块)集成在一起。

[0231] 应注意,本文描述的存储器旨在包括但不限于这些和任意其它适合类型的存储器。

[0232] 本申请实施例提供一种包括本申请实施例的计算引擎的卷积计算芯片。

[0233] 本申请实施例还提供一种计算机可读存储介质,其上存储有指令,当所述指令在计算机上运行时,使得所述计算机执行上述方法实施例的转换数据格式的方法。具体地,该计算机可读存储介质可以为上述线性转换电路、对数转换电路、计算引擎或卷积计算芯片。

[0234] 本申请实施例还提供一种包括指令的计算机程序产品,其特征在于,当计算机运行所述计算机程序产品的所述指令时,所述计算机执行上述方法实施例的转换数据格式的方法。

[0235] 本申请实施例还提供一种包括本申请实施例的卷积计算芯片的计算设备。

[0236] 基于以上各实施例,本实施例提供了一种电子设备,参见图14,为结构示意图。本电子设备中,SoC中可以有多个IP core(知识产权核),还可以包括CPU核(CPU core)SoC通常封闭成一块独立的芯片,例如,华为海思麒麟系列的芯片(如麒麟950,960),高通骁龙系列的SoC芯片(如骁龙650,660)。电子设备中还可以包括存储器,例如内存或闪存(flash)等;输入输出设备,例如显示屏、触摸屏和喇叭等;和通信模块,例如wifi模块、通用串行总

线(Universal Serial Bus,USB)、4G和5G模块等

[0237] 在上述实施例中,可以全部或部分地通过软件、硬件、固件或者其任意组合来实现。当使用软件实现时,可以全部或部分地以计算机程序产品的形式实现。所述计算机程序产品包括一个或多个计算机指令。在计算机上加载和执行所述计算机指令时,全部或部分地产生按照本申请实施例所述的流程或功能。所述计算机可以是通用计算机、专用计算机、计算机网络、或者其他可编程装置。所述计算机指令可以存储在计算机可读存储介质中,或者从一个计算机可读存储介质向另一个计算机可读存储介质传输,例如,所述计算机指令可以从一个网站站点、计算机、服务器或数据中心通过有线(例如同轴电缆、光纤、数字用户线(Digital Subscriber Line,DSL))或无线(例如红外、无线、微波等)方式向另一个网站站点、计算机、服务器或数据中心进行传输。所述计算机可读存储介质可以是计算机能够存取的任何可用介质或者是包含一个或多个可用介质集成的服务器、数据中心等数据存储设备。所述可用介质可以是磁性介质(例如,软盘、硬盘、磁带)、光介质(例如,高密度数字视频光盘(Digital Video Disc,DVD))、或者半导体介质(例如,固态硬盘(Solid State Disk,SSD))等。

[0238] 应理解,本文中涉及的第一、第二以及各种数字编号仅为描述方便进行的区分,并不用来限制本申请的范围。

[0239] 应理解,本文中术语“和/或”,仅仅是一种描述关联对象的关联关系,表示可以存在三种关系,例如,A和/或B,可以表示:单独存在A,同时存在A和B,单独存在B这三种情况。另外,本文中字符“/”,一般表示前后关联对象是一种“或”的关系。

[0240] 应理解,在本申请的各种实施例中,上述各过程的序号的大小并不意味着执行顺序的先后,各过程的执行顺序应以其功能和内在逻辑确定,而不应对本申请实施例的实施过程构成任何限定。

[0241] 本领域普通技术人员可以意识到,结合本文中所公开的实施例描述的各示例的单元及算法步骤,能够以电子硬件、或者计算机软件和电子硬件的结合来实现。这些功能究竟以硬件还是软件方式来执行,取决于技术方案的特定应用和设计约束条件。专业技术人员可以对每个特定的应用来使用不同方法来实现所描述的功能,但是这种实现不应认为超出本申请的范围。

[0242] 所属领域的技术人员可以清楚地了解到,为描述的方便和简洁,上述描述的系统、装置和单元的具体工作过程,可以参考前述方法实施例中的对应过程,在此不再赘述。

[0243] 在本申请所提供的几个实施例中,应该理解到,所揭露的系统、装置和方法,可以通过其它的方式实现。例如,以上所描述的装置实施例仅仅是示意性的,例如,所述单元的划分,仅仅为一种逻辑功能划分,实际实现时可以有另外的划分方式,例如多个单元或组件可以结合或者可以集成到另一个系统,或一些特征可以忽略,或不执行。另一点,所显示或讨论的相互之间的耦合或直接耦合或通信连接可以是通过一些接口,装置或单元的间接耦合或通信连接,可以是电性,机械或其它的形式。

[0244] 所述作为分离部件说明的单元可以是或者也可以不是物理上分开的,作为单元显示的部件可以是或者也可以不是物理单元,即可以位于一个地方,或者也可以分布到多个网络单元上。可以根据实际的需要选择其中的部分或者全部单元来实现本实施例方案的目的。

[0245] 另外，在本申请各个实施例中的各功能单元可以集成在一个处理单元中，也可以是各个单元单独物理存在，也可以两个或两个以上单元集成在一个单元中。

[0246] 以上所述，仅为本申请的具体实施方式，但本申请的保护范围并不局限于此，任何熟悉本技术领域的技术人员在本申请揭露的技术范围内，可轻易想到变化或替换，都应涵盖在本申请的保护范围之内。因此，本申请的保护范围应所述以权利要求的保护范围为准。

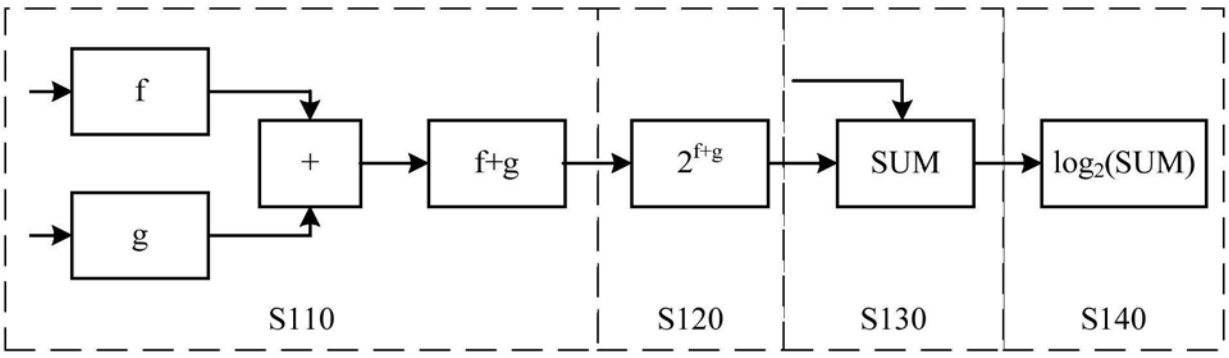

f
g
+
f+g
2^{f+g}
SUM
$\log_2(SUM)$
S110
S120
S130
S140

图1

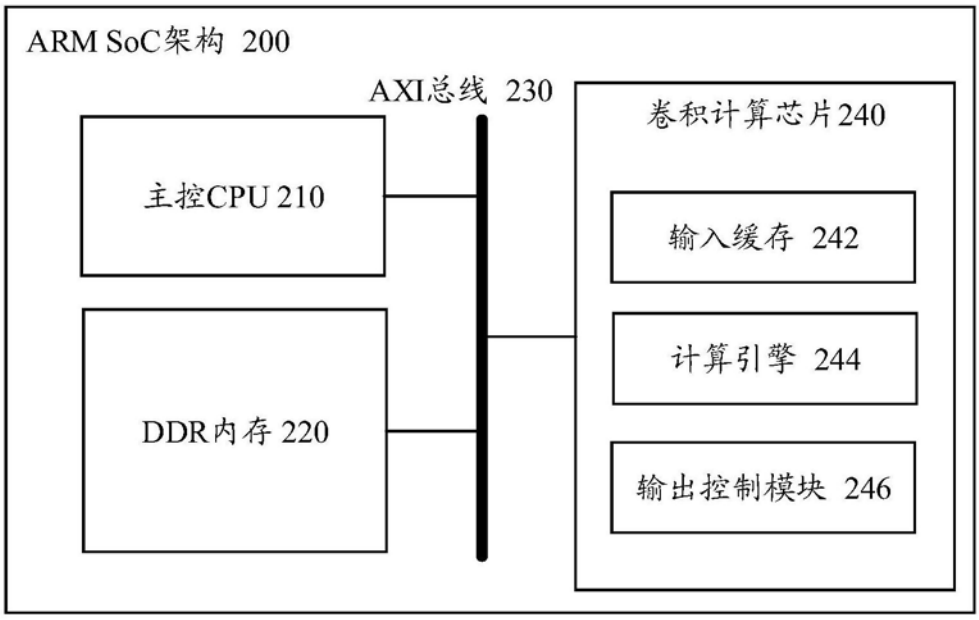

ARM SoC架构 200
AXI总线 230
主控CPU 210
DDR内存 220
卷积计算芯片240
输入缓存 242
计算引擎 244
输出控制模块 246

图2

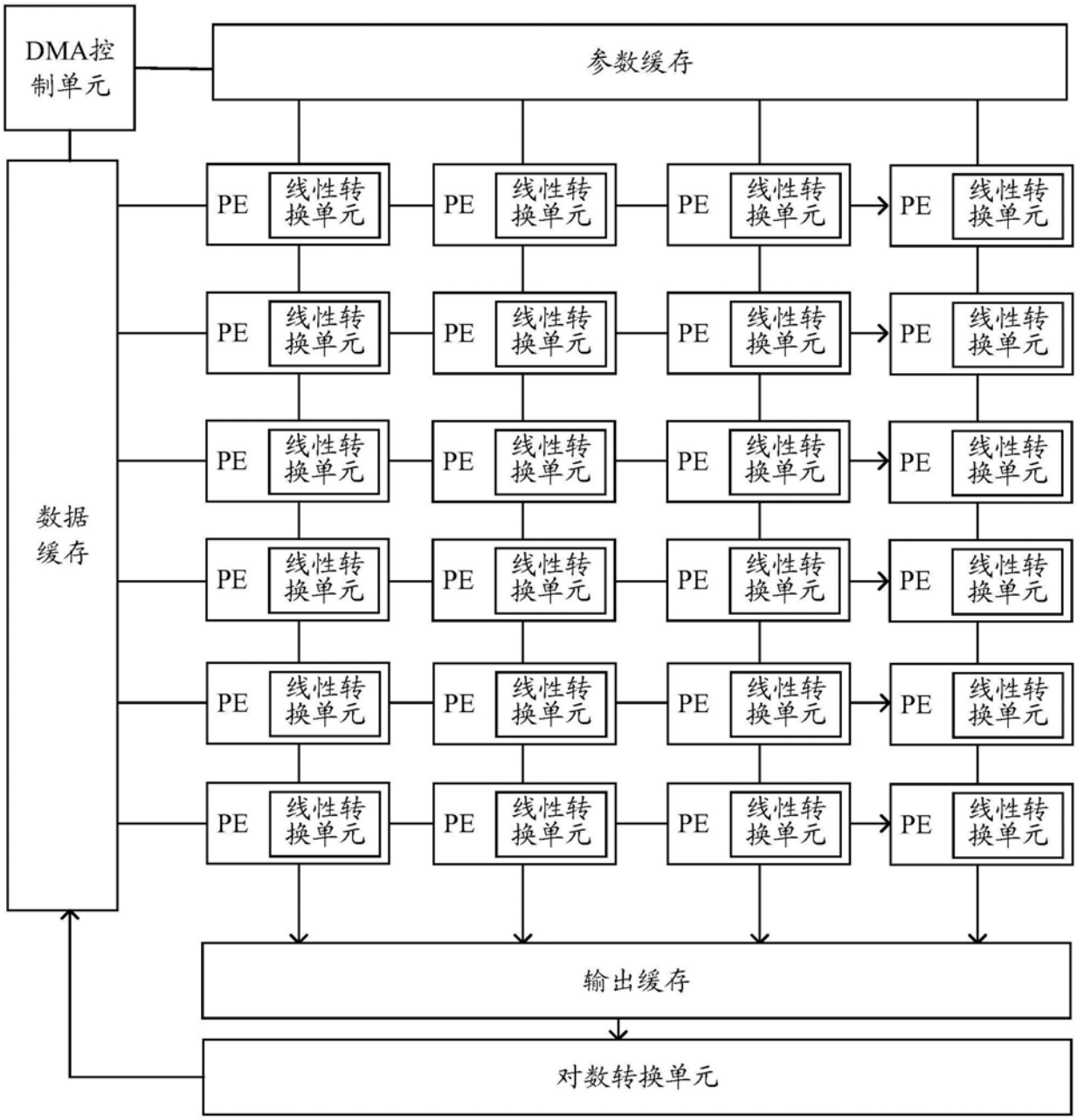


图3

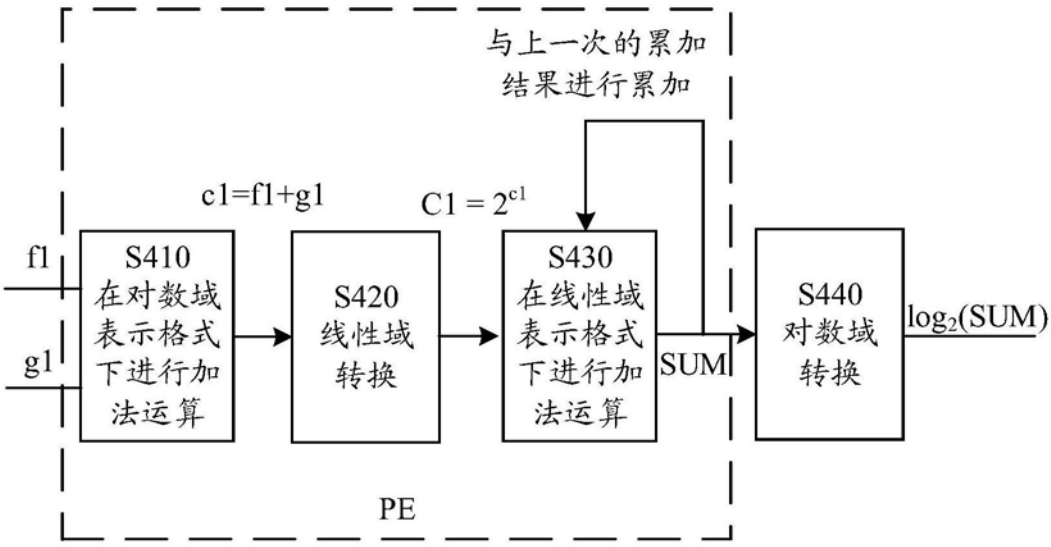

与上一次的累加
结果进行累加
c1=f1+g1
C1 = 2^c1
f1
g1
S410
在对数域
表示格式
下进行加
法运算
S420
线性域
转换
S430
在线性域
表示格式
下进行加
法运算
SUM
S440
对数域
转换
log2(SUM)
PE

图4

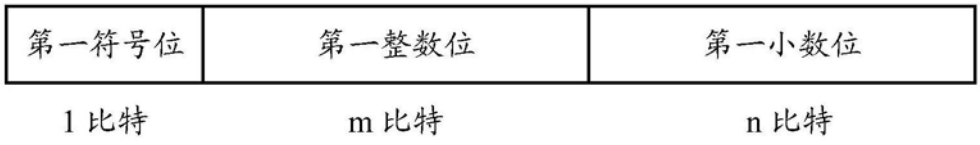

第一符号位
第一整数位
第一小数位
1 比特
m 比特
n 比特

图5

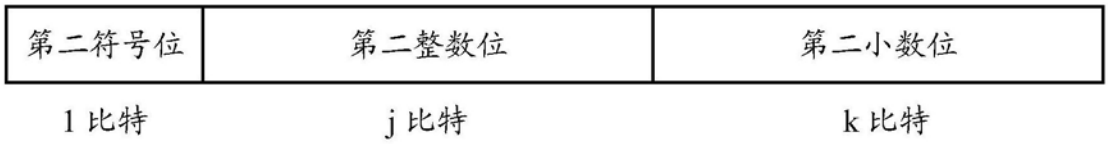

第二符号位
第二整数位
第二小数位
1 比特
j 比特
k 比特

图6

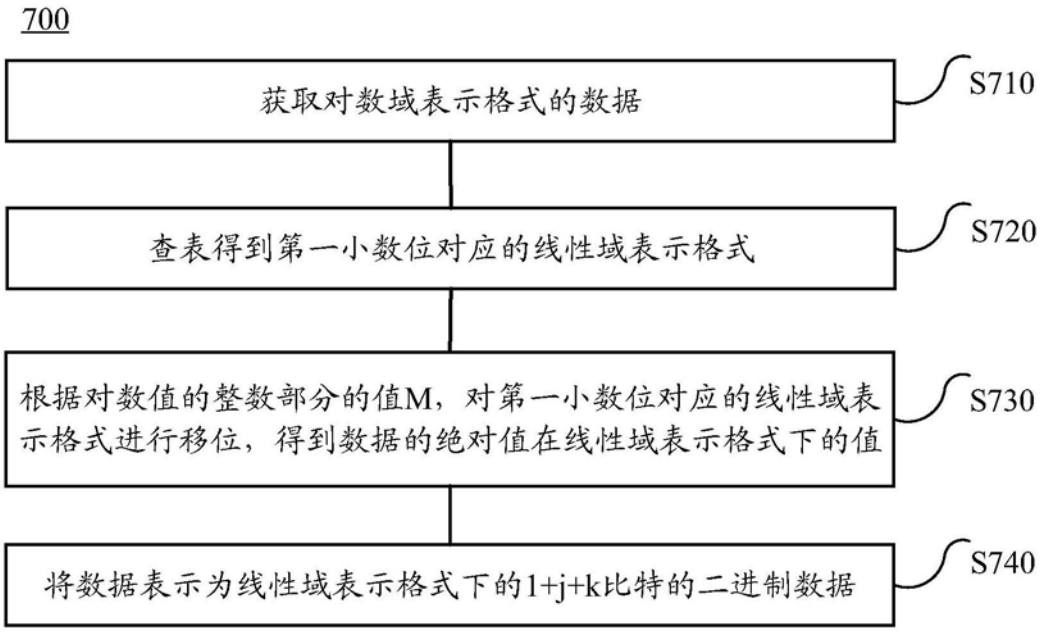

700
获取对数域表示格式的数据
S710
查表得到第一小数位对应的线性域表示格式
S720
根据对数值的整数部分的值M，对第一小数位对应的线性域表示格式进行移位，得到数据的绝对值在线性域表示格式下的值
S730
将数据表示为线性域表示格式下的1+j+k比特的二进制数据
S740

图7

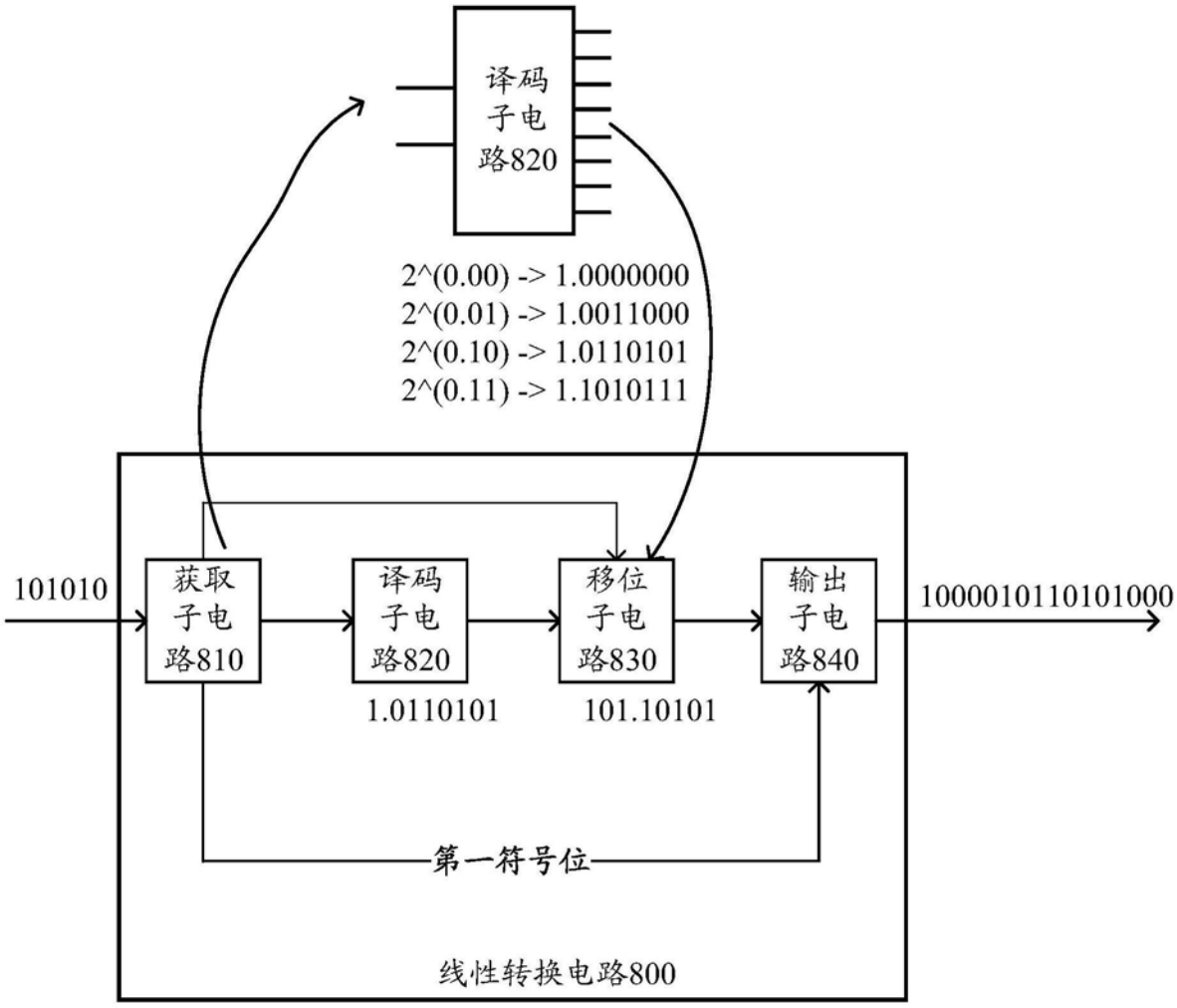

译码
子电
路820
2^(0.00) -> 1.0000000
2^(0.01) -> 1.0011000
2^(0.10) -> 1.0110101
2^(0.11) -> 1.1010111
101010
获取
子电
路810
译码
子电
路820
移位
子电
路830
输出
子电
路840
1000010110101000
1.0110101
101.10101
第一符号位
线性转换电路800

图8

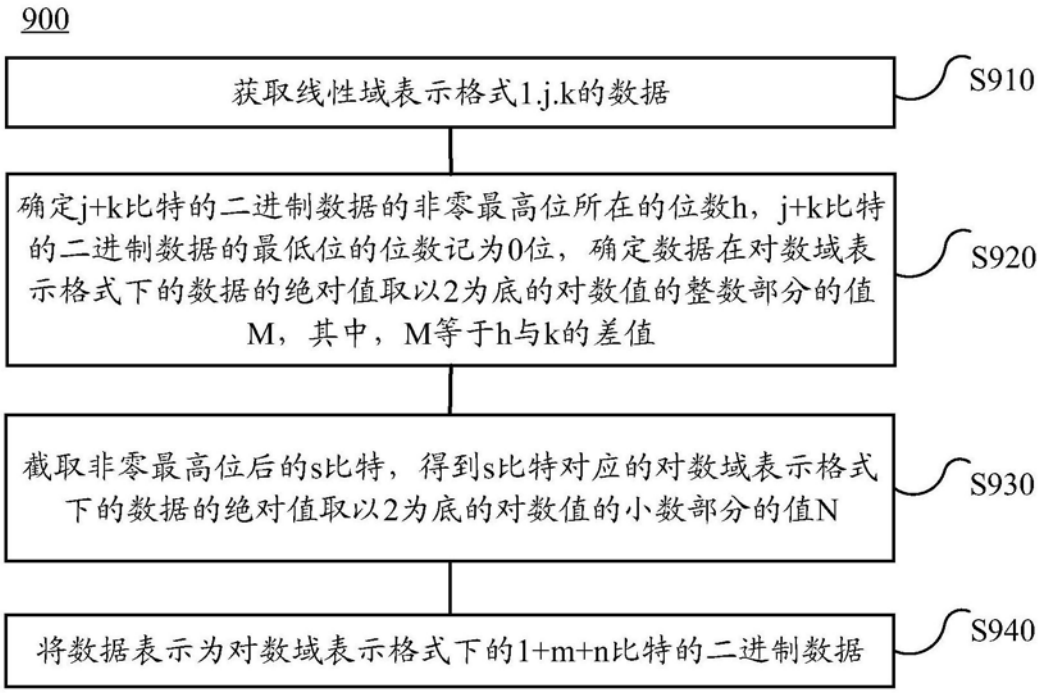

900
获取线性域表示格式1.j.k的数据
S910
确定j+k比特的二进制数据的非零最高位所在的位数h，j+k比特的二进制数据的最低位的位数记为0位，确定数据在对数域表示格式下的数据的绝对值取以2为底的对数值的整数部分的值M，其中，M等于h与k的差值
S920
截取非零最高位后的s比特，得到s比特对应的对数域表示格式下的数据的绝对值取以2为底的对数值的小数部分的值N
S930
将数据表示为对数域表示格式下的1+m+n比特的二进制数据
S940

图9

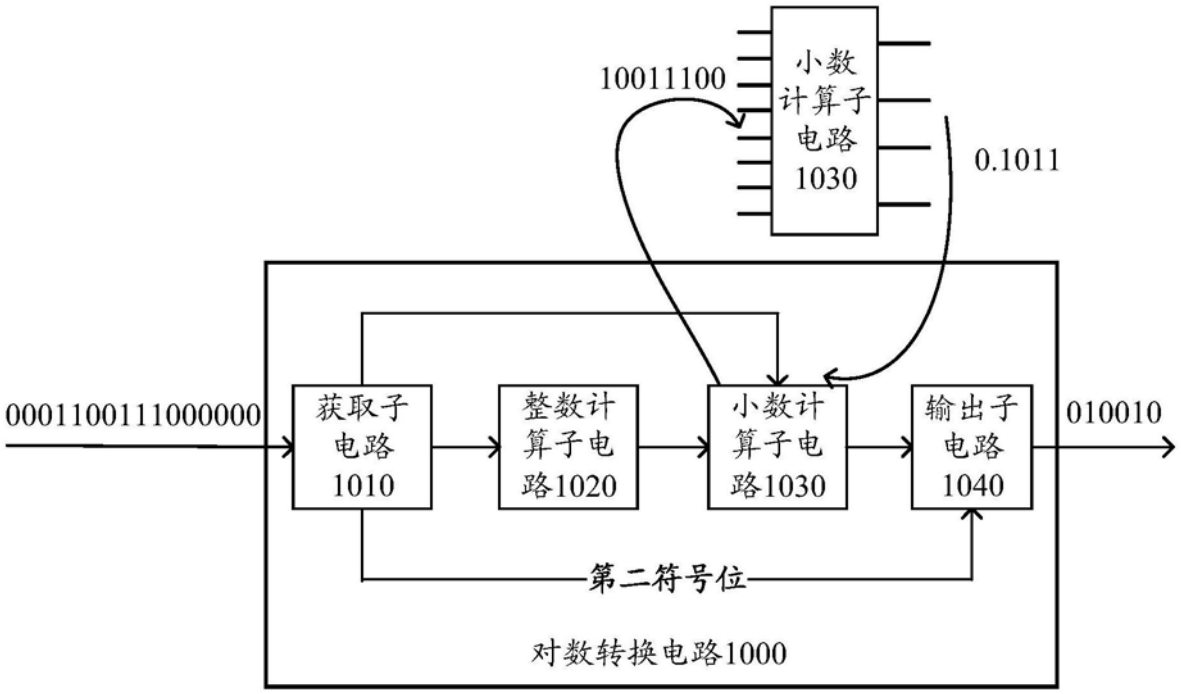

10011100
小数计算子电路1030
0.1011
0001100111000000
获取子电路1010
整数计算子电路1020
小数计算子电路1030
输出子电路1040
010010
第二符号位
对数转换电路1000

图10

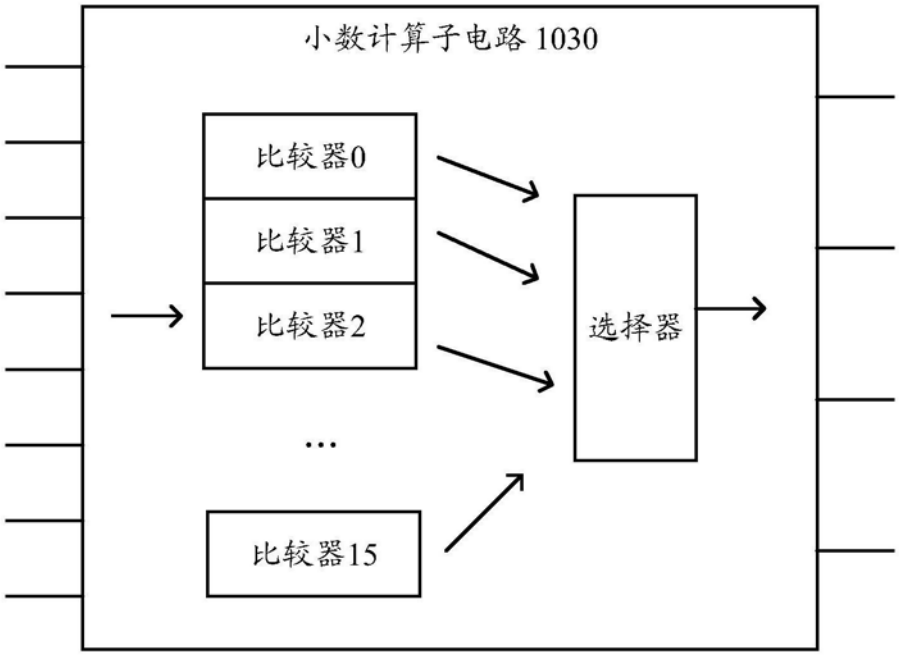


图11

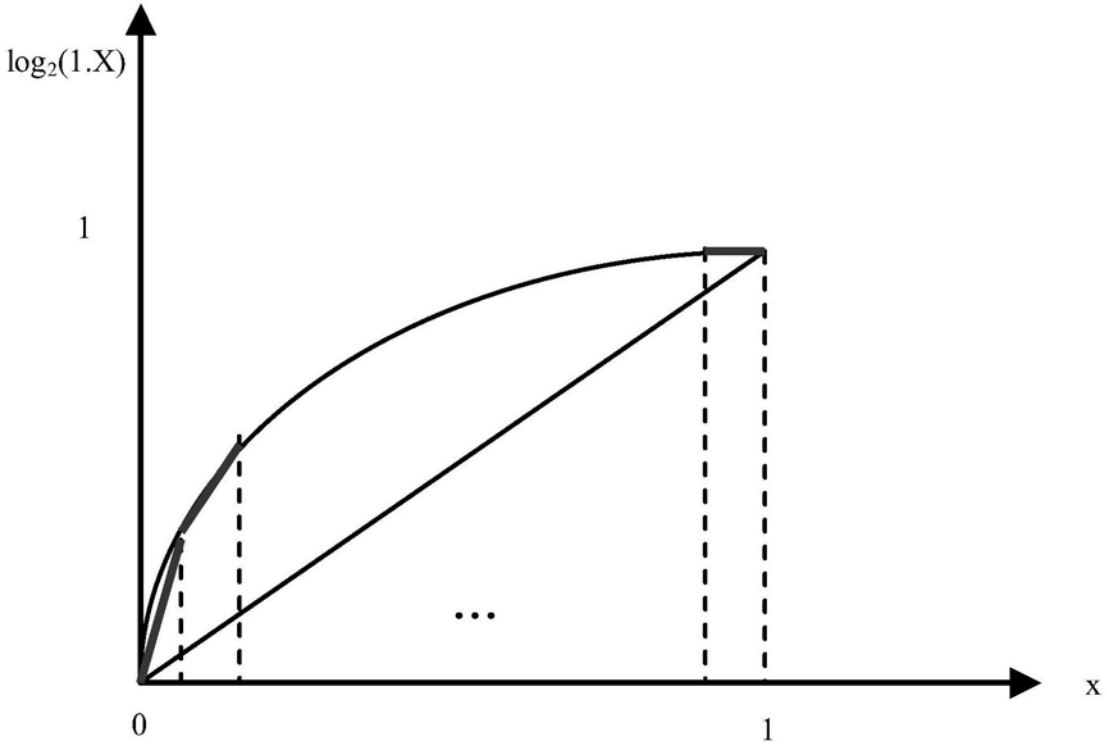


图12

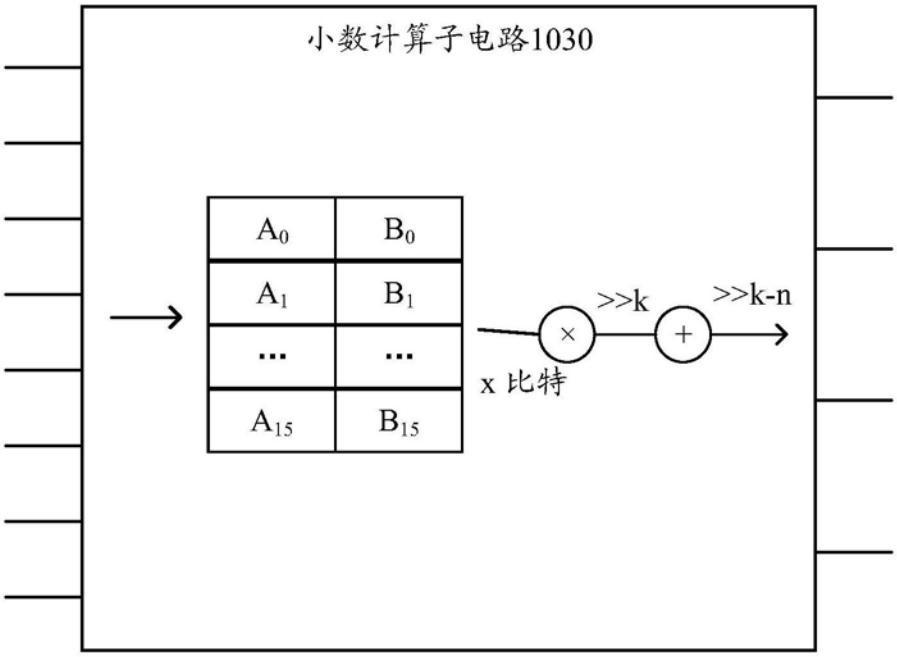


图13

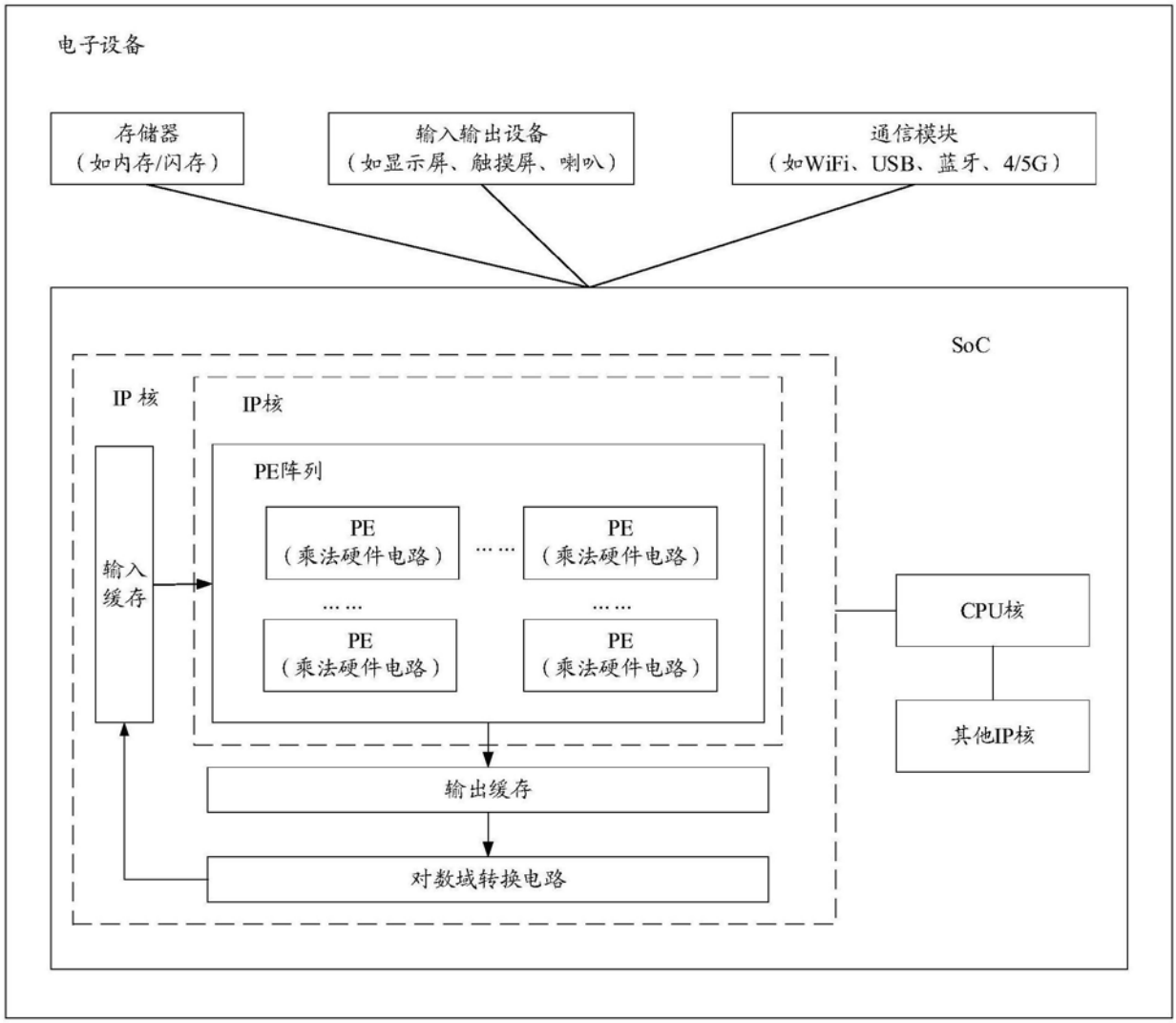

电子设备
存储器
（如内存/闪存）
输入输出设备
（如显示屏、触摸屏、喇叭）
通信模块
（如WiFi、USB、蓝牙、4/5G）
SoC
IP 核
IP核
PE阵列
PE
（乘法硬件电路）
PE
（乘法硬件电路）
PE
（乘法硬件电路）
PE
（乘法硬件电路）
输入
缓存
CPU核
其他IP核
输出缓存
对数域转换电路

图14