(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 110140109 A
(43)申请公布日 2019.08.16

(21)申请号 201780071430.4

(22)申请日 2017.12.15

(30)优先权数据
62/434521 2016.12.15 US
15/841959 2017.12.14 US

(85)PCT国际申请进入国家阶段日
2019.05.17

(86)PCT国际申请的申请数据
PCT/US2017/066677 2017.12.15

(87)PCT国际申请的公布数据
WO2018/112345 EN 2018.06.21

(71)申请人 优创半导体科技有限公司
地址 美国纽约州

(72)发明人 玛雅·穆吉尔 P·赫特利
M·森蒂尔威兰 P·鲍拉佐拉
V·T·拉马杜拉伊

(74)专利代理机构 北京泛华伟业知识产权代理有限公司 11280
代理人 王勇 李科

(51)Int.Cl.
G06F 9/30(2006.01)

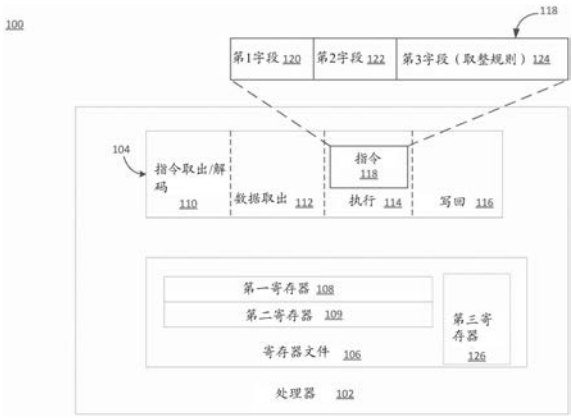
权利要求书2页 说明书8页 附图3页

(54)发明名称

具有嵌入式取整规则的浮点指令格式

(57)摘要

处理器包括:第一存储装置,其存储第一数据项;第二存储装置;以及执行单元,其包括对指令编码的逻辑电路,指令包括:第一字段,其存储第一存储装置的标识符;第二字段,其存储第二存储装置的标识符;以及第三字段,其存储表示取整规则的标识符,其中执行单元执行指令以基于第一数据项来产生第二数据项;根据由指令规定的取整规则来将第二数据项取整;以及将取整后的第二数据项存储在第二存储装置中。



CN 110140109 A

1.一种处理器,包括:

第一存储装置,用于存储第一数据项;

第二存储装置;以及

执行单元,其包括对指令编码的逻辑电路,所述指令包括:

第一字段,用于存储所述第一存储装置的标识符;

第二字段,用于存储所述第二存储装置的标识符;以及

第三字段,用于存储表示取整规则的标识符,

其中所述执行单元执行所述指令以:

基于所述第一数据项来产生第二数据项;

根据由所述指令规定的所述取整规则来将所述第二数据项取整;以及

将取整后的第二数据项存储在所述第二存储装置中。

2.如权利要求1所述的处理器,其中在与所述处理器相关联的指令集架构(ISA)中规定所述指令。

3.如权利要求1所述的处理器,其中所述第一存储装置是第一寄存器或第一存储单元之一,以及其中所述第二存储装置是第二寄存器或第二存储单元之一。

4.如权利要求1所述的处理器,其中所述第一存储装置是不同于所述第二存储装置或与所述第一存储装置相同的存储装置。

5.如权利要求1所述的处理器,其中在所述第一存储装置中存储的所述第一数据项和在所述第二存储装置中存储的所述第二数据项以浮点格式被表示。

6.如权利要求1所述的处理器,其中在所述第一存储装置中存储的所述第一数据项以浮点格式被表示,以及在所述第二存储装置中存储的所述第二数据项以固定点格式被表示。

7.如权利要求1所述的处理器,其中所述取整规则是向最近值取整规则、向零取整规则、向正无限大取整或向负无限大取整之一。

8.如权利要求1所述的处理器,其中所述指令包括加法、减法、乘法或除法操作之一。

9.如权利要求1所述的处理器,其中在所述第一存储装置中存储的所述第一数据项的值由包括符号位、表示指数的位的第一子集和表示分数的位的第二子集的多个位表示。

10.如权利要求1所述的处理器,其中所述第一寄存器和所述第二寄存器是具有相同长度的浮点寄存器。

11.如权利要求1所述的处理器,其中所述第一寄存器和所述第二寄存器是浮点寄存器,以及其中所述第一寄存器包括比所述第二寄存器多的位。

12.如权利要求1所述的处理器,其中所述第一存储装置是用于存储浮点值的浮点寄存器,以及所述第二存储装置是用于存储整数的通用寄存器,以及其中所述指令包括使用在所述指令中规定的所述取整规则的实数到整数转换操作。

13.如权利要求1所述的处理器,其中所述第一存储装置是用于存储整数的通用寄存器,以及所述第二存储装置是用于存储实数值的浮点寄存器,以及其中所述指令包括使用在所述指令中规定的所述取整规则的整数到实数转换操作。

14.如权利要求1所述的处理器,其中所述取整规则是下列情况中至少之一:将未定义数字转换成零、将未定义数字转换成使用多个位可表示的最大数字、或将未定义数字转换

成使用多个位可表示的最小数字。

15.如权利要求1所述的处理器,其中所述第三字段用于存储对所述取整规则编码的立即值或表示第三存储装置的标识符之一,以及其中所述第三存储装置包括指示所述取整规则的标记值。

16.如权利要求1所述的处理器,其中当被执行时,所述处理器使用所述逻辑电路来使用预定数量的处理器时钟周期完成所述指令。

17.一种系统,包括:

存储器;以及

处理器,其通信地耦合到所述存储器,所述处理器包括:

第一存储装置,用于存储第一数据项;

第二存储装置;以及

执行单元,其包括对指令编码的逻辑电路,所述指令包括:

第一字段,用于存储所述第一存储装置的标识符;

第二字段,用于存储所述第二存储装置的标识符;以及

第三字段,用于存储表示取整规则的标识符,

其中所述执行单元执行所述指令以:

基于所述第一数据项来产生第二数据项;

根据由所述指令规定的所述取整规则来将所述第二数据项取整;以及

将取整后的第二数据项存储在所述第二存储装置中。

18.如权利要求17所述的系统,其中所述第一存储装置是第一寄存器或第一存储单元之一,以及其中所述第二存储装置是第二寄存器或第二存储单元之一。

19.如权利要求17所述的系统,其中所述第一存储装置是不同于所述第二存储装置或与所述第一存储装置相同的存储装置。

20.如权利要求17所述的系统,其中在所述第一存储装置中存储的所述第一数据项和在所述第二存储装置中存储的所述第二数据项以浮点格式被表示。

具有嵌入式取整规则的浮点指令格式

[0001] 相关申请

[0002] 本申请要求于2016年12月15日提交的美国临时专利No.62/434,521的优先权,该临时申请的内容通过引用被并入本文。

技术领域

[0003] 本公开涉及处理器,且更特别地,涉及与处理器相关联的指令集架构(ISA),其中ISA的每个浮点指令规定特别适用于那个浮点指令的取整规则。

背景技术

[0004] 处理器(例如中央处理单元(CPU))可执行包括系统软件(例如操作系统)和用户软件应用的软件应用。可根据规定指令的集合的指令集架构(ISA)来设计处理器的微架构。软件程序可被编译成可在处理器的执行流水线上执行的这些指令的集合。在ISA中规定的指令可包括处理浮点值(例如作为输入或作为输出)的指令。这些指令被称为ISA的浮点指令。

附图说明

[0005] 从下面给出的详细描述中和从本公开的各种实施方式的附图中将更充分理解本公开。然而,附图不应被理解为将本公开限制到特定的实施方式,而是仅为了解释和理解。

[0006] 图1示出根据本公开的实施方式的包括处理器102的系统。

[0007] 图2示出根据本公开的实施方式的可包括存储取整规则的标识符的字段的浮点指令。

[0008] 图3示出根据本公开的实施方式的浮点转换指令。

具体实施方式

[0009] 在计算机中,可使用可被解释为实数的表示的多个位来表示浮点值。一个共同的表示是如根据IEEE 754技术标准而定义的二进制32格式。二进制32格式的32位可包括符号位(S)、8个指数位和23个分数位。

[0010] 可使用如在表1中所示的下面的伪代码来将以这种格式编码的32位字转换成实数:

[0011] 表1

[0012]

```
S = Word[31]
Exponent = Word[30:23]
Fraction = Word[22:0]
if( Exponent = 0 )
  if( Fraction = 0 )
    Real = 0.0
  else
    Real = (-1)^S * 2^-126 * 0.Fraction
else if( Exponent != 255 )
  Real = (-1)^S * 2^(Exponent-127) * 1.Fraction
```

[0013] 在这个例子中，符号位S用于确定实数是正(+)数还是负(-)数，其中指数值255(即，全部1)用于表示+/-无限大和其它例外条件。使用有限数量的位的表示可以只表示有限数量的实数值；特别地，存在不能使用该表示来表示的一些实数值。例如，IEEE二进制32格式可表示至多2^{32}个实数值。这意味着某些实数不能被表示。

[0014] 考虑十进制数字33554432(在十六进制中是0x200_0000)和数字1。这两个数字都可以二进制32格式确切地分别被表示为S＝0，Exponent＝152，Fraction＝0以及S＝0，Exponent＝127，Fraction＝0。然而它们的和33554433(0x200_0001)不能以该格式被表示，因为和的表示需要25位分数，其超过被分配到二进制32格式的分数部分的位的数量。

[0015] 当实数值不能确切地以特定的浮点格式被表示时，取整操作可能发生。在一些实现中，取整操作是选择可以那种格式(例如二进制32格式)被表示的可替换实数。一般，所选择的可替换实数可以是下一最大可表示的实数或下一最小可表示的实数。

[0016] 当执行诸如加法、减法、乘法和/或除法的浮点操作时，这些操作的确切结果常常不能以浮点格式被表示。在这种情况下，处理器可能需要执行取整操作以确定可被表示的可替换数字。处理器可基于取整规则来选择特定的取整方法。可被使用的一些取整方法是：

[0017] • 向最近值取整：向最近值取整；如果数字落在中间，则它被取整到具有偶数(零)最低有效位的最近值。

[0018] • 向0取整：向零取整(也被称为截尾取整)。

[0019] • 向+无限大取整：向正无限大取整(也被称为向上或最高限度取整)。

[0020] • 向-无限大取整：向负无限大取整(也被称为向下或最低限度取整)。

[0021] 不同取整规则的应用可产生不同的取整结果。

[0022] 当浮点数转换成整数时，取整也可出现在处理器中。在那种情况下，处理器可使用所确定的取整规则来将由浮点格式表示的实数值转换成最接近的整数。整数结果可根据所使用的取整规则而不同。例如，考虑如表2所示的下面的例子。使用不同的取整规则可产生不同的结果。

[0023] 表2

[0024]

规则/值	+11.5	+12.5	-11.5	-12.5
最近	+12	+12	-12	-12
向0	+11	+12	-11	-12
向+无限大	+12	+13	-11	-12
向-无限大	+11	+12	-12	-13

[0025] 取整也可在整数到浮点转换的情况下产生。例如，当将以整数格式表示的32位整数转换成二进制32位格式时，整数数字0x200_0001不是确切地可表示的且可能在转换之前被取整到0x200_0000或0x200_0004。

[0026] 在处理器的一些实现中，在处理器架构的规定中确定取整规则。在其它实现中，在由处理器通过编程接口可访问的寄存器(被称为浮点控制寄存器)中规定可被使用的取整规则。在这种情形中，当浮点操作(或浮点/整数转换)产生不可表示的结果时，处理器检查浮点控制寄存器以确定要应用哪个取整规则，且结果基于所确定的取整规则被取整。

[0027] 所使用的取整规则的选择可影响浮点操作的序列的总体结果。因此在专门的应用中，挑选不同的取整规则可导致较高或较低质量的最终结果。在这样的应用中，程序可选择适当的取整规则(或多个规则)。然而通常，软件应用并不规定在浮点操作中使用的取整规则。当未规定规则时，使用默认取整规则。默认取整规则通常与编程语言相关联，且常常是向最近值取整规则。

[0028] 当从浮点数转换成整数时的取整方法的选择可明确地被定义为在编程语言的函数库中的取整函数。程序员可接着以代码使用这些取整函数(例如ceiling、round或floor)，以便明确地将期望取整规则应用于特定的数字。

[0029] 在控制浮点取整模式的唯一方式是浮点控制寄存器的情况下，明确地规定取整模式的库函数将增加与管理取整模式有关的开销。例如，在表3中示出ceiling函数的伪代码序列：

[0030] 表3

[0031]

```
CEILING:
  read_rm   $r1  // save old rounding mode
  write_rm  ROUND_TO_NEG_INF
  convert   $r0,$f0
  write_rm  $r1   // restore rounding mode
  return
```

[0032] 其中注意，在这个代码序列中的五行指令中的三行与操纵取整模式有关，这引起计算开销的增加。

[0033] 频繁切换取整模式可能在计算上是昂贵的，特别是在现代无序超标量处理器的情况中。在一些实现中，读取和/或写入取整模式可能消耗多个处理器周期。它常常是序列化操作，禁止并行和无序浮点执行。

[0034] 有出现的情况，其中在代码序列中，取整模式需要频繁地改变。一个例子是在C程

序中，其中浮点操作通常在向最近值取整的情况下被执行，同时浮点到整数转换使用向零取整来被规定。这意味着涉及然后被取整到整数的浮点操作的操作序列将具有频繁的取整模式变化。出现的另一情形是用户明确地希望控制被应用到特定代码区的浮点取整模式。为了正确地支持这个功能，每当控制离开那个代码区时，浮点取整模式需要被重置到语言的默认模式。这也证明是相当昂贵的。

[0035] 本公开的实施方式提供包括指令的指令集架构，指令规定应用于可能需要取整的浮点指令的取整规则。在一个实施方式中，指令可将取整规则直接规定为指令的属性。如果语言需要特定的取整模式，则可在指令中使用立即值来明确地对取整规则编码，因而避免管理浮点寄存器的需要。在另一实施方式中，指令可规定表示指令的取整规则的标识符从浮点控制寄存器被读取。这支持下面的情况：用户希望对所使用的取整规则施加控制，并动态地改变在应用程序中的取整规则。因此，本公开的实施方式提供用于使ISA的浮点指令(包括浮点转换指令)确切地规定期望取整模式或规定由浮点控制寄存器提供的默认取整模式被使用的手段。

[0036] 图1示出根据本公开的实施方式的包括处理器102的片上系统(SoC)100。处理器102可包括在例如SoC 100的半导体芯片集上制造的逻辑电路。处理器100可以是中央处理单元(CPU)、图形处理单元(GPU)或多核处理器的处理核心。如图1所示，处理器102可包括指令执行流水线104和寄存器文件106。流水线104可包括多个流水线阶段，以及每个阶段包括被制造成在多阶段过程中执行特定阶段的操作的逻辑电路，多阶段过程是充分执行在处理器102的指令集架构(ISA)中规定的指令所需的。在一个实施方式中，流水线104可包括指令取出/解码阶段110、数据取出阶段112、执行阶段114和写回阶段116。

[0037] 处理器102可包括寄存器文件106，其还可包括与处理器102相关联的寄存器108、109。在一个实施方式中，寄存器文件106可包括通用寄存器108、109，每个寄存器可包括某个数量(被称为“长度”)的位以存储由在流水线104中执行的指令处理的数据项。例如，根据实现，寄存器108、109可以是64位、128位、256位或512位寄存器。寄存器108、109中的每个可存储一个或多个数据项。可实现寄存器108、109以存储浮点数据项和/或固定点数据项，其中浮点数据项可表示实数而固定点数据项可表示整数。

[0038] 程序的源代码可被编译成在与处理器102相关联的指令集架构(ISA)中定义的一系列机器可执行指令。当处理器102开始执行可执行指令时，这些机器可执行指令可被放置在流水线104上以被顺序地执行。指令取出/解码阶段110可取回放置在流水线104上的指令并识别与指令相关联的标识符。指令标识符可使所接收的指令与在处理器102的ISA中规定的指令118的电路实现相关联。

[0039] 在ISA中规定的指令可被设计成处理存储在通用寄存器(GPR)108、109中的数据项。数据取出阶段112可从GPR 108取回待处理的数据项(例如浮点或固定点)。执行阶段114可包括逻辑电路以执行在处理器102的ISA中规定的指令。

[0040] 在一个实施方式中，与执行阶段114相关联的逻辑电路可包括多个“执行单元”(或功能单元)，每个单元专用于执行一个相应的指令。由这些执行单元执行的所有指令的集合可构成与处理器102相关联的指令集。在执行指令以处理由数据取出阶段112取回的数据项之后，写回阶段116可输出结果并将结果存储在GPR 108、109中。

[0041] 在一个实施方式中，处理器102的ISA可定义浮点指令，且处理器102的执行阶段

114可包括执行单元118,其包括在ISA中定义的浮点指令的硬件实现。浮点指令可包括存储第一寄存器108的标识符的第一字段120(或操作数)、存储第二寄存器109的标识符的第二字段122(或操作数)和存储表示取整规则的标识符的第三字段124(或操作数)。指令当被执行时可包括下列操作:读取第一数据项(浮点数据项或固定点数据项)、基于存储在第一寄存器中的第一数据项来计算结果值(浮点数据项)以及使用在指令的第三字段中规定的取整规则来对结果值取整以将结果存储在第二寄存器109中。以这种方式,本公开的实施方式可允许程序规定按指令取整规则。按指令取整规则实现允许与不同的取整规则相关联的不同指令,而不是对由处理器102执行的所有指令使用一种取整规则(例如默认取整规则)。

[0042] 在一个实施方式中,取整规则可由存储在第三字段124中的立即值识别。例如,立即值可以是整数,且不同的整数值可对应于不同的取整规则。在另一实施方式中,第三字段124可存储寄存器文件106的第三寄存器126的标识符,其中寄存器126可存储对应于特定取整规则的标识符。取整规则的间接规定(例如经由寄存器126)可向程序员提供另外的灵活性以对应用进行编程。

[0043] 图2示出根据本公开的实施方式的可包括存储取整规则的标识符的字段的浮点指令。如图2所示,可在ISA中规定指令200以包括操作字段202、目标寄存器字段204、第一输入寄存器字段206、第二输入寄存器字段208、操作类型字段210和取整规则字段212。操作字段202可存储浮点操作的标识符(例如fadd)。目标寄存器字段204可规定用于存储输出的与处理器相干的浮点寄存器。第一输入寄存器字段206和第二输入寄存器字段208可规定存储所述输入值(或被加在一起的值)的浮点寄存器。操作类型字段210可存储表示浮点类型(例如单精度或双精度)的值。取整规则字段212可存储表示取整规则的类型的标识符(FRM)。

[0044] 例如,在GPTX架构中的指令fadd_s_rzero$f3,$f1,$f2使用向零取整的取整规则来规定$f1与$f2的内容的单精度浮点相加,将结果存储回到$f3中。

[0045] 以FRM值编码的固定取整规则可包括:

[0046] • rnear:向最近值取整(例如与标识符RNEAR相关联),

[0047] • rzero:向零取整(例如与标识符RZERO相关联),

[0048] • rdown:向下取整(例如与标识符RDOWN相关联),

[0049] • rup:向上取整(例如与标识符RUP相关联)。

[0050] 对FRM标识符可采用的其它编码是rdyn,其规定在浮点控制寄存器中规定的取整规则应被使用,因而间接地规定取整规则(而不是使用固定值)。

[0051] 在从整数到浮点数的转换期间,处理器可将整数存储在通用寄存器中并将结果存储在浮点寄存器中。在从通用寄存器到浮点寄存器的复制期间,整数值在取整(如果必要)的情况下转换成等效的浮点表示。当处理器执行将浮点值从浮点寄存器复制到通用寄存器的指令时,取整可类似地出现,其中浮点值基于在指令中规定的取整规则而转换成整数值。

[0052] 在ISA的一个实现中,这些指令是fcvtr(从整数转换到浮点)和rcvtf(从浮点转换到整数)指令,其分别将在通用寄存器中存储的整数转换成在浮点寄存器中存储的浮点值以及将在浮点寄存器中的浮点值转换成在通用寄存器中存储的整数。这些指令的FRM字段可规定在转换期间应用的取整规则的选择。

[0053] 图3示出根据本公开的实施方式的fcvtr指令302和rcvtf指令304。fcvtr指令302的规定可包括存储对浮点寄存器(浮点寄存器存储浮点值)的参考的浮点寄存器字段306和

存储对通用寄存器(通用寄存器存储整数)的参考的通用寄存器字段。fcvtr指令302基于在取整规则字段310中规定的取整规则来将浮点值转换成整数。类似地,rcvtf指令304的规定可包括存储对通用寄存器(其存储整数)的参考的通用寄存器字段312和存储对浮点寄存器(其存储浮点值)的参考的浮点寄存器字段314。rcvtf指令304基于在取整规则字段316中规定的取整规则来将整数转换成浮点值。

[0054] 在可复制到通用寄存器/从通用寄存器复制的指令(例如fcvtr或rcvtf)的上下文中,取整规则可包括被称为raw规则的额外的取整规则。在一个实施方式中,可在具有标识符RAW的fcvtr指令的取整规则字段310(或rcvtf指令的字段316)中规定raw规则。在raw规则下,在源寄存器(通用或浮点)中的位照现状直接复制(例如位到位复制)到目标寄存器(浮点/通用寄存器)而没有转换。raw规则的使用允许浮点值从浮点寄存器复制到相同(或更大)长度的通用寄存器并复制回而不干扰该值。

[0055] 在将浮点数转换成整数的指令的上下文中,本公开的实施方式可提供与未定义数字(NaN)的处理有关的额外取整规则。未定义数字可表示无限大值。这个取整规则可规定NaN到整数转换,其被选择为下列项之一:

[0056] • 所有NaN都被转换成0,

[0057] • +NaN/-NaN被转换成可表示的最正/最负的整数值,

[0058] • 所有NaN被转换成可表示的最正值,或

[0059] • 所有NaN被转换成可表示的最负值。

[0060] 在rcvtf指令的上下文中,如图3所示,rcvtf指令304可包括NaN规则字段318,其中NaN到整数转换规则(如上所述)可被规定。

[0061] 在一个实施方式中,如表3所示的ceiling函数可使用rcvtf指令由表4的下面的代码实现。

[0062] 表4

[0063]

```
CEILING:
        rcvtf_rneg $r0,$f0
        return
```

[0064] 因为指令明确地对取整规则编码,没有操纵浮点控制寄存器的需要,因而减小了与切换取整模式相关联的开销。

[0065] 虽然关于有限数量的实施方式描述了本公开,但本领域中的技术人员将从那里认识到很多修改和变化。意图是所附权利要求涵盖如落在本公开的真实精神和范围内的所有这样的修改和变化。

[0066] 设计可经历从创建到模拟到制造的各种阶段。表示设计的数据可以用很多方式表示该设计。首先,如在模拟中有用的,可使用硬件描述语言或另一功能描述语言来表示硬件。此外,可在设计过程的一些阶段产生具有逻辑和/或晶体管门的电路级模型。此外,大部分设计在某个阶段达到以硬件模型表示各种装置的物理放置的数据的级别。在常规半导体制造技术被使用的情况下,表示硬件模型的数据可以是规定在掩模的不同掩模层上的用于产生集成电路的各种特征的存在或缺乏的数据。在设计的任何表示中,数据可存储在任何

形式的机器可读介质中。存储器或磁性或光学存储装置(例如磁盘)可以是机器可读介质以存储经由光波或电波传输的信息,光波或电波被调制或以另外方式被产生以传输这样的信息。当指示或承载代码或设计的电载波在电信号的拷贝、缓冲或重新传输被执行的程度上被传输时,新拷贝被制造。因此,通信提供商或网络提供商可在有形机器可读介质上至少临时地存储物品,例如被编码到载波内的信息,体现本公开的实施方式的技术。

[0067] 如本文使用的模块指硬件、软件和/或固件的任何组合。作为例子,模块包括硬件,例如微控制器,其与非临时介质相关联以存储适合于由微控制器执行的代码。因此,对模块的引用在一个实施方式中指硬件,其特别地被配置成识别和/或执行在非临时介质上保持的代码。此外,在另一实施方式中,模块的使用指包括代码的非临时介质,代码特别适合于由微控制器执行以执行预定操作。而且如可被推断出的,在又一实施方式中,术语“模块”(在这个例子中)可以指微控制器和非临时介质的组合。通常,被示为分开的模块边界通常改变并可能重叠。例如,第一和第二模块可共享硬件、软件、固件或其组合,同时可能保留某个独立的硬件、软件或固件。在一个实施方式中,术语“逻辑”的使用包括硬件,例如晶体管、寄存器或其它硬件,例如可编程逻辑装置。

[0068] 短语“被配置成”的使用在一个实施方式中指布置、放置在一起、制造、提供出售、导入和/或设计设备、硬件、逻辑或元件以执行所指定或确定的任务。在这个例子中,如果未正在操作的设备或其元件被设计、耦合和/或互连以执行指定任务,其仍然“被配置成”执行所述指定任务。作为纯粹示例性的例子,逻辑门可在操作期间提供0或1。但“被配置成”向时钟提供使能信号的逻辑门不包括可提供1或0的每个潜在逻辑门。替代地,逻辑门是以某种方式耦合的逻辑门,使得在操作期间1或0输出启用时钟。再一次注意,术语“被配置成”的使用并不需要操作,但替代地聚焦于设备、硬件和/或元件的潜伏状态,其中在潜伏状态中,当设备、硬件和/或元件正在操作时,设备、硬件和/或元件被设计成执行特定的任务。

[0069] 此外,短语“以”、“能够”和/或“可操作来”的使用在一个实施方式中指被设计成以特定的方式实现设备、逻辑、硬件和/或元件的使用的一些设备、逻辑、硬件和/或元件。注意如上面,以、能够或可操作来的使用在一个实施方式中指设备、逻辑、硬件和/或元件的潜伏状态,其中设备、逻辑、硬件和/或元件没有正在操作,但被设计成以特定的方式实现设备的使用。

[0070] 如在本文使用的值包括数字、状态、逻辑状态或二进制逻辑状态的任何已知的表示。逻辑电平、逻辑值(logic values)或逻辑值(logical values)的使用常常也被称为1和0,其简单地表示二进制逻辑状态。例如,1指高逻辑电平而0指低逻辑电平。在一个实施方式中,存储单元(例如晶体管或闪存单元)也许能够保持单个逻辑值或多个逻辑值。然而,在计算机系统中的值的其它表示被使用。例如,十进制数字10也可被表示为910的二进制值和十六进制字母A。因此,值包括能够保持在计算机系统中的信息的任何表示。

[0071] 而且,状态可由值或值的部分表示。作为例子,第一值(例如逻辑一)可表示默认或初始状态,而第二值(例如逻辑零)可表示非默认状态。此外,术语“重置”和“置位”在一个实施方式中分别指默认或已更新的值或状态。例如,默认值可能包括高逻辑值,即,重置,而已更新的值可能包括低逻辑值,即,置位。注意,值的任何组合可用于表示任何数量的状态。

[0072] 可经由存储在机器可访问、机器可读、计算机可访问或计算机可读介质上的由处理元件可执行的指令或代码来实现上面阐述的方法、硬件、软件、固件或代码集的实施方

式。非临时机器可访问/可读介质包括以由机器(例如计算机或电子系统)可读的形式提供(即,存储和/或传输)信息的任何机制。例如,非临时机器可访问介质包括:随机存取存储器(RAM),例如静态RAM(SRAM)或动态RAM(DRAM);ROM;磁性或光学存储介质;闪存装置;电气存储装置;光学存储装置;声存储装置;用于保持从临时(传播的)信号(例如载波、红外信号、数字信号)接收的信息的其它形式的存储装置;等等,这些介质与可从那里接收信息的非临时介质区分开。

[0073] 用于对逻辑编程以执行本公开的实施方式的指令可存储在系统中的存储器(例如DRAM、高速缓存器、闪存或其它存储装置)中。此外,指令可经由网络或借助于其它计算机可读介质被分发。因此,机器可读介质可包括用于以由机器(例如计算机)可读的形式存储或传输信息的任何机制,但不限于软盘、光学盘、光盘、只读存储器(CD-ROM)和磁光盘、只读存储器(ROM)、随机存取存储器(RAM)、可擦除可编程只读存储器(EPROM)、电可擦除可编程只读存储器(EEPROM)、磁卡或光卡、闪存、或有形机器可读存储装置,其在信息通过互联网经由电气、光学、声或其它形式的传播信号(例如载波、红外信号、数字信号等)的传输中被使用。因此,计算机可读介质包括适合于以由机器(例如计算机)可读的形式存储或传输电子指令或信息的任何类型的有形机器可读介质。

[0074] 在整个该说明书中对“一个实施方式”或“实施方式”的引用意指结合该实施方式所述的特定特征、结构或特性被包括在本公开的至少一个实施方式中。因此,短语“在一个实施方式中”或“在实施方式中”在整个该说明书中的不同地方中的出现并不一定都指同一实施方式。此外,可在一个或多个实施方式中以任何适当的方式组合特定特征、结构或特性。

[0075] 在前述描述中,关于特定的示例性实施方式给出详细描述。然而,将明显的是,可在不偏离如在所附权利要求中阐述的本公开的更宽的精神和范围的情况下对其做出各种修改和变化。说明书和附图相应地在示例性意义而不是限制性意义上被看待。此外,实施方式和其它示例性语言的前述使用并不一定指同一实施方式或同一例子,但可以指不同和独特的实施方式以及可能同一实施方式。

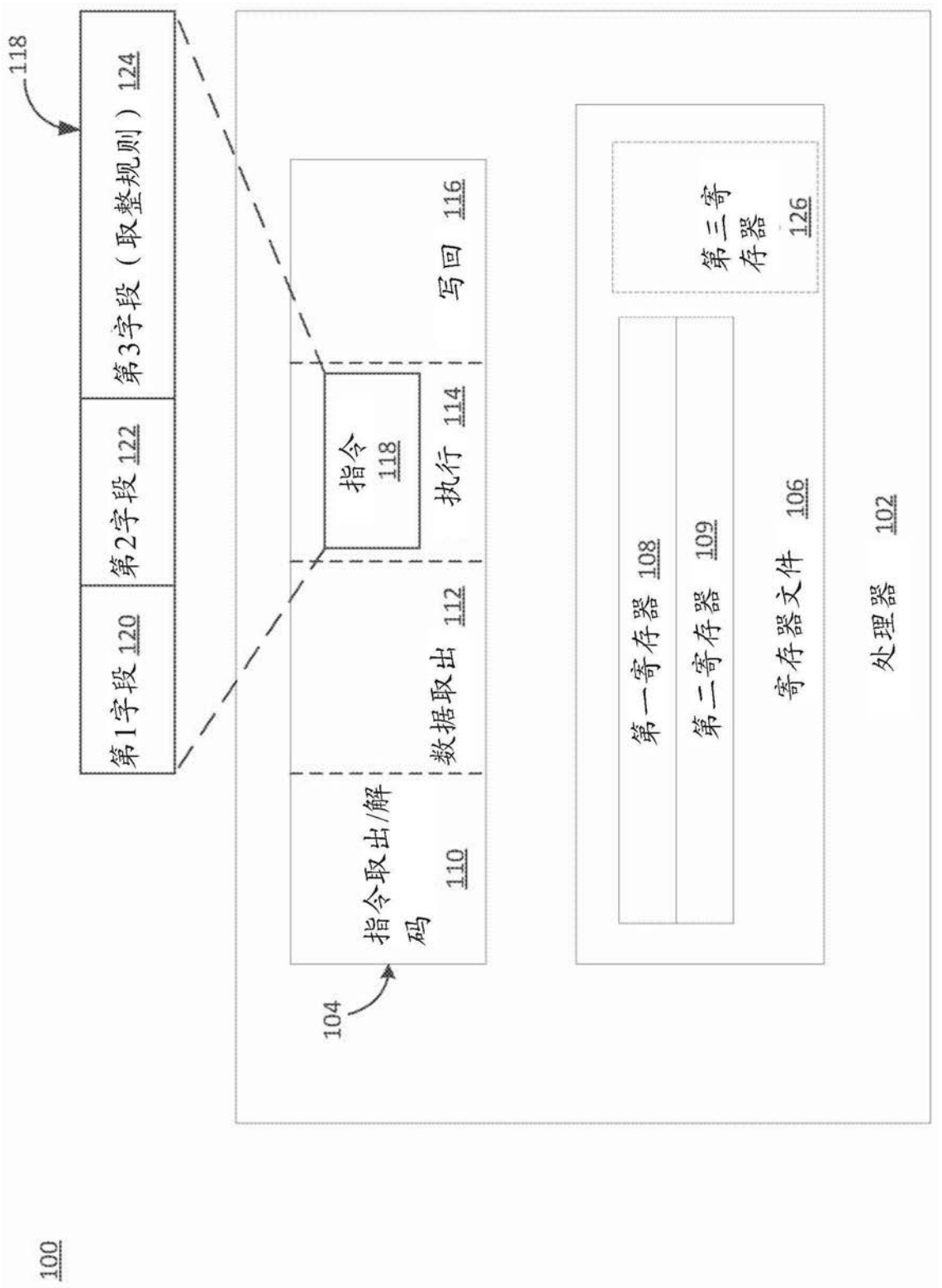

100
处理器 102
寄存器文件 106
第一寄存器 108
第二寄存器 109
第三寄存器 126
104
指令取出/解码 110
数据取出 112
执行 114
写回 116
指令 118
118
第1字段 120
第2字段 122
第3字段（取整规则） 124

图1

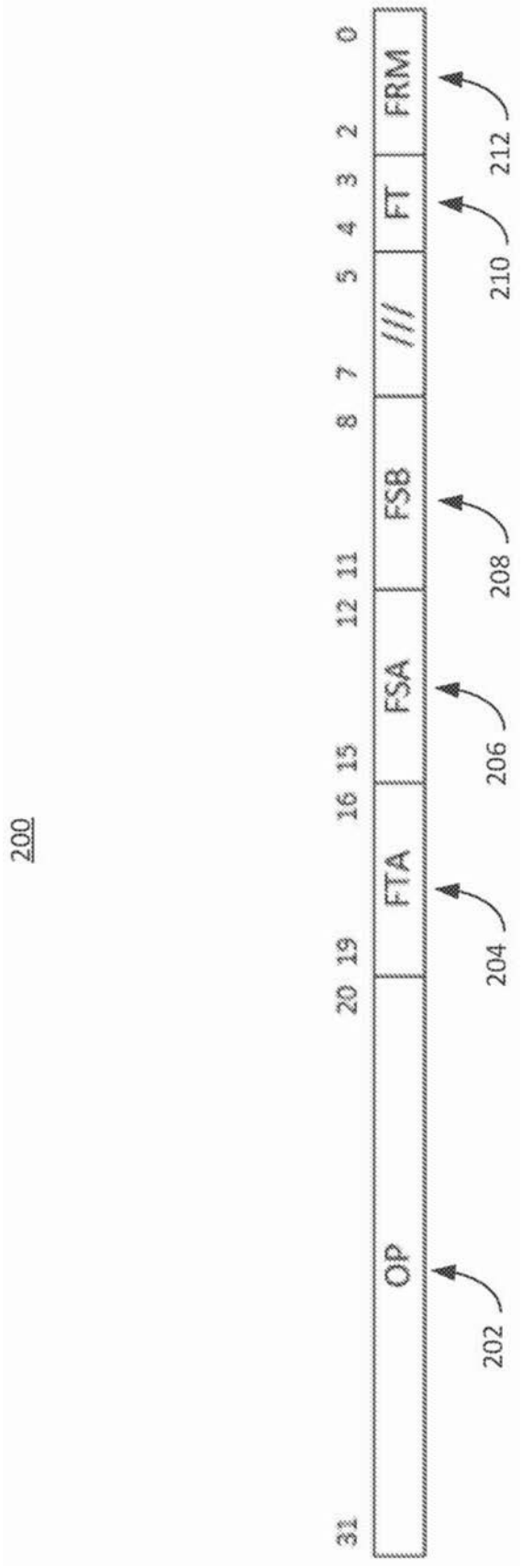

200
0
2
3
4
5
7
8
11
12
15
16
19
20
31
FRM
FT
///
FSB
FSA
FTA
OP
212
210
208
206
204
202

图2

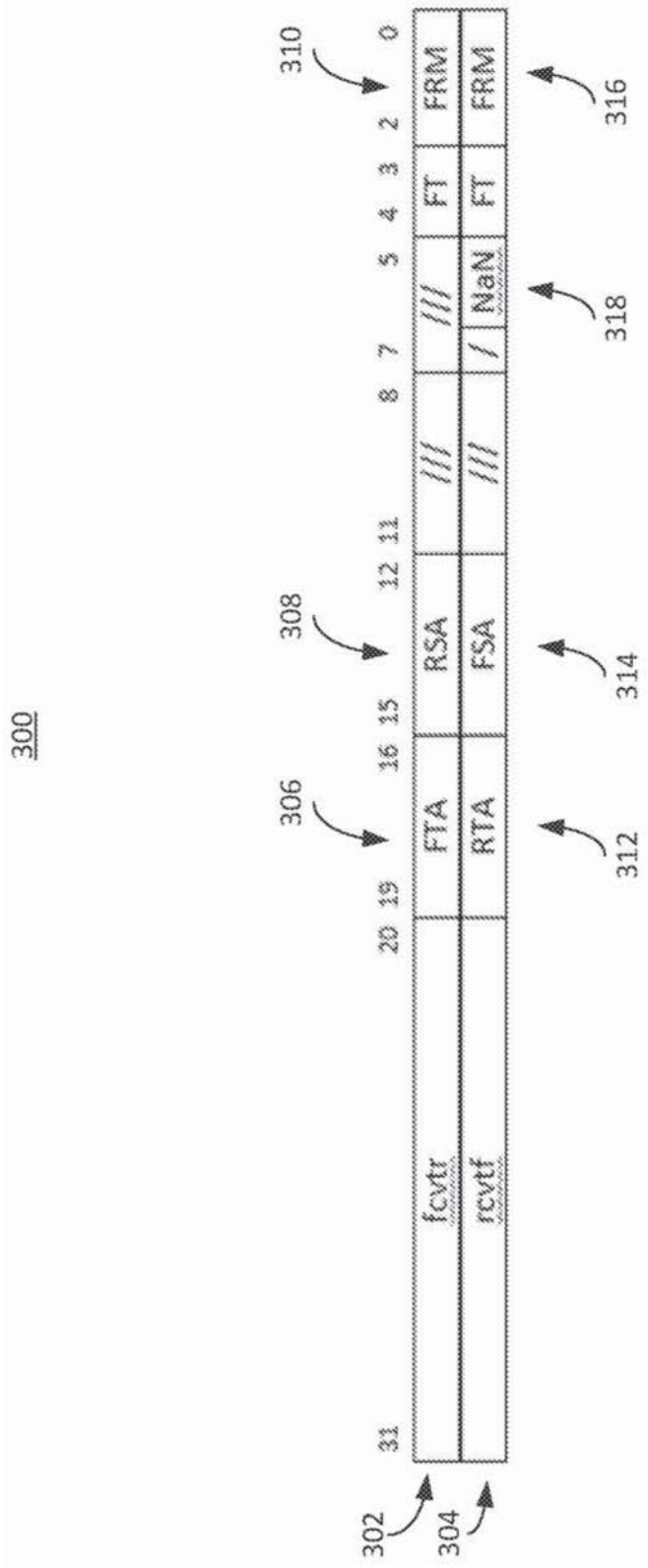

300
310
0
2
3
4
5
7
8
11
12
15
16
19
20
31
FRM
FRM
FT
FT
NaN
316
318
308
RSA
FSA
314
306
FTA
RTA
312
fcvtf
rcvtf
302
304

图3