(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 102508635 A
(43)申请公布日 2012.06.20

(21)申请号 201110319091.1

(22)申请日 2011.10.19

(71)申请人 中国科学院声学研究所
地址 100190 北京市海淀区北四环西路 21 号

(72)发明人 张铁军 王东辉 王琪 洪缨 侯朝焕

(74)专利代理机构 北京亿腾知识产权代理事务所 11309
代理人 陈霁

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

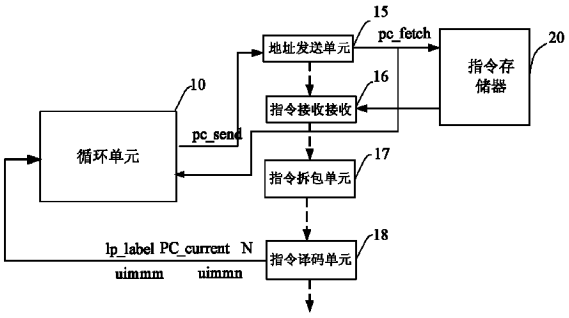
权利要求书 2 页 说明书 9 页 附图 3 页

(54)发明名称

一种处理器装置及其循环处理方法

(57)摘要

本发明公开了一种超长指令字的处理器装置及其循环处理方法，该装置包括循环单元、地址发送单元和指令译码单元，其中循环单元包括循环体数据计算模块、循环计数模块、存储模块和取指地址计算模块。该方法包括：获取循环标志指令；提取循环标志(LP)指令携带的循环参数；根据循环标志指令的地址和循环参数得到并保存循环体数据；将保存的循环体数据作为当前循环体数据；根据所述当前取指地址获取指令并执行；根据当前取指地址得到下一取指地址，并通过比较下一取指地址和循环体数据得到当前取指地址。本发明解决了超长指令字的处理器循环的控制不能完全由硬件实现，且循环执行开销大的问题，从而大大提高了超长指令字的处理器性能。



CN 102508635 A

1. 一种超长指令字的处理器装置，其特征在于包括：循环单元 (10)、地址发送单元 (15) 和指令译码单元 (18)，所述循环单元 (10) 的接收端与所述指令译码单元 (18) 的输出端相连接，所述循环单元 (10) 的发送端与所述地址发送单元 (15) 相连接，当所述指令译码单元 (18) 译码循环标志指令后，将所述循环标志指令携带的循环参数发送给所述循环单元 (10)，所述循环单元 (10) 进一步包括循环体数据计算模块 (11)、循环计数模块 (12)、存储模块 (13) 和取指地址计算模块 (14)，其中

循环体数据计算模块 (11)，用于接收所述指令译码单元 (18) 发送的循环参数，并根据所述循环标志指令的地址和循环参数得到循环体数据：循环起始地址、循环结束地址和循环次数；

存储模块 (13)，与所述循环体数据计算模块 (11) 连接，用于存储所述循环体数据；

取指地址计算模块 (14)，与所述存储模块 (13) 连接，用于读取所述存储模块 (13) 得到当前循环体数据：当前循环起始地址、当前循环结束地址和当前循环次数，根据当前取指地址得到下一取指地址，若所述下一取指地址大于所述当前循环结束地址并且所述当前循环次数非 0，则产生本次循环完成标志信号，更改当前取指地址为所述当前循环起始地址并发给地址发送单元 (15)，否则将所述下一取指地址作为当前取指地址发送给地址发送单元 (15)；

循环计数模块 (12)，连接所述指令取指地址计算模块 (14) 和所述存储模块 (13)，用于读取所述存储模块 (13) 得到当前循环次数，根据所述本次循环结束标志信号更新当前循环次数为所述当前循环次数减 1 并存入所述存储模块 (13)。

2. 根据权利要求 1 所述的装置，其特征在于，

按照先进后出顺序存储和读出所述存储模块 (13) 的循环体数据；

所述取指地址计算模块 (14) 还用于当所述当前取指地址大于所述当前循环结束地址以及所述当前循环次数是 0，产生当前循环结束标志信号；

所述存储模块 (13) 还用于根据所述当前循环结束标志信号将存储的对应循环体数据清除，然后判断所述存储模块 (13) 中是否还有保存的循环体数据，若有，则所述取指地址计算模块 (14) 按照先进后出顺序读取所述存储模块 (13) 得到当前循环体数据。

3. 根据权利要求 1 所述的装置，其特征在于：所述装置还包括指令接收单元 (16)、指令拆包单元 (17) 和指令存储器 (20)，所述指令接收单元 (16) 根据所述地址发送单元 (15) 发送的取指地址从所述指令存储器 (20) 中接收指令，所述指令拆包单元 (17) 对接收的指令进行拆包，将拆包后的指令发送给所述指令译码单元 (18)。

4. 根据权利要求 1 所述的装置，其特征在于：所述循环参数包括循环体中指令数目和循环次数。

5. 根据权利要求 4 所述的装置，其特征在于：所述循环标志指令携带的循环体中指令数目由链接器计算并赋给。

6. 根据权利要求 4 所述的装置，其特征在于：

所述循环标志指令是其所在指令执行包的最后一条指令；

根据所述循环标志指令地址和所述循环参数得到循环体数据是：通过循环标志指令地址 + 指令编码长度 / 处理器最小寻址单元长度得到循环起始地址，通过循环标志指令地址 +（循环体中指令数目 +1）×（指令编码长度 / 处理器最小寻址单元长度）得到循环结束地

址，循环次数是所述循环标志指令携带的循环次数。

7. 一种超长指令字处理器的循环处理方法，其特征在于包括以下步骤：

1) 获取循环标志指令；

2) 提取所述循环标志指令携带的循环参数；

3) 根据所述循环标志指令的地址和所述循环参数得到并保存循环体数据：循环起始地址、循环结束地址和循环次数；根据所述循环标志指令的地址得到当前取指地址；

4) 将所述保存的循环体数据作为当前循环体数据，得到当前循环起始地址、当前循环结束地址和当前循环次数；

5) 根据所述当前取指地址获取指令并执行；根据所述当前取指地址得到下一取指地址，判断所述下一取指地址是否大于所述当前循环结束地址，如否，则将所述下一取指地址作为当前取指地址，重复本步骤；若是，则判断所述当前循环次数是否是 0，若不是 0，则更新当前循环次数为所述当前循环次数减 1，将所述当前循环起始地址作为当前取指地址，重复本步骤，若是 0，则将所述下一取指地址作为当前取指地址。

8. 根据权利要求 7 所述的方法，其特征在于，

步骤 4) 进一步包括：按照先进后出顺序将所述保存的循环体数据作为当前循环体数据，得到当前循环起始地址、当前循环结束地址、当前循环次数；

步骤 5) 进一步包括：若所述下一取指地址大于所述当前循环结束地址并且所述当前循环次数是 0，则清除保存的对应循环体数据；

步骤 5) 之后还包括步骤：判断是否有保存的循环体数据，若有，则转到步骤 4)。

9. 根据权利要求 7 所述的方法，其特征在于：所述循环参数包括循环体中指令数目和循环次数。

10. 根据权利要求 9 所述的方法，其特征在于：所述循环标志指令携带的循环体中指令数目由链接器计算并赋给。

11. 根据权利要求 9 所述的方法，其特征在于：

所述循环标志指令是其所在指令执行包的最后一条指令；

根据所述循环标志指令地址和所述循环参数得到循环体数据是：通过循环标志指令地址 + 指令编码长度 / 处理器最小寻址单元长度得到循环起始地址，通过循环标志指令地址 +（循环体中指令数目 +1）×（指令编码长度 / 处理器最小寻址单元长度）得到循环结束地址，循环次数是所述循环标志指令携带的循环次数。

一种处理器装置及其循环处理方法

技术领域

[0001] 本发明涉及一种处理器技术，尤其涉及一种处理器装置及其循环处理方法。

背景技术

[0002] 随着计算机和信息技术的飞速发展，数字信号处理(Digital Signal Processing，DSP)技术应运而生并得到迅速的发展。DSP处理器则广泛的应用于数据密集型计算类应用，如图像、视频编码等。这些应用的计算复杂性和实时性特点对处理器的性能提出了更高的要求。而这些应用大部分计算都集中在循环以及循环嵌套中完成。

[0003] 在DSP的应用中，如果DSP处理器能够实现零开销循环，将可以大大提高DSP处理器的性能。所谓零开销循环是DSP处理器在执行循环时，不用花时间去检查循环计数器的值就能执行一组指令，由硬件完成循环跳转和循环计数器的增减。

[0004] 现有的零循环开销技术大多不支持超长指令字(Very Long Instruction Word，VLIW)结构的DSP处理器，而且增加的循环指令众多，硬件实现方法复杂。所支持的循环体内指令数目和循环次数有限，循环的控制不能完全由硬件实现，循环开销依然存在。

[0005] 如2010年12月6-8号在成都举办的第18届智能信号处理与通信系统国际会议(ISPACS)上Zhenqi Wei，Peilin Liu，Ji Kong，Rendong Ying等人发表的《Low-Power Microarchitecture of Zero-Overhead Nested Loops in Embedded Processors》文章中公开了通过增加专用的循环指令：LOOP.C，LOOP.B，LOOP.R，LOOP.BR来通知硬件接下来循环体的指令数目和循环次数，或者刷新硬件中循环计数器等专用寄存器的值。通过增加硬件单元，包括一个循环寄存器堆栈和循环缓存器，其中循环寄存器堆栈用于存储循环体的起始地址、结束地址、当前循环PC、循环次数和循环体是否可用、是否使用循环缓冲的标志位，循环缓存器用于存储小循环的循环指令。文章所公开的方法虽然能较好的解决零循环开销的问题，但是增加的指令较多，虽然文章提出其支持循环嵌套，但是没有提出具体的硬件实现方式。而且该方法只适用于单发射的DSP处理器结构，不适用于具有VLIW结构的高性能DSP处理器。

[0006] 如计算机期刊IEEE Transactions on computers，Vol 57，NO.2，February 2008中Nikolaos Kavvadias and Spirdon Nikoladis等人发表的《Elimination of overhead operations in complex loop structures for embedded microprocessors》文章中提到通过任务控制循环图(Task Control-flow Graph，TCFG)优化应用程序的循环节点，编译产生优化后的代码。增加的循环硬件模块包括：循环参数表、索引计算单元和任务选择单元。可以支持循环嵌套，以及多入口和多出口循环。多入口和多出口循环主要用于控制密集型循环，而大多数程序的计算密集型循环控制逻辑简单，不存在多入口和多出口。而且这种方法只支持单发射的处理器结构，而且不能被传统编译器优化，硬件结构复杂。

[0007] 如德州仪器(Texas Instruments，TI)公司DSP芯片C64X+系列支持VLIW，主要采用软件流水机制减少循环开销。硬件部分包括1个缓冲器和两个计数器，另外还有7条相关指令。其中，缓冲器可以用来存储最多14个指令执行包的循环体代码，使得循环时不

用每次访问内存重新取值，节省功耗和存储访问带宽。在程序编译过程中，编译器会自动将可以放入缓冲器的循环指令用特殊指令标识，用 MVC 指令将循环次数装载到内部循环计数器，循环代码的开始和结束分别用指令 SPLOOP 和 SPKERNEL 指令标识。这样循环体执行结束时，其硬件机制可以控制代码从循环体开始处继续执行，减少了分支指令的开销。但是该方法中，对计数器的写入和读取，循环体的开始和结束都需要通过指令实现，浪费指令周期。循环次数由循环计数器的值决定，而且从加载循环计数器值到它的内容可以使用需要 4 个时钟周期，因而要求在 SPLOOP 指令前 4 个周期加载循环计数器。这种方法对指令顺序要求严格，而且这 4 个周期的间隔通道需要通过插入空指令来实现，浪费指令周期。

发明内容

[0008] 本发明的目的是通过在超长指令字的处理器中增加一条指令和一个循环单元，解决超长指令字的处理器循环控制不能完全由硬件实现，循环执行开销大的问题。

[0009] 为实现上述目的，本发明一方面提供了一种超长指令字的处理器装置，包括循环单元、地址发送单元和指令译码单元，其中循环单元的接收端与译码单元的输出端相连接，循环单元的发送端与地址发送单元相连接，当指令译码单元译码到循环标志指令（L P 指令）时，则将循环标志指令携带的循环参数信息发送给循环单元，循环单元进一步包括循环体数据计算模块、循环计数模块、存储模块和取指地址计算模块，其中循环体数据计算模块用于接收指令译码单元发送的循环参数，并根据循环标志指令的地址和循环参数得到循环体数据：循环起始地址、循环结束地址和循环次数；存储模块与循环体数据计算模块连接，用于存储循环体数据；取指地址计算模块与存储模块连接，用于读取存储模块得到当前循环体数据：当前循环起始地址、当前循环结束地址和当前循环次数，根据当前取指地址得到下一取指地址，若下一取指地址大于当前循环结束地址并且当前循环次数非 0，则产生本次循环完成标志信号，更改当前取指地址为当前循环起始地址并发送给地址发送单元，否则将下一取指地址作为当前取指地址发送给地址发送单元；循环计数模块连接指令取指地址计算模块和存储模块，用于读取存储模块得到当前循环次数，根据本次循环结束标志信号更新当前循环次数为当前循环次数减 1 并存入所述存储模块。

[0010] 本发明另一方面提供了一种超长指令字处理器的循环处理方法，该方法的步骤包括：(1) 获取循环标志指令；(2) 提取循环标志指令携带的循环参数；(3) 根据循环标志指令的地址和循环参数得到并保存循环体数据：循环起始地址、循环结束地址、循环次数；根据循环标志指令的地址得到当前取指地址；(4) 将保存的循环体数据作为当前循环体数据，得到当前循环起始地址、当前循环结束地址、当前循环次数；(5) 根据当前取指地址获取指令并执行；根据当前取指地址得到下一取指地址，判断下一取指地址是否大于当前循环结束地址，如否，则将下一取指地址作为当前取指地址，重复本步骤；若是，则判断当前循环次数是否是 0，若不是 0，则更新当前循环次数为所述当前循环次数减 1，将当前循环起始地址作为当前取指地址，重复本步骤，若是 0，则将下一取指地址作为当前取指地址。

[0011] 根据本发明的处理器装置及其循环处理方法，循环执行完全由硬件实现，无需通过指令实现循环计数器的赋值和修改，能够实现超长指令字的处理器循环零开销，可以大大提高超长指令字处理器的性能。

附图说明

[0012] 通过以下结合附图以举例方式对本发明的实施方式进行详细描述后，本发明的其他特征、特点和优点将会更加明显。

[0013] 图 1 是循环标志指令示意图；

[0014] 图 1A 是本发明一实施例循环标志指令编码示意图；

[0015] 图 1B 是本发明另一实施例循环标志指令编码示意图；

[0016] 图 2A 是本发明一个实施例超长指令字的处理器结构示意图；

[0017] 图 2B 是本发明另一个实施例超长指令字的处理器结构示意图；

[0018] 图 3 是本发明实施例处理器的循环处理方法流程图。

具体实施方式

[0019] 本发明通过在现有技术的超长指令字的处理器中增加一条循环标志指令 (LP 指令) 和一个循环单元实现了超长指令字的处理器循环执行零开销。

[0020] 针对 LP 指令的编码格式以及该指令的定义做如下描述。

[0021] LP 指令语法中带有一个标号 (label)，一个无符号参数 (uimmn) 和一个并行标志位符号。标号 (label) 代表该循环体结束后下一个指令执行包的标号，无符号参数 (uimmn) 代表该循环体内的循环次数，并行标志位符号代表本条指令和下一条指令是否同属于一个指令执行包，即是否可以并行执行。程序中每个指令执行包内可包括 1 到 N 条指令，N 为该超长指令字处理器支持的多发射数目。如图 1 所示，可以通过自动或手动标识循环体内容，循环次数，在循环体结束后下一个指令执行包前插入相应的标号 (label)，并且在循环体前插入指令 LP label，uimmn. 。其中参数 label 等于循环体结束后下一个指令执行包前插入的标号名称。在链接过程中，汇编器和链接器通过 (label 指令包的首地址 -LP 指令的地址)/(指令编码长度 / 处理器最小寻址单元长度)-1 得到循环体内指令数目，并将其赋给 LP 指令编码中无符号参数 (uimmm)。无符号参数 (uimmn) 代表该循环体内的循环次数。“.”是并行标志位符号，可自己定义。此处“.”表示该指令与后面的指令不属于一个指令执行包，不能并行执行。

[0022] 图 1A 是本发明一实施例循环标志指令编码示意图。如图所示，LP 指令编码总长度为 L_{inst}，其中指令标识位占 a 位；并行位等占 b 位；LP 指令第一操作数 uimmm 占 m 位，用于存储循环体中的指令的数目，该指令的数目由公式 (uimmm+1)*(L_{inst}/Step)+lp_pc ＝ PC_label 计算得到，其中 lp_pc 代表 LP 指令的地址，PC_label 代表循环体结束后下一个指令执行包的首指令地址；第二操作数 uimmn 占 n 位，用于存储循环体的循环次数；Step 代表处理器的最小寻址单元位宽，例如按字节寻址的处理器的最小寻址单元位宽为 8 比特，按字寻址的处理器的最小寻址单元位宽为 32 比特。

[0023] 在指令译码阶段，通过译码可以将循环体中的指令数目 uimmm 信息和循环的次数 uimmn 信息传递给循环单元做相应的处理。针对不同的处理器，其 LP 指令编码总长度 Z，以及标识位和并行位等所占的长度都不一致，但都是固定的，而 m+n ＝ L_{inst}-a-b，具体 m 和 n 的长度可以根据不同的处理器编码格式进行调节，一般 n ＞ m。由于超长指令字的处理器应用中，密集计算的循环体内容都不会很大，但循环次数却可能会很大。可以支持的循环体内指令条数为 2^n，支持循环次数为 2^m。要求 LP 指令必须为其所在指令执行包中的最后一条指

令。

[0024] 在一个实施例中，以某一超长指令字的处理器设计为例。LP 指令编码如图 1B 所示，LP 指令编码总长度为 32 比特，其中标识位占 5 比特；pr 占 3 比特，代表推断寄存器的索引；p 占 1 比特，代表并行标志位；操作数 imm9 占 9 比特，代表循环体内的指令数目高达 2^9；操作数 imm14 占 14 比特，代表循环次数高达 2^{14}。根据操作数 imm9、操作数 imm14 和当前 LP 指令所在指令执行包的首指令地址和该指令执行包内指令数目可以计算出循环体的开始的指令地址和结束的指令地址。

[0025] 以上针对超长指令字的处理器中增加的 LP 指令做了相应描述，以下结合附图针对 DSP 处理器进行描述。

[0026] 图 2A 是本发明一个实施例超长指令字的处理器结构示意图。如图 2A 所示，该处理器包括循环单元 10、地址发送单元 15、指令接收单元 16、指令拆包单元 17、指令译码单元 18 和指令存储器 20，其中循环单元 10 的接收端与指令译码单元 18 的输出端相连接，循环单元 10 的发送端与地址发送单元 15 相连接。

[0027] 指令接收单元 16 根据地址发送单元 15 发送的取指地址从指令存储器 20 中接收指令，由指令拆包单元 17 对指令接收单元 16 接收的指令进行拆包处理，并将拆包处理后的指令发送给指令译码单元 18。当指令译码单元 18 译码到循环标志指令后，则将循环标志指令携带的循环参数发送给循环单元 10 进行处理，循环参数包括循环体中指令数目 (uimmm) 和循环次数 (uimmn)。其中循环体中指令数目和循环次数通过循环标志指令编码中的第一操作数和第二操作数获取。循环单元 10 根据接收到的循环参数信息以及接收的指令译码单元 18 当前正在译码的指令执行包中的指令数目 (N)、正在译码的指令执行包首指令地址 (PC_current)、LP 指令标志 (lp_lable)。根据循环标志指令所在指令执行包的首指令地址和所在指令执行包中指令数目得到该循环标志指令的地址，根据循环标志指令的地址和循环参数得到并保存循环体数据：循环起始地址、循环结束地址和循环次数。按照先进后出顺序将保存的循环体数据读出作为当前循环体数据：当前循环起始地址、当前循环结束地址和当前循环次数；根据当前循环体数据和地址发送单元 15 发送的取指地址 (pc_fetch)，计算并获得下一周期指令的取指地址再发送给地址发送模块 15。

[0028] 图 2B 是本发明另一个实施例超长指令字的处理器结构示意图。如图 2B 所示，处理器的循环单元 10 进一步包括循环体数据计算模块 11、循环计数模块 12、存储模块 13 和取指地址计算模块 14。

[0029] 循环体数据计算模块 11 与存储模块 13 和指令译码单元 18 连接，接收来自指令译码单元 18 发送的循环参数，上述循环参数包括循环体中指令数目 (uimmm) 和循环次数 (uimmn)。此外循环体数据计算模块 11 还接收译码单元 18 当前正在译码的指令执行包中的指令数目 (N)、正在译码的指令执行包首指令地址 (PC_current)、LP 指令标志 (lp_lable)，此外循环体数据计算模块 11 还接收存储模块 13 的信号 counter[0] 寄存器的值。循环体数据计算模块 11 通过上述接收的循环参数信息生成循环使能信号 (loop_en)，当 LP 指令标志 (lp_lable) 有效时，置循环使能信号有效，当从存储模块 13 中读出的 counter[0] 寄存器为 0 时，说明循环（对于嵌套循环而言指最外层循环）执行完毕，置循环使能信号无效，其余情况下循环使能信号保持不变。循环体数据包括循环起地址、循环结束地址和循环次数，其中根据公式：$lp\_pc = PC\_current+(N-1)*(L_{inst}/Step)$，计算出 LP 指令的地址 (lp_pc)；

根据公式：lpc_start = lp_pc+(L_{inst}/Step)，计算出循环体的起始地址（lpc_start）；根据公式：lpc_end = lp_pc+(uimmm+1)*(L_{inst}/Step)，计算出循环体的结束地址（lpc_end）；根据公式：counter = uimmn，得到循环次数（counter）。循环体数据计算模块 11 将得到的循环体数据和循环使能信号和 LP 指令标志一起发送给存储模块 13，同时将循环使能信号发送给循环计算模块 12。

[0030] 存储模块 13 连接循环体数据计算模块 11 和取指地址计算模块 14，存储由循环体数据计算模块 11 发送的循环体数据：循环体的起始地址、循环体的结束地址和循环体的循环次数。

[0031] 存储模块 13 选用 FILO(First In Last Out，先进后出）存储模块，通过该存储模块可支持多层嵌套循环，设支持的嵌套循环层数为 N，那么存储模块的深度就必须为 N。以 N = 4 为例，如表一所示，支持 4 层嵌套循环。存储模块内含 3 组寄存器，分别为 lpc_start 寄存器、lpc_end 寄存器和 counter 寄存器，分别用于存储循环体起始地址、循环体结束地址和循环体循环次数。

[0032] 表一：

[0033]

N	0	1	2	3
lpc_start				
lpc_end				
counter				

[0034] 存储模块 13 的数据读出和写入分别受到读指针 rp 和写指针 wp 的控制，读指针 rp 指向当前循环应该读取的存储模块的位置，代表了目前正在执行的循环体所在的循环嵌套层次，写指针 wp 指向下一次遇到循环标志 lp_label 时，应该写入的存储模块的位置，代表下一个循环嵌套的层次。读指针和写指针的数值是 0 至 N-1。本发明通过修改读指针的方法实现存储内容清除或无效的功能。

[0035] 当存储模块 13 接收的 LP 指令标志 lp_lable 信号有效时，则将接收到的循环体数据：循环体的起始地址、循环体的结束地址和循环次数分别写入到存储模块 13 中的 lpc_start 寄存器、lpc_end 寄存器和 counter 寄存器，然后修改读写指针，此时正常情况下，读写指针分别加 1，但是以下三种情况除外：(1) 如果当前读指针 rp 和写指针 wp 都为 0，则表明之前没有循环执行，或之前的循环已经完成，即将开始一个新的循环，这时如果有数据写入存储模块 13，那么读指针 rp 保持 0 不变，写指针 wp 加 1。(2) 如果当前写指针为 N-1，表明当前循环标志 lp_label 所在的循环嵌套层次是该处理器所能支持的最大的第 N 层嵌套循环，所以写指针保持不变，读指针正常加 1。(3) 如果当前读指针为 N-1，表明当前正在执行的循环的循环嵌套层次是该处理器所能支持的最大的第 N 层嵌套循环，而此时循环标志 lp_label 所在的循环嵌套层次是 N+1，超出了处理器所能支持的范围，属于例外情况，这种情况应该由编译器根据处理器的参数来避免，不应该出现。这种情况下设定读写指针都保持不变。

[0036] 如果当前嵌套层的循环执行完毕，即当前读指针所指的循环次数寄存器

counter[rp] 为 0，那么正常情况下，读写指针应该减 1。但下述情况除外：(1) 如果当前写指针为 0，说明还未有循环参数写入存储体，因而读写指针应该保持为 0；(2) 如果当前读指针为 N-1，表明正在执行的是该处理器所能支持的最大的第 N 层嵌套循环，因而当第 N 层嵌套循环执行完毕时，等待下一次写入的是第 N 层嵌套循环，所以写指针保持不变，读指针减 1；(3) 如果当前读指针为 0，说明正在执行的是第 1 层嵌套循环，即循环的最外层，那么当第 1 层嵌套循环执行完毕时，说明当前循环执行完毕，下一次写入的将是一个新的循环，因而读指针应该保持为 0，写指针减 1。

[0037] 在一个例子中，存储模块 13 的内容写入控制代码如下所示：

[0038]

```
always @(posedge clk or negedge rst)
  begin
  if(!rst)
     begin
```

[0039]

```
        LPC_START[0] <= 32'b0;   LPC_END[0] <= 32'b0; COUNTER[0] <= 32'b0;
        LPC_START[1] <= 32'b0;   LPC_END[1] <= 32'b0; COUNTER[1] <= 32'b0;
        LPC_START[2] <= 32'b0;   LPC_END[2] <= 32'b0; COUNTER[2] <= 32'b0;
        LPC_START[3] <= 32'b0;   LPC_END[3] <= 32'b0; COUNTER[3] <= 32'b0;
    end
  else if(lp_label)   //write the lp parameter in the LP_FIFO//
    begin
        LPC_START [wp] <= lp_start;
        LPC_END [wp] <= lp_end;
        COUNTER [wp] <= lp_counter;
     end
   else                    //change the COUNTER[wp]
        COUNTER [wp] <= counter_next;
  end
```

[0040] 在上述代码中，当复位信号 rst 信号有效时，将存储模块 13 中的内容复位为全 0。当 LP 指令标志 (lp_lable) 信号有效时，则将循环体数据计算模块 11 发过来的循环起始地址、循环结束地址和循环次数分别写入到写指针 wp 指向的寄存器 lpc_start[wp]、寄存器 lpc_end[wp] 和寄存器 counter[wp] 中，否则将循环计数模块 12 发送的下一个周期的循环次数 (counter_next) 信号根据写指针写入到相应的循环次数寄存器 counter[wp] 中。

[0041] 在一个例子中，存储模块 13 的内容读出控制代码如下所示：

[0042]

```
always @(loop_en,LPC_START,LPC_END,COUNTER,rp)
begin
  if(loop_en)
    begin
          lpc_start_now = LPC_START[rp];
           lpc_end_now = LPC_END[rp];
          counter_now = COUNTER[rp];
    end
  else
    begin
           lpc_start_now = 32'b0;
           lpc_end_now = 32'b0;
           counter_now = 32'b0;
    end
end
```

[0043] 在上述代码中，当来自循环体数据计算模块 11 的循环使能信号 loop_en 有效时，根据读指针 rp 将存储模块 13 中的 lpc_start[rp] 寄存器，lpc_end[rp] 寄存器，counter[rp] 寄存器中的信息读出，信号名分别为当前循环的循环次数 (counter_now)、当前循环的循环体起始地址 (lpc_start_now) 和当前循环的循环体结束地址 (lpc_end_now)，否则输出信号“32′ b0”，上述代码中“32′ b0”代表 32 比特都为 0。

[0044] 取指地址计算模块 14 连接存储模块 13 和循环计数模块 12，从存储模块 13 读出当前循环的循环次数 (counter_now)、当前循环的循环体起始地址 (lpc_start_now) 和当前循环的循环体结束地址 (lpc_end_now)，并且根据以上参数和当前取指地址计算出下一取指地址，下一取指地址＝当前取指地址 (pc_fetch)+指令包的宽度 / 寻址单元位宽 (Step)。判断下一取指地址是否大于当前循环结束地址，如果不是，说明当前循环体尚未执行完毕，则下一取指地址作为当前取指地址 (pc_send)；如果是，说明当前循环体执行完毕，产生本次循环完成标志信号 (loops_end)，并发送给循环计数模块 12，进而判断当前循环次数是否是 0，若不是 0，则说明当前循环尚未执行完毕，应从循环体开始处再进行取指，将当前循环起始地址作为当前取指地址；若是 0，说明当前循环执行完毕，产生循环结束标志信号 (loop_end)，程序应该顺序往下执行，则将下一取指地址作为当前取指地址。

[0045] 循环计数模块 12 用于循环计数器的计算，分别与循环体数据计算模块 11、存储模块 13 和取指地址计算模块 14 连接，从存储模块 13 中读取当前循环的循环次数 (counter_now) 信号，并接收来自取指地址计算模块 14 的循环结束信号和来自循环体数据计算模块 11 的循环使能信号。循环计数模块 12 通过上述信号产生下一个周期的循环次数 (counter_next) 信号，如果循环使能信号有效，并且本次循环完成标志信号 (loops_end) 有效，更新下一个周期的循环次数 (counter_next) 为当前循环次数 (counter_now) 减 1，并将下一个周期的循环次数 (counter_next) 信号发送给存储模块 13，用于更新当前循环次数 counter[rp] 寄存器。

[0046] 图 3 是本发明实施例超长指令字处理器的循环处理方法流程图。该方法的步骤包括 301-311：

[0047] 在步骤 301，获取循环标志指令。

[0048] 循环标志指令带有两个操作数，用于存储循环参数，该循环参数包括循环体中指

令数目和循环次数，其中第一操作数 (uimmm) 用于存储循环体中的指令数目，循环体中指令数目是由链接器根据循环标志指令包的首地址和循环标志指令的地址相减再除以一条指令所占地址的步长再减 1 得到；第二操作数 (uimmn) 用于存储循环体的循环次数。处理器在指令取指阶段根据当前取指地址从指令存储器中获取指令，并对指令进行译码操作，当译码到循环标志指令后，处理器获取该循环标志指令。

[0049] 在步骤 302，提取循环标志指令携带的循环参数。

[0050] 当处理器获取循环标志指令后，通过循环标志指令的第一个操作数 (uimmm) 提取循环体中指令数目，通过第二操作数 (uimmn) 提取循环次数。

[0051] 在步骤 303，根据循环标志指令的地址和循环参数得到并保存循环体数据，并根据循环标志指令的地址得到当前取指地址。

[0052] 处理器根据公式：lp_pc ＝ PC_current+(N-1)*(L_{inst}/Step) 得到循环标志指令的地址，其中 lp_pc 代表循环标志指令地址、PC_current 代表正在译码的指令执行包首指令地址、N 代表当前正在译码的指令执行包中的指令数目、L_{insn} 代表指令编码长度、Step 代表处理器的最小寻址单元位宽。

[0053] 处理器根据得到的循环标志指令的地址和在步骤 302 提取的循环参数得到并保存循环体数据，该循环体数据包括循环起始地址、循环结束地址和循环次数，其中循环起始地址根据公式：lpc_start ＝ lp_pc+(L_{inst}/Step) 得到，lpc_start 代表循环起始地址；循环结束地址根据公式：lpc_end ＝ lp_pc+(uimmm+1)*(L_{inst}/Step) 得到，lpc_end 代表循环结束地址，公式中 (uimmm+1) 代表循环体中指令数目 +1；循环次数是循环标志指令第二操作数 (uimmn) 携带的循环次数。

[0054] 在步骤 304，将保存的循环体数据作为当前循环体数据，得到当前循环起始地址、当前循环结束地址和当前循环次数。

[0055] 优选地，处理器按照先进后出的顺序将保存的循环体数据作为当前循环体数据，得到当前循环起始地址、当前循环结束地址和当前循环次数。

[0056] 在步骤 305，处理器根据当前取指地址从指令存储器中获取指令并执行相应的操作。

[0057] 在步骤 306，处理器根据当前取指地址计算出下一取指地址。

[0058] 在步骤 307，处理器将下一取指地址与当前循环结束地址进行比较，如果下一取指地址不大于当前循环结束地址，说明当前循环体尚未执行完毕，流程进入步骤 308；如果下一取指地址大于当前循环结束地址，说明当前循环体执行完毕，产生本次循环完成标志信号，流程进入步骤 309。

[0059] 在步骤 308，处理器将下一取指地址作为当前取指地址，流程转而执行步骤 305。

[0060] 在步骤 309，处理器进一步判断当前循环次数是否是 0，如果当前循环次数不是 0，则说明当前循环尚未执行完毕，流程进入步骤 310；如果循环次数是 0，说明当前循环执行完毕，产生循环结束标志信号，流程转而进入步骤 308。

[0061] 在步骤 310，处理器更新当前循环次数为当前循环次数减 1。

[0062] 在步骤 311，处理器将当前循环起始地址作为当前取指地址，流程转而进入步骤 305。

[0063] 在本发明实施例中，处理器的循环处理方法进一步包括：当下一取指地址大于当

前循环结束地址并且当前的循环次数是 0 时，处理器清除保存的对应循环体数据，如果处理器还有保存的循环体数据时，则流程跳转至步骤 304 继续执行，否则循环处理结束。

[0064] 本发明实施例循环处理方法的循环执行完全由硬件实现，无需通过指令实现循环计数器的赋值和修改，能够实现超长指令字的处理器循环零开销，可以大大提高超长指令字处理器的性能。

[0065] 显而易见，在不偏离本发明的真实精神和范围的前提下，在此描述的本发明可以有许多变化。因此，所有对于本领域技术人员来说显而易见的改变，都应包括在本权利要求书所涵盖的范围之内。本发明所要求保护的范围仅由所述的权利要求书进行限定。

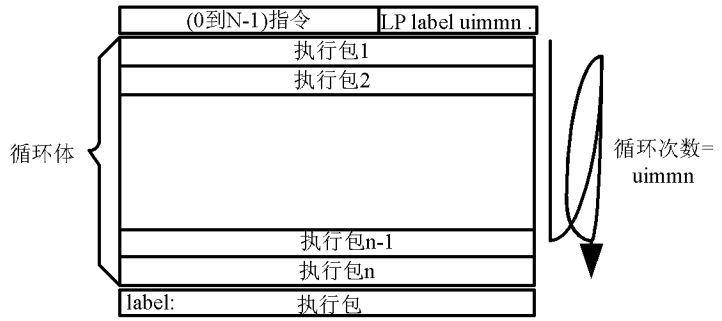

(0到N-1)指令
LP label uimmn .
执行包1
执行包2
循环体
循环次数=
uimmn
执行包n-1
执行包n
label:
执行包

图 1

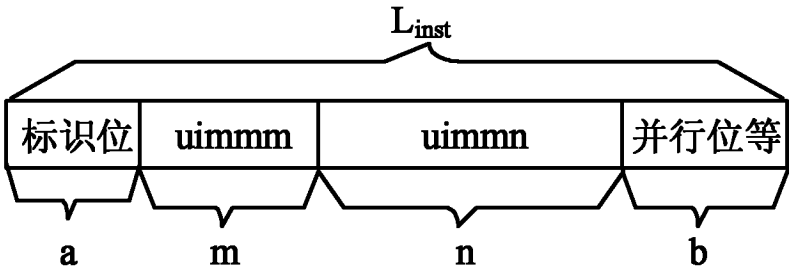

Linst
标识位
uimmm
uimmn
并行位等
a
m
n
b

图 1A

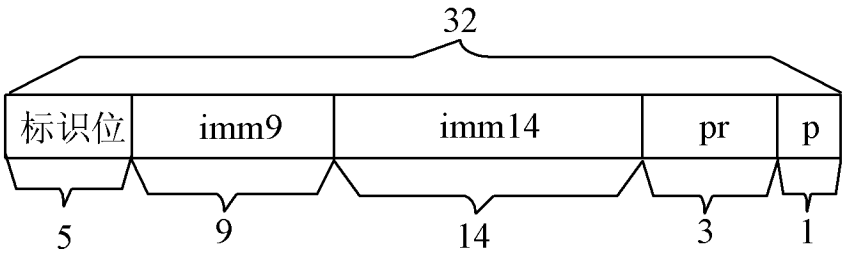

32
标识位
imm9
imm14
pr
p
5
9
14
3
1

图 1B

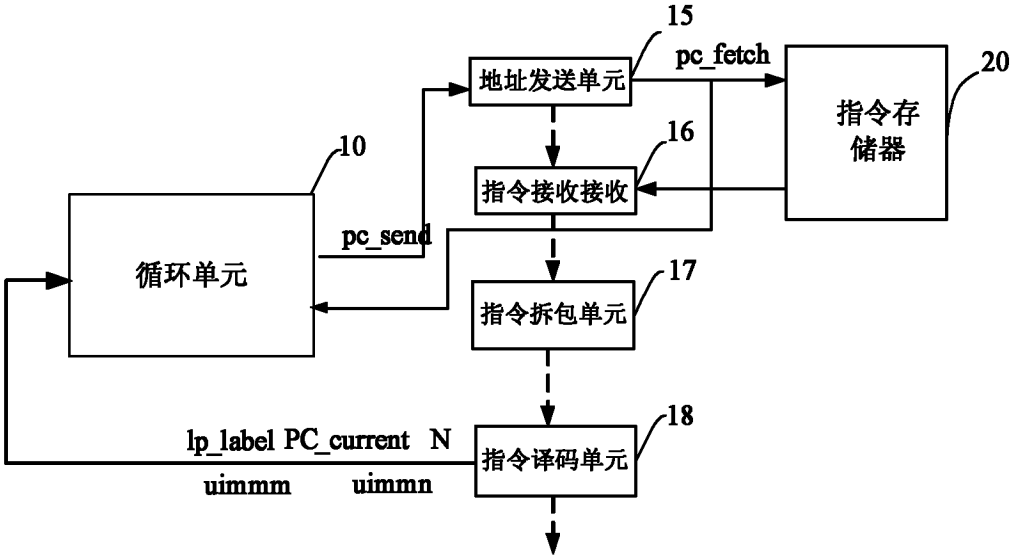

15
地址发送单元
pc_fetch
20
指令存储器
16
指令接收接收
10
循环单元
pc_send
17
指令拆包单元
18
lp_label PC_current N
uimmm uimmn
指令译码单元

图 2A

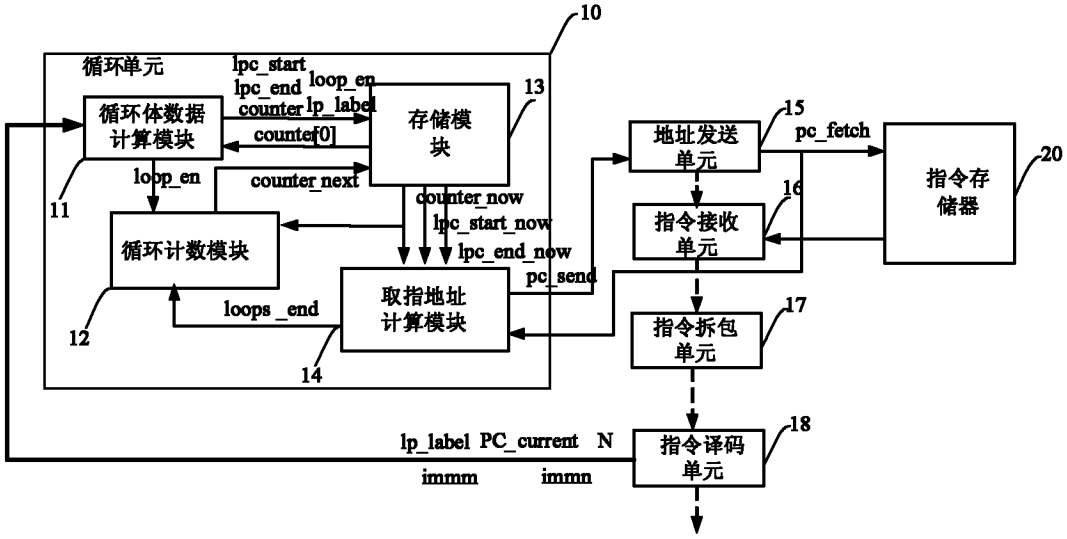

10
循环单元
lpc_start
lpc_end
loop_en
counter
lp_label
counter[0]
存储模块
13
循环体数据计算模块
11
loop_en
counter_next
counter_now
lpc_start_now
lpc_end_now
循环计数模块
12
loops _end
取指地址计算模块
14
pc_send
地址发送单元
15
pc_fetch
20
指令存储器
16
指令接收单元
17
指令拆包单元
18
lp_label PC_current N
immm immn
指令译码单元

图 2B

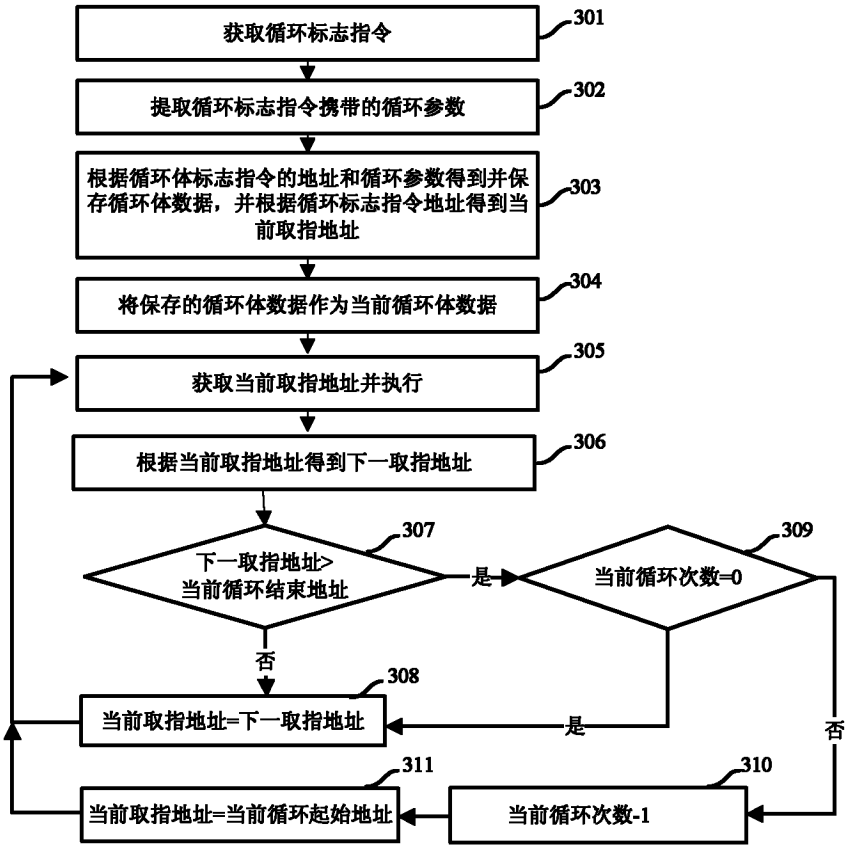

获取循环标志指令
301
提取循环标志指令携带的循环参数
302
根据循环体标志指令的地址和循环参数得到并保存循环体数据，并根据循环标志指令地址得到当前取指地址
303
将保存的循环体数据作为当前循环体数据
304
获取当前取指地址并执行
305
根据当前取指地址得到下一取指地址
306
下一取指地址>
当前循环结束地址
307
是
当前循环次数=0
309
否
当前取指地址=下一取指地址
308
是
否
当前取指地址=当前循环起始地址
311
当前循环次数-1
310

图 3