

risv_v vector 指令集 vsetli 指令的实现方法及系统

技术领域

本发明涉及 CPU 技术领域，具体涉及一种 risv_v vector 指令集 vsetli 指令的实现方法及系统。

背景技术

isc_v 指令集才近期公布了完整的指令集，目前可参考的实现方式基本没有。为了实现简单，最简单的办法就是，vsetli 指令毕业时就需要刷新流水，同时执行单元不管 unactive element 部分，都送到执行单元去执行，导致执行周期的增加。

现有的 vsetli 指令毕业时就需要刷新流水，导致 CPU 执行效率低，Vector 指令对 unactive element 部分也在执行单元中执行了，最后再通过 mask 的方式来对数据进行选择，其实 mask 的数据本来是不需要进入执行单元的，这样导致的结果就是让费功耗，同时还增加了指令的执行周期。

发明内容

针对现有技术的不足，本发明公开了一种 risv_v vector 指令集 vsetli 指令的实现方法及系统，用于解决上述存在的问题。

本发明通过以下技术方案予以实现：

第一方面，本发明公开了一种 risv_v vector 指令集 vsetli 指令的实现方法，包括以下步骤：

S1CPU 乱序执行时，在 rename 模块中分配 vectag[n:0]信息，判断指令是否为 vsetli；

S2 若指令是 vsetli，则将 vectag+1，若是非 vsetli 指令，则 vectag 保持不变；

S3 发射到执行单元，对 vsetli 指令分发到 csr 模块；对应其它的 vector 指令分发到 vpu 模块；

S4 判断指令 vectag 与 ROB 广播的 vectag 一致时,指令从 reserver station 发射到执行单元;

S5 指令执行完成,在 ROB 模块中,按顺序毕业,并在毕业时更新寄存器 vectag,执行结束。

更进一步的,所述方法中,每个 cycle 发射 0-5 条指令。

更进一步的,所述方法中,若接受 vsetli 指令,则该 cycle 只发射 vsetli,每个 cycle 分配一个 vectag,其它指令等到下一 cycle 再发射。

更进一步的,所述方法中,将 unactive 发射到执行单元,执行 2*n cycle 完成,将 unactive 不发射到执行单元,执行 n cycle 完成。

更进一步的,所述方法中,vpu 模块 reserve station 中的指令 vectag 都需要和寄存器 vectag 比较,只有一致的情况下,该指令才能发射到执行单元。

更进一步的,所述方法中,在 rename 中分配 vectag[n:0],作为 vpu 指令发射到执行单元的一个条件,用以在执行 vsetli 指令时不刷新流水线。

第二方面,本发明公开了一种 risv_v vector 指令集 vsetli 指令的实现系统,所述系统用于执行第一方面所述的 risv_v vector 指令集 vsetli 指令的实现方法,包括 rename 模块、dispatch 模块、vpu 模块和 ROB 模块。

本发明的有益效果为:

本发明非 vsetl{i} Vector 指令只需根据比自己更老的 vsetl{i} 中最年轻的指令执行完才能进入执行单元,比目前的刷新流水效率高很多。

本发明 Vector 指令对 unactive element 部分也在执行单元中执行了,最后再通过 mask 的方式来对数据进行选择,可以减少功耗,同时可以减少执行周期,减少 latency。

附图说明

为了更清楚地说明本发明实施例或现有技术中的技术方案，下面将对实施例或现有技术描述中所需要使用的附图作简单地介绍，显而易见地，下面描述中的附图仅仅是本发明的一些实施例，对于本领域普通技术人员来讲，在不付出创造性劳动的前提下，还可以根据这些附图获得其他的附图。

图 1 是一种 `risv_v vector` 指令集 `vsetli` 指令的实现方法的原理步骤图；

图 2 是本发明实施例乱序 CPU 基本框图；

图 3 是本发明实施例 `unactive` 发射对比框图。

具体实施方式

为使本发明实施例的目的、技术方案和优点更加清楚，下面将结合本发明实施例中的附图，对本发明实施例中的技术方案进行清楚、完整地描述，显然，所描述的实施例是本发明一部分实施例，而不是全部的实施例。基于本发明中的实施例，本领域普通技术人员在没有作出创造性劳动前提下所获得的所有其他实施例，都属于本发明保护的范围。

实施例 1

本实施例公开如图 1 所示的一种 `risv_v vector` 指令集 `vsetli` 指令的实现方法，包括以下步骤：

S1CPU 乱序执行时，在 `rename` 模块中分配 `vectag[n:0]` 信息，判断指令是否为 `vsetli`；

S2 若指令是 `vsetli`，则将 `vectag+1`，若是非 `vsetli` 指令，则 `vectag` 保持不变；

S3 发射到执行单元，对 `vsetli` 指令分发到 `csr` 模块；对应其它的 `vector` 指令分发到 `vpu` 模块；

S4 判断指令 `vectag` 与 ROB 广播的 `vectag` 一致时，指令从 `reserver`

station 发射到执行单元;

S5 指令执行完成, 在 ROB 模块中, 按顺序毕业, 并在毕业时更新寄存器 vectag, 执行结束。

本实施例中, 每个 cycle 发射 0-5 条指令。若接受 vsetli 指令, 则该 cycle 只发射 vsetli, 每个 cycle 分配一个 vectag, 其它指令等到下一 cycle 再发射。

本实施例中, 将 unactive 发射到执行单元, 执行 $2*n$ cycle 完成, 将 unactive 不发射到执行单元, 执行 n cycle 完成。

本实施例中, vpu 模块 reserve station 中的指令 vectag 都需要和寄存器 vectag 比较, 只有一致的情况下, 该指令才能发射到执行单元。

本实施例中, 在 rename 中分配 vectag[n:0], 作为 vpu 指令发射到执行单元的一个条件, 用以在执行 vsetli 指令时不刷新流水线。

本实施例 vsetli 指令毕业时不用刷新流水线, unactive element 部分不用发射到执行单元去执行, 这样可以减少功耗和执行周期。

实施例 2

本实施例参照乱序 CPU, 其基本框如图 2 所示, 本实施例公开一种 risv_v vector 指令集 vsetli 指令的实现系统包括 rename、dispatch、ROB 和 vpu 四个模块。

本实施例的 rename 模块, 在 rename 模块中会分配一个 vectag[n:0] 信息, 如果是 vsetli, 则将 vectag+1, 非 vsetli 指令的 vectag 保持不变, 这样做的目的是为了在 vpu 单元执行的指令, 可以通过把 reserve station 中指令的 vectag 和 csr 广播出来的 vectage 一致的情况下, 才可能发射到执行单元中去执行。

本实施例的 dispatch 模块, 功能是按照指令的类型, 将指令分发到不同的 datapath, 对应 vsetli 指令, 则分发到 csr 模块; 对应其它的 vector 指令, 则分发到 vpu 模块。其中每个 cycle 可以发射 5 条指

令，如果遇到 vsetli 指令，则该 cycle 只发射 vsetli，其它指令等到下一 cycle 再发射，所以每个 cycle 只需要分配一个 vectag。

本实施例的 vpu 模块，vector 指令 datapath，指令从 reserver station（保留站）中发射到执行单元的一个重要的条件就是需要本 entry 的指令 vectag 能够和 ROB 广播出来的 vectag 一致，才能发射到执行单元。如图 3 所示将 unactive 发射到执行单元，需要 2*n cycle 完成，如果将 unactive 不发射到执行单元，只需要 n cycle 完成。可减少 latency 的同时，减少功耗。

本实施例的 ROB 模块，每条指令执行完后，都要顺序的毕业，在毕业时，同时更新寄存器 vectag。

vectag 分配更新 vectage 寄存器，vector 指令可以发射的条件的时间轴表如下

时间	cycle1	cycle2	cycle3	...	cycle_m		..	cycle_n
指令 1		vsetli	vec_instr0					
指令 2			vec_instr1					
分配 vectag 寄存器	n	n+1	n+1					
毕业指令					vsetli			vec_instr0 和 vec_instr

								1
更新 vectag					更新			
指令可以发 射						vec_i nstr 0 和 vec_i nstr 1 可 以发 射		

综上，本发明非 $vset1\{i\}$ Vector 指令只需根据比自己更老的 $vset1\{i\}$ 中最年轻的指令执行完才能进入执行单元，比起目前的刷新流水效率高多了。刷新流水是需要从新从取指开始的，而不用刷新只是在保留站里等，等到比自己更老的 $vset1\{i\}$ 中最年轻的指令执行完就能到执行单元中执行了。

本发明 Vector 指令对 unactive element 部分也在执行单元中执行了，最后再通过 mask 的方式来对数据进行选择，可以减少功耗，同时可以减少执行周期，减少 latency。

以上实施例仅用以说明本发明的技术方案，而非对其限制；尽管参照前述实施例对本发明进行了详细的说明，本领域的普通技术人员应当理解：其依然可以对前述各实施例所记载的技术方案进行修改，或者对其中部分技术特征进行等同替换；而这些修改或者替换，并不使相应技术方案的本质脱离本发明各实施例技术方案的精神和范围。

权 利 要 求 书

1、一种 risv_v vector 指令集 vsetli 指令的实现方法，其特征在于，所述方法包括以下步骤：

S1CPU 乱序执行时，在 rename 模块中分配 vectag[n:0]信息，判断指令是否为 vsetli；

S2若指令是 vsetli，则将 vectag+1，若是非 vsetli 指令，则 vectag 保持不变；

S3 发射到执行单元，对 vsetli 指令分发到 csr 模块；对应其它的 vector 指令分发到 vpu 模块；

S4 判断指令 vectag 与 ROB 广播的 vectag 一致时，指令从 reserver station 发射到执行单元；

S5 指令执行完成，在 ROB 模块中，按顺序毕业，并在毕业时更新寄存器 vectag，执行结束。

2、根据权利要求 1 所述的 risv_v vector 指令集 vsetli 指令的实现方法，其特征在于，所述方法中，每个 cycle 发射 0-5 条指令。

3、根据权利要求 2 所述的 risv_v vector 指令集 vsetli 指令的实现方法，其特征在于，所述方法中，若接受 vsetli 指令，则该 cycle 只发射 vsetli，每个 cycle 分配一个 vectag，其它指令等到下一 cycle 再发射。

4、根据权利要求 1 所述的 risv_v vector 指令集 vsetli 指令的实现方法，其特征在于，所述方法中，将 unactive 发射到执行单元，执行 2*n cycle 完成，将 unactive 不发射到执行单元，执行 n cycle 完成。

5、根据权利要求 1 所述的 risv_v vector 指令集 vsetli 指令的实现方法，其特征在于，所述方法中，vpu 模块 reserve station 中的指令 vectag 都需要和寄存器 vectag 比较，只有一致的情况下，该指令才能发射到执行单元。

6、根据权利要求 1 所述的 risv_v vector 指令集 vsetli 指令的实现方法，其特征在于，所述方法中，在 rename 中分配 vectag[n:0]，作为 vpu 指令发射到执行单元的一个条件，用以在执行 vsetli 指令时不刷新流水线。

7、一种 risv_v vector 指令集 vsetli 指令的实现系统，所述系统用于执行如权利要求 1-6 任一项所述的 risv_v vector 指令集 vsetli 指令的实现方法，其特征在于，包括 rename 模块、dispatch 模块、vpu 模块和 ROB 模块。