

IMPLEMENTATION METHOD AND SYSTEM OF RISV_V VECTOR

INSTRUCTION SET VSETLI INSTRUCTION

TECHNICAL FIELD OF THE INVENTION

The invention relates to the technical field of CPUs, in particular to a
5 method and system for implementing a *risv_v* vector instruction set *vsetli*
instruction.

BACKGROUND OF THE INVENTION

The *isc_v* instruction set has only recently published the complete
instruction set, but there is basically no implementation method available for
10 reference at present. In order to achieve simplicity, the simplest way is that the
vsetli instruction needs to be refreshed upon graduation, and the execution unit,
regardless of the unactive element part, is sent to the execution unit for
execution, resulting in an increase in the execution cycle.

The existing *vsetli* instructions need to be refreshed when they graduate,
15 resulting in low efficiency of CPU execution. The unactive element part of the
Vector instruction is also executed in the execution unit, and finally the data is
selected by the way of mask. In fact, the data of mask does not need to enter the
execution unit, which leads to power consumption and increases the execution
cycle of the instruction.

SUMMARY OF THE INVENTION

In view of the deficiency of the prior art, the invention discloses a method and a system for realizing *risv_v* vector instruction set *vsetli* instruction, which is used for solving the existing problems.

5 The invention is realized through the following technical proposal:

First, the invention discloses a method for realizing *risv_v* vector instruction set *vsetli* instruction, which comprises the following steps:

When the S1CPU is executed out of order, the *vectag* [n: 0] information is allocated in the rename module to determine whether the instruction is *vsetli*.

10 S2 if the instruction is *vsetli*, then *vectag*+1, if it is not *vsetli* instruction, then *vectag* remains unchanged.

S3 is transmitted to the execution unit, *vsetli* instructions are distributed to the csr module, and other vector instructions are distributed to the vpu module.

15 When S4 determines that the instruction *vectag* is consistent with the *vectag* broadcast by ROB, the instruction is transmitted from reserver station to the execution unit.

The execution of S5 instruction is completed, in the ROB module, graduate in order, and update the register *vectag* when graduation, the execution ends.

Further, in the method, each cycle emits 0-5 instructions.

5 Further, in the method, if the *vsetli* instruction is accepted, the cycle only transmits *vsetli*, each cycle allocates one *vectag*, and other instructions are not transmitted until the next cycle.

Further, in the method, the unactive is transmitted to the execution unit, 2cm cycle completion is performed, the unactive is not transmitted to the
10 execution unit, and n cycle completion is executed.

Further, in the method, the instruction *vectag* in the vpu module reserve station needs to be compared with the register *vectag*, and only if the instruction is consistent can the instruction be transmitted to the execution unit.

Further, in the method, the *vectag* [Nvpu 0] is allocated in the rename as
15 a condition for the vpu instruction to be transmitted to the execution unit without refreshing the pipeline when the *vsetli* instruction is executed.

In the second aspect, the invention discloses a system for realizing *risv_v* vector instruction set *vsetli* instruction. The system is used for executing the realization method of *risv_v* vector instruction set *vsetli* instruction described in

the first aspect, which comprises rename module, dispatch module, vpu module and ROB module.

The beneficial effects of the invention are:

5 The non-*vsetl {i}* Vector instruction of the invention only needs to be executed according to the youngest instruction in the older *vsetl {i}* before entering the execution unit, which is much higher than the current refresh pipeline efficiency.

10 The Vector instruction of the invention also executes the unactive element part in the execution unit, and finally selects the data by the way of mask, which can reduce the power consumption, at the same time reduce the execution cycle and reduce the latency.

BRIEF DESCRIPTION OF THE DRAWINGS

15 In order to more clearly illustrate the technical scheme in the embodiment of the invention or the prior art, the following will briefly introduce the drawings that need to be used in the embodiment or the prior art description, obviously, the drawings described below are only some embodiments of the invention, and for ordinary technicians in the art, other drawings can be obtained according to these drawings without creative work.

Fig. 1 is a principle step diagram of a method of implementing *vsetli* instructions in *risv_v* vector instruction set.

Fig. 2 is a basic block diagram of the out-of-order CPU of the embodiment of the present invention.

5 Fig. 3 is a unactive transmission comparison block diagram of an embodiment of the present invention.

DETAILED DESCRIPTION OF THE EMBODIMENTS

In order to make the purpose, technical scheme and advantages of the embodiment of the invention more clear, the technical scheme in the
10 embodiment of the invention will be described clearly and completely in combination with the drawings in the embodiment of the invention. Obviously, the described embodiments are some embodiments of the invention, not all embodiments. Based on the embodiments of the invention, all other embodiments obtained by ordinary technicians in the field without creative
15 work fall within the scope of the protection of the invention.

Embodiment 1

The present embodiment discloses a method for implementing *risv_v* vector instruction set *vsetli* instructions as shown in fig. 1, which includes the following steps:

When the S1CPU is executed out of order, the *vectag*[*n*:0] information is allocated in the rename module to determine whether the instruction is *vsetli*.

S2 if the instruction is *vsetli*, then *vectag*+1, if it is not *vsetli* instruction, then *vectag* remains unchanged.

5 S3 is transmitted to the execution unit, *vsetli* instructions are distributed to the csr module, and other vector instructions are distributed to the vpu module.

When S4 determines that the instruction *vectag* is consistent with the *vectag* broadcast by ROB, the instruction is transmitted from reserver station to the execution unit.

10 The execution of S5 instruction is completed, in the ROB module, graduate in order, and update the register *vectag* when graduation, the execution ends.

In the present embodiment, each cycle emits 0-5 instructions. If the *vsetli* instruction is accepted, the cycle only transmits *vsetli*, each cycle allocates one *vectag*, and the other instructions are not sent until the next cycle.

15 In the present embodiment, the unactive is transmitted to the execution unit, the execution of 2n cycle is completed, the unactive is not transmitted to the execution unit, and the execution of n cycle is completed.

In the present embodiment, the instruction *vectag* in the vpu module reserve station needs to be compared with the register *vectag*, and only if the
20 instruction is consistent can the instruction be transmitted to the execution unit.

In the present embodiment, the *vectag* [*n:0*] is allocated in the rename as a condition for the vpu instruction to be transmitted to the execution unit without refreshing the pipeline when the *vsetli* instruction is executed.

The *vsetli* instruction of the present embodiment does not need to refresh
 5 the pipeline when graduating, and the unactive element part does not need to be transmitted to the execution unit for execution, which can reduce power consumption and execution cycle.

Embodiment 2

The embodiment refers to the out-of-order CPU, and its basic frame is
 10 shown in figure 2. The embodiment discloses a system for realizing *risv_v* vector instruction set *vsetli* instructions, which includes four modules: rename, dispatch, ROB and vpu.

The rename module of the present embodiment allocates a *vectag* [*vsetli* 0] information in the rename module, and if it is a *vsetli*, the *vectag* of the *vectag*+1,
 15 non-*vsetli* instruction remains unchanged, so that the instruction executed by the vpu unit can be transmitted to the execution unit only if the *vectag* of the instruction in the reserve station is consistent with the *vectage* broadcast by the csr.

The function of the dispatch module of the embodiment is to distribute the
 20 instruction to different datapath according to the type of instruction, corresponding to the *vsetli* instruction to the csr module, and to the other vector

instruction to the vpu module. Each cycle can send five instructions. If the *vsetli* instruction is encountered, the cycle only launches the *vsetli*, and the other instructions wait until the next cycle, so each cycle only needs to allocate one *vectag*.

5 The vpu module of the present embodiment, the vector instruction datapath, an important condition for the instruction to be transmitted from the reserver station (reservation station) to the execution unit is that the instruction *vectag* of the entry is required to be consistent with the *vectag* broadcast by the ROB before it can be transmitted to the execution unit. As shown in figure 3,

10 transmitting unactive to the execution unit requires 2n cycle to complete, and if the unactive is not transmitted to the execution unit, only n cycle is needed to complete. It can reduce latency and power consumption at the same time.

 In the ROB module of the present embodiment, after each instruction is executed, it is necessary to graduate sequentially and update the register *vectag*

15 at the same time.

Vectag allocates the update *vectage* register, and the timeline table of the conditions under which the vector instruction can be issued is as follows:

Time	cycle1	cycle2	cycle3	...	cycle_m		..	cycle_n
Instruction 1		<i>vsetli</i>	vec_instr0					
Instruction 2			vec_instr1					
Allocate the <i>vectag</i> register	n	n+1	n+1					
Graduation Instruction					<i>vsetli</i>			vec_instr0 and vec_instr1
update <i>vectag</i>					Update			

command can be issued						vec_instr0 and vec_instr1 can emit		
--------------------------	--	--	--	--	--	---	--	--

In summary, the non-*vsetl {i}* Vector instruction of the invention only needs to be executed according to the youngest instruction in the older *vsetl {i}* before entering the execution unit, which is much more efficient than the current refresh pipeline. Refreshing the pipeline needs to start with a fresh
5 finger fetch, instead of just waiting in the reservation station until the youngest instruction in the older *vsetl {i}* has been executed.

The Vector instruction of the invention also executes the unactive element part in the execution unit, and finally selects the data by the way of mask, which can reduce the power consumption, at the same time reduce the
10 execution cycle and reduce the latency.

The above embodiments are only used to illustrate the technical scheme of the invention, not to limit it; although the invention is described in detail with reference to the aforementioned embodiments, ordinary technicians in the field should understand that they can still modify the technical scheme recorded in
15 the above-mentioned embodiments, or equivalent replacement of some of the technical features. These modifications or replacements do not deviate the essence of the corresponding technical scheme from the spirit and scope of the technical scheme of the embodiments of the present invention.

CLAIMS

1. A method for realizing *risv_v* vector instruction set *vsetli* instruction, which is characterized in that the method comprises the following steps:
 - 5 when the S1CPU is executed out of order, the *vectag* [n: 0] information is allocated in the rename module to determine whether the instruction is *vsetli*;
 - S2 if the instruction is *vsetli*, then *vectag*+1, if it is not *vsetli* instruction, then *vectag* remains unchanged;
 - 10 S3 is transmitted to the execution unit, *vsetli* instructions are distributed to the csr module, and other vector instructions are distributed to the vpu module;
 - when S4 determines that the instruction *vectag* is consistent with the *vectag* broadcast by ROB, the instruction is transmitted from reserver
 - 15 station to the execution unit;
 - the execution of S5 instruction is completed, in the ROB module, graduate in order, and update the register *vectag* when graduation, the execution ends.

2. A method for realizing *risv_v* vector instruction set *vsetli* instructions according to claim 1, wherein each cycle emits 0-5 instructions in the method.
3. The realization method of the *risv_v* vector instruction set *vsetli* instruction according to claim 2, wherein, if the *vsetli* instruction is accepted in the method, the cycle only transmits the *vsetli*, each cycle allocates one *vectag*, and the other instructions are not transmitted until the next cycle.
4. The method for realizing the *risv_v* vector instruction set *vsetli* instruction according to claim 1, wherein the unactive is transmitted to the execution unit, the execution of 2n cycle is completed, the unactive is not transmitted to the execution unit, and the execution of n cycle is completed.
5. The realization method of the *risv_v* vector instruction set *vsetli* instruction according to claim 1, wherein the instruction *vectag* in the vpu module reserve station needs to be compared with the register *vectag*, and only if the instruction is consistent, the instruction can be transmitted to the execution unit.
6. The implementation method of the *risv_v* vector instruction set *vsetli* instruction according to claim 1, wherein the *vectag* [*vectag* 0] is allocated in the rename as a condition for the vpu instruction to be

transmitted to the execution unit, so that the pipeline is not refreshed when the *vsetli* instruction is executed.

7. A system for realizing *risv_v* vector instruction set *vsetli* instruction, the system is used for implementing *risv_v* vector instruction set *vsetli* instruction described in any of claims 1-6, wherein it includes rename module, dispatch module, vpu module and ROB module.