recent update at opensbi

Xiang W
wxjstz@126.com
wangxiang@iscas.ac.cn

Intro

- Support for discrete hart id
- Fix bug of inter-processor interrupt
- Introduction to SSE

Support for discrete hartid

- Commits: [1] [2] [3] [4] [5] [6] [7] [8]
- original issue
  - A scratch is a fixed-length array indexed by the hartid, and since hartid can be discrete, the scratch array may be underutilized
  - sbi_hartmask is used to record a set of harts, it is a bitmap where the bits identify the id of the hart, because the hartid is discrete so the bitmap may be underutilized.

Support for discrete hartid

Improvement:

- Building a table that looks up the hart id by hart index
- Modify the index of the scratch array to hart index
- The bit of sbi_hartmask is modified to correspond to the hart index

Support for discrete hartid

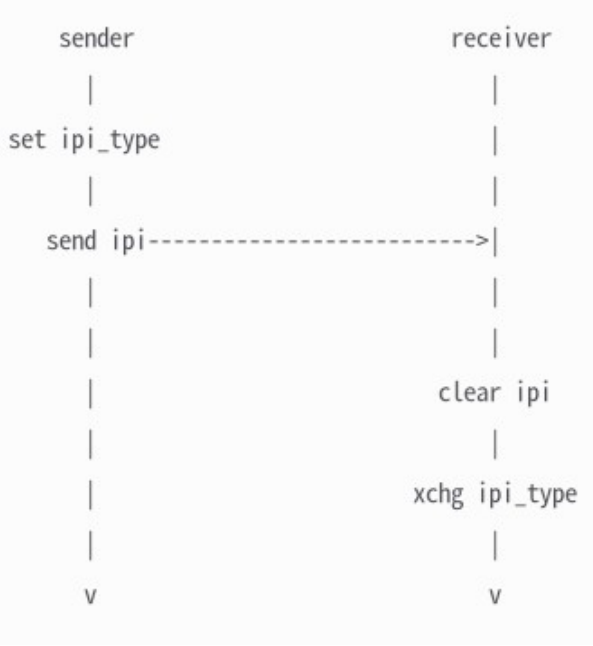
Note: Since the above lookup table is constructed in sbi_scratch_init, some operations cannot be accessed until sbi_scratch_init is called. as follows:

```
sbi_hartid_to_hartindex
sbi_hartindex_to_hartid
sbi_hartindex_to_scratch
sbi_hartid_to_scratch
sbi_hartid_valid
sbi_scratch_last_hartindex
```

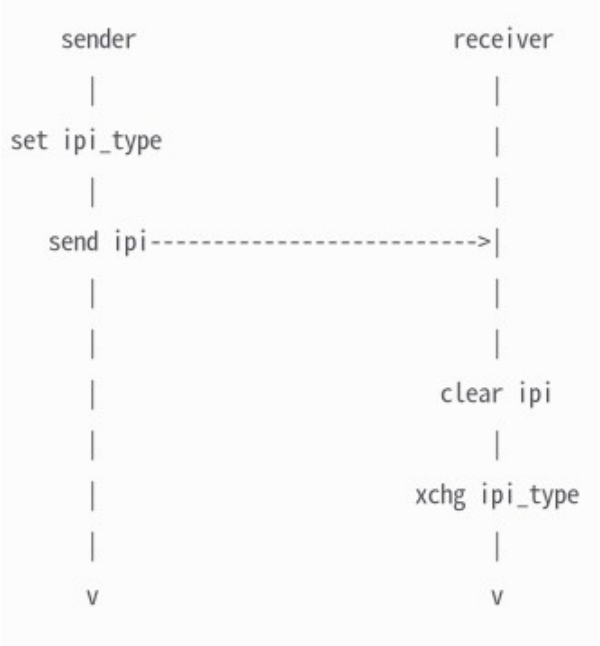
```
SBI_HARTMASK_INIT_EXCEPT
sbi_hartmask_set_hartid
sbi_hartmask_clear_hartid
sbi_hartmask_test_hartid
...
```

Fix bug of inter-processor interrupt

- There are two serious errors about ipi
  - Lost ipi event
  - Going into exception handling without ipi event
- Before understanding, need to know how ipi event are sent and received. Refer to the chart on the right

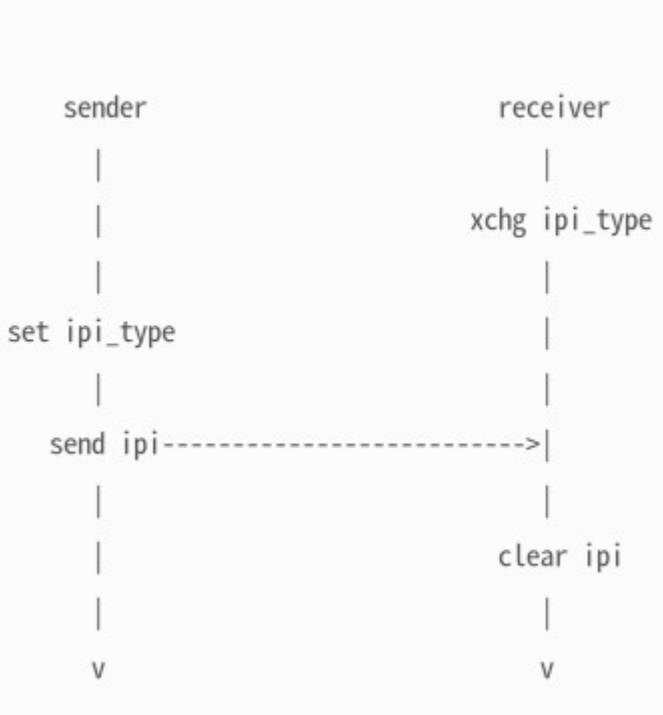


Fix bug of inter-processor interrupt



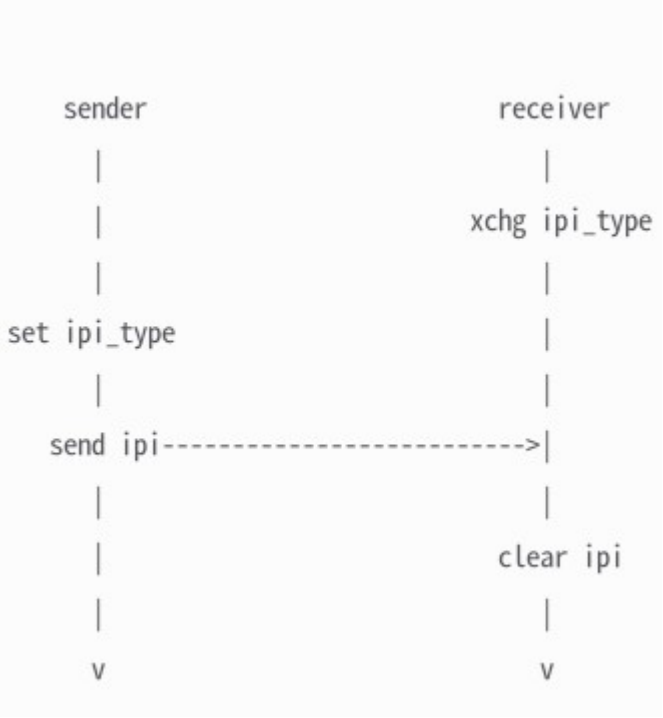
- **ipi_type** is a bitmap of the ipi event received by the receiver.
- **set** is an atomic operation to set bit in ipi_type.
- **xchg** is an atomic operation used to read and clear ipi_type.
- **send/clear ipi** are io operations, implemented via writel. There is a memory barrier instruction(fence w,o) before write.

Fix bug of inter-processor interrupt: first case



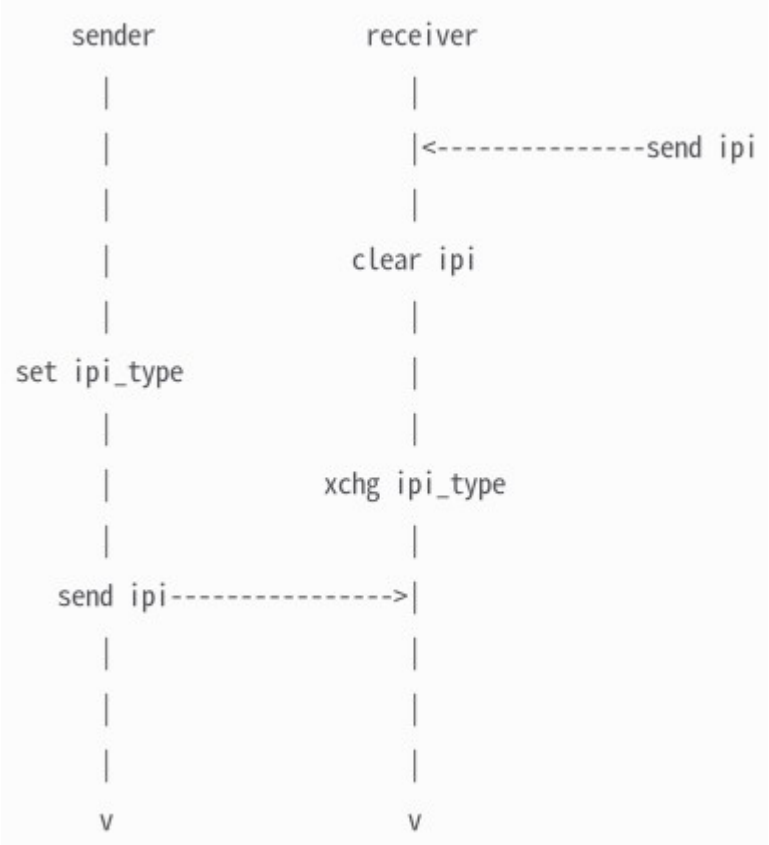
- Commits: [1] [2]
- The memory order of atomic operations is only valid for memory or io
- Because fence w,o does not guarantee that subsequent memory writes occur before the memory barrier. So, a case such as the left picture may occur.

Fix bug of inter-processor interrupt: first case



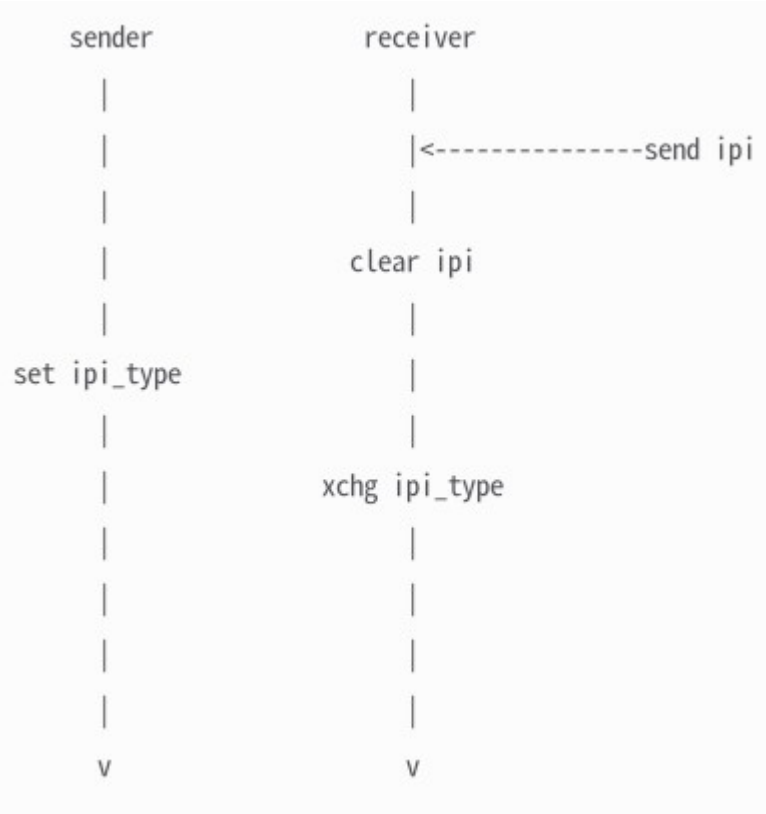
- Fix:
  - Add memory barrier(fence ow,ow) before send ipi
  - Add memory barrier(fence ow,ow) after clear ipi
  - In order to improve efficiency, the mmio operation in the ipi driver has been modified and the memory barrier has been removed(use writel_relaxed replace writel).

Fix bug of inter-processor interrupt: second case



- Commits: [1]
- The second case is caused by sending an ipi interrupt every time an ipi event is sent, as shown on the left

Fix bug of inter-processor interrupt: second case



- Fix:
  - Reading the old value when setting bit of ipi_type and sends an ipi interrupt only if the original ipi_type value is 0.
  - The modified effect is shown on the left

Introduction to SSE

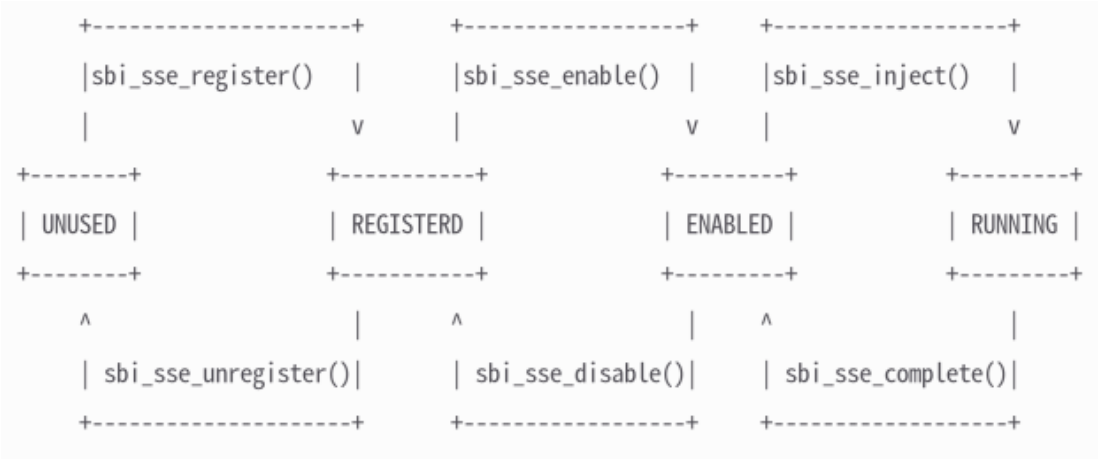
- Commits: [1] [2]
- SSE(Supervisor Software Events) extension provides a mechanism to send a SSE event to interrupt supervisor execution and go to the event's handler.
- SSE event is identified by an integer.
- SSE event has two type: global and local. A global event can be handled by specified hart, while a local event can only be handled by the current hart.
- SSE event has priority

Introduction to SSE

- SSE event has the following properties:
  - Event id
  - State
  - Priority
  - Associated hart id
  - Context of event handler

Introduction to SSE

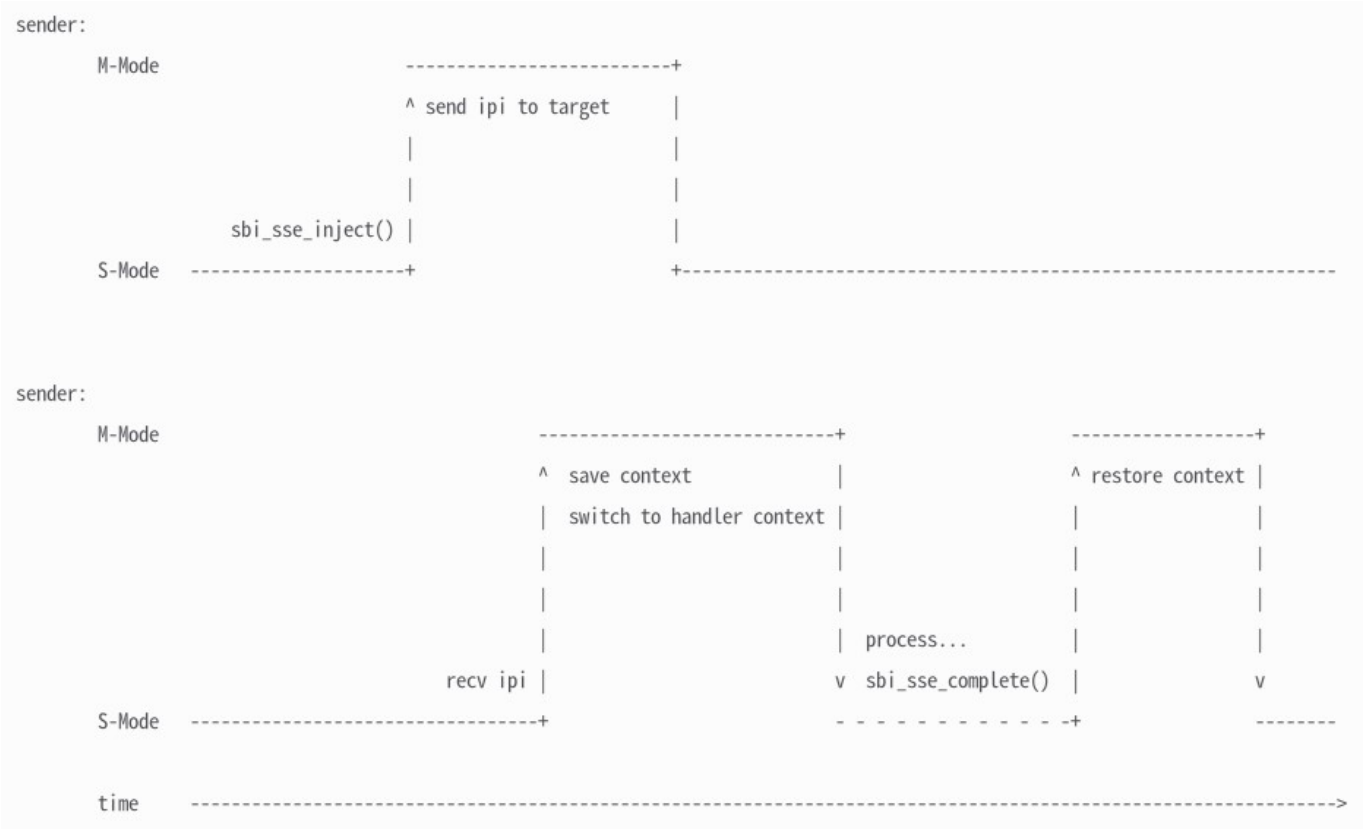
- The status of SSE is as follows picture



- sbi_sse_register is used to set the context of the event handler
- Priority can be modify under UNUSED/REGISTED
- The event can be triggered after it is enabled, and the triggered event enters the running state.
- After running, the event handler calls sbi_sse_complete to release hart.

Introduction to SSE

- The processing process is shown in the figure



Reference

- opensbi source code
- SBI Supervisor Software Events Extension

Thanks!
Q&A