Date: 2021/01/19
Task Group: Fast Interrupts
Chair: Krste Asanovic
Co-Chair: Kevin Chen
Number of Attendees: ~10
Current issues on github: https://github.com/riscv/riscv-fast-interrupt/issues
Previous meeting minutes:
https://github.com/riscv/riscv-fast-interrupt/tree/master/minutes

Issue discussed:

#106 Enhance support co-routines and thus allow level change already saved registers.

Per request from our previous meeting, the author David sent out a detailed pdf document and diagram (can be found in the attachments in issue #106). He also presented and explained his proposal in details. His proposal basically describes how to add HW support to track updated registers in order to minimize unnecessary register saving/restoring for co-routines and interrupts.

Some member pointed out that the worst case scenario won't get improved with this approach. This is because, even though the "window" to avoid repeated saving/restoring of interrupt context has been extended, the worst case will occur when an interrupt arrives right after this extended window.

Also, some member was concerned about the extra HW cost. Alternatively, a pure soft approach may achieve similar goal by using compiler to reserve a fixed group of registers for interrupts or only save registers that are actually used.

Lastly, the fastest approach should be using pure HW-assisted context saving/restoring, and it is not clear how this proposed scheme can work with pure HW approach. Also, we may need to examine if there are any potential security issues in the usage model.

#31 Does Interrupt Level Affect WFI?

This issued was re-opened because of the following problem:

In spec we described: In CLIC, similar to original mode, Wait for Interrupt instruction (WFI) is unaffected by the global interrupt-enable bits in mstatus.xie but should honor clicintie[i] and xintthresh. Also, the threshold is valid only for the current privilege mode (while ignored for other privilege
modes).

However, this definition creates counter-intuitive cases. For example:

- When WFI occurs in M-mode, a pending M-mode interrupt level = 3, mintthresh = 4, WFI won't wake up (because it is masked by m-mode threshold)

In contrast,

- When WFI occurs in M-mode, if there is a U-mode interrupt with similar level = 3, and uintthresh = 4, WFI will wake up in this case (because threshold is only valid for the current privilege mode (M-mode) and ignored for U-mode interrupts)

This seems really strange because a lower privilege interrupt can wake WFI but a current higher privilege mode interrupt cannot.

Krste thought this counter-intuitive behavior was not a bug and hence acceptable. So we would not change the definition but only explain in the spec.

Regarding whether WFI should wake up when level == threshold, we concluded that we would define it in a way such that hardware cost is minimized. Therefore, we would re-use the existing circuit for normal usage, which is detecting preemption when level > threshold. Similarly, we would change the definition to allow WFI wake up when an interrupt level > threshold (but not for == case).

Lastly, we would also make a note in the spec to remind users that WFI should not be placed inside an ISR.