NOTE: updated zoom info for next meeting and beyond.

Meeting link: https://zoom.us/j/95666422067

Passcode: 174425

Join link: https://zoom.us/j/95666422067?pwd=aHM1ZEJDcFJ3cGE0UWdTRmtaUXhsQT09

Date: 2023/08/01

Task Group: Fast Interrupts

Chair: Dan Smathers

Co-Chair: Jean-Baptiste Brelot

Number of Attendees: 6

Meetings Disclaimers Video :
https://drive.google.com/file/d/1y_XWJus8M5ZwSQ2cvEOzCjlOmsmXOnN4/view

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

Fast Interrupt DoD (Definition of Done) Status:
https://wiki.riscv.org/display/HOME/Fast+Interrupts+TG

Next meeting agenda (08/15/23)

Issue #314/pull #342 – shv==0 requirement for xnxti. more discussion needed.

Issue #308/pull #325/pull #343– xnxti service loop – fix pull #343 based on discussion.

Issue #307/pull #331 – Parameter cleanup (compare with pull #297?)

Issue #303 – How are CLIC-only CSRs and fields handled in CLINT mode

Meeting minutes:

Accepted pull #341, closed #333. big change to spec with xinhv. xinhv now indicates if xepc is address of instruction or address of table.

Discussed pull #342 – doubling size of xtvt table. has conflicts. is this what we want? implies 2 sets of code. handlers without save restore. handlers with save restore. doesn't totally double code of handlers. Is this the best way to solve this? tricky thing is full routine has a bunch of save registers. restores are later. if every handler has sting of saves, string of loads. in sw version, do that once. code indirect jumps to body of code. with hw vectoring, if want to do both. hw vectoring jumps to different place that does save, jumps to common code. has to remember it needs to restore. one way of doing that (worry about static code size of handlers). version with save restore. version that saves, jump and link to core routine, then jumps back to retore. the other mode we had, hw interrupt could use interrupt attribute model. pay as you go.

that was vision before. hw would be interrupt attribute. sw would do full abi and go thru that trampoline. if you want to have the save abi.

There are 3 versions of the handler. interrupt attribute version. Inline saves as it needs. can be invoked by hw.

Sw vectoring code. code doesn't have to save restore because factored down to common code (trampoline).

3rd case – replicate. so hw vectored but need to save and restore like the sw one did. can do that with code. instead of doubling table size, we could have a state bit and a single hw entry point separate from the sw entry point. if in service loop, just jump to core using table for sw vectoring. if outside and get hw interrupt, go to different place that does core and then returns back. so common interrupt handler has to know what the table entry is going to jump to. have to make hw vectoring come into this save/restore code, when haven't done save/restore.

The idea is there is only a sinlge static code body and somehow have mechanism, if have hw vectored outside of this. so only need one entry in the table if we can engineer this. if in service loop, don't do save restore. otherwise do save restore. like set a bit in csr and test it. it assumes all hw are using the same abi model. so don't need twice the table entries and don't need twice the static code. will investigate.

Discussed Issue #308/pull #325/pull #343:

Discussing in pull whether to write to csrrw form of mnxti, would we always want to set it so OR. rw is a little funky because it writes a value based on setting an interrupt enable. so always OR in the interrupt enable. so show csrrs in pseudo code and don't update mcause.pil. so could be csrrs or csrrc. So update pull #343 and try again next time.

For pull #331 – parameter clean up. separate into different PRs.

Github updates since last minutes:

Specification updates since last minutes:

Open issue status:

Issues that can be closed?

#307/pull #331 – NUM_INTERRUPT listed in two sections (text cleanup)

Need spec updates:

#158 – change CLICCTRLBITS to CLICMLPBITS. (resolve #226 first?)

#171 – CLICCFGLBITS parameter - related to #80, #158. (resolve #226 first)

Pull #286 – text update fix for #96/158/171/226/49 - allow multiple implementations for priv mode/level.

Need more discussion:

#102 - preemptible interrupt handler code (for section 7.2)

#226 – replace clicintattr.mode with xcliccfg.xlvl proposal (formerly #96)

#303 - How are CLIC-only CSRs and fields handled in CLINT mode?

#308/pull #325 – xnxti service loop behavior

#314 – shv==0 requirement for xnxti

Issues need to be worked:

#91 – DTS entry – have linux group review DTS example. If add CLINT compatibility option, just use same DTS entry as clint?

#107 - heritage of features. keep researching and adding references to bibliography.

#185 – SAIL model implementation of CLIC

#186 – CLIC architecture tests

#187 – QEMU CLIC implementation update

#221 – compressed clicintip/clicintie status registers (1-bit per interrupt) similar to AIA eip0-eip63? Add justification and create a pull with a proposal. Pull #270. add prior art reference.

#242 – SPIKE model implementation of CLIC

Issues punted for rev1, keep open for future enhancements:

#92/Pull #281 – hypervisor compatibility. still punted for rev1?

#99 – horizontal interrupt window. punted for rev1

#101 - xnxti to trigger on equal level. punted for rev1

#106 – allow level change. Punted for rev1

#108 – pushint/popint? defer to broader discussion on providing hardware stacking of interrupt contexts. Punted for rev1

#192 – allow mix of CLIC/CLINT at different priv modes punted for rev1

#248 – CLIC hypervisor mode (related to #92).

#**329 -** support for hw register save.