Date: 2020/7/7
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/blob/master/clic.adoc

Issues discussed:

#33, #85, #88 Why are read-only xintstatus CSRs assigned in the read-write address range?

These CSRs are temporarily allocated in the read-write address range because we may have more read-write fields in the future. We may change this before we ratify the spec.

Also, as an insightful member pointed out in #85:
“at some point in booting an OS the current interrupt levels will need to be set to 0 (so that interrupts can be serviced) as they are currently undefined at reset. The only current way to do this is to write xcause.pil and then execute an xRET instruction - it's particularly difficult for m-mode to do this for say u-mode.

I think it would be useful to make xintstatus writeable - it makes OS setup easier (along with writing test cases)”

#51 Add xscratchcsw/xscratchcswl read and write pseudocode

As pointed out in the discussion thread “Behavior of xscratchcsw and xscratchcswl”:
If we follow the same spirit of “conditional swap”, read access should become a conditional read:

```
csrrs rd, mscratchcsw, x0

// Pseudocode operation.
if (mcause.mpp!=M-mode) then {
    rd = mscratch;
} else {
    // rd unchanged
}
```

However, a senior member indicated that keeping the value unchanged (in false clause) will require an extra GPR read-port in an out-of-order CPU. This is quite expensive so another member suggested to always write the destination with x0 (i.e., rs1/source GPR). Nevertheless, for set/clear CSR instructions (csrrs/csrrc), directly copying the source GPR to the destination GPR will require an complete new datapath/MUX which did not exist before. This extra datapath and circuit may slow down the CPU max frequency.

In practice, only the read-modify-write (swap/CSRRW) operation is useful for xscratchcsw/xscratchcswl. In contrast, the “pure read” and “pure write” operations are meaningless and not useful at all. Therefore, we could just possibly simply declare the non CSRRW variants as "reserved". In the meantime, we will still try to write out all the behaviors of the CSR variants for further discussion.

#90 clicint* reserved and custom bits and software use of inactive entries

“ip” is not frequently read by software - the main function is to cause a hardware interrupt. It will be more often written, either by hardware or software.

Because ip byte is written by hardware and software, we want to avoid packing bits unrelated to signaling an interrupt in that byte. any additional bits should be related to the pended interrupt. The text should be updated to indicate that ip=zero means no interrupt pending, and ip!=0 (not just 1) indicates an interrupt is pending.

A minor point, but writing a 1 is easier in RISC-V software than writing 0x80, as 16b instruction can create a 1 but need 32b instruction to create 0x80.

#47 CLIC reset behavior

The policy remains unchanged: General model in RISC-V has been mandatory reset state is minimized, platforms or company policy might add additional reset requirements.

Will add more clarification in the spec.

#50 xcause save privilege modification bits

Since the hypervisor spec has been updated significantly after the creation of this issue, the author would like to take more time to re-evaluate the original proposal.

#89 Proposal: reorder lower 16 interrupt IDs

The author suggested to remove the backward-compatible layout of the lower 16 interrupt IDs. There are multiple options with different pros and cons. As this topic is big and we ran out of time, we decided to further discuss and get more feedback in our workgroup discussion.