Date: 2020/10/06
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:

#105 Is mcause inhv writeable?

This bit is writable, so the value will stay if firmware writes to it. During the normal operation, it is set and cleared by the HW state machine for vectored mode interrupts. It is not updated for exceptions. xret does not clear it (to maintain backward compatibility). When this bit is set, it enters exception but other parts of xcause does not get updated/cleared.

Proposal: Need to determine which of the following options works the best. If HW fetch for the vector table is not successful, the machine will execute the SW exception routine with mcause reflecting one of the following:

mcause.interrupt = 1, exception code = vector number, inhv = 1
mcause.interrupt = 0, exception code = some existing fault, inhv = 1
mcause.interrupt = 0, exception code = a new type (group) of interrupt fetch faults, inhv = N/A (this information is already encoded in the exception code)

More questions and thoughts:

Is this type of fault even recoverable at all?

As long as SW does not clear this interrupt, the same interrupt will trigger again after the fault is handled/resolved by SW.

Can we ignore the vector lookup failure and fall back to pure SW ISR? Can we fall back to pure SW mode if HW vector mode fails? Do we need 2 versions of ISR in

this case? Will this mode change create any corner cases when there are nesting interrupts?

Without vector number (in option 2 and 3), it may be difficult to figure out which part of the vector table went wrong as the vector table can be quite large (if we have 4K interrupts). Can we capture the vector number info in mtval?

Will check with Krste about his original intention and thoughts.

Need to clarify the details in the spec.