Jan 26, 2023 | RISC-V Control Transfer History TG Meeting

Attendees: tech.meetings@riscv.org Beeman Strong Bruce Ableidinger

Notes

- **Attendees**: Beeman, Mark, Bruce, JohnS, Stas, Robert, Anup
- **Slides/video** here
- Aiming for TSC approval of the TG in ~2 weeks, once we have permanent chairs
    - **If you are interested in applying to be a permanent chair, see here**
- Continuing requirements discussion
- Freeze
    - Recap: ability for HW to stop recording transfers, to preserve history leading up to sample point
    - May want to freeze on counter overflow, or on LCOFI, depending on implementation. Could support separate bits.
    - Mark: James Ball working on multi-level M-mode, for debug & trace. Talk to him, may have a TG shortly.
    - Robert: believe freeze should be required
    - Bruce: will want to "freeze" when entering debug mode, so need the HW anyway
    - Stas: would freeze support an event on CTR overflow, to get interrupt and support complete trace via CTR?
        - Slow, but could be poor-man's trace if no trace available
        - Not related to freeze, but can discuss
    - Bruce: may want freeze on breakpoint exception too
        - Beeman: LBRs always freeze on breakpoint, FWIW
    - **Agreed to require some freeze, but implementations can choose which one(s)**
- Virtualization support
    - Do we want to require that CTRs can be directly used in VS, without trap & emulate?
    - Counters require trap & emulate, aim to allow direct access for CTR
    - How does Intel do it?
        - Intel has different approach to virtualization vs RISC-V, switch LBRs on/off on transitions between host & guest. Priv mode (CPL) filtering doesn't factor in virtualization.
    - Bruce: how would context filtering work?
        - Beeman: good question. If VM selects as ASID/VMID value, need to think about how to virtualize that. Could require trapping.
    - **Will aim to add ISA to support VM access to CTR without traps**
- Mark: If we come up with something novel, need to share on a public list. Not a RV members list. So there is a record of it, for prior art.
- Tolerance for slowdown

    - High-performance CPUs can retire >1 transfer per cycle. Cheaper CTR implementations may slow down retire to support fewer transfers per cycle => fewer write ports to CTR array.
    - Do we want to require no slowdown? Nice for tools/users, hard on implementations
    - Robert: vote for no slowdown
    - Bruce: such a requirement may cause someone not to implement it
    - Beeman: tools/users can mitigate CTR slowdown using sample rate, to limit total slowdown
    - **Will recommend no slowdown but not require it, and include some indication of whether an implementation has slowdown (config struct?)**
- **Out of time, will continue next week**

Action items

- [ ]