Apr 25, 2024 | RISC-V Performance Events TG

Attendees: Beeman Strong Dmitriy Ryabtsev tech.meetings@riscv.org

Notes

- **Attendees**: Dmitriy, Beeman, Guillem, WilliamM, ChunL, DaiH, ShashankN, Ved
- **Slides/video** here (sorry, started video late)
- Continue groups discussion
- Separated groups by RETIRED (non-spec) vs SPEC
- Looking at CACHE_RETIRED vs CACHE_SPEC
    - An implementation might implement only 3 of ACCESS, HIT, MISS, MERGE, such that the unimplemented event would be a formula
    - Metrics are currently kept in separate files, and are ratios (e.g., MPKI)
    - Believe that file structure doesn't really matter, tools will consume all events
        - Beeman Strong to check with Atish Kumar Patra
        - Update: confirmed that file structure doesn't matter
    - Could have events, formulas, and metrics mixed together
- *RETIRED counts should match across implementations, but *SPEC counts might not
    - TG will decide which if any events will be required. Some events may not apply for some implementations.
- CACHE events now apply to all caches, but can break them down by cache
- Typo: TOPDON events
- What is a DIRECT_JUMP.MISPRED?
    - Since direct jumps have a fixed direction and fixed target
    - FE could misidentify jump/branch as a non-control-transfer inst, which will clear
        - This could also happen on non-control-transfers misidentified as jumps/branches…
    - But such clears likely have a lesser recovery latency, should probably be a separate event
    - Keep MISPRED as the classic case of predicting the wrong direction/target
- Started recording here
- For RECOVERY_CYCLES, need to decide when recovery ends
    - Could be much longer than the minimum recovery latency for, e.g., a mispredict, since could miss I$, ITLB, …
    - Means we could have overlapping events, e.g. Icache miss & clear recovery
    - For top-down, treat such cases as BAD_SPEC rather than FE_BOUND, since misses on recovery path will be completely exposed, while other misses may be partially or completely hidden
    - Since this event is used in top-down formulas, should count until clear's target inst is delivered from FE to BE
- Tma_bad_spec is listed as metric, but some implementations could implement it as an event

    - Less likely, but possible. Should leave the option anyway, while suggesting the formula as an alternative
    - SPEC.INST_ISSUED should be uops, since counting slots
    - Spec will need some text clarifying inst vs uop, since only some implementations will differentiate between the two
- Perf modelling SIG is discussing fusion/fracture, can interfere with counts here
    - So each slot can hold <1 inst (uop), 1 inst, or >1 inst (fused insts)
    - Maybe better to refer to SLOTS, rather than UOPs (or INSTs, for top-down)
- Show pipeline_width as a metric, with integer value. Is that how other implementations do it?
    - Many just hardcode it in the formulas, like this better
    - Don't we want it to be FE-to-BE width (issue/dispatch) vs retire width?
        - It says max retire "sustained", which is limited to the narrowest pipeline width. So should be issue width.
- What is ScaleUnit? Copied from perf but not sure
    - Beeman Strong to check
- How to handle SMT?
    - Need separate counters per HW thread, which count events only for the associated HW thread
        - There are security risks for any events that count across threads, if we plan to have any they need to be able to be disabled
    - How to count cycles?
        - Only count cycles per-thread based on which thread is retiring?
        - Intel counts all cycles on both threads
- Out of time, will continue offline and next time
- May want to collaborate with perf modelling SIG, for POC

Action items

- [ ] Beeman Strong check on how perf uses ScaleUnit