Aug 29, 2024 | RV Performance Event Sampling TG

Attendees: Beeman Strong Snehasish Kumar tech.meetings@riscv.org

Notes

- **Attendees**: Beeman, Bruce, Punit, Snehasish, RobertC, Atish, Chun
- **Slides/video** here
- Updates:
  - TG approval vote by TSC ends tonight, don't expect any issue
  - Next is ratification plan at tech-chairs
- Instr sampling interface (cont'd)
  - CSR approach is less complex, but more overhead, and requires more CSRs context switched
  - Memory approach is more complex, but less overhead for usages that don't need call-stack, and fewer CSRs to switch
  - To collection SW the two are more or less equivalent, can read from memory or CSR
  - What about sending instr sample data to trace?
    - As discussed, may be valuable but don't want base capability to depend on availability of trace
    - Will leave integration into trace to future TG led by trace experts
    - But do expect that output to memory would share the ISA (and HW?) with self-hosted trace
      - So HW complexity is really shared with self-hosted trace
  - Why didn't we do CTR to memory?
    - CTR updates on every control transfer, which is every few insts. Would be too expensive to store so frequently.
    - Instr sampling would only store a sample every 1ms, or maybe 1us, but much less often. So overhead of stores is much less.
  - What about an internal register buffer, ala CTR?
    - So can store multiple samples internally, rather than just one?
    - Est sample data for each sample will use 4-6 CSRs
    - Latest LBR/BRBE implementations are 32-deep, 3 64-bit MSRs each. Assuming similar buffer size, could fit 16-24 samples before needing an interrupt
    - But no overhead per sample, unlike memory
    - How does perf overhead of this compare to memory buffer?
    - SW is going to store the record to memory anyway, so arguably the memory overhead is the same
      - Though HW store to memory likely involves pipeline flush per sample
  - Usage options: Interrupt every sample (CSR), vs every 16/24 (CSR buffer), vs every ~1000 (memory)

        - Google usages today: interrupt every sample (for call-stack) or every 100s or 1000s of samples
        - Don't see where CSR buffer fits in
    - Similar option to CSR buffer is an internal memory buffer that is only flushed when full, or on demand. Just means HW stores to memory instead of SW.
    - Memory allows future expansion
        - PEBS includes options to include GPRs, LBRs, etc
            - MikeC's talk last week referred to using PEBS GPRs to get memcpy function args, to know that each call was only 4B. Replaced with moves.
        - If we ever want the option to add such state components, output to memory is the only practical option
            - Would require a larger CSR buffer just to hold one record with GPRs and CTR
        - CSR allows expansion too, since SW can collect whatever state it wants if interrupting on every sample
    - Maybe base capability should be CSR, but also define extension to send samples to memory?
        - CSR option allows a low-complexity implementation, ala IBS
        - Memory option allows a low-overhead sampling option
        - If output to memory is enabled, sample data CSRs should be read-only 0, so don't have to save/restore them
        - Do we have a customer for this simpler CSR implementation?
            - Most in this TG are high-perf, likely to implement memory
            - But imagine that smaller form-factors would like it
            - Could allow implementations to hardcode enable for memory output, so don't have to implement CSR option as well
    - Whatever we decide now is not set in stone, can be re-evaluated later if new information comes to light. But hoping to have a planned direction.
- **Out of time**. Will meet next week in Perf Analysis SIG slot. Let's either close on this or defer it to after we define the rest of the details.

Action items

- ☐ Aug 1, 2024 - Beeman Strong - Propose method for evaluating tradeoffs of CSR vs memory buffer