[RISC-V] [HW-Designer] Is the "Quick Access" feature supported?

2026-09-30 - Comments (0) - RISC-V

Yes, the Lauterbach RISC-V debugger does support and utilize the “Quick Access” hardware feature in certain scenarios, if the target’s debug IP fulfills the respective requirements.

Quick Access is an optional feature of the RISC-V debug hardware on the chip. The Quick Access abstract command allows the hardware to stop the core for a very short time, then let the core perform a certain operation (e.g. memory access or register access) via program buffer execution, and then immediately resume the core again. As the sequence of all these steps is executed in one atomic hardware operation, the time during which the core is halted is as short and unintrusive as possible. This allows to execute certain operations while the core is running.

The Lauterbach RISC-V debugger can use Quick Access in several scenarios:

Run-time memory access via Quick Access:

- Enabled via “*SYStem.MemAccess QuickAccess*”
- Hardware requirements:
  - Quick Access abstract command supported
  - abstractcs.progbufsize >= 7 (see *note #1* below)
    - If progbufsize is at least 8, the debugger will additionally execute a “*FENCE.I*” for program memory accesses (if HW supports it), or execute a “*FENCE*” for data memory accesses (see *note #2* below).
    - If progbufsize is at least 9, the debugger will additionally execute a “*FENCE*” + “*FENCE.I*” (if HW supports it) for program memory accesses (see *note #2* below).
  - For RV32 cores:
    - abstractcs.datacount >= 3
    - hartinfo.datasize >= 3
  - For RV64 cores:
    - abstractcs.datacount >= 6
    - hartinfo.datasize >= 6

Run-time program counter snooping via Quick Access:

- Hardware requirements:
  - Quick Access abstract command supported
  - abstractcs.progbufsize >= 4 (see *note #1* below)
  - For RV32 cores:
    - abstractcs.datacount >= 2
    - hartinfo.datasize >= 2
  - For RV64 cores:
    - abstractcs.datacount >= 4
    - hartinfo.datasize >= 4

Note #1: The required sizes for "abstractcs.progbufsize" mentioned above do *not* take into account the required *ebreak* at the end of each program buffer instruction sequence. This means if the HW design does not support implicit ebreak (dmstatus.impebreak), then the required program buffer sizes all need to be increased by 1.

Note #2: In certain hardware setups with instruction- and/or data-caches, it may be necessary for the debugger to issue a *FENCE* + *FENCE.I* instruction. See our RISC-V FAQ entry "Instruction/data cache handling in TRACE32" for details.