Fast Interrupts TG

Status at a glance:

- [RVS-1017] Fast Interrupts (CLIC) - RISC-V Jira (riscv.org)
    - CLIC ratification status is now tracked using Jira which includes links to google docs ratification plan and status checklist.

Charter

- Current Charter at https://github.com/riscv/groups/tree/main/Fast-Interrupts
- Calendar: https://calendar.google.com/calendar/u/0/embed?src=tech.meetings@riscv.org
- Develop a low-latency, vectored, priority-based, preemptive interrupt scheme for interrupts directed to a single hart, compatible with the existing RISC-V standards. Provide both hardware specifications and software ABIs/APIs. Standardize compiler conventions for annotating interrupt handler functions.
- Meetings Disclaimers Video

Specification

- Latest Draft Fast Interrupt Specification (v0.9-draft-20240314)
- `What's next:`
    - 0 outstanding pre-ratification issues.
        - Freeze Checklist Completed
    - Major items on Freeze Checklist:

        - **RISC-V SAIL**

        SAIL Implementation Completed

        - **RISC-V Tests**

        ACT tests created. Missing SHV tests.

        - **RISC-V Tests Input**

        riscv-software-src/riscv-config: RISC-V Configuration Validator (github.com)

        Schema_isa.yaml - gives allowed configuration on risc-v and allowed values. WARL fields give allowed ranges. 10k lines in file.

        Examples/rivc32i_isa.yaml is compared against schema_isa.yaml to see if it is allowed.

        When add new CSRs, add PR to add to schema_isa.yaml and then will be checked against example implementation.

        Schemas/schema_platforml.yaml - memory mapped registers (like clicintctl, etc.)

        Run python scripts that look at your implementation and see if it is valid. e.g., if have d then have f- extenstion, etc. checks if WARL field is valid.

        Write anything, read legal. mapping from what is illegal to what is legal. mapping is arbitrary. so prefer (easiest) if implementers implement when write something illegal, don't write. that works easiest for describing in this file.

Encoding/OpCode consistency review

- Need to propose new CLIC CSR Registers and addresses

- What's next: when outstanding issues are reduced, start planning for review
- tech-chairs@lists.riscv.org - when spec is solid but not a final spec - primarily want to nail down opcode/CSR assignment and have a solid draft spec (but not a final spec ready for official Arch Review)
- Also, to remind people of what gets reviewed (as is appropriate for a given extension), see the following list. In addition to the extension spec, please submit information about the PoCs and about utility/efficiency (although we don't need all the gory detail - a paragraph or so for each can be fine). For items considered to not be consequential, a sentence or so explaining why should suffice.
    - Consistency with the RISC-V architecture and philosophy
    - Documentation clarity and completeness
        - Including proper distinction between normative and non-normative text
    - Motivation and rationale for the features, instructions, and CSRs
    - Utility and efficiency (relative to existing architectural features and mechanisms)
        - Is there enough value or benefit to justify the cost of implementation
        - Is the cost in terms of area, timing, and complexity reasonable
    - Proof of Concept (PoC)
        - Software PoC to ensure feature completeness and appropriateness for intended use cases
        - Hardware PoC to demonstrate reasonable implementability
    - Inappropriate references to protected IP (i.e. covered by patents, copyright, etc.)

Architecture Tests

- Deterministic Test plan for the fast-interrupt is available. Discussion on-going on how to add async/undeterministic testing of interrupts.
- YAML config needs to be created. See info here.
- Discussion in Arch tests group to add automation (docker?) to validate check-in so that arch-test-suite is run against sail, spike, gcc/toolchain, with versions used recorded. So sail and spike will need to work for CLIC before CLIC tests can be added to riscv-config github.

Compilers / Toolchains

GCC and Binutils

- No new instructions are added. Needs to be aware CSR names? Need to choose arch string. Given that the key CSRs are in M-mode, it should probably be named something like "Smclic"

LLVM

- No new instructions are added. Needs to be aware of CSR names?

Simulators

Though all listed under "simulators", these are actually a collection of formal model / virtual machine / architectural simulators / DV simulators etc.

SAIL

- https://github.com/riscv/sail-riscv is the official location

Spike

- TBD

riscvOVPSimPlus

- Imperas Commercial Simulator
- Freeware version

QEMU

- QEMU implementation of CLIC-0.9 specification (Version 0.9-draft-20210217)

Proof-of-Concept implementations

Hardware

Project Name	Base Architecture	Level of implementation	Notes
area-optimized core	RV32/64	RTL simulation, FPGA Implementation, Synthesis	closed / commercial source https://www.seagate.com/innovation/risc-v/
high-performance core	RV32	RTL simulation, FPGA Implementation, Synthesis	closed / commercial source https://www.seagate.com/innovation/risc-v/
microcontroller-class core	RV32IMAFC	RTL, fully synthesizable	Apache License, Version 2.0 https://github.com/T-head-Semi/opene906/blob/main/doc/opene906_datasheet.pdf
E2/S2 series	RV32/64	RTL, fully synthesizable	https://www.sifive.com/core-designer
N22	RV32	RTL, fully synthesizable	http://www.andestech.com/en/products-solutions/andescore-processors/riscv-n22/
BM-310/BI-651	RV32/64	RTL, fully synthesizable	https://cloudbear.ru/bm_310.html https://cloudbear.ru/bi_651.html
n200/n900/nx900/ux900	RV32/64	RTL, fully synthesizable (ECLIC)	https://www.nucleisys.com/product.php?site=n200 https://www.nucleisys.com/product.php?site=n900
R9A02G021	RV32	samples available	R9A02G021 Datasheet (renesas.com)

Software

Project/Maintainer	Description

ABI Extensions (no new ABI required)

- Regular C function that save/restores all caller-save registers
- Inline handler gcc interrupt attribute to always callee-save every register (save as you go)
- psABI Task Group - https://github.com/riscv-non-isa/riscv-elf-psabi-doc
    - https://github.com/riscv-non-isa/riscv-elf-psabi-doc/blob/eabi/eabi.adoc