OpenHW Group

Proven Processor IP

RISC-V CSR Access Modes

Mike Thompson

mike@openhwgroup.org

Objectives

- Most digital hardware functional blocks support a set of Control and Status Registers.
- We need to verify the power-on-reset value, function and access behavior of all CSRs:
  - **PoR**:
    - Value stored in the CSR after release of reset.
  - **CSR function**:
    - Verify that the control fields change the behavior of the DUT as expected.
    - Verify that the status fields accurately reflect the the state of the DUT over time.
  - **CSR access behavior**:
    - Verify that writable fields can be written and will store the expected value.
    - Verify that non-writable fields do not change when written.
- This presentation discusses the verification of CSR access behavior for RISC-V compliant cores.

Verification of CSR Access Modes

- There are a lot of CSRs with a lot of modes, so test-programs to verify them are typically long.
  - The CV32E40X CSR access mode test-program is ~42KLoC (assembler).

- Fortunately, much of the effort can be automated if we have a machine readable specification of the CSRs.
  - We already have a prototype of this in place.
  - The machine readable CSR spec can also be used to create a human readable User's Manual.

- The machine readable specification needs to be complete and able to generate comprehensive tests.
  - The current format (used by CV32E40) is close to complete, but not quite there.

CSR Attributes

- The verification of CSR PoR values, access behavior and function is impacted by per-CSR “attributes”.

- The number and features of “CSR attributes” are typically unique to the DUT.

- In RISC-V cores, CSRs have one of two attributes:
  - **Volatile**: the value stored in the CSR is updated by the hardware logic, without software action (e.g. mcycle).
  - **Non-volatile**: the value stored in the CSR is static and will not change its value unless software writes it.

- Verification of the access mode behavior of volatile CSRs is particularly challenging.

Verification: it all starts with a Specification

- The CSR specification should provide the following information:
  - Volatile or not.
  - Address.
  - PoR value.
  - Fields:
    - Bit positions.
    - Access mode.
    - Description (what does it do?)

Machine Scratch (`mscratch`)

CSR Address: 0x340

Reset Value: 0x0000_0000

Detailed:

Bit #	R/W	Description
31:0	RW	Scratch value

Access Model Specification

- Typical access mode behavior in non-RISC-V designs:
  - Read-Only (**RO**): writes to CSR will have no affect.
  - Read-Write (**RW**): a write will change the value stored in the CSR. Subsequent reads will return the previously written value.
  - Write-1-to-Clear (**W1C**): Writing all 1's will clear the CSR. Otherwise the CSR is RO.

- The access mode behavior of RISC-V CSRs also depends on the privilege mode of the core:

  The RISC-V ISA explicitly provides this in the "Privilege" column of the CSR Address tables (see next slide…)

CSR Privilege

- Section 2.2, *CSR Listing*, of the Privileged Spec has a “Privilege” column which looks a lot like a machine-mode-specific access behavior:
  - e.g. **MRO** *may* read-only in machine mode.
- These privilege labels are never defined in the ISA.
- What is the access behavior of a “Machine CSR” when not in machine mode?
- Accessing the dcsr (DRW) when not in Debug Mode raises an illegal instruction:
  - This adds an extra dimension to access mode verification.

Number	Privilege	Name	Description
Machine Information Registers			
0xF11	MRO	mvendorid	Vendor ID.
0xF12	MRO	marchid	Architecture ID.
0xF13	MRO	mimpid	Implementation ID.
0xF14	MRO	mhartid	Hardware thread ID.
0xF15	MRO	mconfigptr	Pointer to configuration data structure.
Machine Trap Setup			
0x300	MRW	mstatus	Machine status register.
0x301	MRW	misa	ISA and extensions
0x302	MRW	medeleg	Machine exception delegation register.
0x303	MRW	mideleg	Machine interrupt delegation register.
0x304	MRW	mie	Machine interrupt-enable register.
0x305	MRW	mtvec	Machine trap-handler base address.
0x306	MRW	mcounteren	Machine counter enable.
0x310	MRW	mstatush	Additional machine status register, RV32 only.
Machine Trap Handling			
0x340	MRW	mscratch	Scratch register for machine trap handlers.
0x341	MRW	mepc	Machine exception program counter.
0x342	MRW	mcause	Machine trap cause.
0x343	MRW	mtval	Machine bad address or instruction.
0x344	MRW	mip	Machine interrupt pending.
0x34A	MRW	mtinst	Machine trap instruction (transformed).
0x34B	MRW	mtval2	Machine bad guest physical address.
Machine Configuration			
0x30A	MRW	menvcfg	Machine environment configuration register.
0x31A	MRW	menvcfgh	Additional machine env. conf. register, RV32 only.
0x747	MRW	mseccfg	Machine security configuration register.
0x757	MRW	mseccfgh	Additional machine security conf. register, RV32 only.
Machine Memory Protection			
0x3A0	MRW	pmpcfg0	Physical memory protection configuration.
0x3A1	MRW	pmpcfg1	Physical memory protection configuration, RV32 only.
0x3A2	MRW	pmpcfg2	Physical memory protection configuration.
0x3A3	MRW	pmpcfg3	Physical memory protection configuration, RV32 only.
		⋮	
0x3AE	MRW	pmpcfg14	Physical memory protection configuration.
0x3AF	MRW	pmpcfg15	Physical memory protection configuration, RV32 only.
0x3B0	MRW	pmpaddr0	Physical memory protection address register.
0x3B1	MRW	pmpaddr1	Physical memory protection address register.
		⋮	
0x3EF	MRW	pmpaddr63	Physical memory protection address register.

Table 2.5: Currently allocated RISC-V machine-level CSR addresses.

Specifications are never perfect

- RISC-V CSR Field Specifications are specified from the perspective of the ABI, not the RTL:
  - The impact of CSR Field Specs on access mode is not made clear.

- Examples of ABI-centric descriptions of access mode behavior, for example:
  - Write Any Read Legal (**WARL**)
  - Reserved Writes Preserve Values, Reads Ignore Values (**WPRI**)

- It is not clear how to develop an access mode test based on these access mode descriptors:
  - For example, a designer could choose to implement a WPRI field as RO.
  - In CV32E40P, WARL assumed to be equivalent to RW.

…and a couple of more things!

- In RISC-V and CORE-V projects, whether or not a CSR is volatile or not must be inferred from the Description.
  - We should explicitly add the attribute to the User Manual.

- The RISC-V ISA may also provide for implementation-specific behaviors:
  - The core's User Manual must specify these.
  - Open a GitHub issue if it does not!

What to do with all this Information?

Start with the YAML CSR description we already have:

- Update privilege_mode field to match Section 2.2, *CSR Listing*, of the Privileged Spec.
- Include type in User Manual's CSR Table.
- Fully specify all 64/32 bits of a CSR:
  - Unused fields should be fully specified as type=RO and reset_val=0x0.
- Add a volatile field.

```
- csr: misa
  description: >
    Machine ISA Register
  address: 0x301
  privilege_mode: M
  rv32:
    - field_name: MXL
      description: >
        Encodes native base ISA width
      type: WARL
      reset_val: 1
      msb: 31
      lsb: 30
    - field_name: Extensions
      description: >
          Encodes all supported ISA extensions
      type: WARL
      reset_val: 0x104
      msb: 25
      lsb: 0
```

Prototype Algorithm for Access Mode Tests

```
for csr in csrs:
  if (csr.machine_mode):
    if (!volatile):
      # will need one of these for each mode supported by the DUT
      check_access(user_mode)

      # The key to access mode testing will be proper parsing of
      # CSR Privilege (MRO, MRW,etc.) and CSR Field Specification (WPRI, WLRL, etc.)
      # to create a read-write mask.
      mask = build_mask(csr)

      # 'patterns' should include all ones, all zeros, 0x55, 0xAA and a "walking one"
      for pattern in patterns:
        write_csr(csr.addr, pattern)
        if ((mask || read_csr(csr.addr)) != pattern):
          error(csr, pattern)
    else:
      # Volatile CSRs each require unique handling for access mode testing
      if (csr.name == 'mcycle'):
        do_mcycle_access_test()

  elif (csr.user_mode):
    # access testing for URO and URW CSRs
```

Next Steps

- Determine what the *Privilege* column in the CSR address tables in the privilege spec means.
  - The CV32E40P only supports Machine and Debug modes so we assumed the following:
    - MRO and MRW means the CSR access modes are RO and RW.
    - DRO and DRW mean:
      - CSR access modes are RO and RW in Debug mode.
      - Access to these CSRs in machine mode were trapped as illegal instructions.

- Confirm that CSR Field Specifications (WPRI, etc.) do not affect RTL-level access mode testing.

- Update the CSR YAML description.

- Create a generator to produce CSR documentation and access mode tests:
  - Write it ourselves or use Jade.

Thank You