IOPMP Task Group Meeting

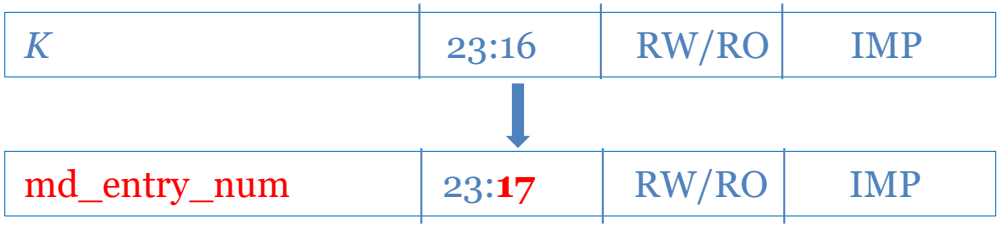
November 7, 2024

Video link

Minutes

- The feedback of r0.9.2-RC1:
  - wrong *k*-value position/# of bits: fixed
  - MSI support or not? Yes, taking 3 new fields/registers
  - clarification: SRCMD table fmt=2: the spec need a figure to help readers
  - SRCMD table fmt=2 for overwriting permission within an MD: explanation
  - SRCMD table fmt=2 for MDSTALL: simplified implementation

Adjust HWCFG0.rsv



For MDCFG_FMT=:

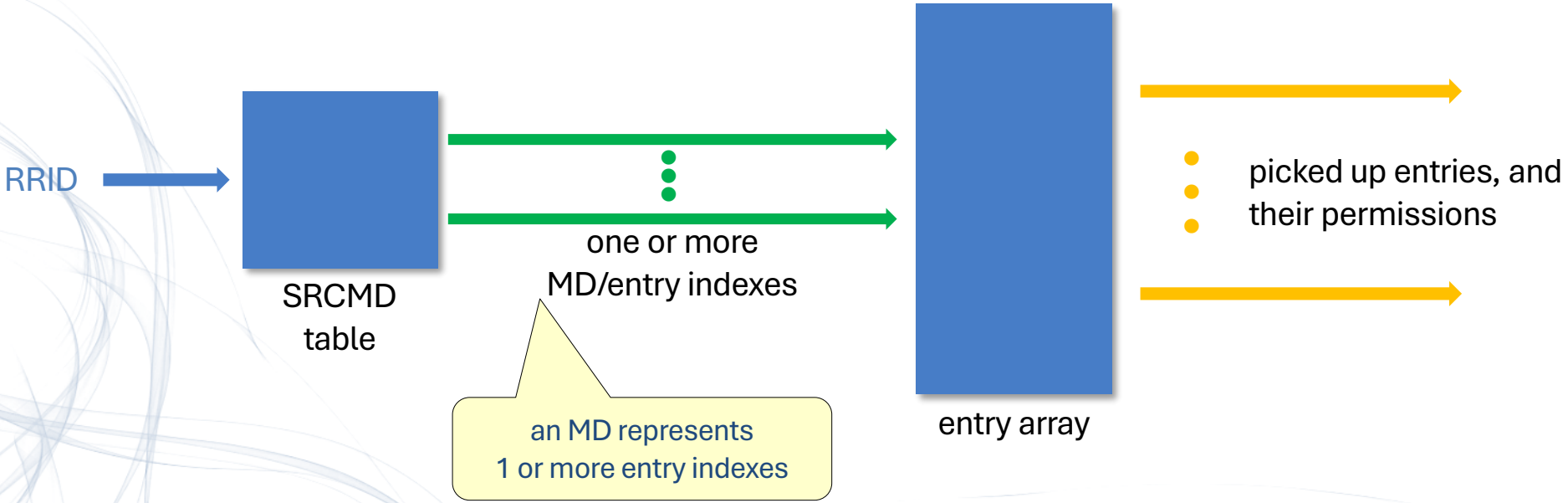
- 0: ZERO [0] for full model; //MD is configured by MDCFG
- 1: RO [*k* value - 1] for rapid-*k*; //No MDCFG
- 2: RW/RO [*k* value - 1] for dynamic-*k*; //No MDCFG

It can still support up to 128x63 = 8,064 entries for the rapid-*k*.
For even larger requests, the full model can support up to 65,535 entries.

Shall the IOPMP spec support MSI?

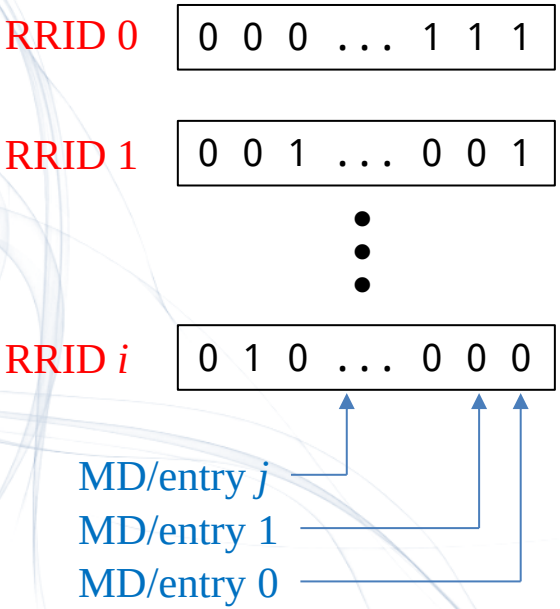
- Current spec only defines a single interrupt signal, no MSI supported. To use MSI, one should use APLIC now.
- Shall we direct supporting MSI?
  - Registers needed:
    - A 11-bit or 32-bit MSI data to write
      - EIID at APLIC: 11 bits
      - IOMMU uses 32 bits
    - A 32-bit and 64-bit MSI address to write
    - A bit indicates which mode (single wire or MSI) is implemented.
  - A lock mechanism (maybe we can reuse ERR_CFG.l)

Entries pick up of IOPMP (SRCMD.fmt=0)



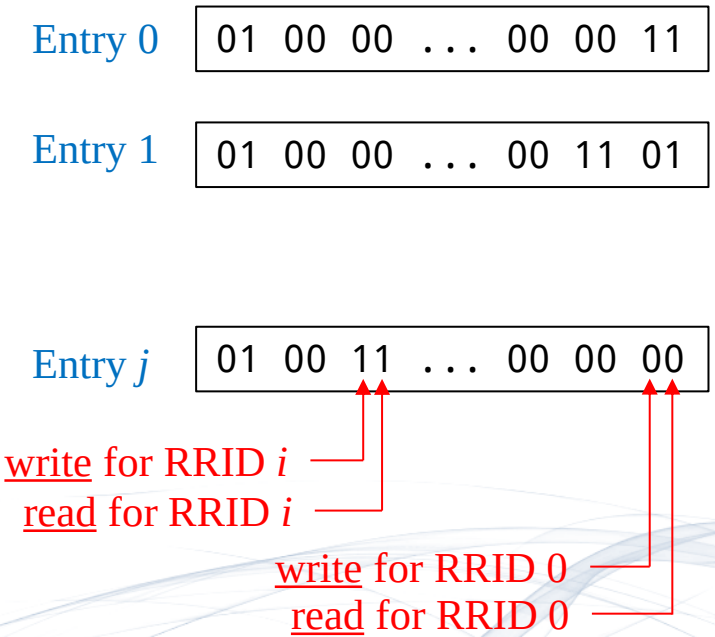
Format 0 vs Format 2

Format 0:
the table was index by RRID

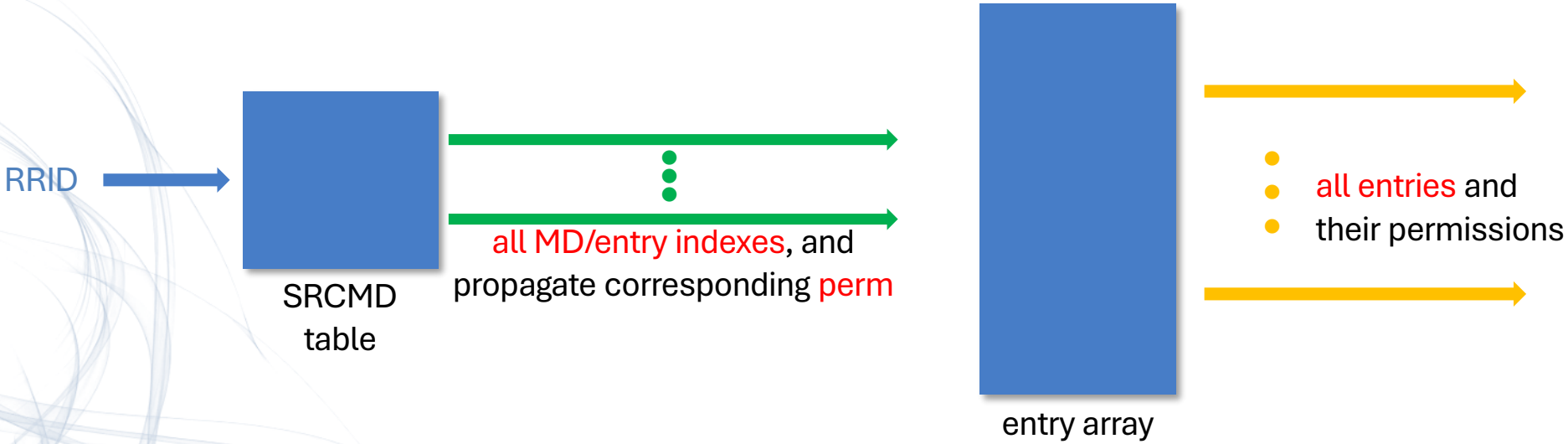


SRCMD table

Format 2:
the table is index by entry

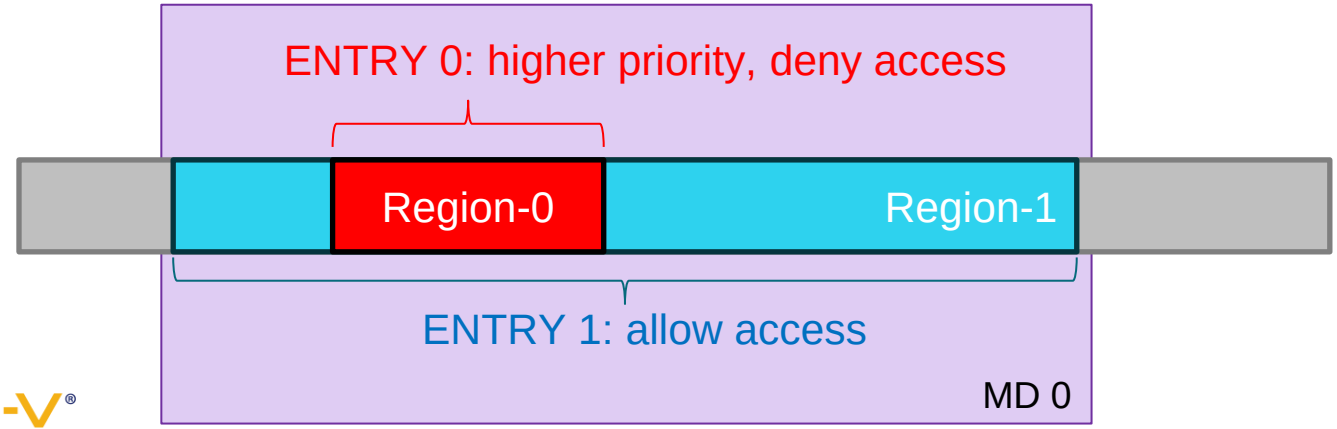


Entries pick up of IOPMP (SRCMD.fmt=2)



Permission exception within an MD

- Drill a permission exception by higher-priority rules within an MD, for SRCMD.fmt ≠ 2
- EX: MD0 = (ENTRY 0, ENTRY 1)
  - ENTRY_CFG(0).r/w=0b00 for region 0 // narrow range, block access
  - ENTRY_CFG(1).r/w=0b11 for region 1 // wider range, allow access



Priority rules under SRCMD.fmt = 2

- Drill a permission exception by higher-priority rules for SRCMD.fmt = 2:
- Problem: MD[0]= (ENTRY[0], ENTRY[1])
  - ENTRY_CFG(0).r/w=0b00 for region 0
  - ENTRY_CFG(1).r/w=0b11 for region 1
  - Add SRCMD(0).PERM[rrid].r/w=0b00 //that is, let ENTRY_CFG decide

r/w permission for RRID r == PERM[r].r/w OR ENTRY_CFG.r/w

MDSTALL for SRCMD.fmt = 2

- MDSTALL.md stalls the related RRIDs by selecting MDs; however, every MD is related to all RRIDs. That is, all RRIDs will be stalled together as long as you select any MDs.
- Thus, in MDSTALL(H), only the bit 0 is effective : the field exempt (for write) or is_stalled (for read), no matter how many MDs are implemented.
- Proposal: in SRCMD.fmt=2, if one wants to implement MDSTALL(H), one can implement bit 0 in MDSTALL only no matter how many MDs are implemented.
  - Let MDSTALL ← 1 // stall all RRIDs
  - Let MDSTALL ← 0 // resume all RRIDs

✓ RRIDSCP: not need to change.