IOPMP TG Meeting

October 13th, 2022

Agenda

- Disclosure:
- Atomicity issues on programming IOPMP
  - Andes' proposal

Only RISC-V Members May Attend

- Non-members are asked to please leave except for Joint Working Groups (JWG).
- Members share IP protection by virtue of their common membership agreement. Non-members being present jeopardizes that protection. Joint working groups (JWG) agree that any IP discussed or worked on is fully open source and unencumbered as per the policy.
- It is easy to become a member. Check out riscv.org/membership
- If you need work done between non-members or or other orgs and RISC-V, please use a joint working group (JWG).
    - used to allow non-members in SIGs but the SIGs purpose has changed.
- Please put your name and company (in parens after your name) as your zoom name. If you are an individual member just use the word “individual” instead of company name.
- Non-member guests may present to the group but should only stay for the presentation. Guests should leave for any follow on discussions.

Antitrust Policy Notice

RISC-V International meetings involve participation by industry competitors, and it is the intention of RISC-V International to conduct all its activities in accordance with applicable antitrust and competition laws. It is therefore extremely important that attendees adhere to meeting agendas, and be aware of, and not participate in, any activities that are prohibited under applicable US state, federal or foreign antitrust and competition laws.

Examples of types of actions that are prohibited at RISC-V International meetings and in connection with RISC-V International activities are described in the RISC-V International Regulations Article 7 available here: https://riscv.org/regulations/

If you have questions about these matters, please contact your company counsel.

Collaborative & Welcoming Community

RISC-V is a free and open ISA enabling a new era of processor innovation through open standard collaboration. Born in academia and research, RISC-V ISA delivers a new level of free, extensible software and hardware freedom on architecture, paving the way for the next 50 years of computing design and innovation.

We are a transparent, collaborative community where all are welcomed, and all members are encouraged to participate. We are a continuous improvement organization. If you see something that can be improved, please tell us. help@riscv.org

We as members, contributors, and leaders pledge to make participation in our community a harassment-free experience for everyone.

https://riscv.org/community/community-code-of-conduct/

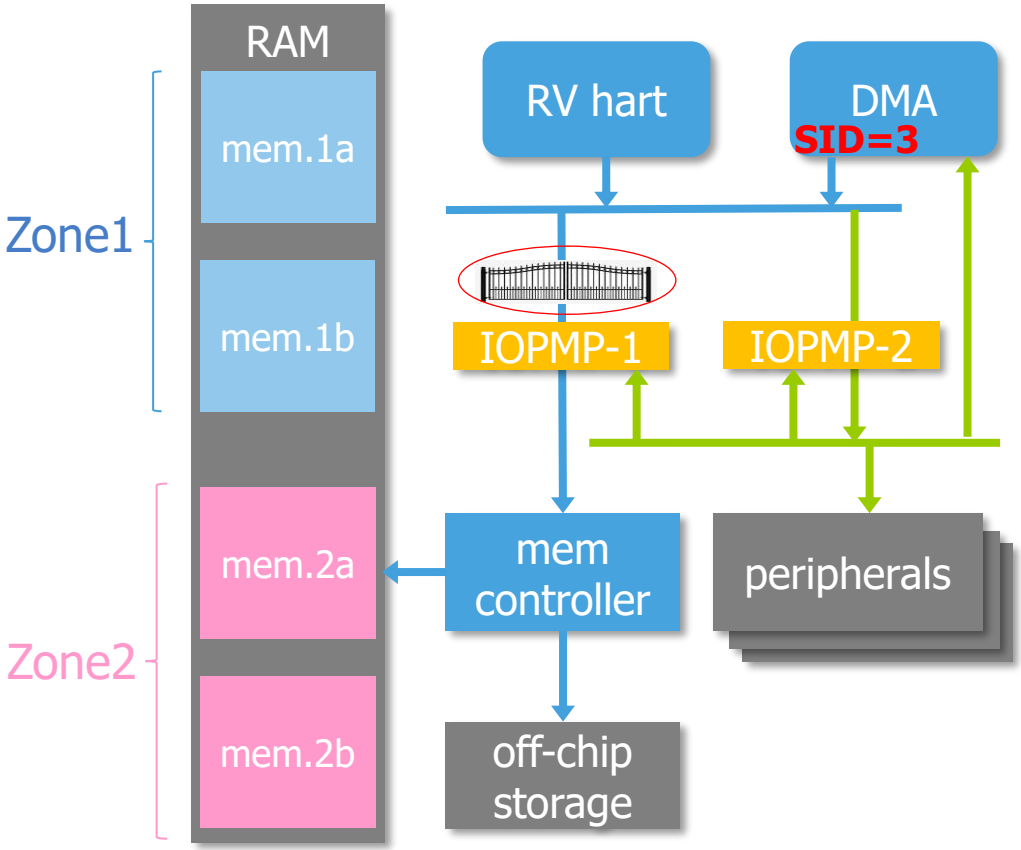
Conventions

- **For one hour meetings, please start at 5 after the start time** in order to allow people going to other meetings have time for a short break between meetings. 30 minute meetings start on time.
- Unless it is a scheduled agenda topic, we don’t solve problems or detailed topics in most meetings unless specified in the agenda because we don’t often have enough time to do so and it is more efficient to do so offline and/or in email. We identify items and send folks off to do the work and come back with solutions or proposals.
- If some policy, org, extension, etc. can be doing things in a better way, help us make it better. Do not change or not abide by the item unilaterally. Instead let's work together to make it better.
- Please conduct meetings that accommodates the virtual and broad geographical nature of our teams. This includes meeting times, repeating questions before you answer, at appropriate times polling attendees, guide people to interact in a way that has attendees taking turns speaking, …
- Where appropriate and possible, meeting minutes will be added as speaker notes within the slides for the Agenda

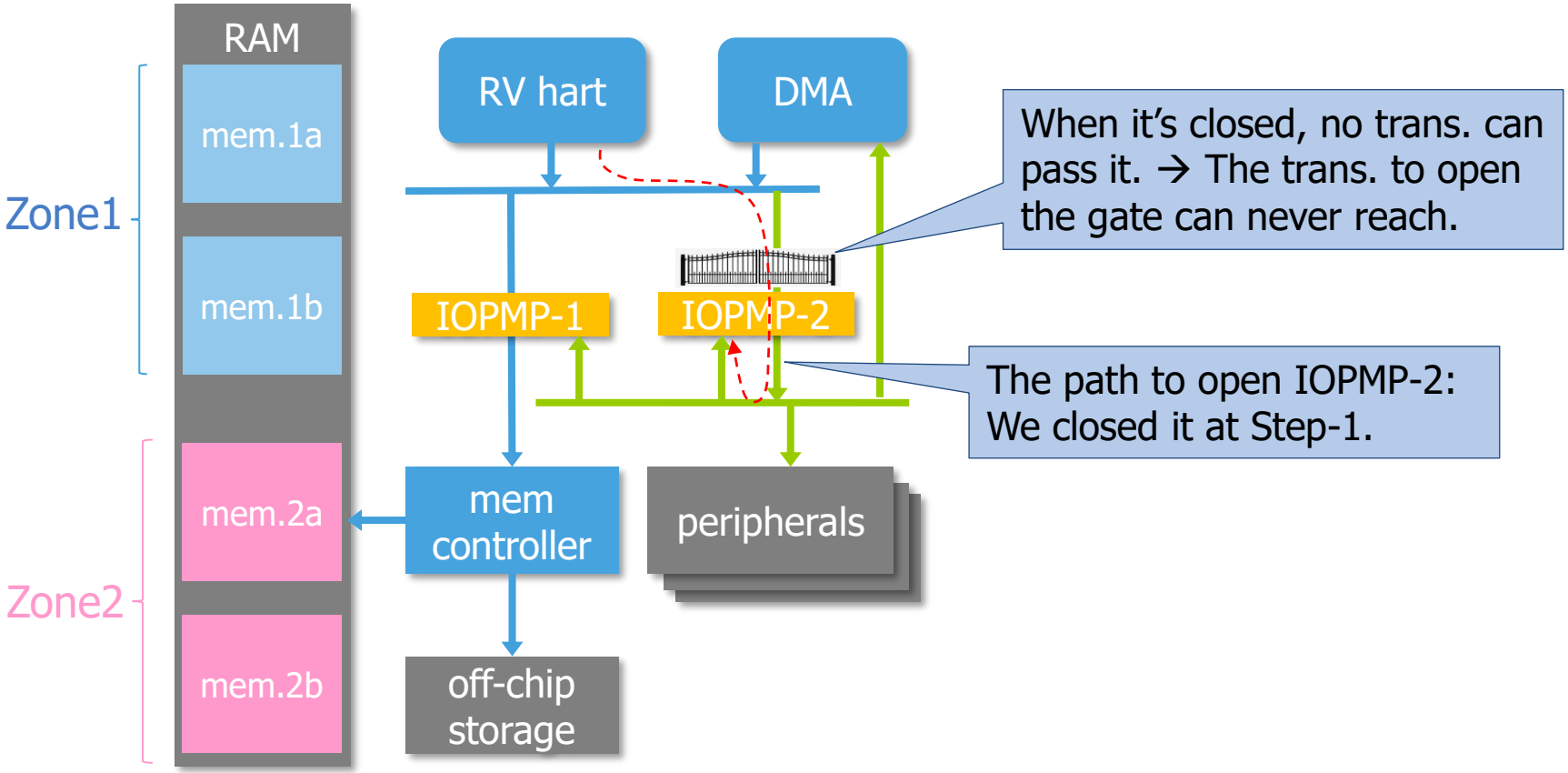
Atomicity Issues

- On programming multiple rules in the run-time, there could be a moment that IOPMP works under incomplete settings. → It could transiently create a security hole.

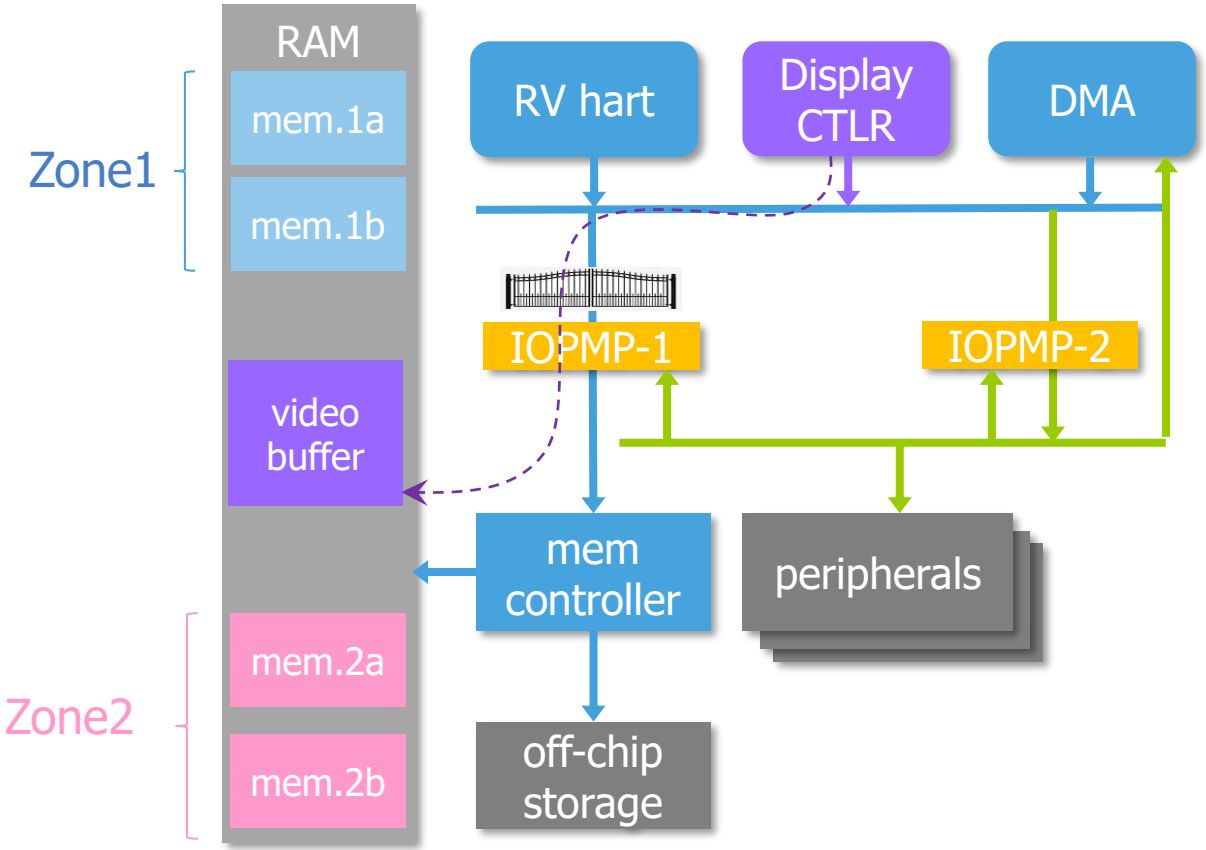
Add a Gate


RAM
mem.1a
mem.1b
mem.2a
mem.2b
Zone1
Zone2
RV hart
DMA
SID=3
IOPMP-1
IOPMP-2
mem controller
peripherals
off-chip storage

Deadlock Created by Adding a Gate



A System with a Display Controller



Drill a Little Hole in the Gate

- Allow an exception while IOPMP is stalled:
  - **Per-SID**:
    - **SID-Stall**: Specify one or more SID whose transactions should wait.
    - **SID-exempt**: Specify one or more SIDs whose transactions can still go through the IOPMP.
  - **Per-MD**:
    - **MD-Stall**: Specify one or more MDs that transactions target them should wait.
    - **MD-exempt**: Specify one or more MDs that transactions target them exempt the stall.

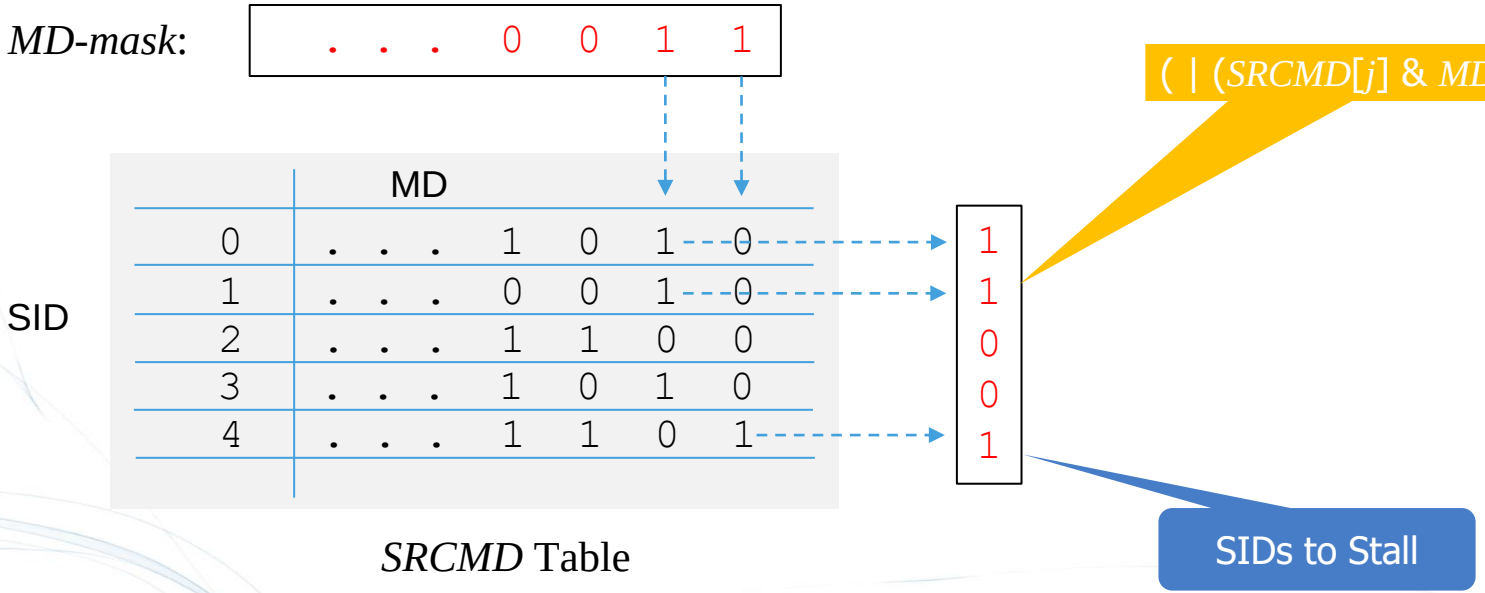
Classification Table

	Per-SID	Per-MD
Stall	SID-Stall	MD-Stall
Stall-Exempt	SID-Exempt	MD-Exempt
	introduced	talk today

Per-MD Stall

- Stall MDi → stall any transaction matching MDi.
- We specify MDs when we want to stall a transaction:
- Each $\mathrm{MD}i$ (for $0 \le i \le 62$) has a bit, $MD\text{-}mask[i]$:
  - Write 1: try to stall the transactions associated to $\mathrm{MD}i$.
  - Write 0: resume the transactions
- $MD\text{-}mask[63]$:
  - Return 1 on read: all targeted $\mathrm{MD}i$ are stalled
  - Return 0 on read: all targeted $\mathrm{MD}i$ are not stalled yet.

Per-MD Stall



SRCMD Table

Invisible *SID-stalling* bits

- Every SID=*j* has a bit, *SID-stalling*[*j*]:
  - A transaction with SID=*j* should be stalled if *SID-stalling*[*j*]==1;
  - All *SID-stalling*[*j*] are updated only when *MD-mask* is written.
    - =1 if any *i*, *MDi* associated to SID=*j*, where *MD-mask*[*i*]==1;
    - =0, otherwise;
    - No update when *SRCMD* table is updated. ➔ *SID-stalling* keeps stable while updating IOPMP. ➔ Stall or not will be based on the 'old' *SRCMD* table. ➔ Atomicity Requirement is met.
    - That is, *SID-stalling* is a snapshot of the *SRCMD* table at the moment of writing *MD-mask*.
  - The bit is invisible for programmers.
  - Users shouldn't care about the bits.

Per-MD Stall Steps

1) Set $MD\text{-}mask[i]$ to 1 for all MDi about updating.
2) Read back $MD\text{-}mask$ and wait until $MD\text{-}mask[63] == 1$.
3) Update IOPMP.
4) Set all $MD\text{-}mask$ to 0.

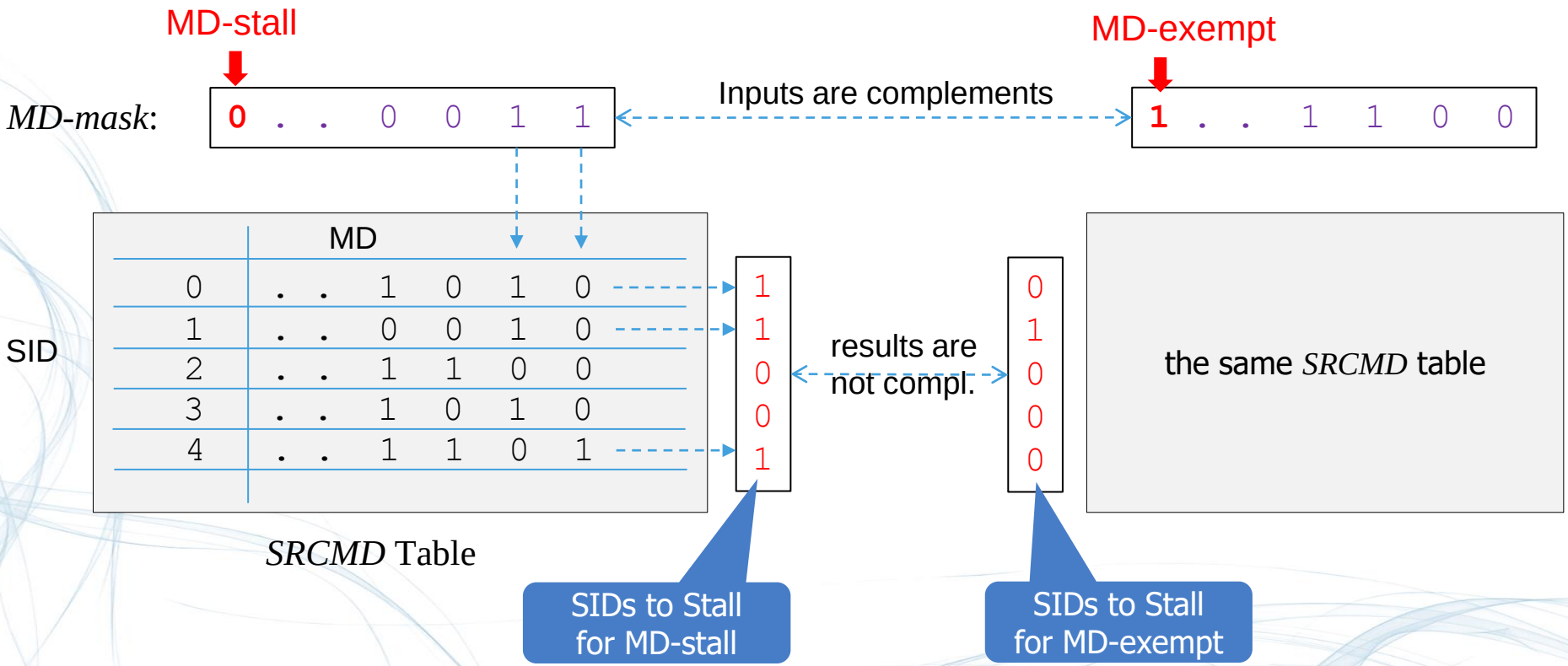
Per-MD Exempt

- In stall-exempt, *MD-mask*[*i*] specify the *MDi*-related transaction can keep going through IOPMP without stalling.
- *stall-exempt* bit, *MD-mask*[63], controls if *MD-mask* works in stall-exempt or stall:
  - *SID-stalling*[*j*] <= *stall-exempt* ^ (| (*SRCMD*[*j*] & *MD-mask*));

Complement here!

Same as MD-stall

Per-MD Stall vs Exempt



Why Per-MD Scheme?

- # of MD is the small number, ≤63
  - One register is enough, no uncertain number of registers
  - ➢ Easier for MMIO space allocation and easier for users.
- All *MD-mask* can be accessed in one transaction:
  - One transaction update (write *MD-mask*),
  - One transaction polling (read back *MD-mask*), and
  - One instruction to compare (test if stall successfully)
  - ➢ Lower latency

Scenarios for Per-MD or Per-SID

- Scenario-1: Update Entries for one or more MD*i*:
  - to stall all transactions to related MD(s). (per-MD more convenient.)
- Scenario-2: Update one or more MDTOP registers:
  - to stall all transactions to related MD(s). (per-MD)
- Scenario-3: Update *SRCMD* for SID(s):
  - Update 1 SID: By nature, it meets the atomicity requirement.
    - No stall is needed.
  - Update 2 or more SIDs: to stall all transactions:
    - Per-SID scheme: specify the SIDs to stall. (per-SID)
    - Per-MD scheme: specify the MD to exempt the stalling. (per-MD)

Thank You