IOPMP Updates:
The Protection of IOPMP

Paul Shan-Chyun Ku, Ph.D.
Deputy Technical Director, Architecture
Andes Technology
2021/12/8

Biography of Dr. Paul Shan-Chyun Ku

Technical Areas	• SoC Architect, Platform Security • Parallel Algorithms, System-level Performance Analysis	
Industry Experience	RISC-V, 2021	Vice Chair of TEE Task Group
	Andes, 2019	Deputy Technical Director of Architecture
	Realtek, 2009	Manager of SoC, VoIP, and BSP
	Cadence, 2006	Member of Consulting Staff
	Faraday, 2001	Deputy Manager of Core Technology
Education	• PhD, CS, National Tsing-Hua University (Taiwan) • BS, CS, National Tsing-Hua University (Taiwan)	

Agenda

1 What Is IOPMP for

2 RISC-V IOPMP and Its Models

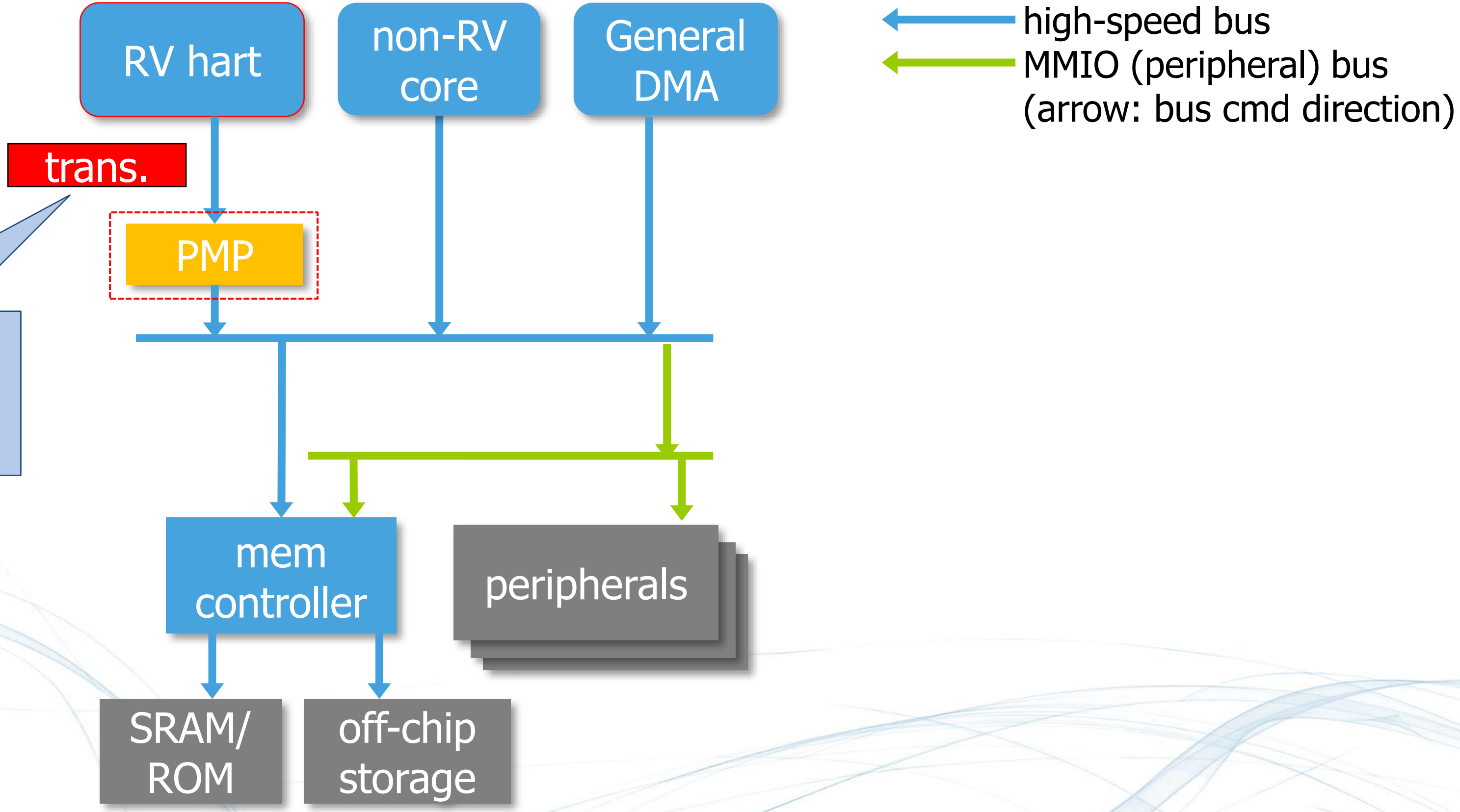
3 Protect IOPMP by Its Nature

4 Fine-grained Protection

5 Concluding Remarks

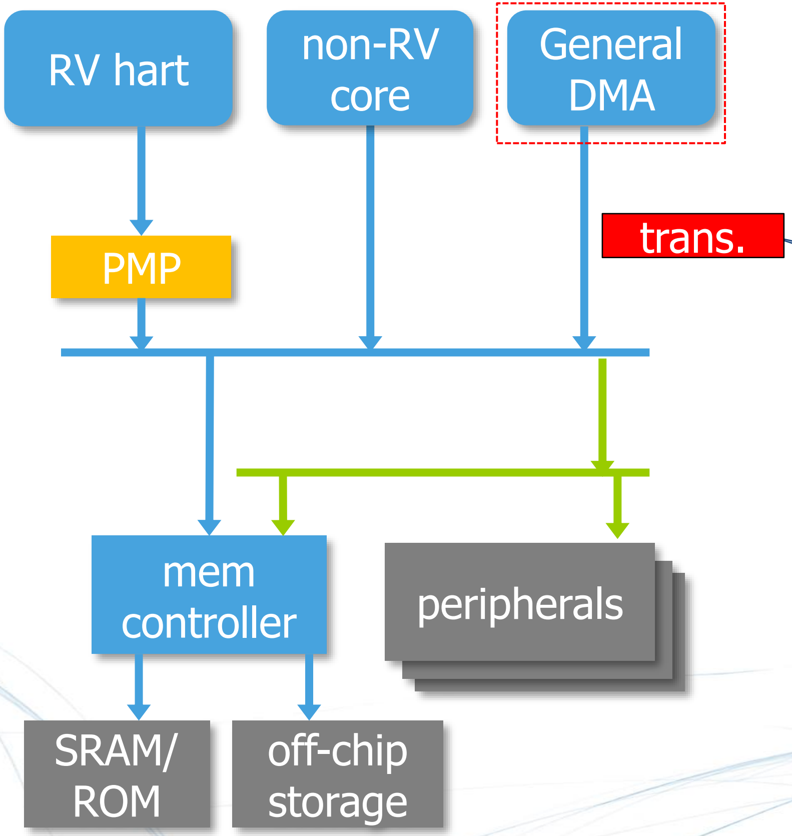
What Is IOPMP for?

A Platform without IOPMP



transactions issued from RV hart: checked by PMP

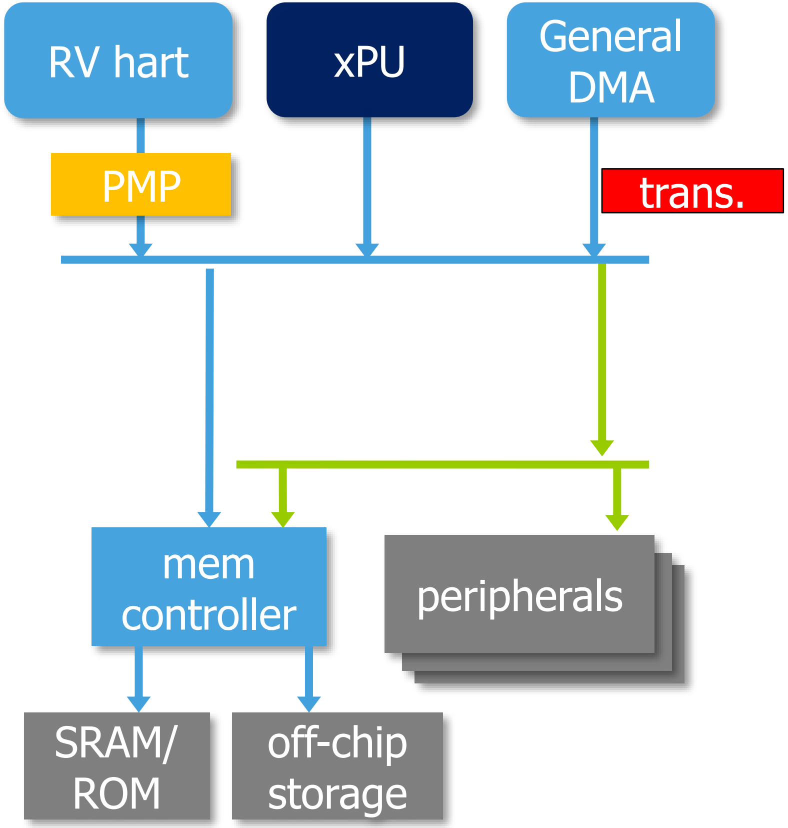
A Platform without an IOPMP



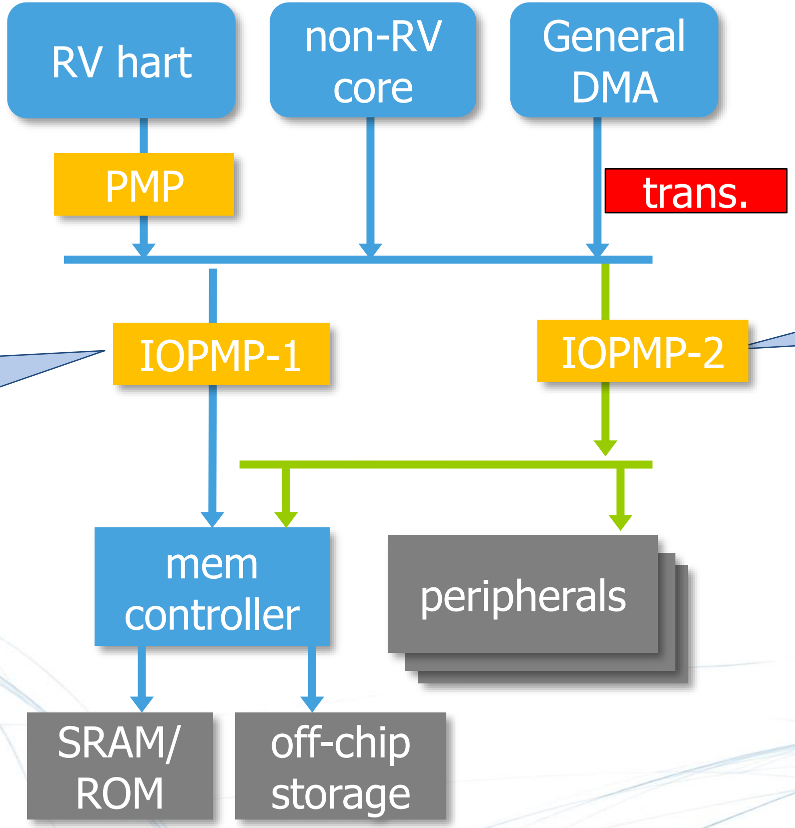
transactions issued from DMA:
Never check

Malicious SW can utilize DMA to access any data.

A Platform with GPU/PPU/NPU/…



A Platform with IOPMP



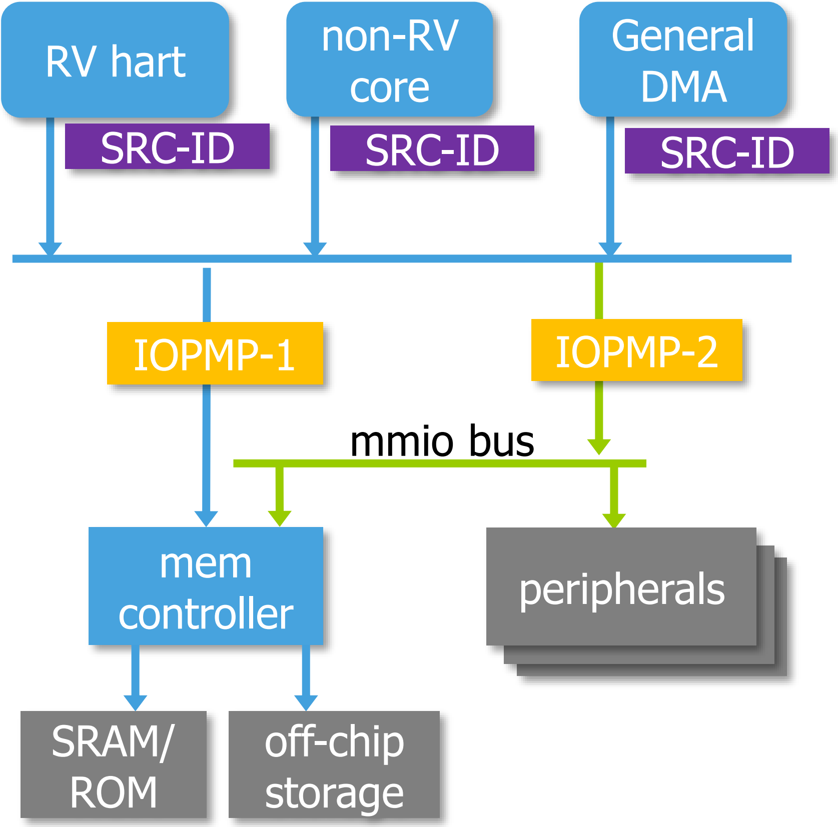
transactions issued from other masters: checked by IOPMP

transactions issued from other masters: checked by IOPMP

RISC-V IOPMP and its Models

Source ID (SID)

- A source-ID represents a bus master or a group of bus masters with the same permission.
- A bus master has one, but could be the same as another.
- A bus master with multi-channel, multi-VM or multi-mode may need 1+ SIDs.

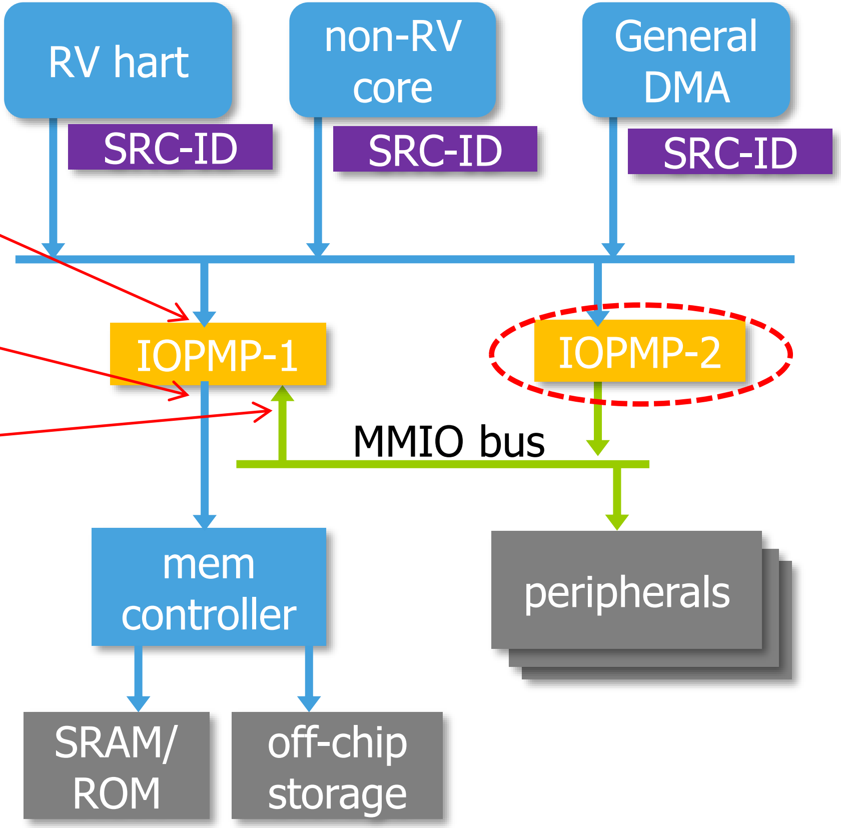


IOPMP: Rule-based Checker

- An IOPMP has an ordered list of entries, and every entry has
  - A memory region: defines the area related to this entry
  - Permission: read, write, both, or none
- Entry with the lower the index has the higher priority.
- A transaction crossing the region boundaries is illegal!
- Map tables: map from a SID to its IOPMP entries.

IOPMP Interface to the Platform

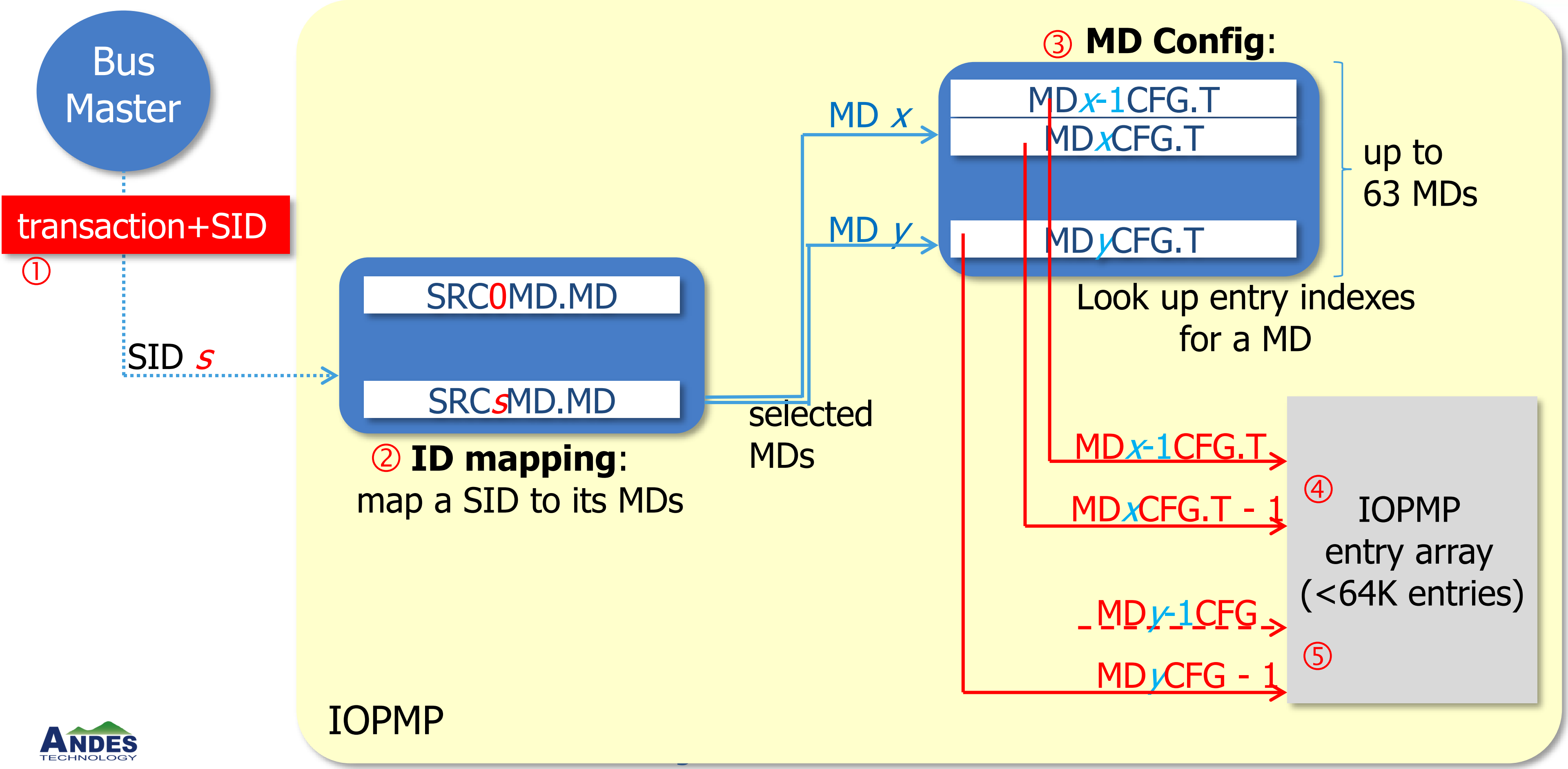
- The master port:
  - Where the transaction flow gets into an IOPMP
- The slave port:
  - Where the transaction flow leaves an IOPMP
- The control port:
  - Control the IOPMP
  - Usually connect to a MMIO bus
- Act as a bus bridge crossing two types of buses:
  - EX: IOPMP-2



Two IOPMP Variations

- Source Enforcement:
  - Its master port connect to the slave port of a bus master. Serves the bus master only.
  - No source-ID at all.
  - Has IOPMP entries only, so protect them only.
- Destination Enforcement:
  - Serves multiple bus masters.
  - Source-ID is needed.
  - Has the mapping from a SID to its IOPMP entries.
  - More setting protections.

Retrieve IOPMP Entries (Full Model)



Check by IOPMP Entries

- Selected IOPMP entries:
  - Ordered; the lower order, the higher priority
  - Every entries contains:
    - One region of memory, and
    - the permission on the region.
- If a permission violation is caught, we may have:
  - An asynchronous interrupt is issued,
  - A bus error,
  - A record of the violation,
  - Ignoring the transaction silently, or
  - Forge the response.

The Other Models

- The full model
  - Manage a large number of IOPMP entries flexibly, and
  - Share the entries among MDs or SIDs,
  - But at the cost of:
  - the latency,
  - the area, and
  - the energy.
- The other models are induced from the Full Model: replacing one or two tables by simple logics.

The Other Models

- Other 3 models are derived:
  - The Isolation Model, the Rapid-*k* Model and the Compact-*k* Model.

		ID Map (SRC-MD table)	
		Table lookup	Reduced
MD-CFG	Table lookup	**Full Model**	**Isolation Model**
	Shifter	**Rapid-*k* Model**	**Compact-*k* Model**

MD to entries map:
by a const shifter.

Fix: SRC ID = MD ID

Protect IOPMP

Why We Need to Protect IOPMP?

- IOPMP is used to regulate the transactions: “who can access which”
- Its settings tamper with maliciously; just like make it close eyes
- The unwanted transactions can access the sensitive data by manipulating a DMA. → The IOPMP-based system protection collapses.

What "Protecting IOPMP" Means?

- The two levels requirements:

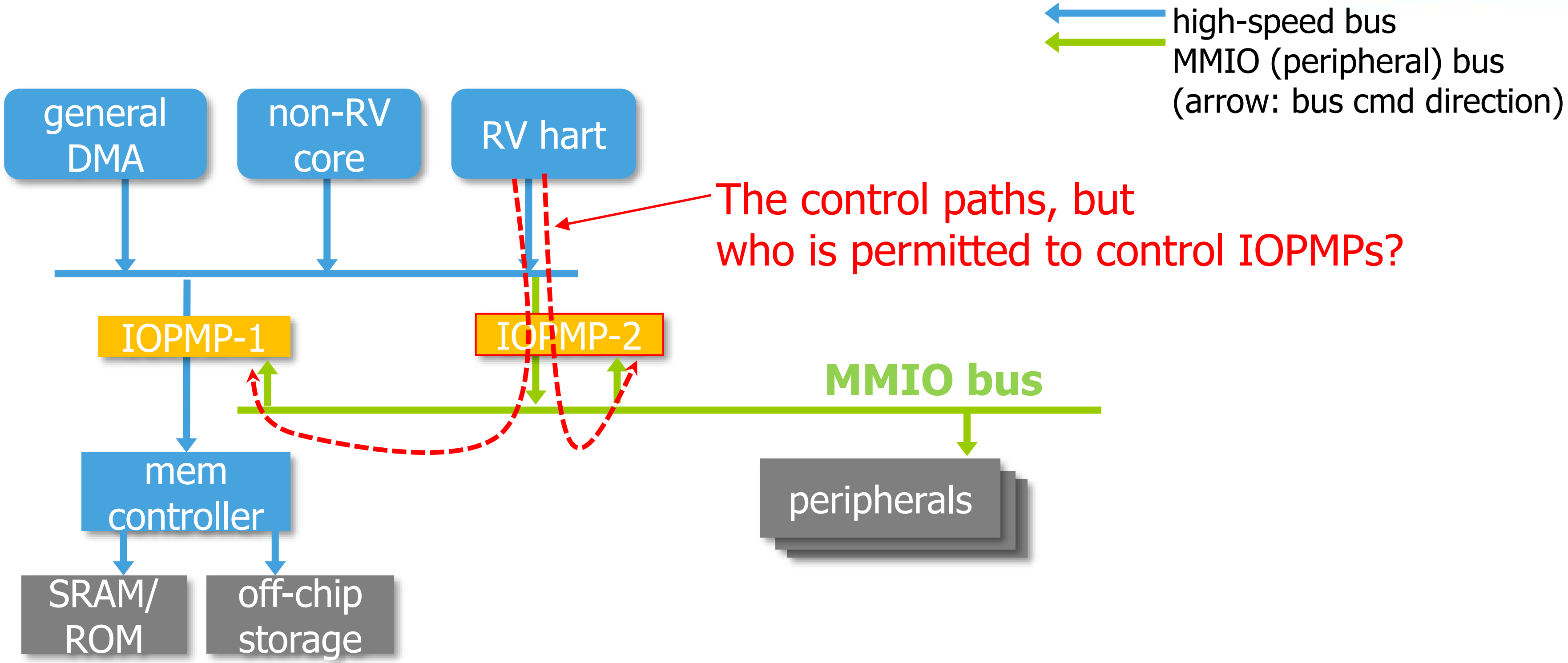
  The first requirement: control who can control IOPMPs

  - In the previous figure, all IOPMP control registers are hung on the MMIO bus, so IOPMP-2 can take the responsibility.
  - IOPMP should be controlled by *trusted* and *predefined* roles, such as "secure boot" or "security monitor."

  The second requirement: mitigate once trusted software is compromised

  - For example: an anti-rollback counter stored inside the chip can be accessed only during the boot-time, afterward all accesses are denied. A locked rule can help to enforce it even when the security monitor is compromised in the runtime.
  - A hardware mechanism to "lock" IOPMP fully or partially until rest. Like the lock feature of PMP entry, it provides one more layer of protection.

Protect IOPMPs

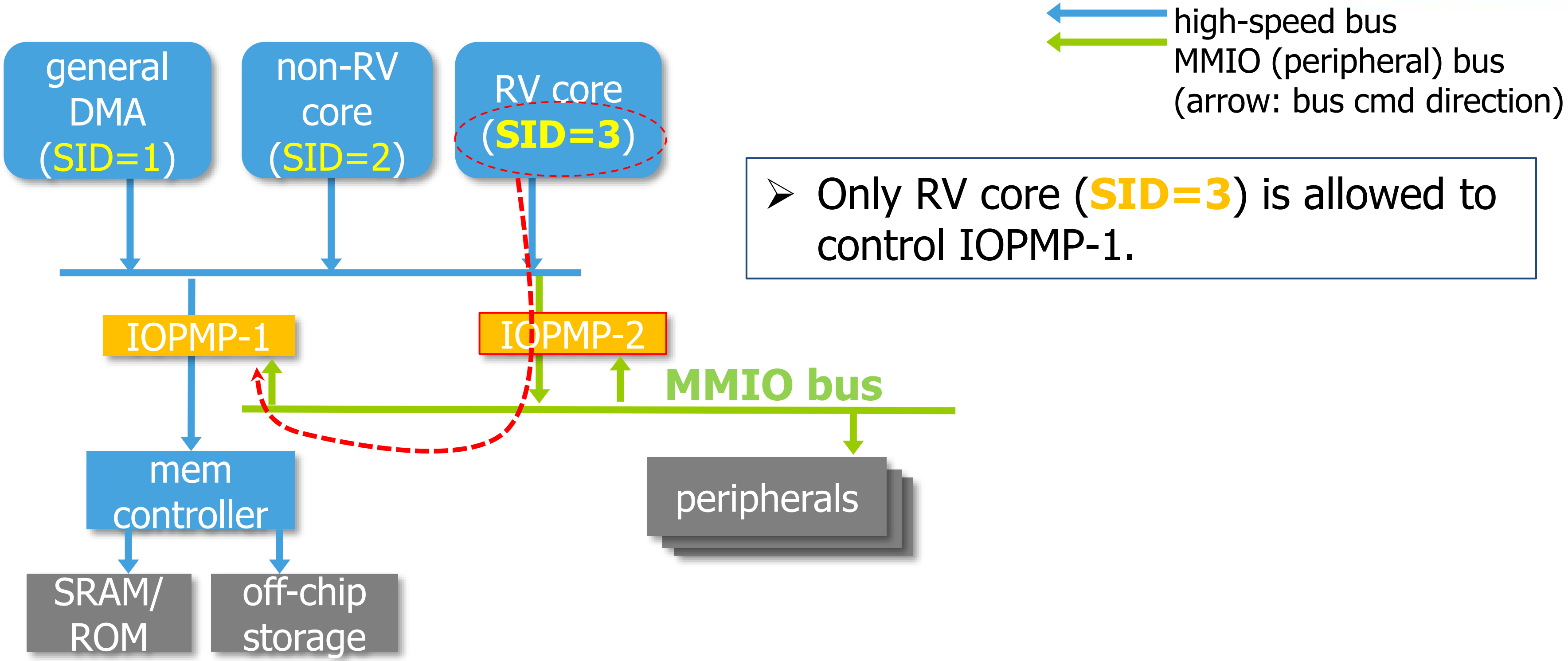


The control paths, but who is permitted to control IOPMPs?

The IOPMP Controls Who Can Control IOPMPs

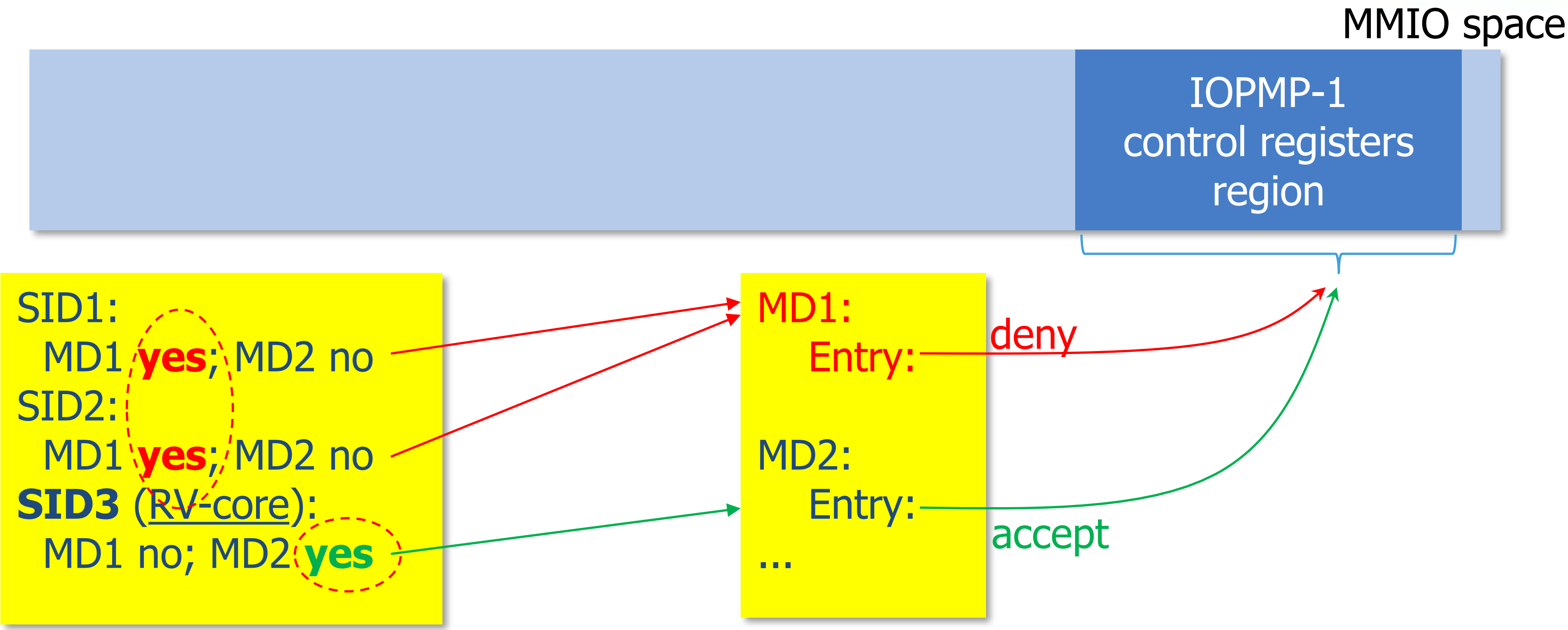
- The first requirement: control who can control IOPMPs:
  - Setting steps for the IOPMP of bus bridge to MMIO (e.g., IOPMP-2).
  - 1) Let MD 1 deny accesses to the control registers of the target IOPMP.
  - 2) Let MD 2 accept accesses to the same region.
  - 3) Let the permitted SID(s) associate with MD 2.
  - 4) Let the rest of SID(s) associate with MD 1.

An Example



- Only RV core (**SID=3**) is allowed to control IOPMP-1.

IOPMP-2 Controls Who Can Control IOPMP-1



MMIO space

IOPMP-1
control registers
region

SID1:
MD1 **yes**; MD2 no
SID2:
MD1 **yes**; MD2 no
SID3 (RV-core):
MD1 no; MD2 **yes**

MD1:
Entry: deny

MD2:
Entry: accept

...

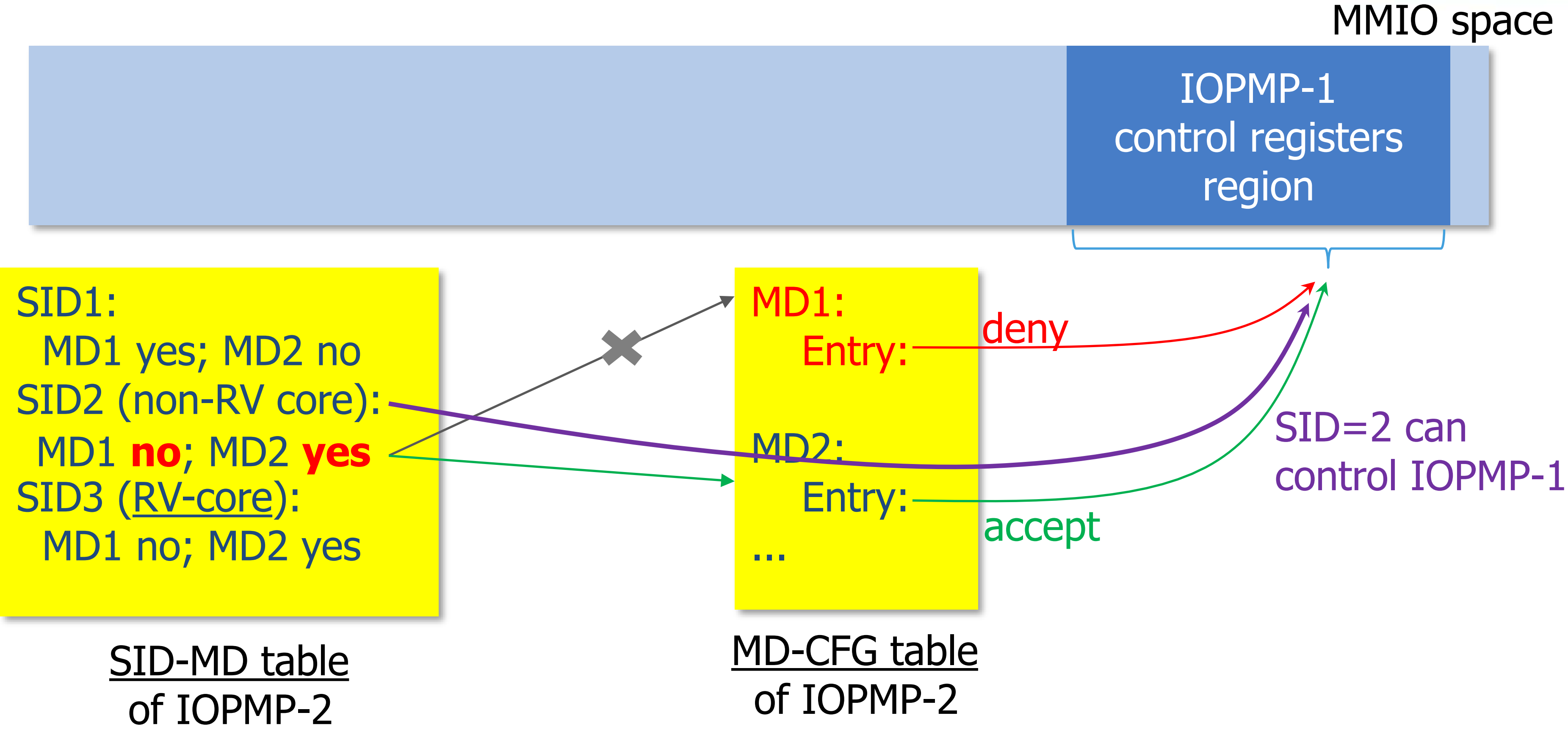
SID-MD table
of *IOPMP-2*

MD-CFG table
of *IOPMP-2*

What if IOPMP-2 is modified unwantedly?

- IOPMP-2 may be modified unwantedly due to
  - Bugs of the security monitor
  - The malicious code lures security monitor
- It could incidentally create a unwanted control path.

What if IOPMP-2 is modified unwantedly?



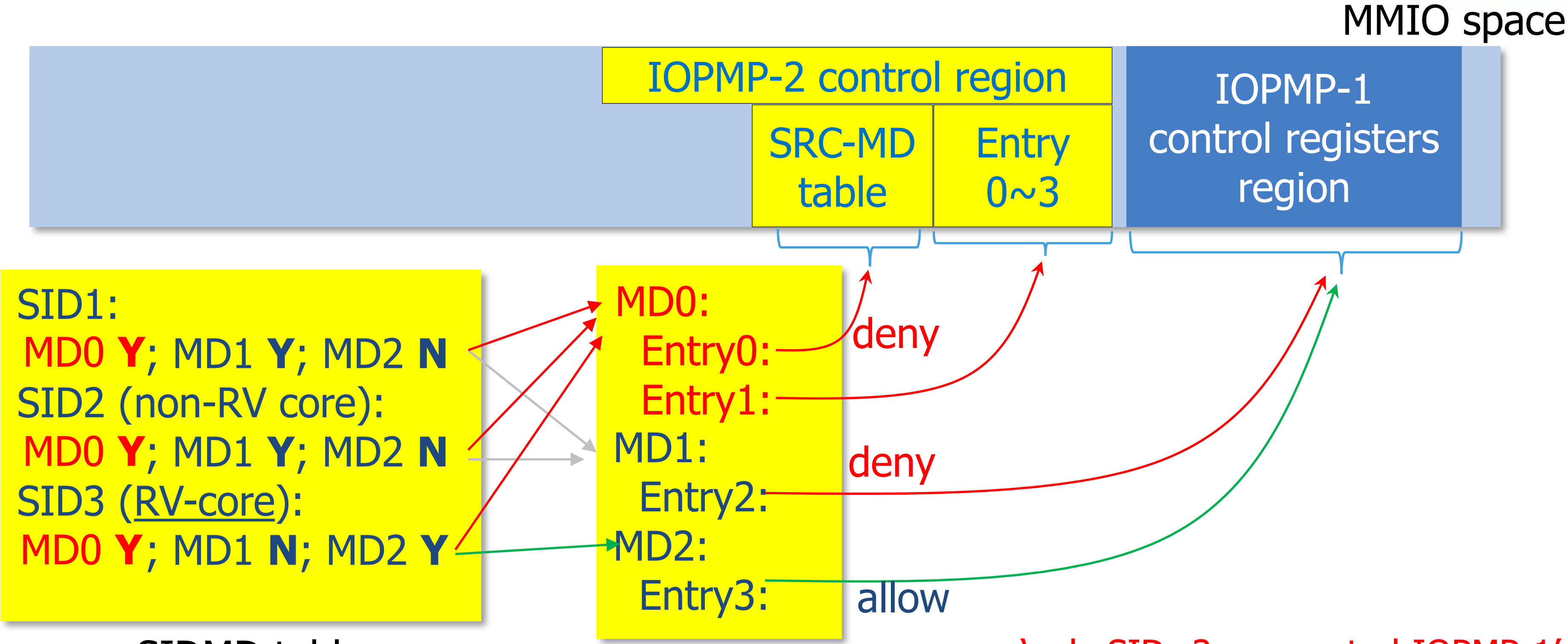
Lock The Control Path

- The second requirement: mitigate once trusted software is compromised
  - Here, the sensitive data is the above setting which enforces only the RISC-V core allowed to control IOPMP-1.
  - IOPMP-2 can lock itself by MD 0:
    1) Let MD 0 have two entries:

       Entry 0: deny accesses to the SRC-MD table of IOPMP-2.

       Entry 1: deny accesses to Entry 0, 1, 2 (of MD 1) and 3 (of MD 2).
    2) Fill up all other settings in the SRC-MD table of IOPMP-2.
    3) Let every SID associate with MD 0 in IOPMP-2.
  - One can extend it to lock more data.

IOPMP-2 Locks Settings of all IOPMPs



MMIO space

IOPMP-2 control region

SRC-MD table

Entry 0~3

IOPMP-1 control registers region

SID1:
MD0 **Y**; MD1 **Y**; MD2 **N**
SID2 (non-RV core):
MD0 **Y**; MD1 **Y**; MD2 **N**
SID3 (RV-core):
MD0 **Y**; MD1 **N**; MD2 **Y**

MD0:
Entry0:
Entry1:
MD1:
Entry2:
MD2:
Entry3:

SIDMD table
of IOPMP-2

MDCFG table
of IOPMP-2

'only SID=3 can control IOPMP-1'
has been locked!

Protect IOPMPs by an IOPMP

- IOPMPs themselves can construct a structure to satisfy the fundamental requirements.
  - The first requirement: control who can control IOPMP
  - The second requirement: lock IOPMP

- However, the methods is lack of flexibility.
  - To lock 'who can control IOPMP-1', we locked whole SRC-MD table.
  - The minimal grain of IOPMP checking is 4 bytes. It is too coarse to provide enough flexibility.

The limitation of Protect IOPMP by MD

- For example: we want to lock ‵*who can control IOPMP-1*’ but leave ‵*the other MDs’ association programmable*.’
- However, when using the above strategy, we have to enforce “every SID associates with MD 0.” By far, we have to lock the whole SRCMD table.

Protect IOPMP Settings

- Protecting IOPMP settings is to protect the following components:
  - ID Mapping: *SRCMD* Table
  - Memory Domain Configuration: *MDCFG* Table
  - IOPMP *entries*
- *SID* is an important part; however, it does not belong to the IOPMP spec.
  - SID protection will reply on the implementation or the other spec.
  - In many cases, hardwiring SID could be a good choice.

Per-MD Protection of The SRCMD Table

- Optional MDMSK: (per-MD locker)
  - If MDMSK.*MD*[*m*]=0, SRC*s*MD.MD[*m*] is programmable for all SID *s*.
  - MDMSK.*L*: a bit of sticky lock to the MDMSK.
- Lock the mapping of MD *m*.
  - S1: Initialize MD *m*.
  - S2: Set/clean SRC*s*MD.MD[*m*] for all SID *s*.
  - S3: Set MDMSK.MD[*m*]=1 // make SRC*s*MD.MD[*m*] read-only for all *s*.
  - S4: Lock MDMSK by setting MDMSK.L=1
- IOPMP without MDMSK
  - Wire MDMSK.L=1 and MDMSK.MD=0

Use Case of MDMSK

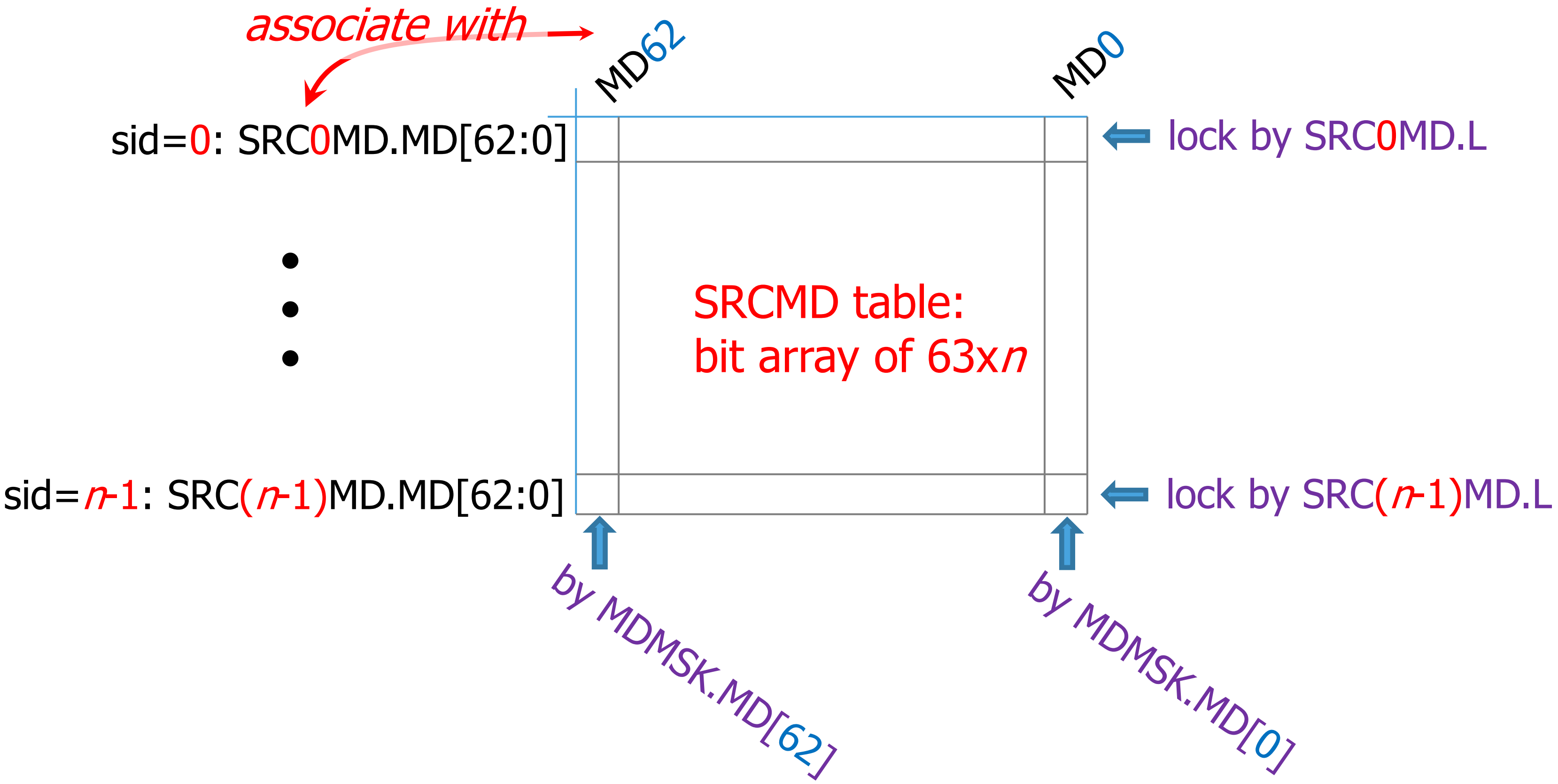
- The above example: we want to lock '*who can control IOPMP-1*' but leave '*the other MD's association programmable*.'
  1) put IOPMP-1 control registers in MD 0 <u>without write</u> permission.
  2) put IOPMP-1 control registers in MD 1 <u>with write</u> permission.
  3) let SID=3 associate with MD 1, and the other SIDs associate with MD 0.
  4) let MDMSK.MD=0b11 (MD 0 and 1) and MDMSK.L=1

- The SRCMD table are now partially locked; (only the mapping between SIDs to MD 0 and MD 1 is locked.)

Per-SID Protection of The SRCMD Table

- Optional SRC*s*MD.L: (per-SID)
  - A sticky lock to make the entry read-only until reset.
  - SRC*s*MD.L==1, the stickily lock of SRC*s*MD.
  - SRC*s*MD.L==0, SRC*s*MD.MD[*m*] is programmable as long as MDMSK.MD[*m*]=0.

The SRCMD table Protection (cont.)



associate with

MD62

MD0

sid=0: SRC0MD.MD[62:0]

sid=*n*-1: SRC(*n*-1)MD.MD[62:0]

SRCMD table:
bit array of 63x*n*

lock by SRC0MD.L

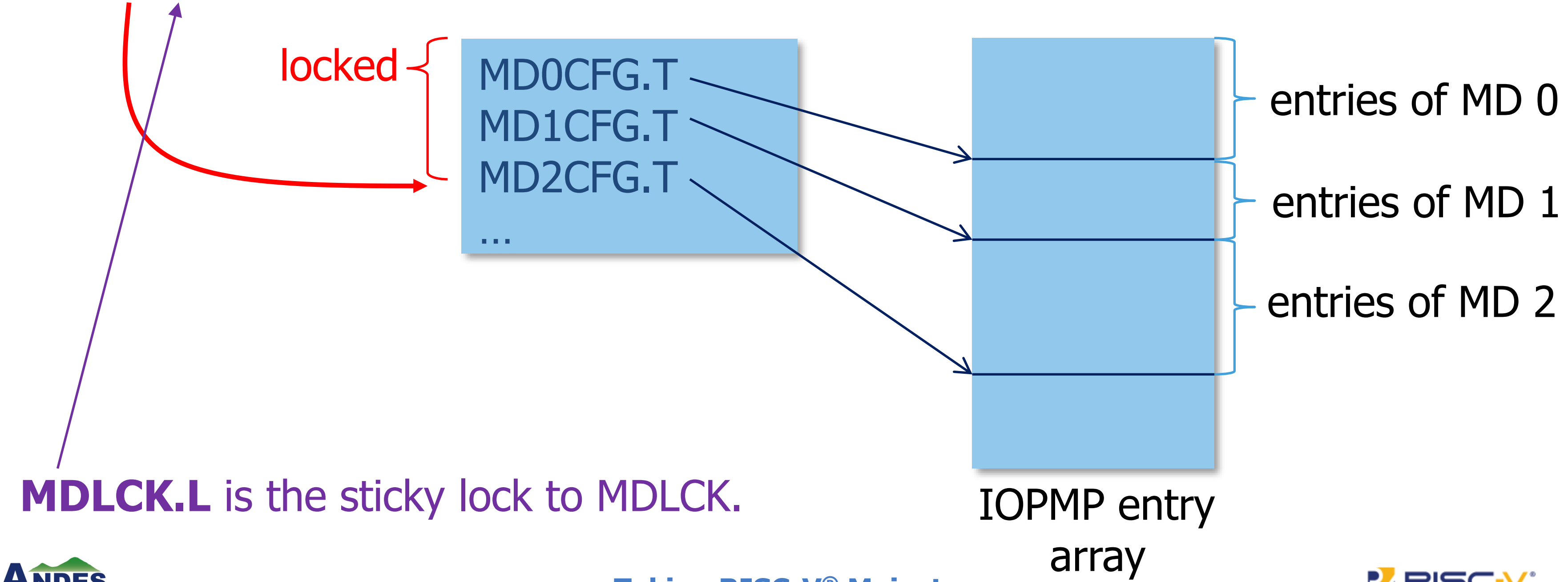
lock by SRC(*n*-1)MD.L

by MDMSK.MD[62]

by MDMSK.MD[0]

The MDCFG Table Protection

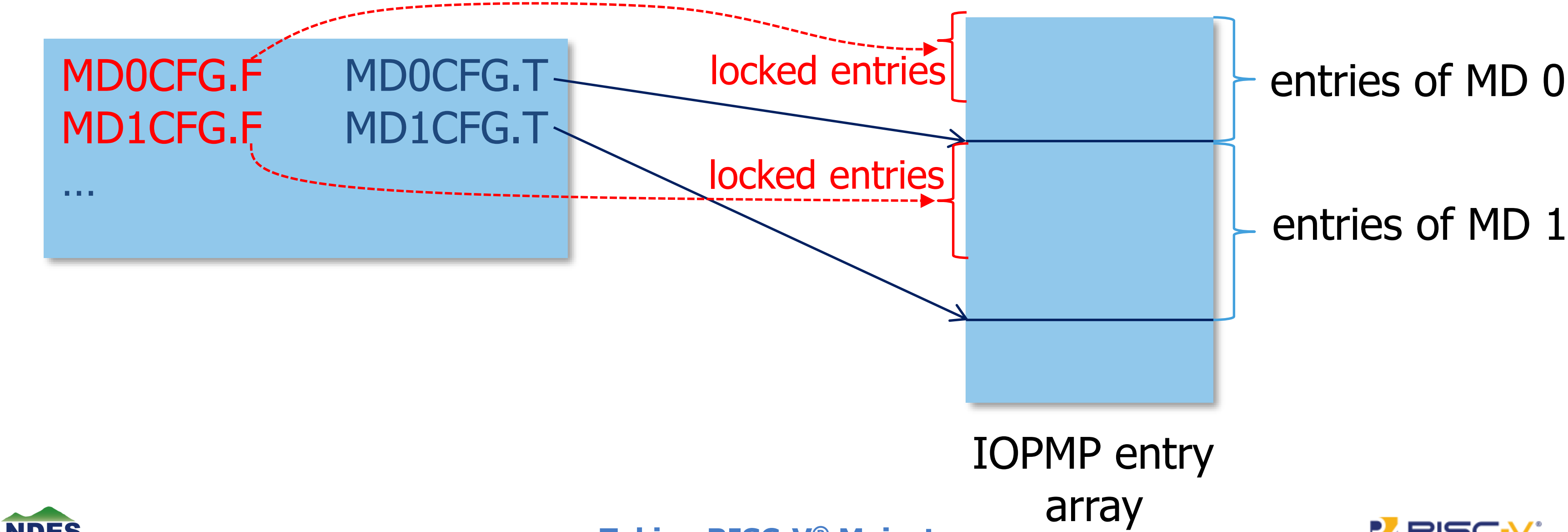
- MD*m*CFG is used to define the MD *m*:
  - MD*m*CFG.T: defines the top entry belongs to the MD.
- **MDLCK.F** defines the number of MD*m*CFG is locked.



MDLCK.L is the sticky lock to MDLCK.

IOPMP Entry Protection

- For MD *m*,
  - An optional field, MD*m*CFG.F, defines the number locked IOPMP entries belonging to the MD.
  - An optional bit, MD*m*CFG.L, is the sticky lock to MD*m*CFG.F.



Concluding Remarks

Concluding Remarks

- Explained why we need IOPMPs to secure a platform.
  - Software solution can't perfectly solve all kinds of cases.
- Glanced at what an IOPMP is.
  - Source-ID, entry, ports, matching rules and violation responses.
- Looked into the protection by a loop of denying
  - Use existed properties to protect IOPMP setting
  - Coarse-grained protection
- Introduced several advanced protection mechanisms
  - Optional features
  - Fine-grained protection

Thank You