High Area-Efficiency IOPMP Architecture for Large Systems

August 23, 2024

Dr. Paul Shan-Chyun Ku
Andes Technology

Speaker: 辜善群

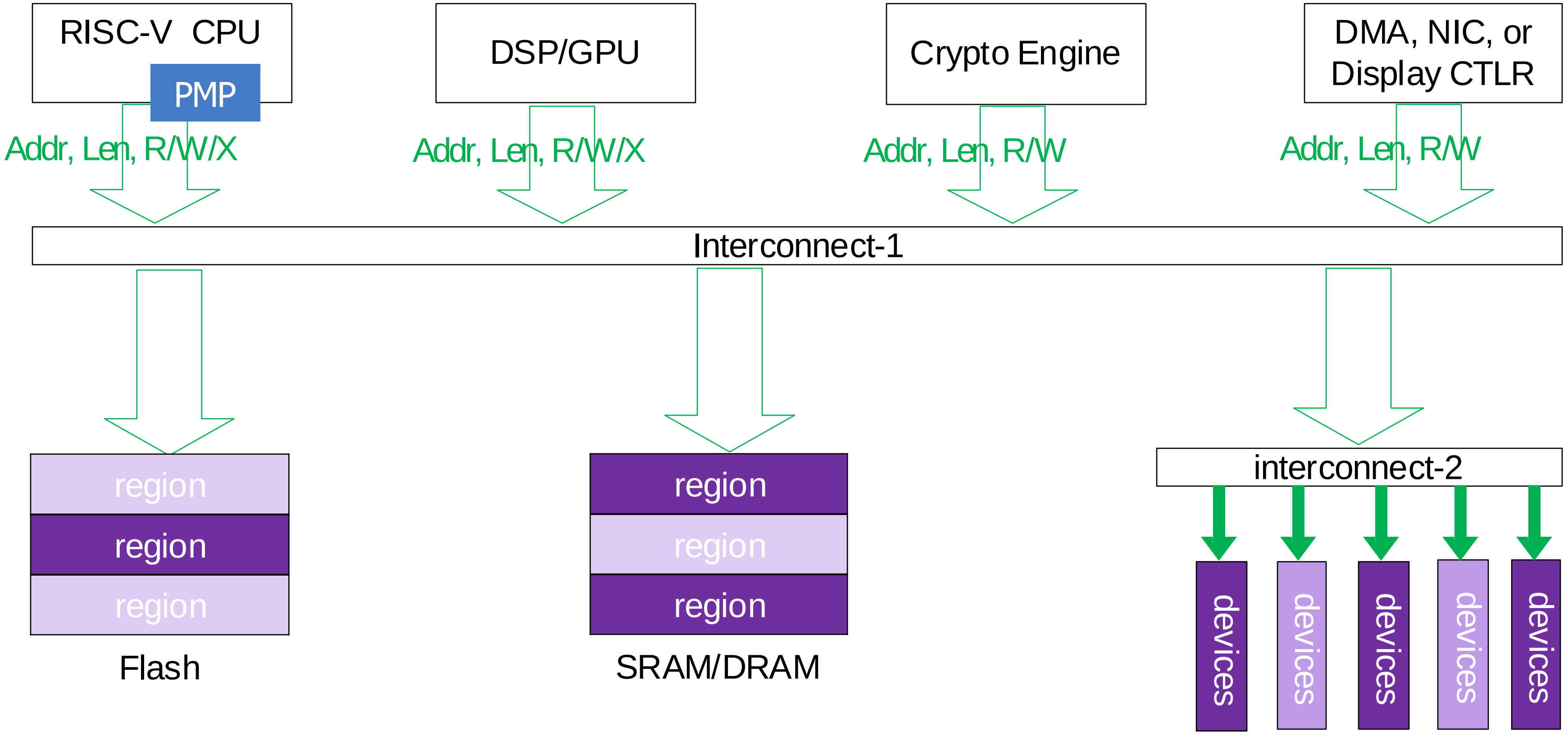
Experience:

- The Chair of IOPMP Task Group (2022-)
- The Vice-chair of TEE TG (2021-2022)
- Deputy Director, Andes Technology

Agenda

- A brief on the IOPMP
- The problem of IOPMP's scalability
- IOPMP non-priority rules and cacheability
- Area-effective architecture
- Experiment and remarks

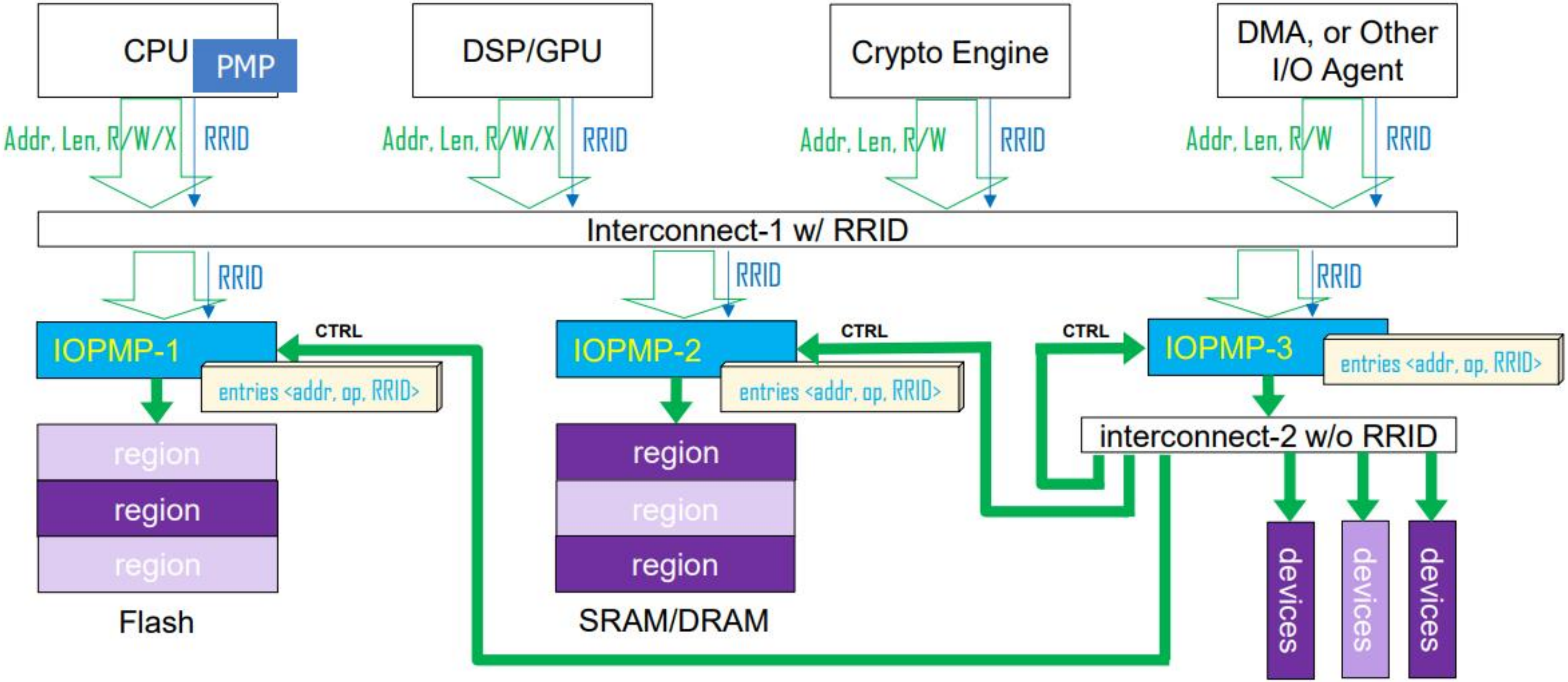
A typical platform



Vulnerability and threat

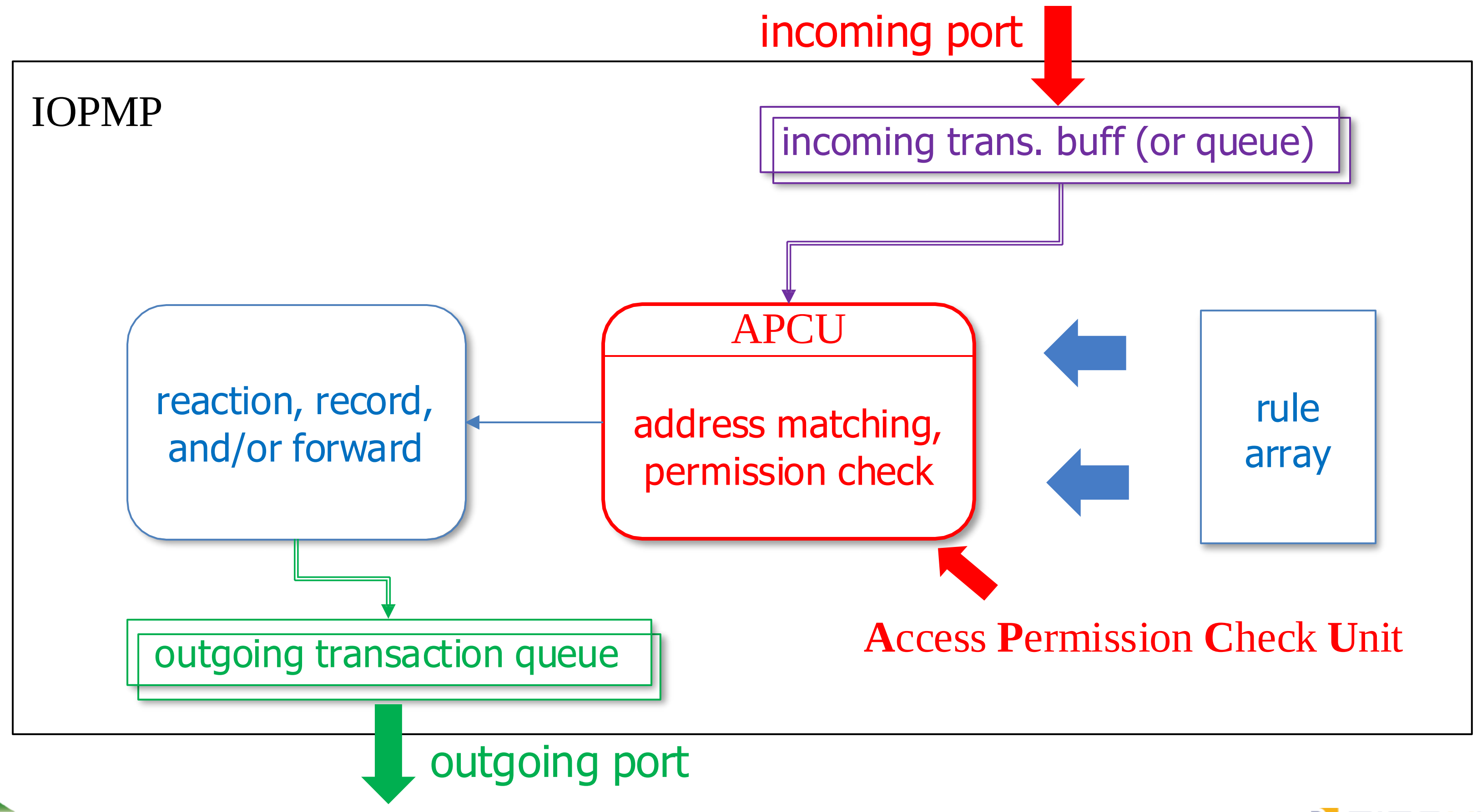
- RISC-V CPU’s transactions are checked by PMP/ePMP:
  - By (1) CPU mode, (2) memory region, and (3) operation
- The other I/O agents: DSP, GPU, DMA, NIC, LCDC…
  - Transactions from them are <u>NOT CHECKED</u> ➔ vulnerability!
  - A malicious SW that can control the I/O agents to access anywhere becomes the threats.
  - EX: an attack asks the I/O agent to read the sensitive asset without PMP/ePMP’s check and store it to its own legal space.
- IOPMP is the tool to mitigate the such a threat.
  - The IOPMP task group under RISC-V international is working on the architecture spec.

A platform with IOPMPs

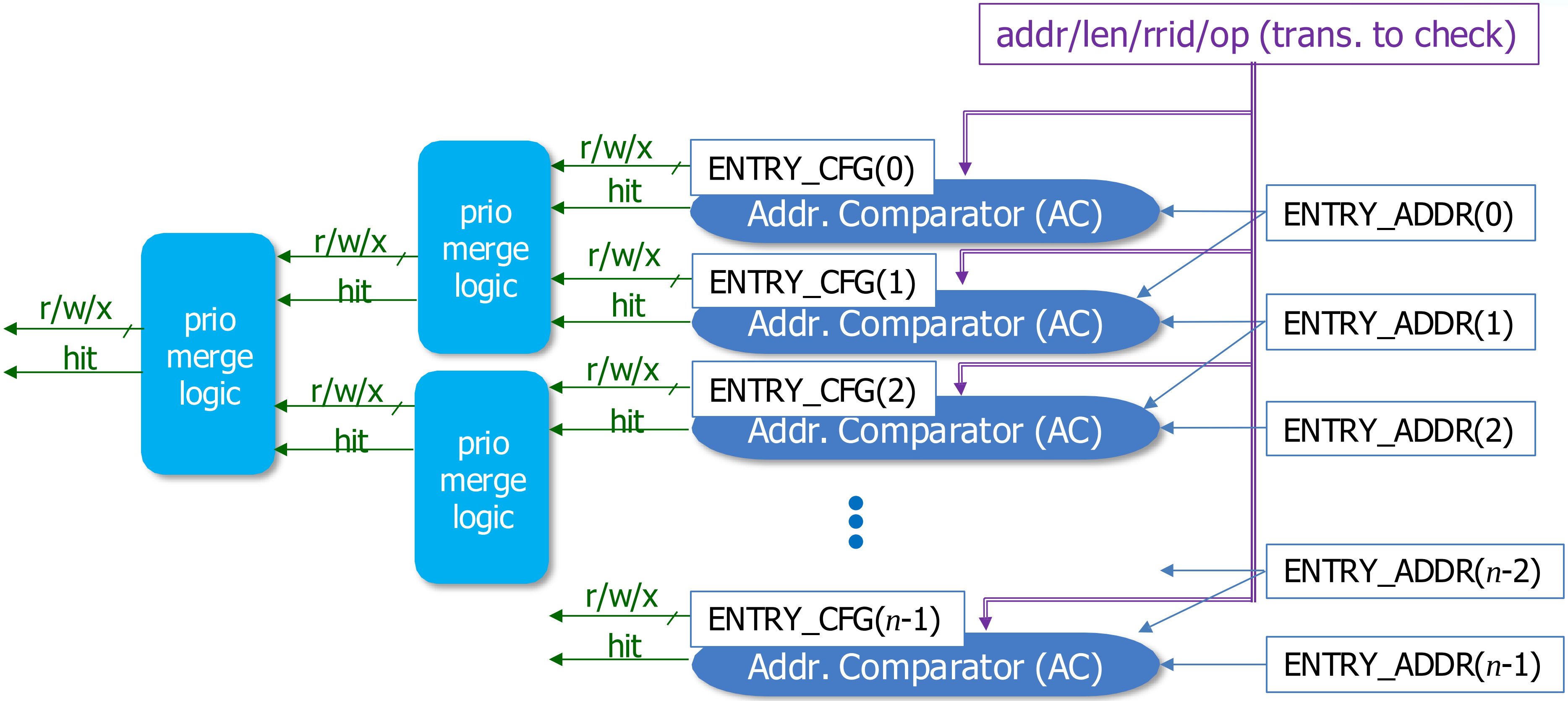


IOPMP's Implementation and Scalability

IOPMP block diagram



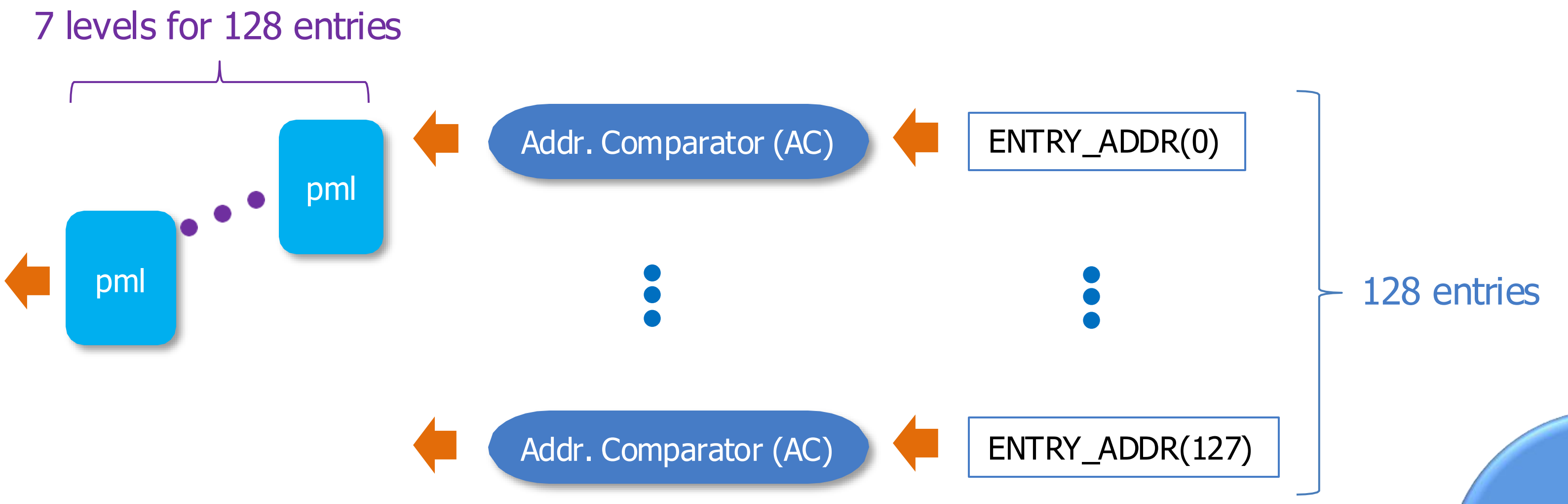
Conceptual structure of APCU



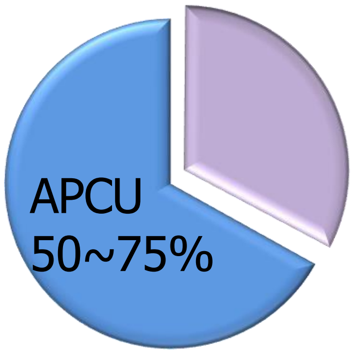
The need for rules increases quickly

- Along with SoCs becoming more and more complex, the need for the rules increases sharply:
  - NVidia shown a project using over 100 entries last year.
  - SJTU implemented an sIOPMP with couples of thousand entries.
  - Andes provides an IOPMP IP having about 500 entries.
- Scalability is important for now!

Ex. 128 entries: 128 ACs plus 7-level ML.



APCU's size increases linearly with the number of rules.



area of an IOPMP

Non-priority Rule and Cacheability

Priority and non-priority rules

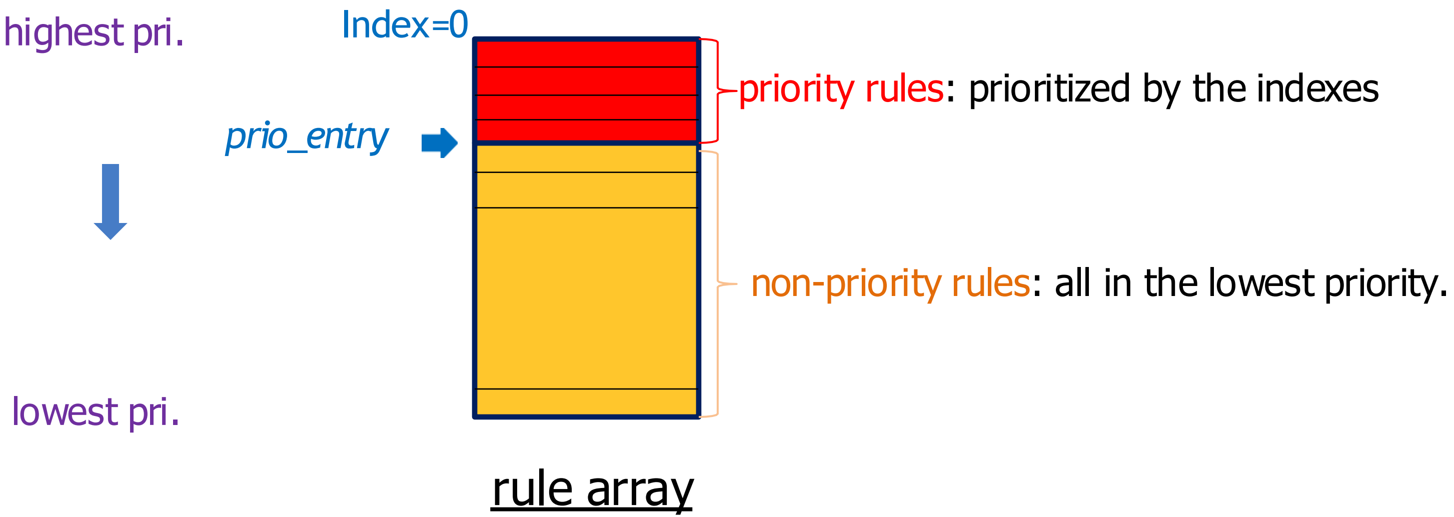
- Priority rules (ex, PMP/ePMP):
  - Higher secure: locked high-priority rules are overwritten by no means.
  - More flexible: no need to lock all rules in the boot time
  - Cost/power-consumption increases faster as #(rules) goes up.
- Non-priority rules:
  - Less secure: the isolation can be breached by a single malicious rule.
  - Cost/power-consumption increase slower as #(rules) goes up.
  - May need to lock all rules at the boot time → less flexible

Cacheability

- Caching priority rules is challenge:
  - A high priority rule not in the cache can overwrite the result from the cached but lower priority rules.
  - In both ways: permitted and unpermitted.
- Non-priority rules are cacheable:
  - As long as a rule grants permission, no other rules can revoke it.
  - Caching the rule granting permission is workable.
  - But not vice versa: if a transaction does NOT get the permission from cached rules, all non-priority rules should be scanned. ➔ cache miss!

Support priority and non-priority rules

- Register *prio_entry*, indicates the number of priority rules.

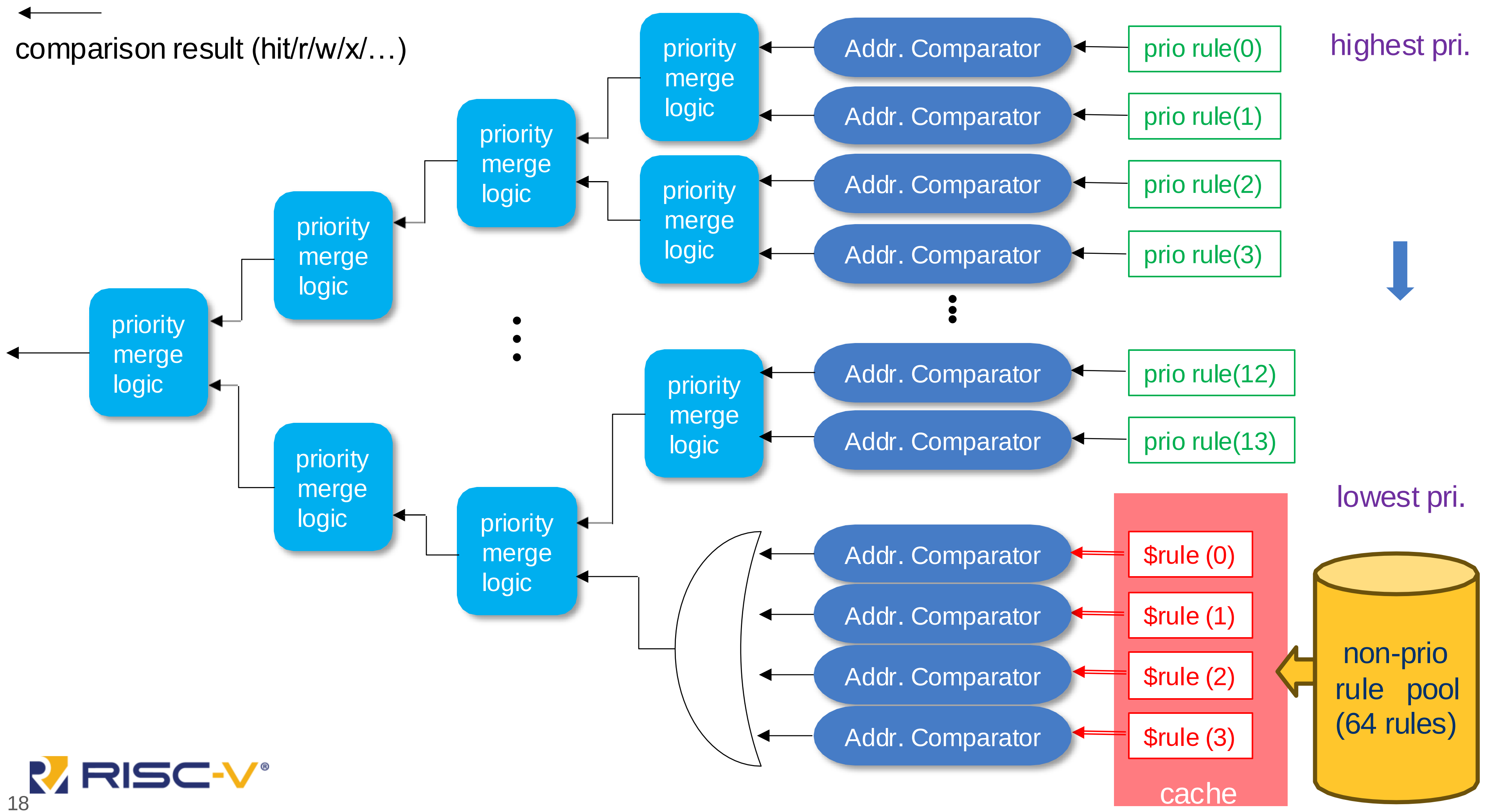


Area-effective Architecture of IOPMP

Combining priority and non-priority rules

- A typical system has
  - A few number of sensitive data: ex. anti-roll back counter, device keys, private keys, secure monitor's data/code, etc
  - ➢ A <u>few number of priority</u> rules with higher priority.

  - A large number of: execution environments' data/code
  - ➢ A <u>large number of non-priority</u> rules with lower priority.



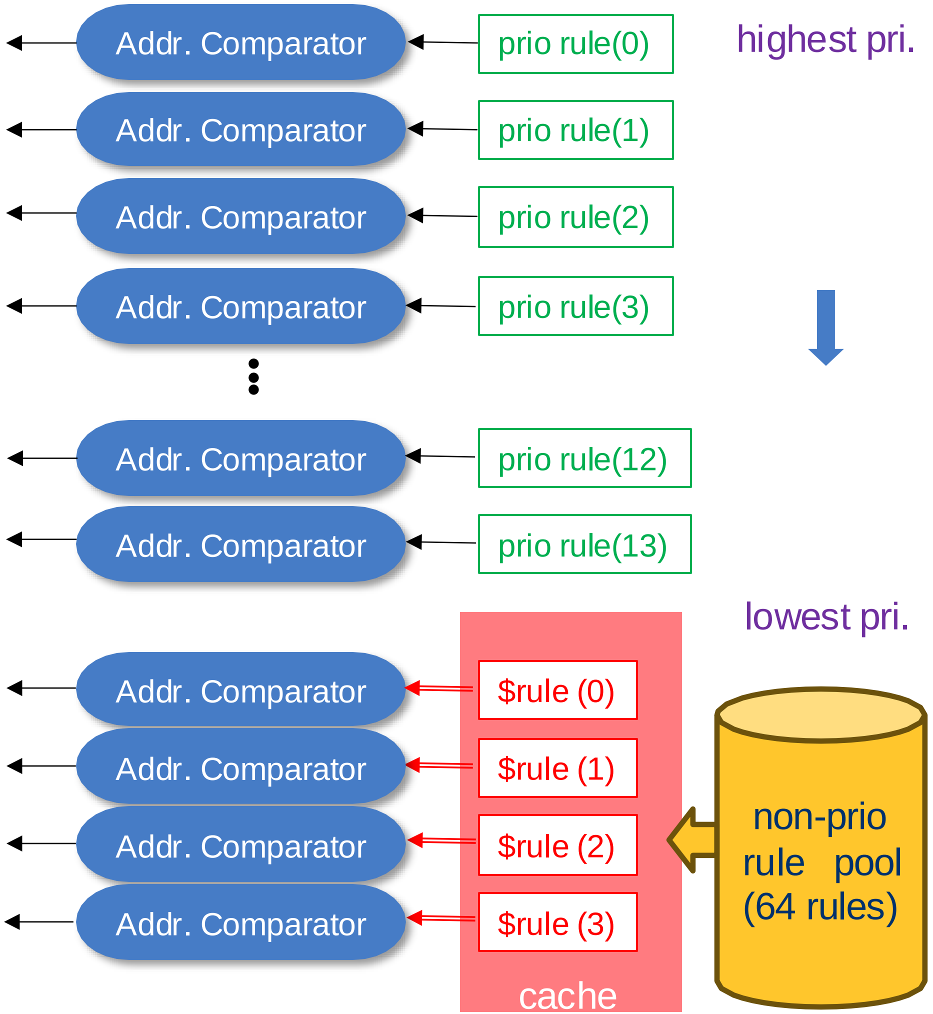
comparison result (hit/r/w/x/…)
priority merge logic
Addr. Comparator
prio rule(0)
prio rule(1)
prio rule(2)
prio rule(3)
prio rule(12)
prio rule(13)
highest pri.
lowest pri.
$rule (0)
$rule (1)
$rule (2)
$rule (3)
cache
non-prio rule pool (64 rules)

Combining priority and non-priority rules

- IOPMP' configuration:
  - Priority rules: 14
  - Non-priority rules: 64
- APCU's resource:
  - Address comparators: 18 // 78 ACs if using only priority rules
  - Priority merge logics: ~17
  - Cache size: 4
- It creates a possible to use SRAM to store non-priority rules when needed.
  - Access rules only when cache miss
  - Scanning all rules in multiple cycle is acceptable

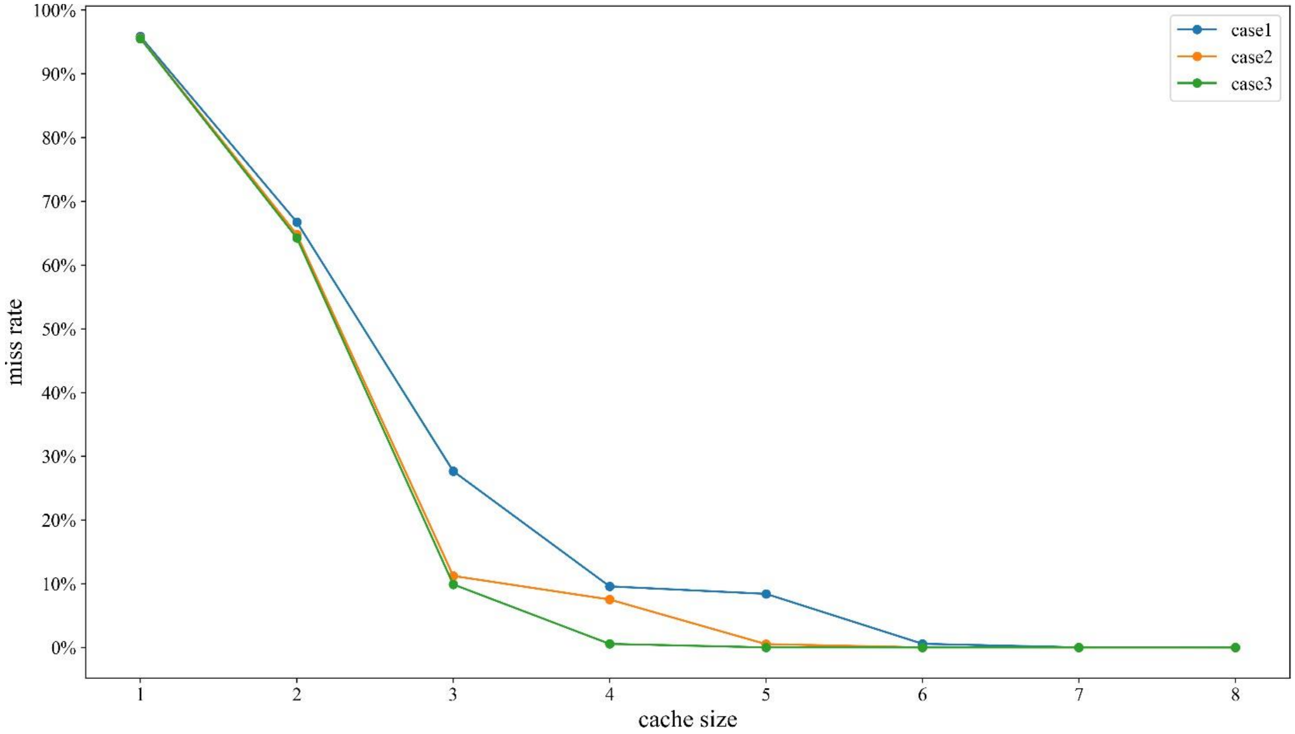
Cache Performance

- Cache miss analysis:
  - The first hit
  - Illegal access caught
  - Cache contention
- Locality can be very good:
  - Pick a good cache size, equals to the maximum number of concurrent access streams



Cache miss rates

720P video playing on a setup box with <u>4 ~ 7 concurrent access streams</u>.



7 streams: 720P w/ decryption + network-upload

6 streams: 720P w/ decryption

4 streams: 720P

Linux as REE on QEMU

Remarks

- By taking advantage of priority rules and non-priority rules, without sacrificing security, we can achieve good scalability, maintain a certain degree of flexibility, and minimize the gate counts.
- With a proper setting, the cache mechanism delivers an excellent amortized throughput.

Thank You

RISC-V®