Open-source RISC-V Input/Output Physical Memory Protection (IOPMP) IP

Luís Cunha, Francisco Marques, Manuel Rodriguez, Tiago Gomes, Bruno Sá, Sandro Pinto

RISC-V

RISC-V Summit Europe 24 @Munich

FCT Fundação para a Ciência e a Tecnologia

CENTROALGORITMI

UNIVERSIDADE DO MINHO

TII Technology Innovation Institute

ZERO DAY LABS

Agenda

01 Introduction
Memory Protection Mechanisms & Specification Overview

02 Implementation
Overview

03 Evaluation
Functional Validation & Hardware Results

Introduction

01

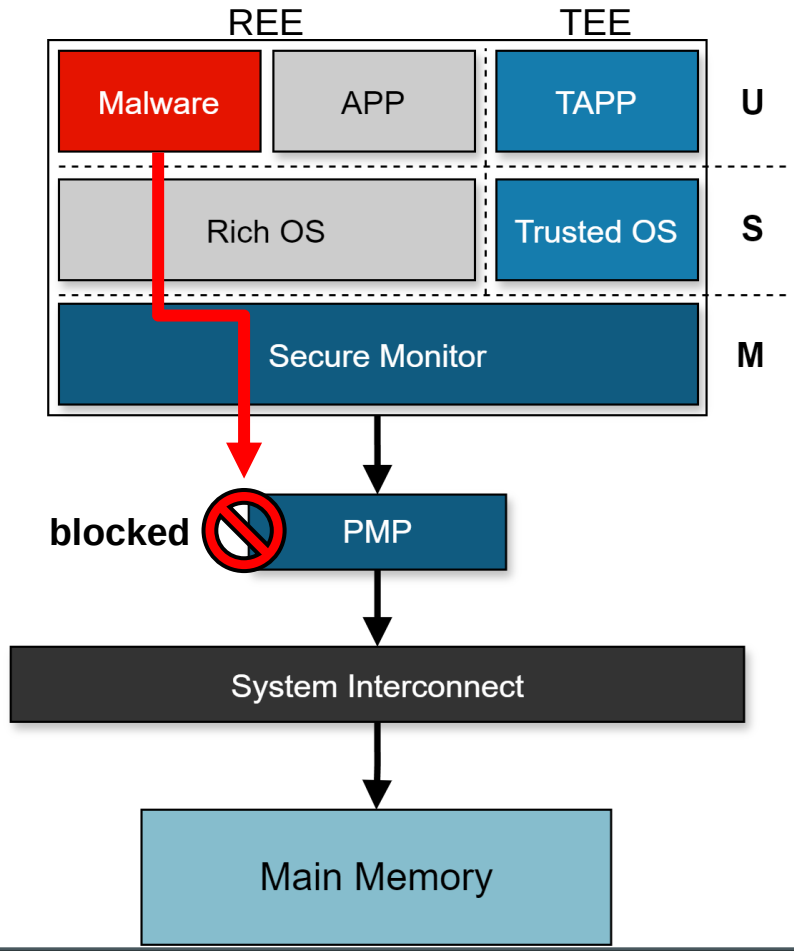
Memory Protection

Memory protection is a recognized security capability

MAIN PURPOSE:

- Prevent a process from accessing memory that has not been allocated to it

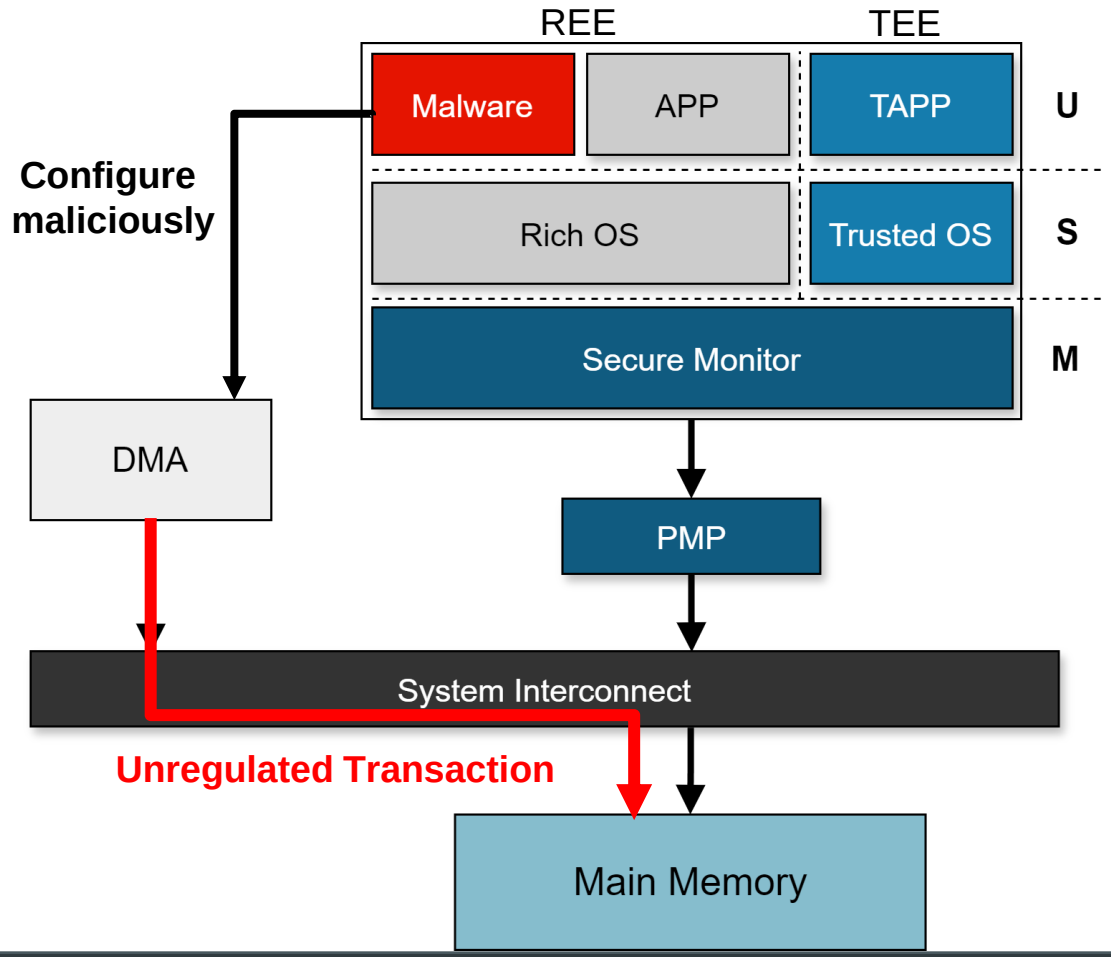
- The **Physical Memory Protection (PMP)** (part of the privileged specification) enforces memory protection at the hart level



Memory Protection

Memory protection is a recognized security capability

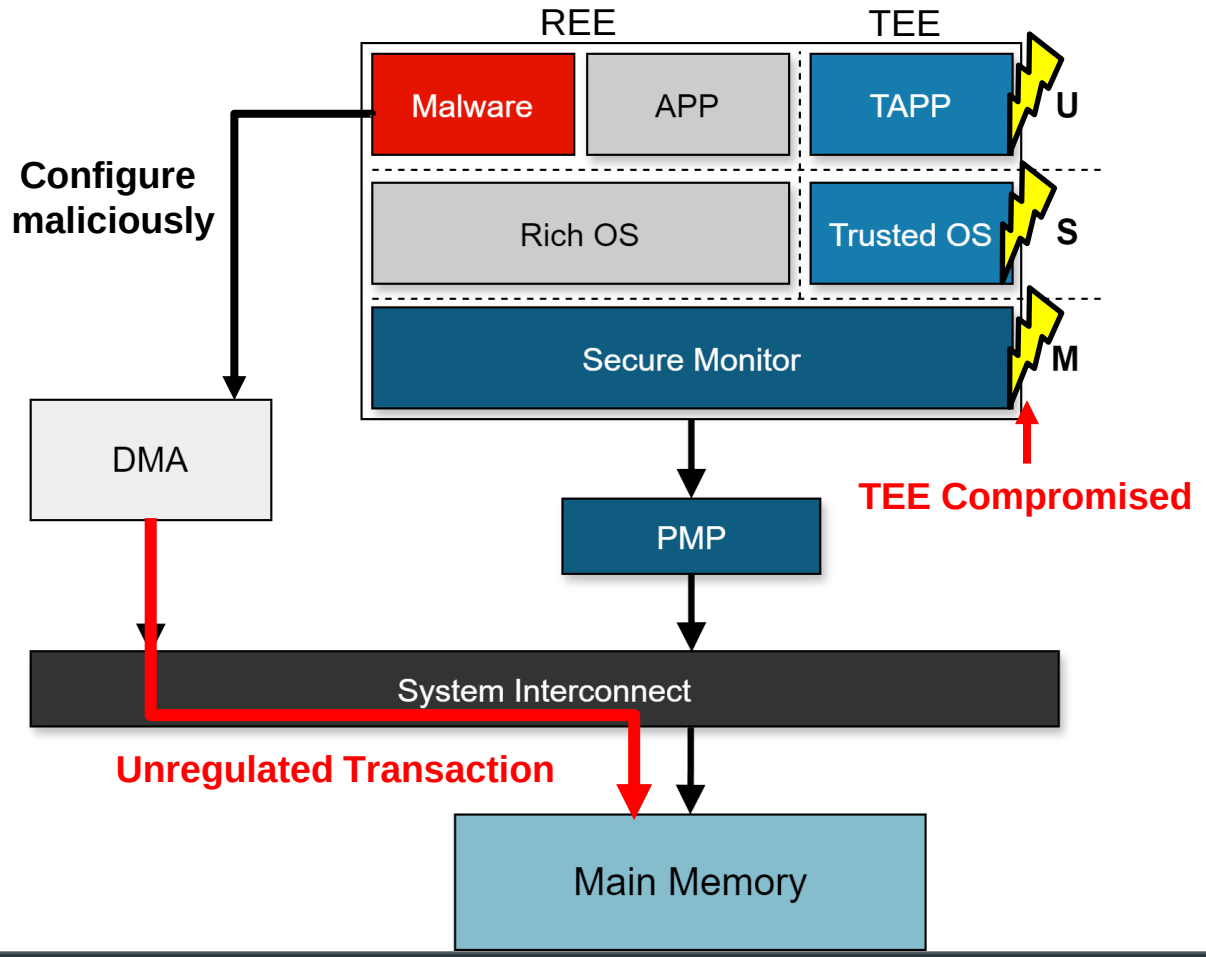
- The PMP **ONLY** enforces memory protection at the hart level
- Other devices are **unsupervised**
- Unsupervised devices can potentially compromise overall System-on-Chips



Memory Protection

Memory protection is a recognized security capability

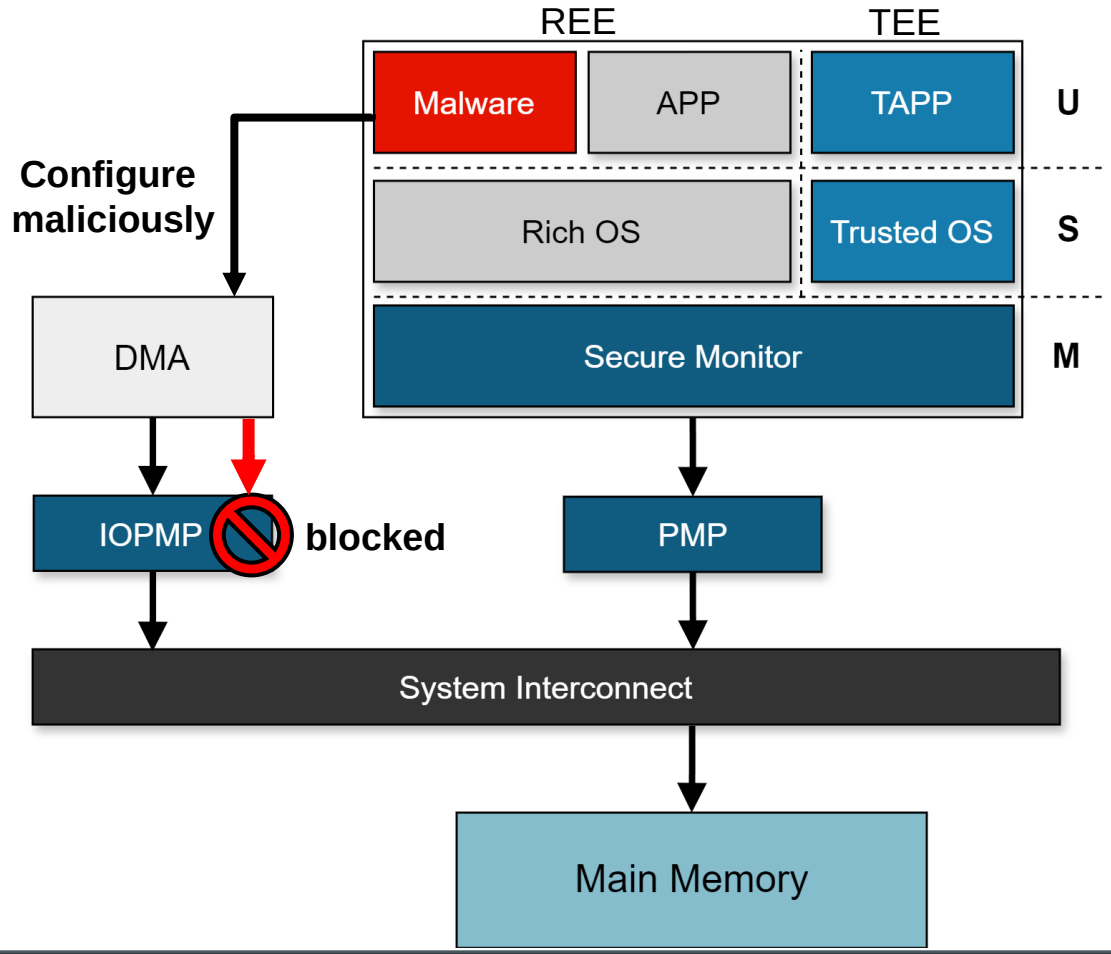
- The PMP **ONLY** enforces memory protection at the hart level
- Other devices are **unsupervised**
- Unsupervised devices can potentially compromise overall System-on-Chips



Memory Protection

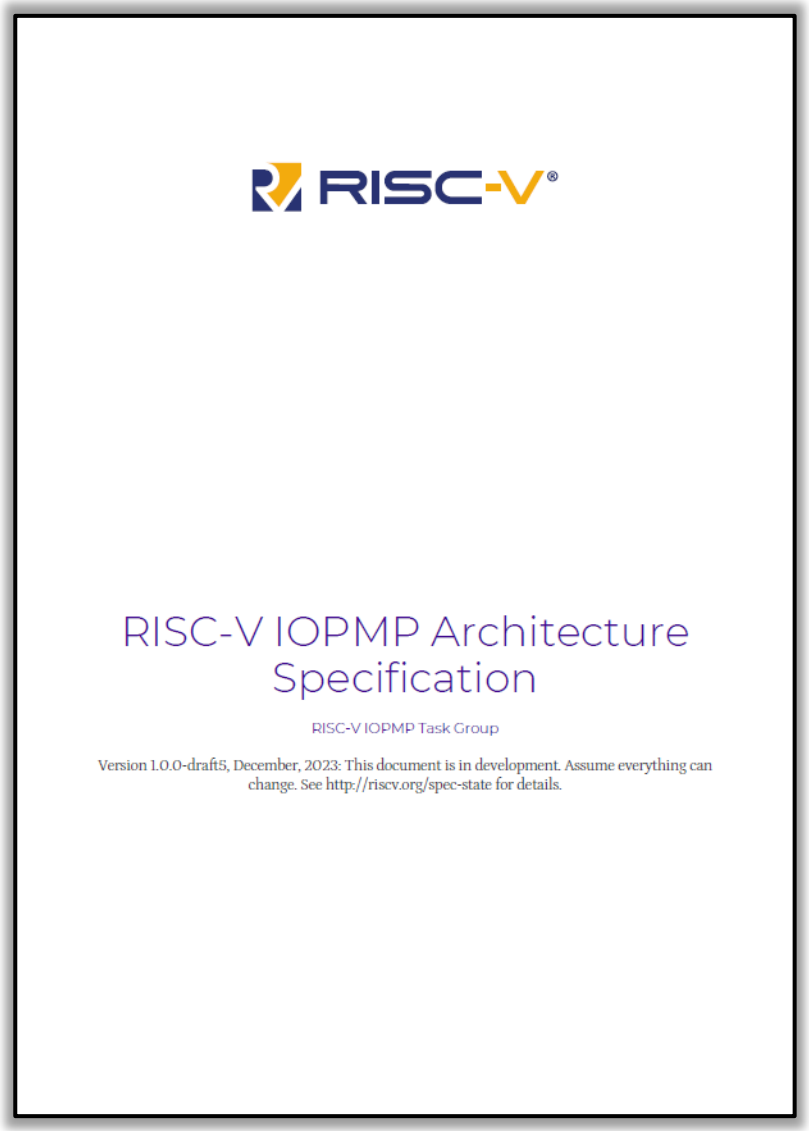
Memory protection is a recognized security capability

- The RISC-V community has been working towards the specification of the **Input/Output Physical Memory Protection (IOPMP)**
- **Mediate and manage** device accesses to memory by performing permission checks



IOPMP Specification

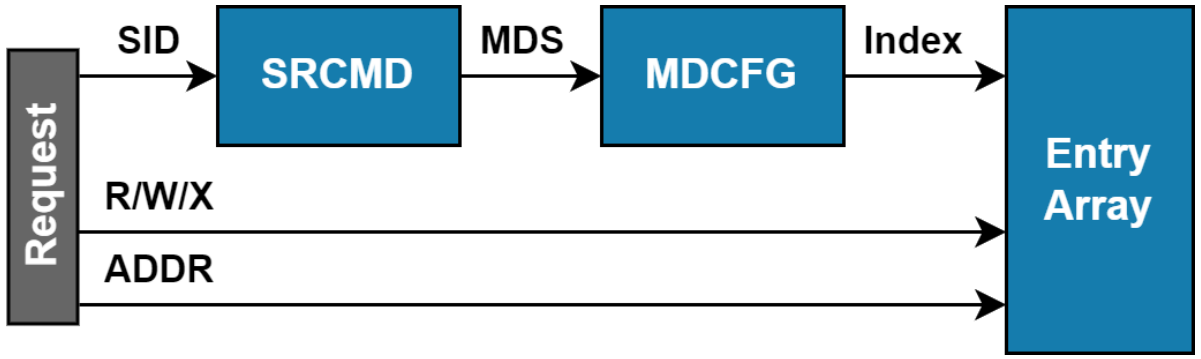
- **Mediate bus access from devices**
- **Multiple device Support**
- **5 Operation Models:**
  - Full
  - Rapid-K
  - Dynamic-K
  - Isolation
  - Compact-K
- **Priority and non-priority rules**



IOPMP Specification

Three structures to assess the validity of a transaction:

- **Entry** – defines a physical address range and a set of rules
- **Memory Domain** – Used to define groups of entries
- **Source Identifier** – Identifies the device making the transaction request (recently changed to RRID)

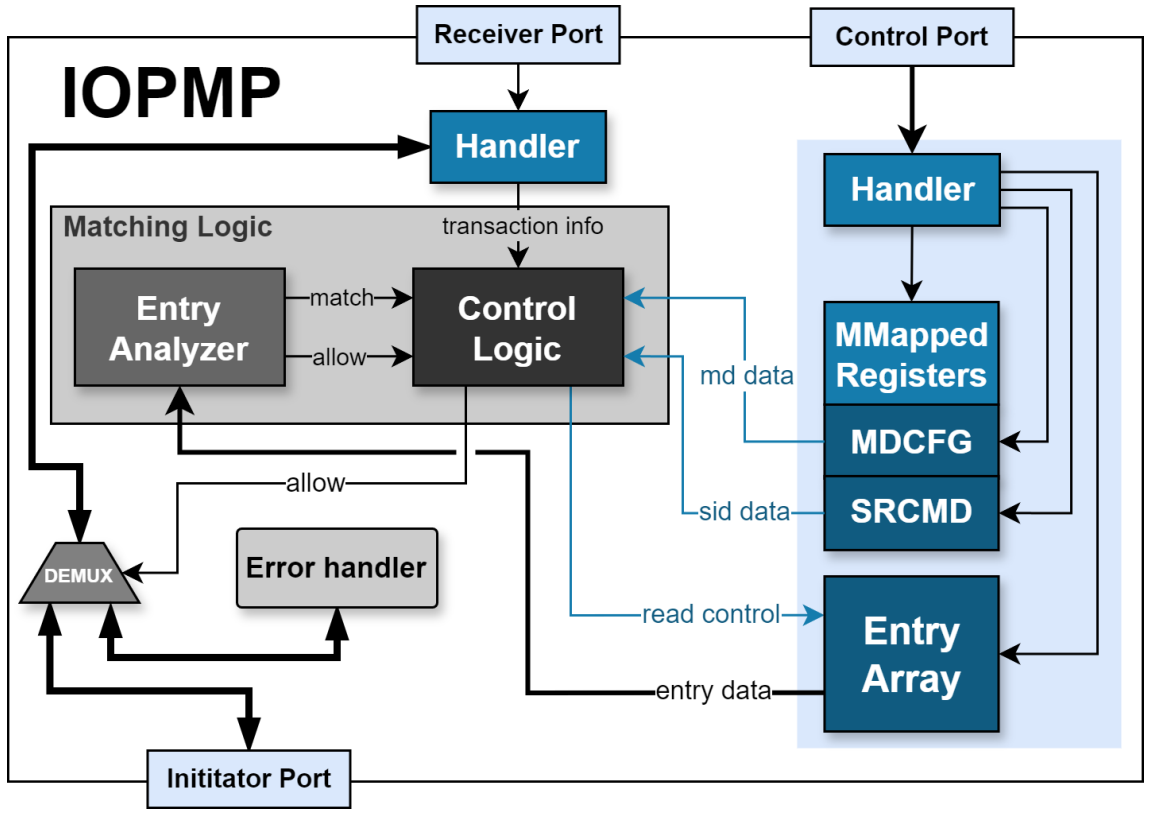


Implementation

02

IP Overview

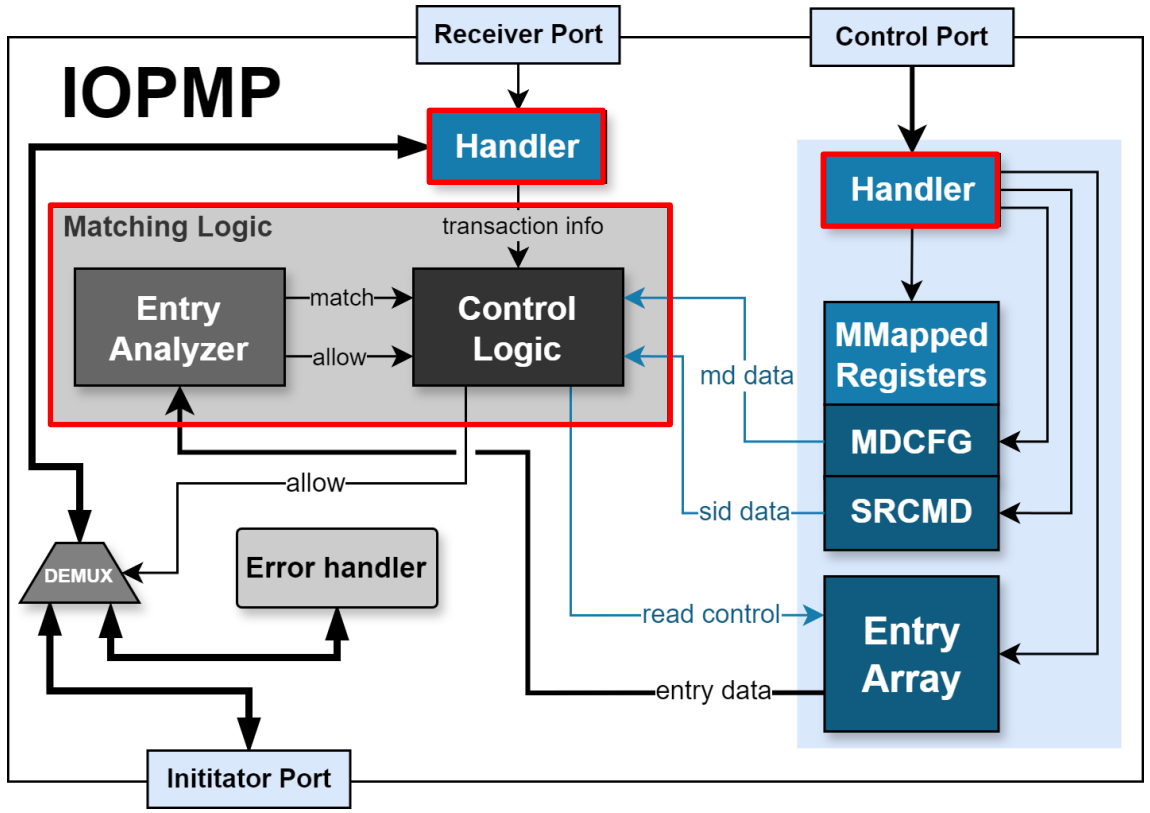
Feature	Notes
Model	Full Model
Configuration Protection	Only mandatory implemented
Programming Protection	Registers not implemented but the IP is stalled while configuring entries
Error Reporting	Only mandatory implemented
TOR Support	Implemented
Programmable Priority Entries	Implemented
Source Enforcement	Implemented



IP Microarchitecture

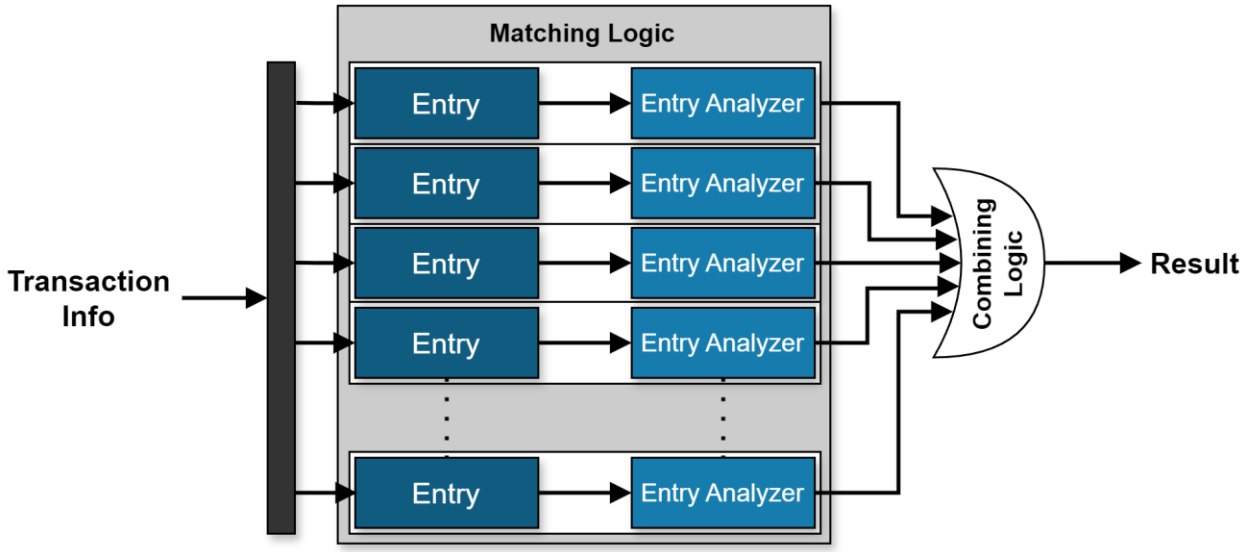
- The **Communication Handler** extracts transaction's information
  - SID
  - Base Address
  - Transaction's Length
  - Transaction Type (R,W,X)

- The **Matching Logic** calculates the corresponding Entry indexes and validates the transaction



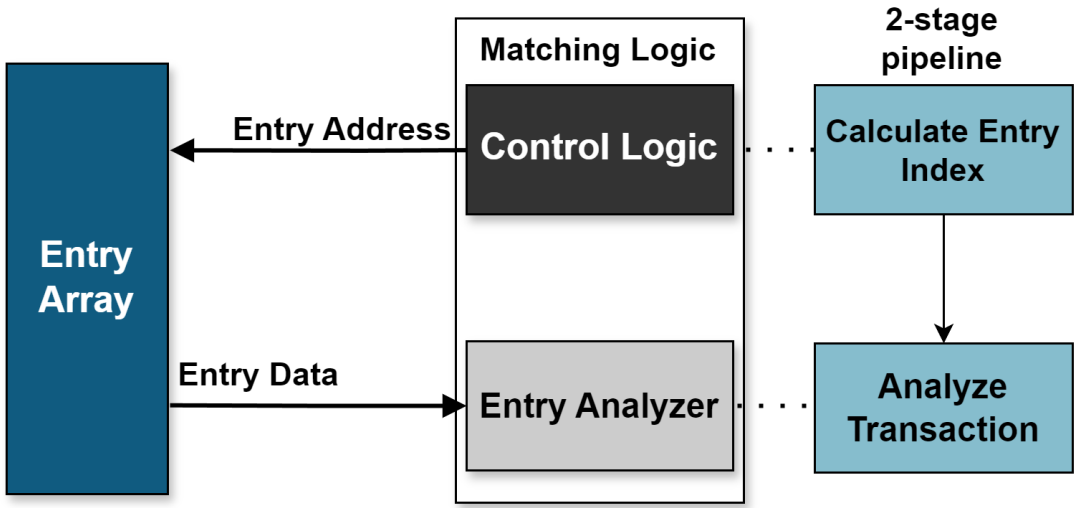
Matching Logic

Parallel Approach



- + Performance
- - Scalability

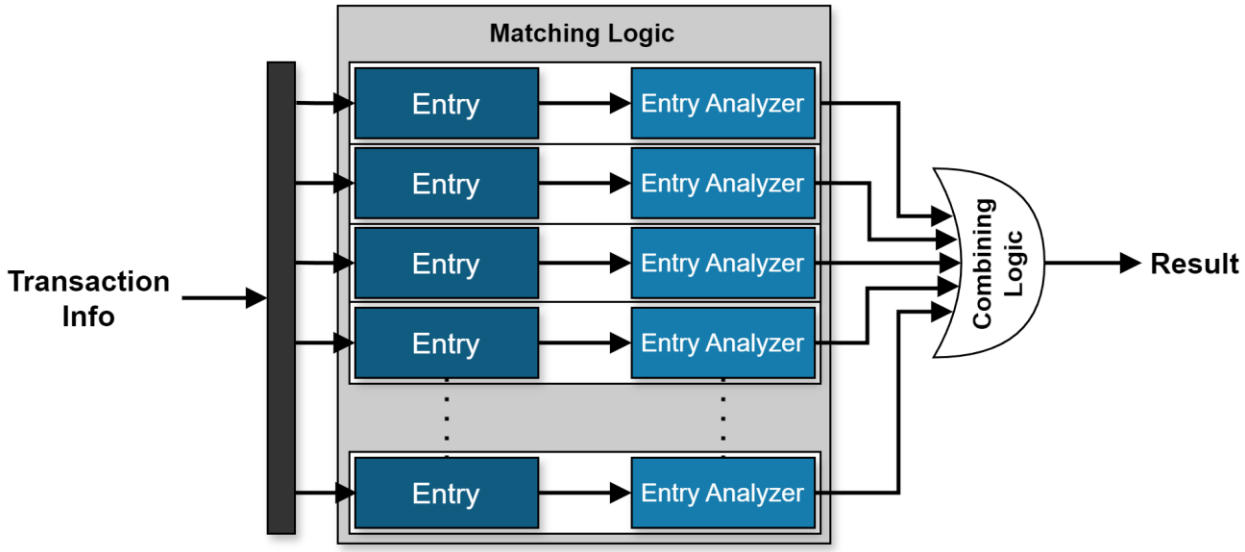
Sequential Approach



- - Performance
- + Scalability
- + Modularity

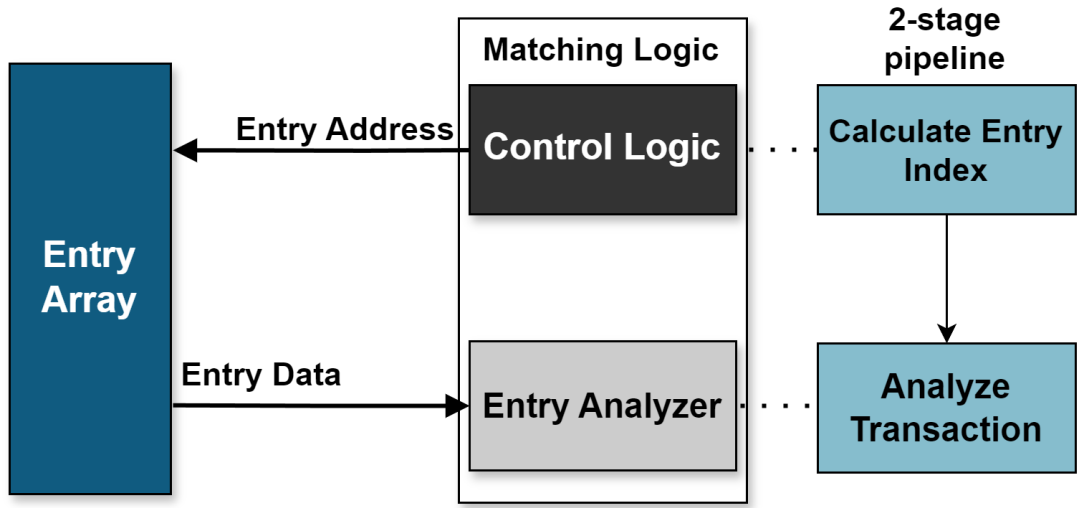
Matching Logic

Parallel Approach



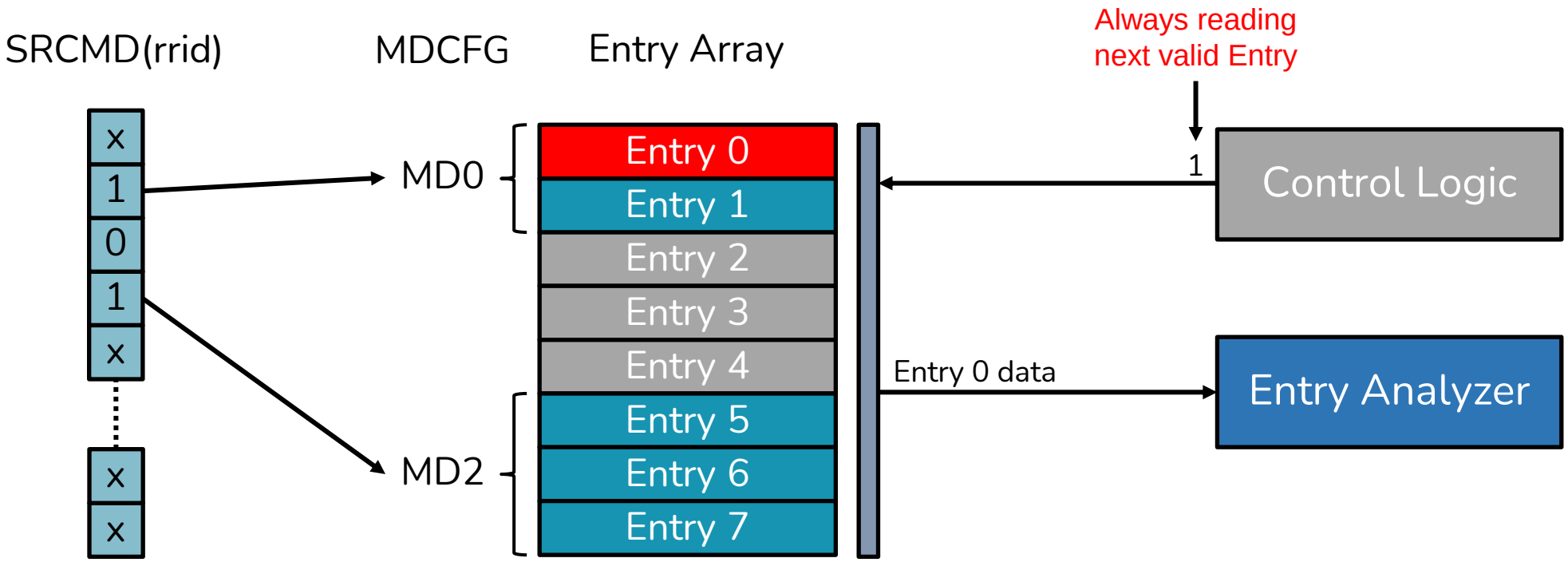
- + Performance
- - Scalability

Sequential Approach

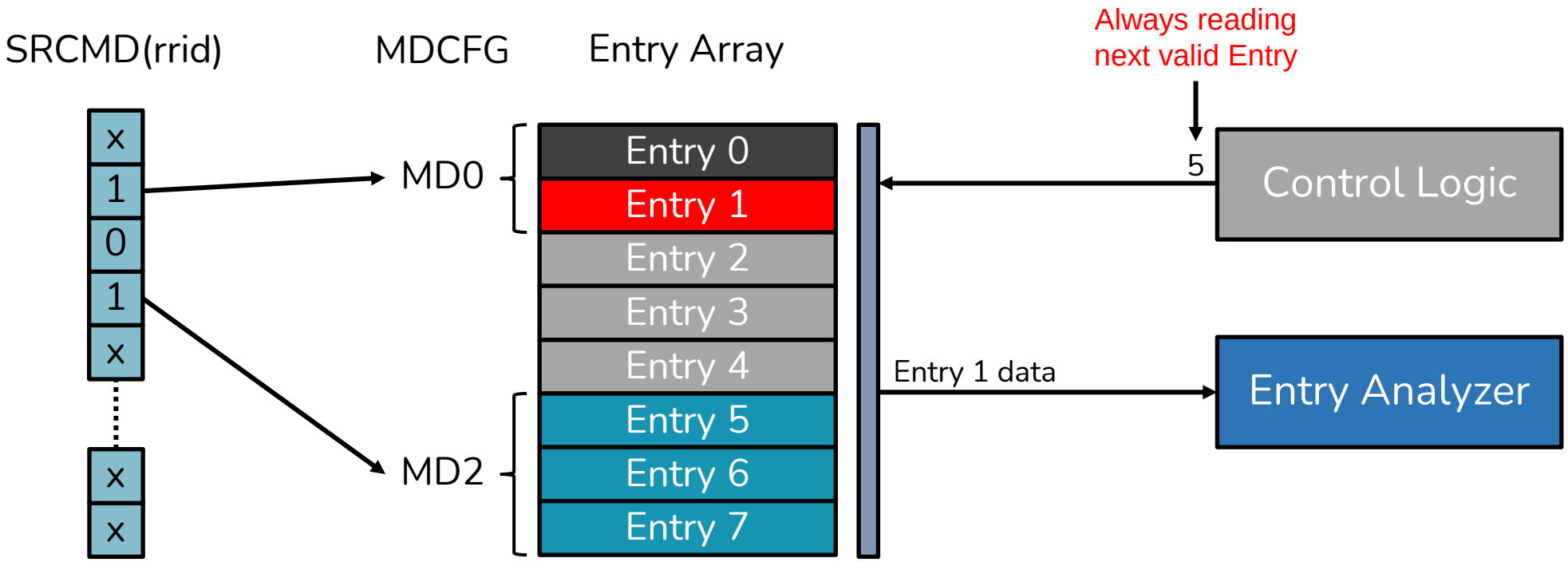


- - Performance
- + Scalability
- + Modularity

Entry Navigation

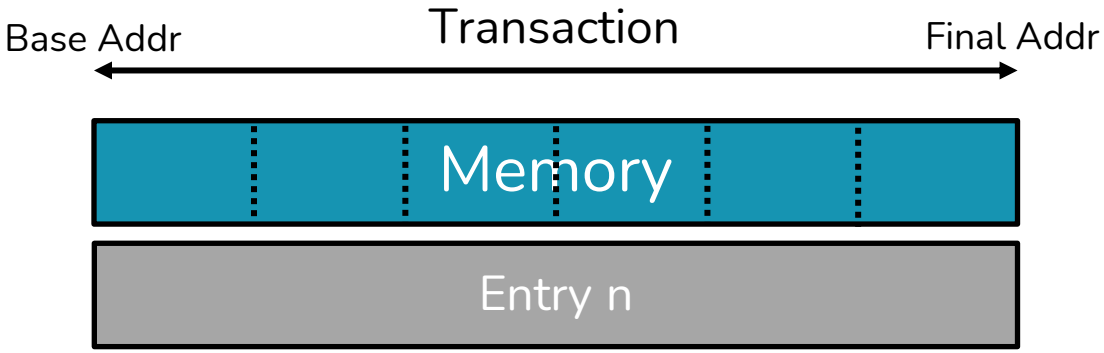


Entry Navigation



Transaction Validation

1. Entry Analyzer detects a match with Entry n with the base address of the transaction
2. Entry Analyzer checks if the entire length of the transaction is validated by Entry n
3. Entry Analyzer checks the permissions
4. As Entry n refers to the entire memory region of the transaction, the Entry analyzer signals a **match and allow**



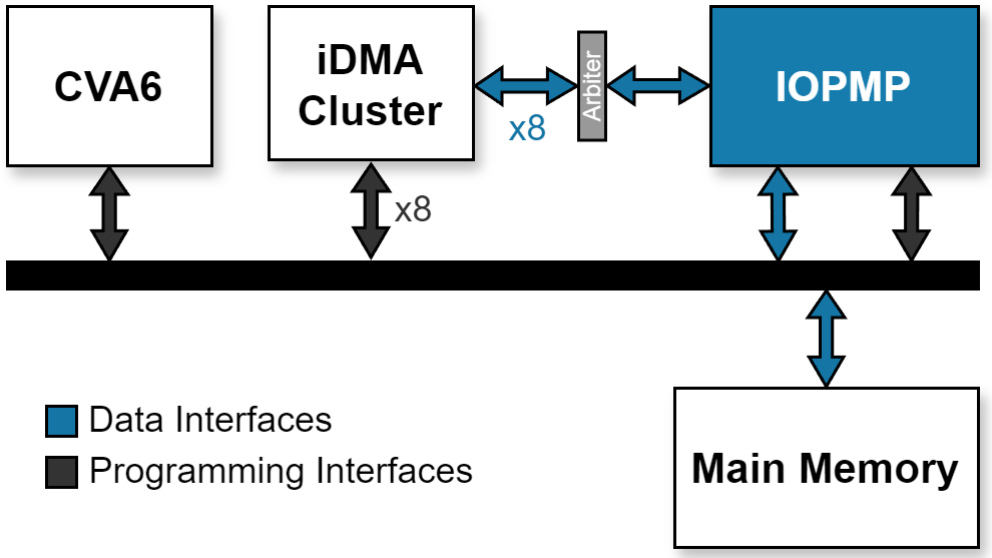
The entire transaction is validated within a clock cycle, after finding corresponding entry

Evaluation

03

Functional Evaluation

- Genesys2 FPGA @100MHz
- Two configurations of a single core CVA6-based SoC:
  - IOPMP in Source Enforcement managing 1 PULP iDMA device
  - IOPMP managing 8 PULP iDMA devices
- Validated SW stacks:
  - Custom Baremetal Framework
  - OpenSBI + Baremetal
  - OpenSBI + Linux



Hardware Resources

Configurations feature:

- Same number of MDs and Supported Sources
- Contains 8 entries for each MD/Source

	Entries	Sources / MDs	LUT	FF	BRAM
Baseline	8	1	2179	1285	4
x8	64	8	2878 *(32%)*	1862 *(45%)*	4 *(0%)*
x16	128	16	3533 *(62%)*	2414 *(88%)*	4 *(0%)*
x32	256	32	5000 *(129%)*	3597 *(180%)*	4 *(0%)*
x64	512	64 / 63*	7149 *(228%)*	6346 *(393%)*	4 *(0%)*

*Specification Maximum

Hardware Resources

Configurations feature:

- Same number of MDs and Supported Sources
- Contains 8 entries for each MD/Source

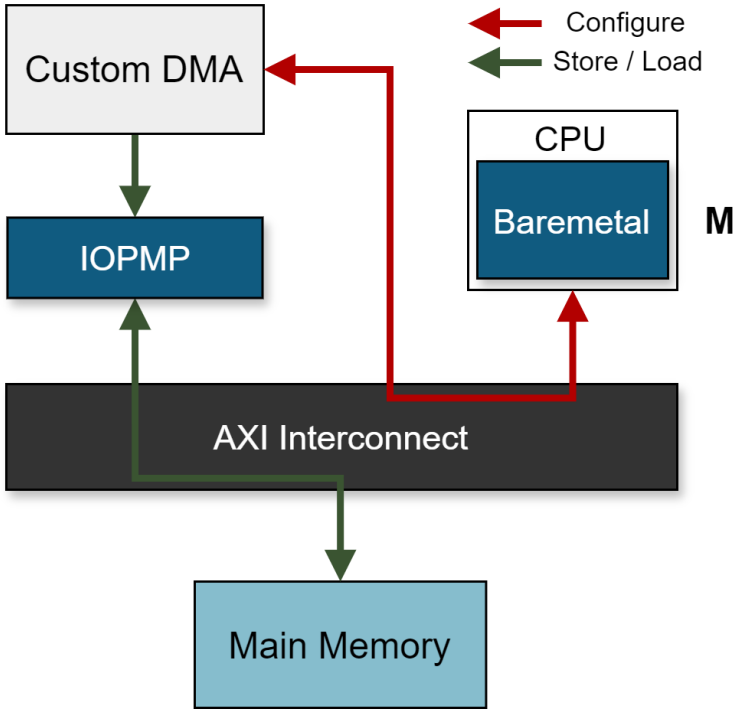
	Entries	Sources / MDs	LUT	FF	BRAM
Baseline	8	1	2179	1285	4
x8	64	8	2878 *(32%)*	1862 *(45%)*	4 *(0%)*
x16	128	16	3533 *(62%)*	2414 *(88%)*	4 *(0%)*
x32	256	32	5000 *(129%)*	3597 *(180%)*	4 *(0%)*
x64	512	64 / 63*	7149 *(228%)*	6346 *(393%)*	4 *(0%)*

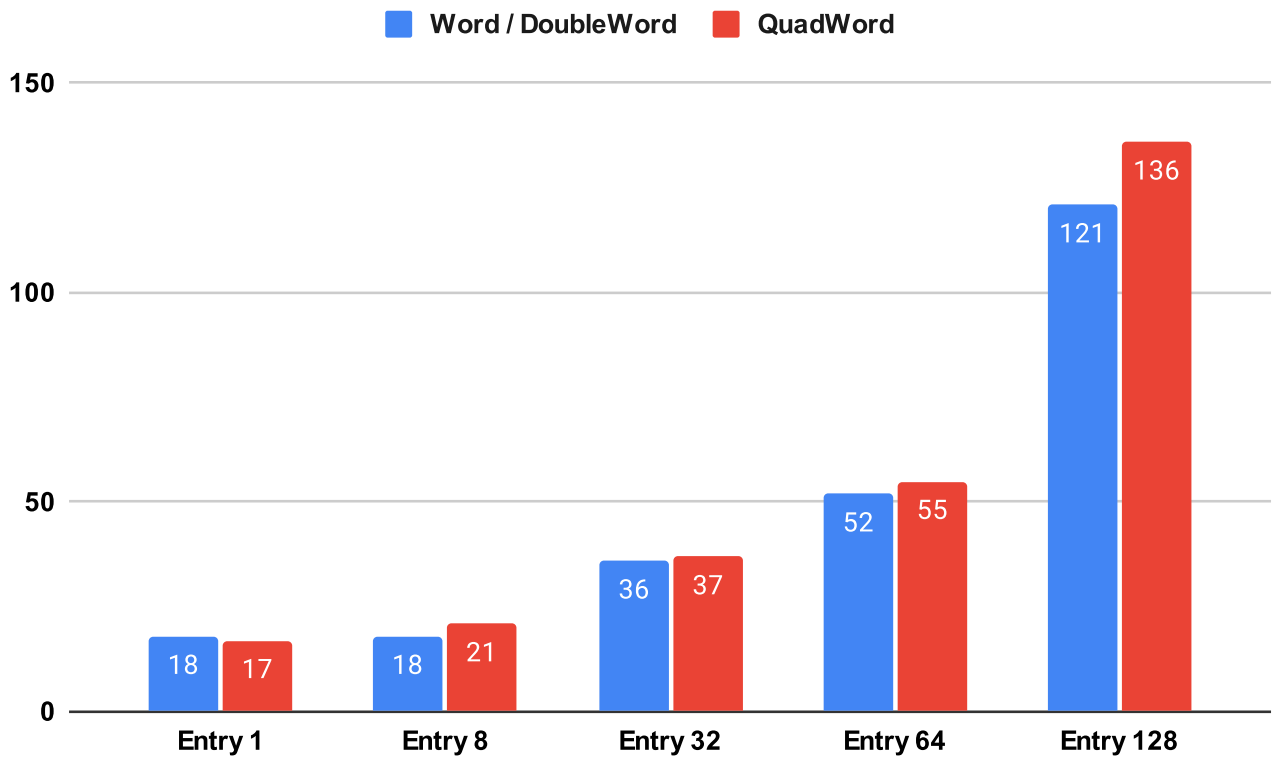
3.5% LUT, *1,6% FF*, and *0,9% BRAM* increase on a CVA6-based SoC

*Specification Maximum

Latency Results

- Device performing independent store/load operations
- Results refer to the entry that matches the transaction

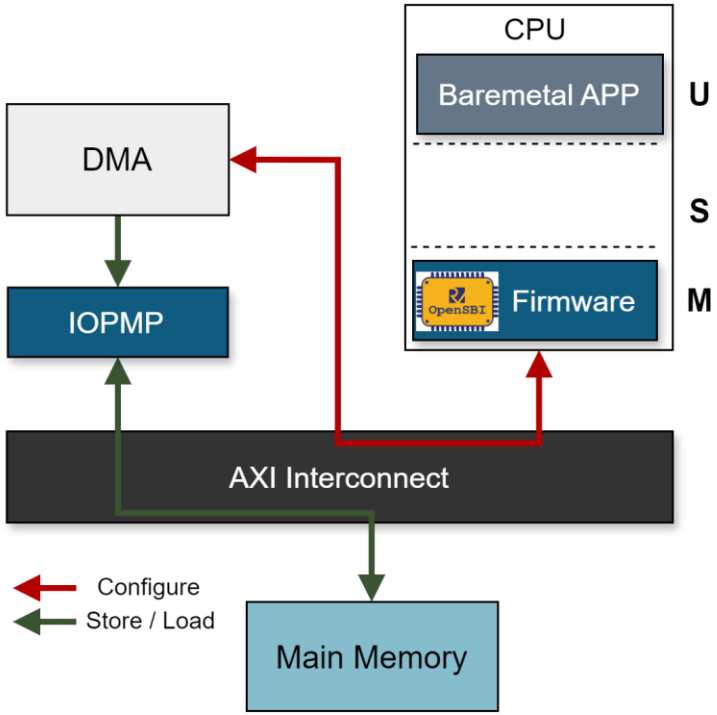




Clock Cyle penalty
(average over 1000 transactions)

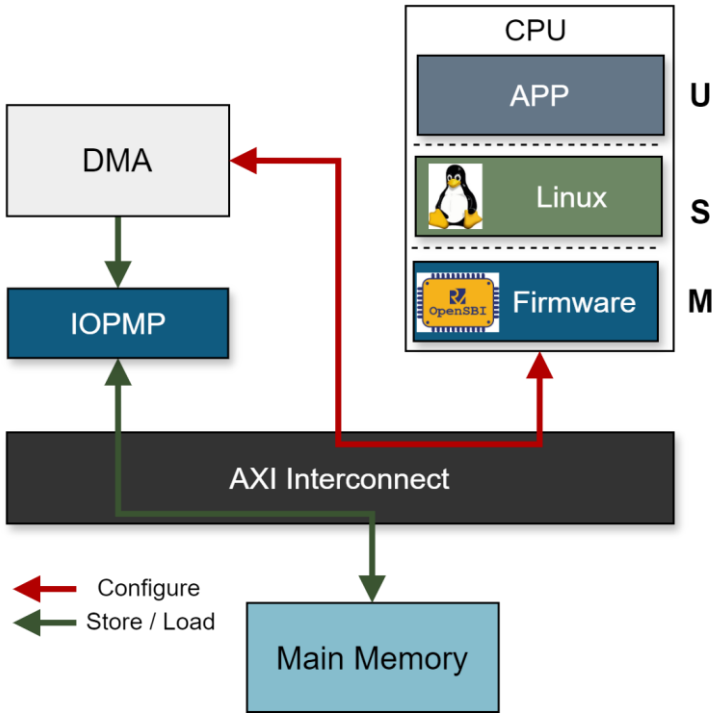
Latency Results

Baremetal + DMA



Performance penalties only up to 3% with 64 entries

Linux + DMA

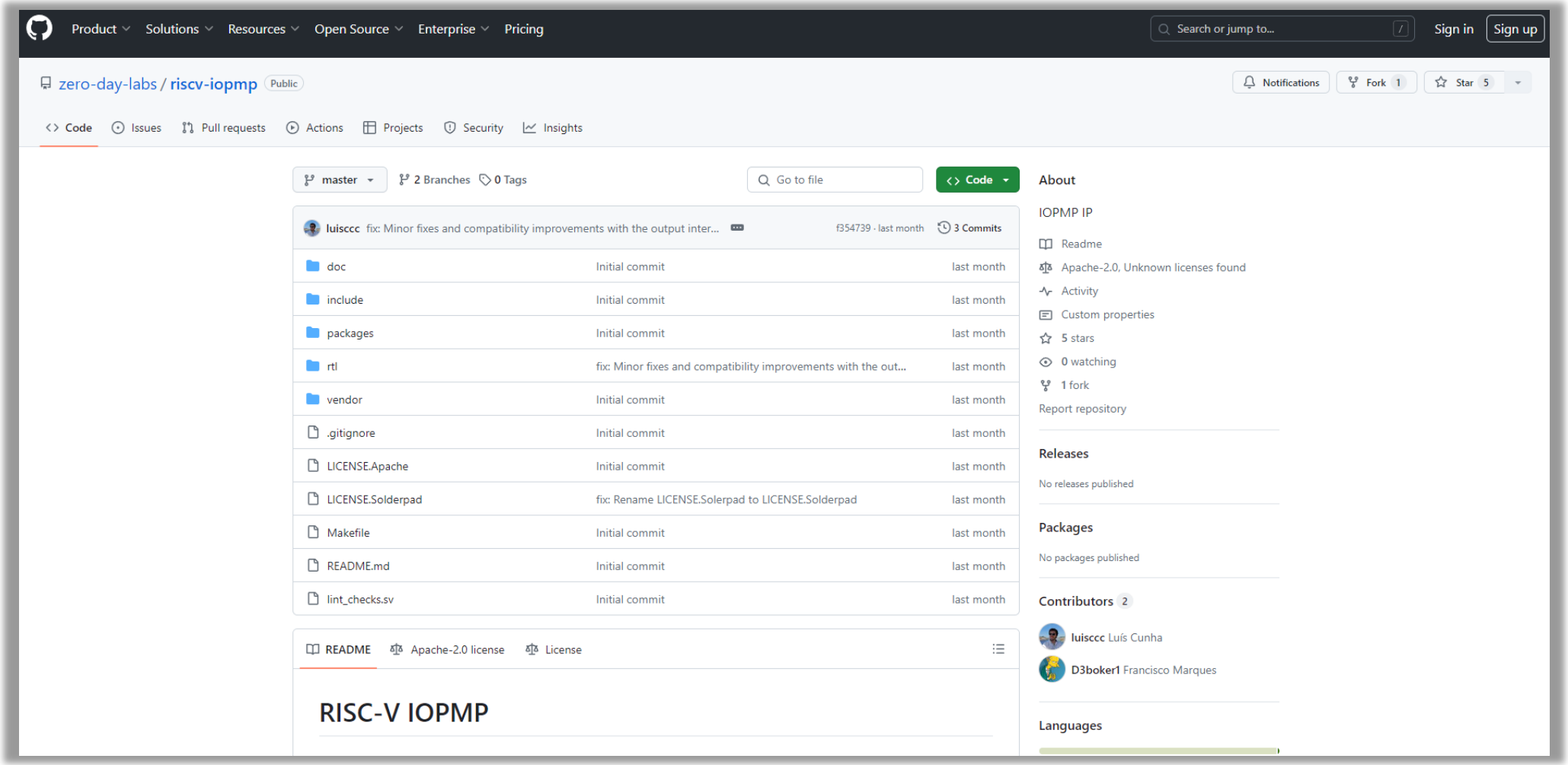


With more complex SW stacks and more complex workloads, the performance penalty is less noticeable!

TAKEAWAY

Design and implementation of the first OPEN-SOURCE IOPMP IP compliant with the version 1.0.0-draft5

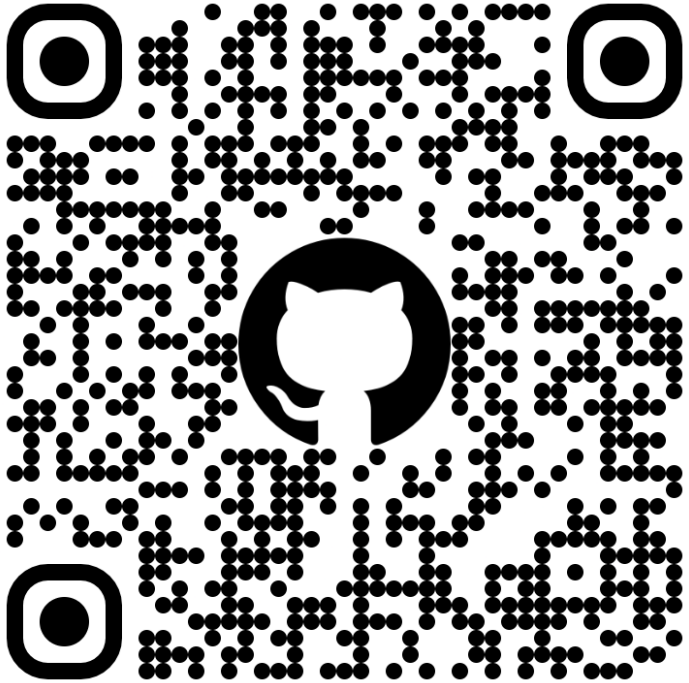
CALL TO ACTION!



CALL TO ACTION!

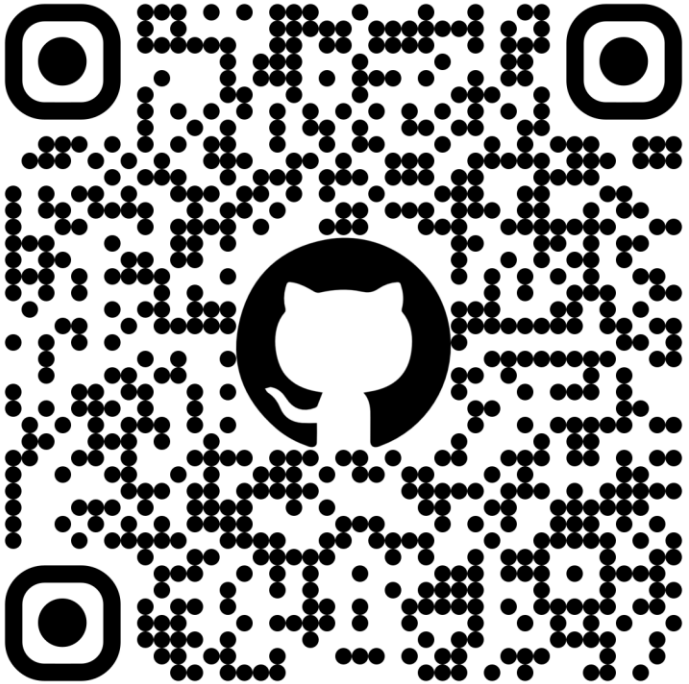
Basic IP

https://github.com/zero-day-labs/riscv-iopmp



CVA6 with IOPMP

https://github.com/zero-day-labs/cva6/tree/feat/iopmp



THANK YOU!

Luís Cunha (UMinho)

id11207@alunos.uminho.pt