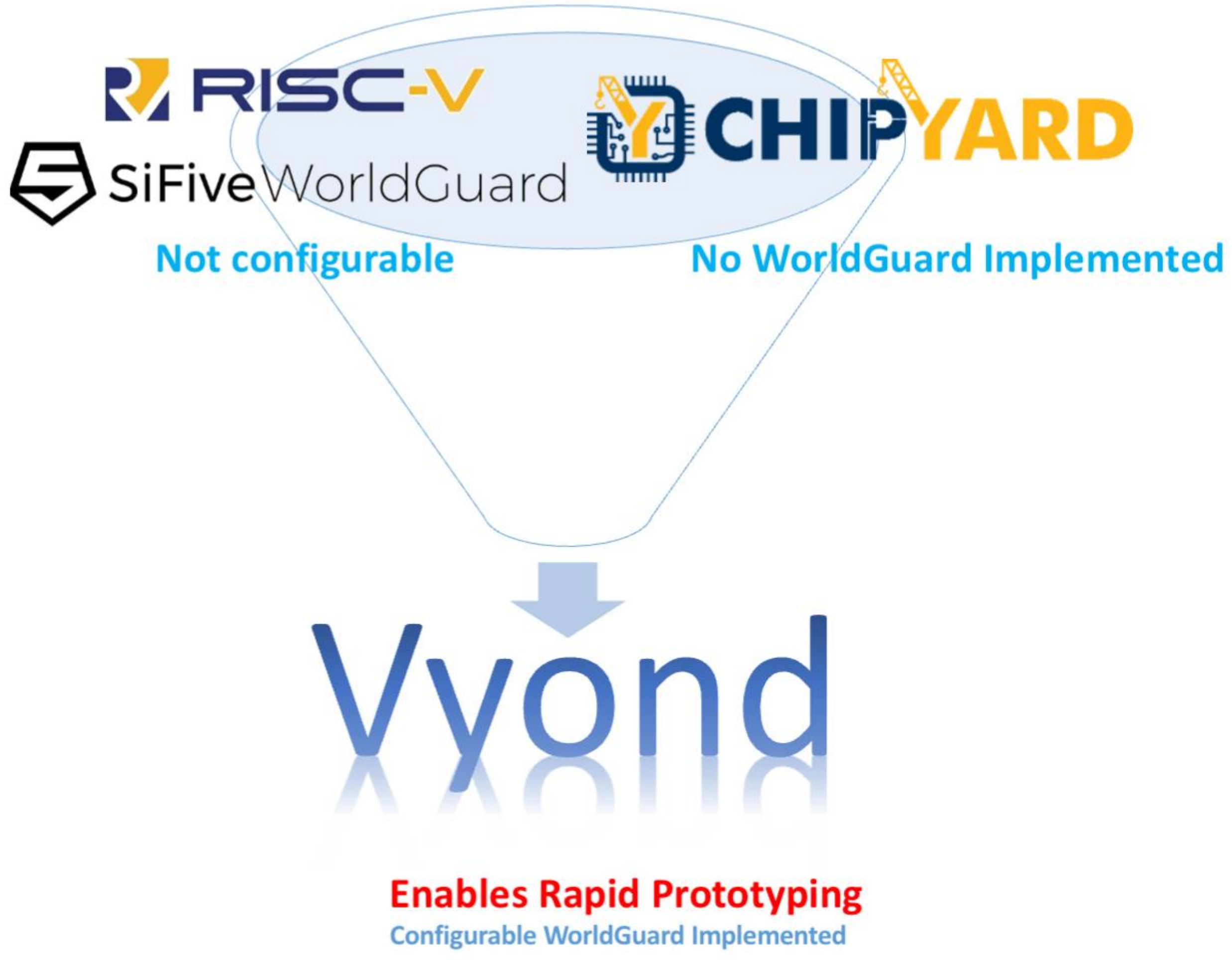
WorldGuard: Software Isolation in RISC-V and Vyond

Sungkeun Kim
Security&Privacy Team
Samsung Research
July31, 2025

About me

- Working on security in computer systems and architecture
  - Thesis – Cache Design and Analysis for Mitigating Hardware Security in Multicore Systems
- Developed on Galaxy phones before Ph.D. study
  - Android framework
- Like open-source projects
  - H2database, Gem5, Intel RAAD, Chipyard
  - Now vyond
- Like Running

Vyond: Flexible and Rapid WorldGuard-based Security Prototyping using Chipyard


RISC-V
SiFive WorldGuard
CHIPYARD
Not configurable
No WorldGuard Implemented
Vyond
Enables Rapid Prototyping
Configurable WorldGuard Implemented

Agenda

- Software (Runtime) isolation for Trusted Execution Environment
- WorldGuard Introduction
- WorldGuard Implementation on Chipyard
- Related techniques and future potential

Trusted Execution Environment

What's wrong with my phone?

- What's wrong with my phone and server?
  - We can do so many things with it.
    - Banking, games, videos, etc.
  - Might be something malicious together..
    - A larger attack surface
- It may happen the same thing in clouds services
  - Many services from different companies run in the physically the same machine
- Sharing the physical resource is efficient **but dangerous!**

Trusted Execution Environment

- Guarantees Integrity and Confidentiality
- Examples
  - Intel – SGX/TDX
  - Arm – TrustZone/Realms
  - RISC-V – Keystone/Pengai
  - Samsung – TrustWare (Tursted OS)
- WorldGuard provides isolation

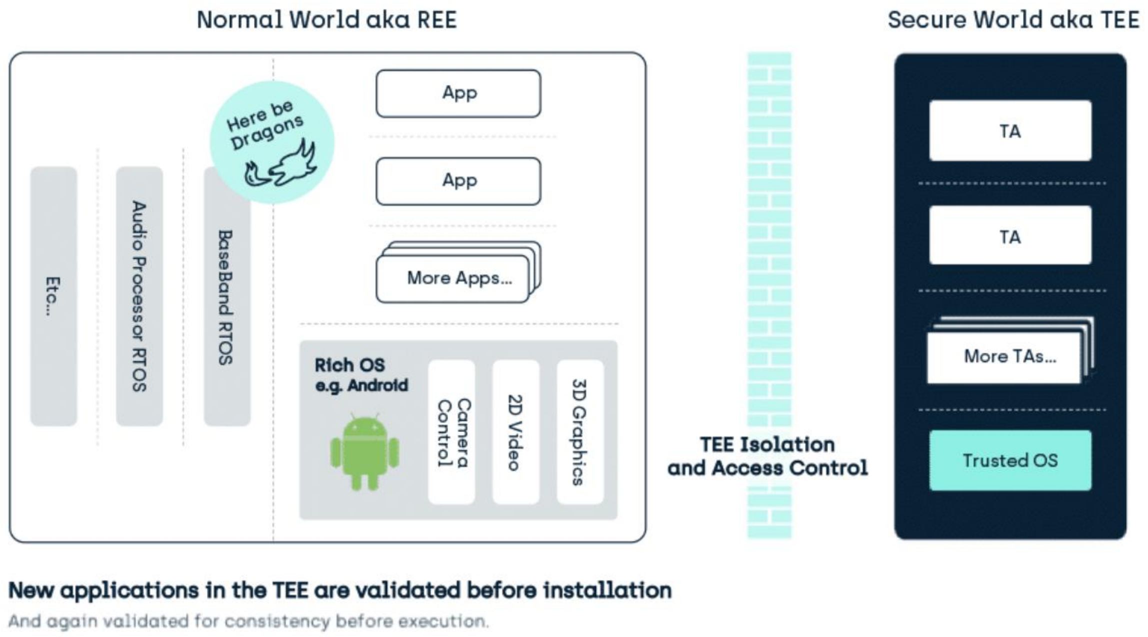


Figure 1 – TEE alongside traditional REE containing Rich OS

https://www.trustonic.com/technical-articles/what-is-a-trusted-execution-environment-tee/

Taxonomy of Isolation strategies

- Partitioning Resources

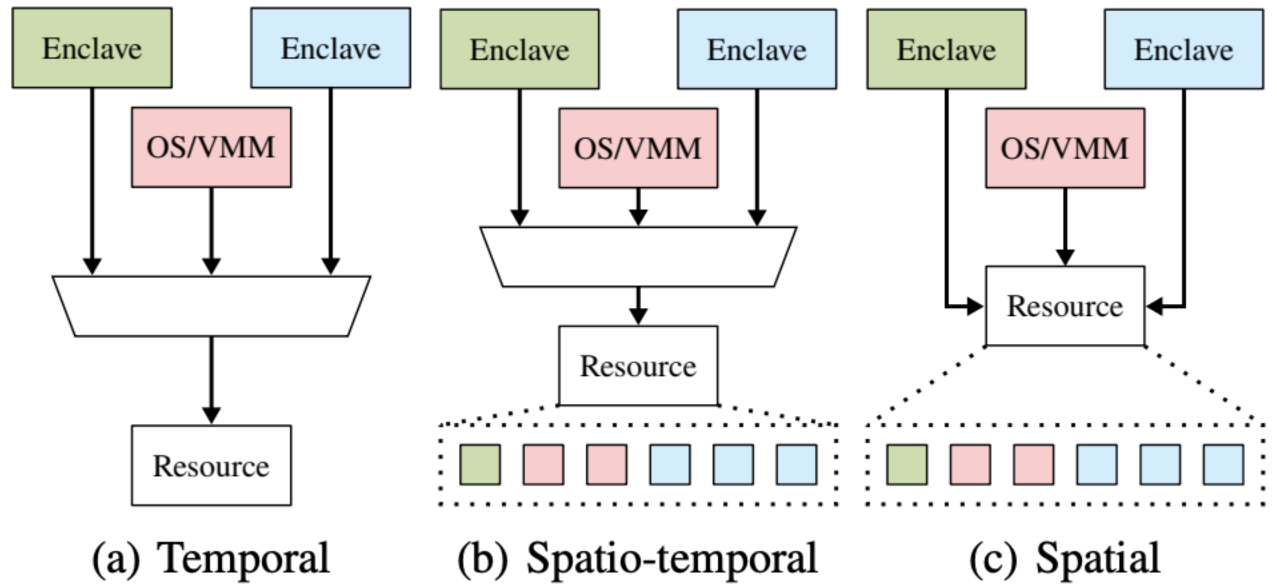


Fig. 5: Resources can be partitioned temporally, spatially, or a mix thereof (spatio-temporal).

https://arxiv.org/pdf/2205.12742

Taxonomy of Isolation strategies

- Enforcement
  - Logical
  - Cryptographical

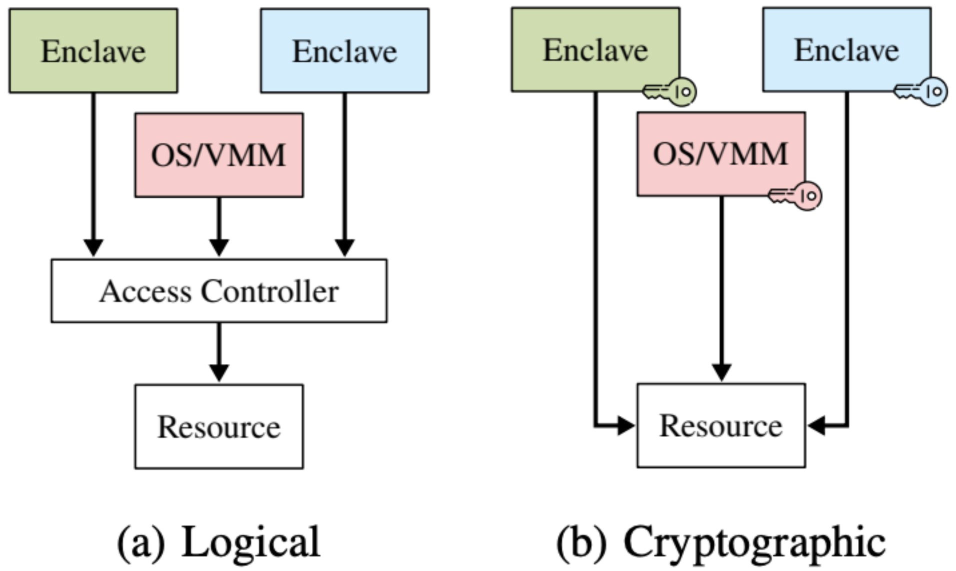


Fig. 6: Isolation enforcement strategies.

https://arxiv.org/pdf/2205.12742

WorldGuard

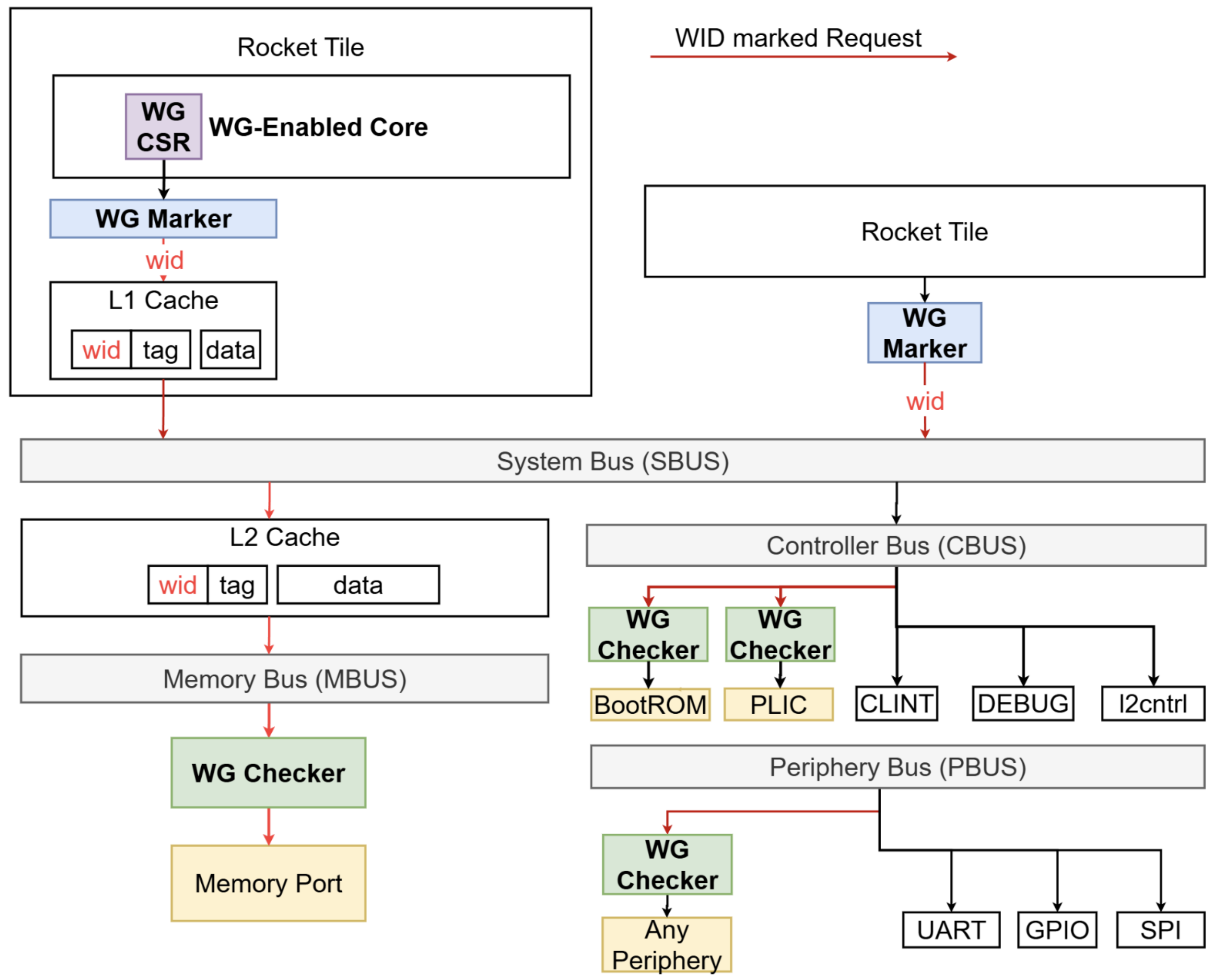
Introduction to WorldGuard (WG)

- A hardware-based software isolation solution for RISC-V cores
- It leverages logical enforcement strategy to isolate
- WG provides software execution contexts known as "worlds", beyond which software cannot reach.
  - World is referred to World ID (WID)

Key Concepts

- A wg-aware core
  - Contains WID registers needed to mark transactions with WID associated with each active privilege level.
- A wgMarker
  - Adds the active WID to bus transaction passing through them.
- A wg checker
  - Accept or reject transactions based on:
    - Physical address, WID, transaction type (read or write)
    - Accepted transaction are passed downstream to the target resource.
    - Rejected transaction generates an interrupt or exception.
- The trusted World ID
  - WID authorized to access WG configuration registers.

Overview


Rocket Tile
WG CSR
WG-Enabled Core
WG Marker
wid
L1 Cache
wid
tag
data
WID marked Request
Rocket Tile
WG Marker
wid
System Bus (SBUS)
L2 Cache
wid
tag
data
Controller Bus (CBUS)
WG Checker
WG Checker
BootROM
PLIC
CLINT
DEBUG
l2cntrl
Memory Bus (MBUS)
WG Checker
Memory Port
Periphery Bus (PBUS)
WG Checker
Any Periphery
UART
GPIO
SPI

A wg-aware core (CSR+Marker)

- A WG-aware core integrates wgMarker with additional internal registers and CSRs so that different privilege modes use different WIDs.

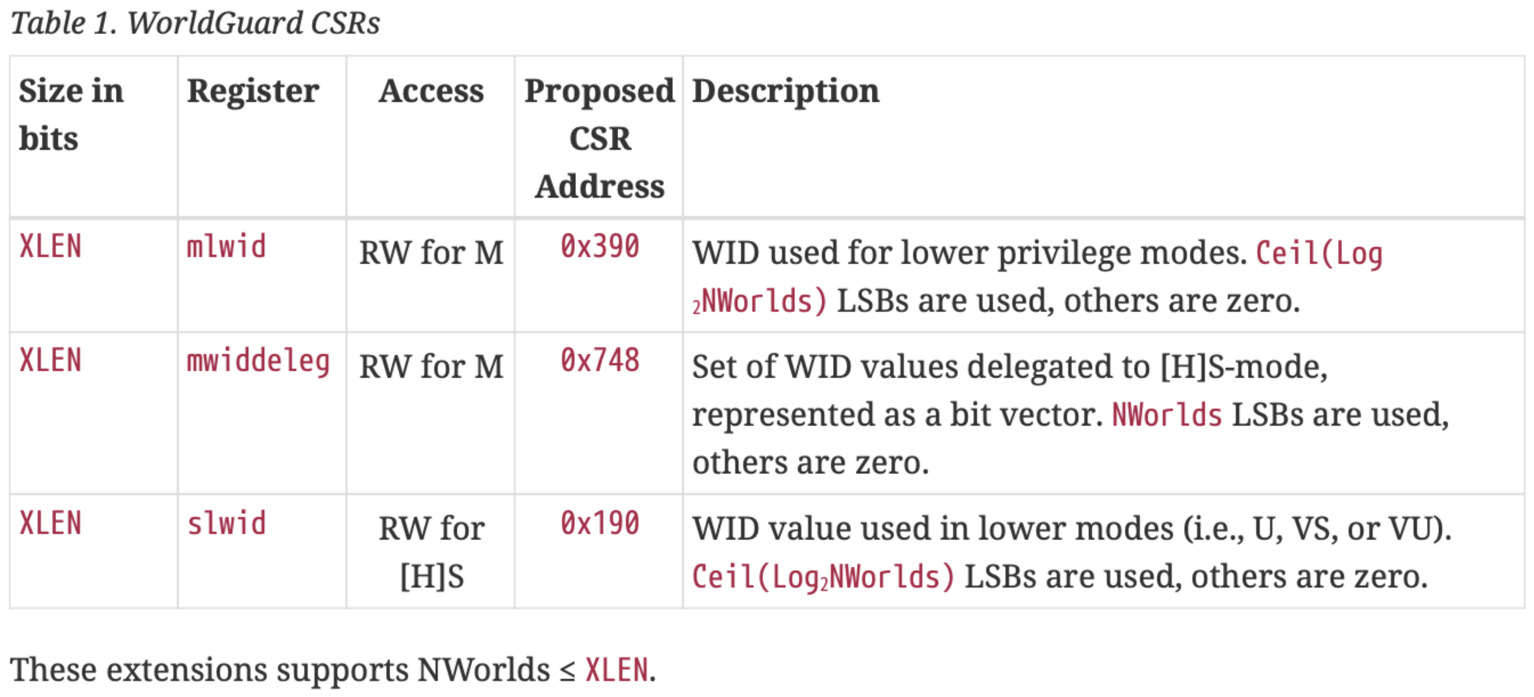
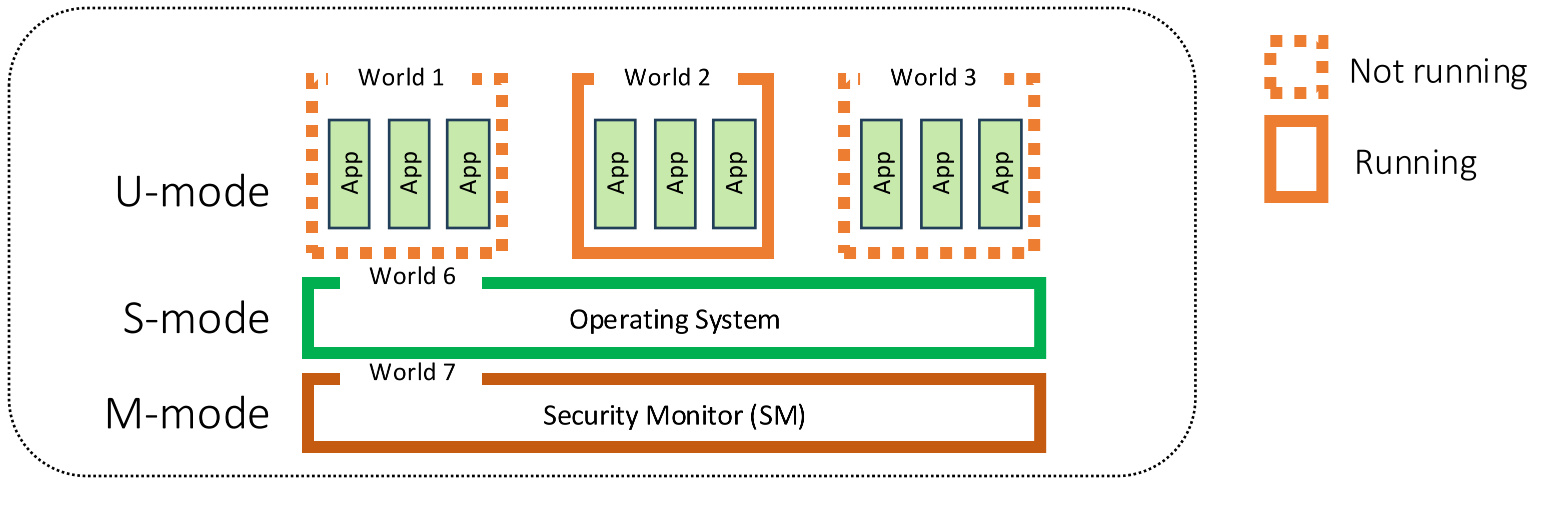


Table 1. WorldGuard CSRs

Size in bits	Register	Access	Proposed CSR Address	Description
XLEN	mlwid	RW for M	0x390	WID used for lower privilege modes. Ceil(Log$_2$NWorlds) LSBs are used, others are zero.
XLEN	mwiddeleg	RW for M	0x748	Set of WID values delegated to [H]S-mode, represented as a bit vector. NWorlds LSBs are used, others are zero.
XLEN	slwid	RW for [H]S	0x190	WID value used in lower modes (i.e., U, VS, or VU). Ceil(Log$_2$NWorlds) LSBs are used, others are zero.

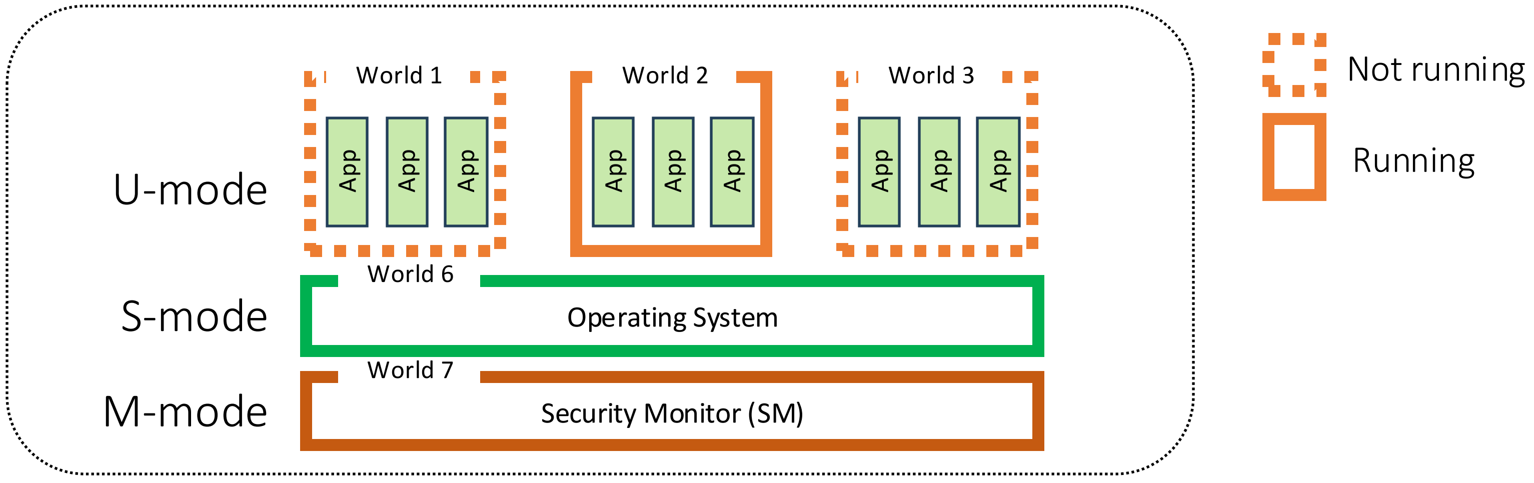
These extensions supports NWorlds ≤ XLEN.

Worlds allocation example



S/W

Worlds allocation example

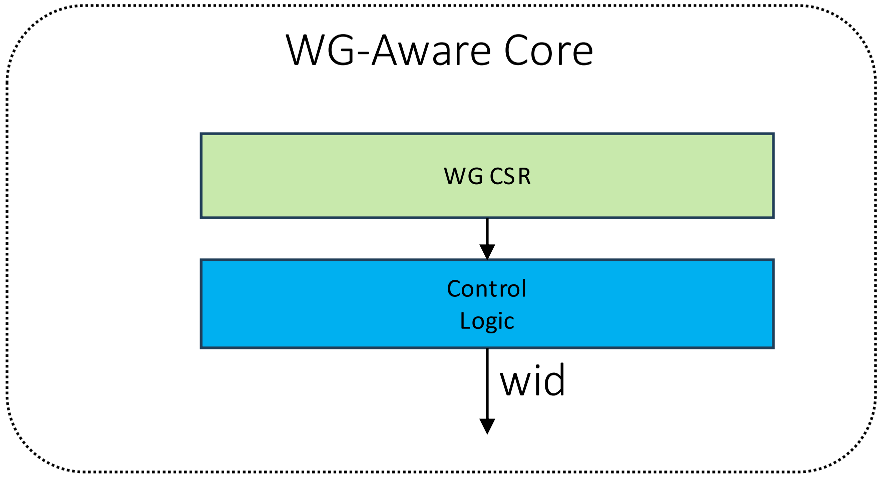


S/W

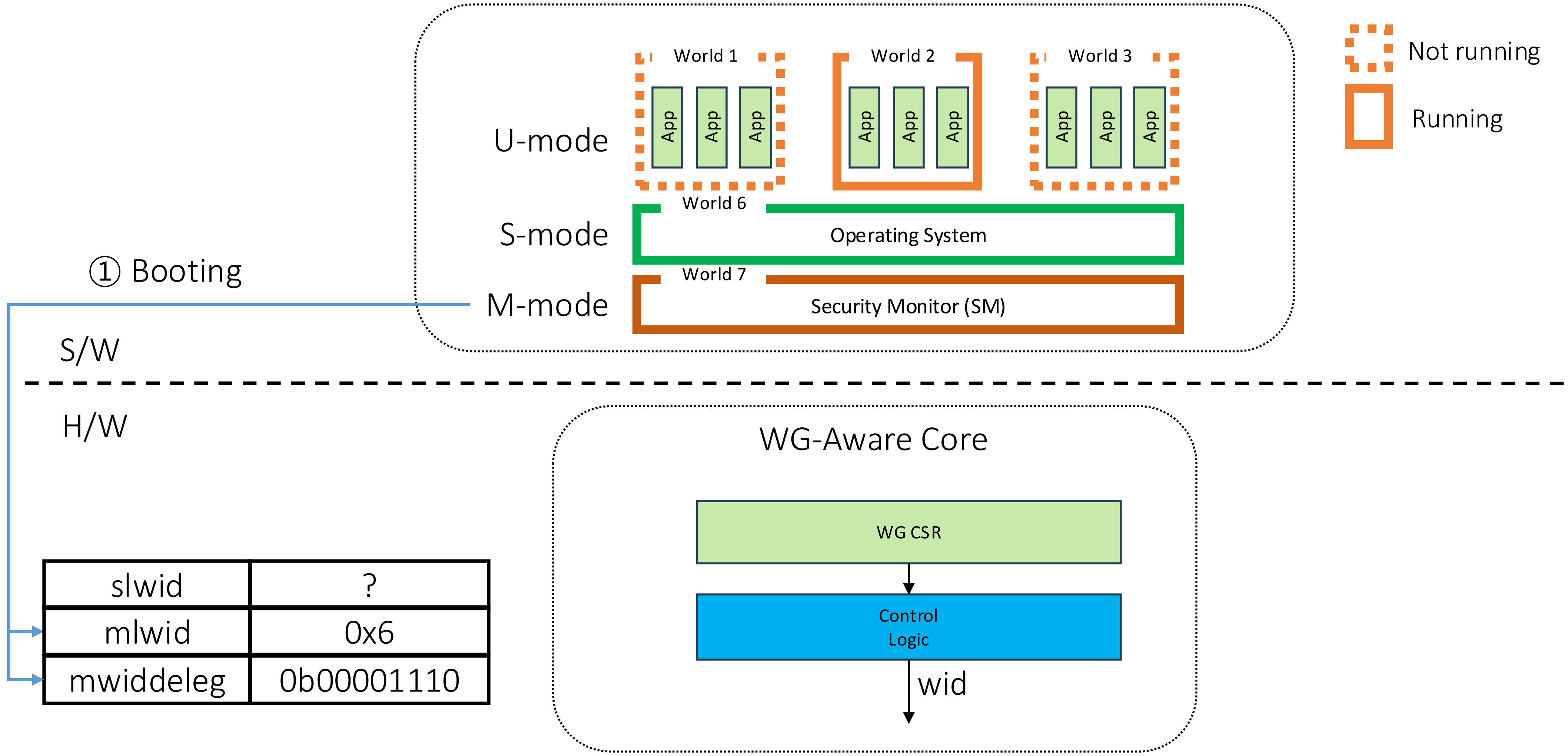
H/W

WG CSRs

slwid	?
mlwid	?
mwiddeleg	?

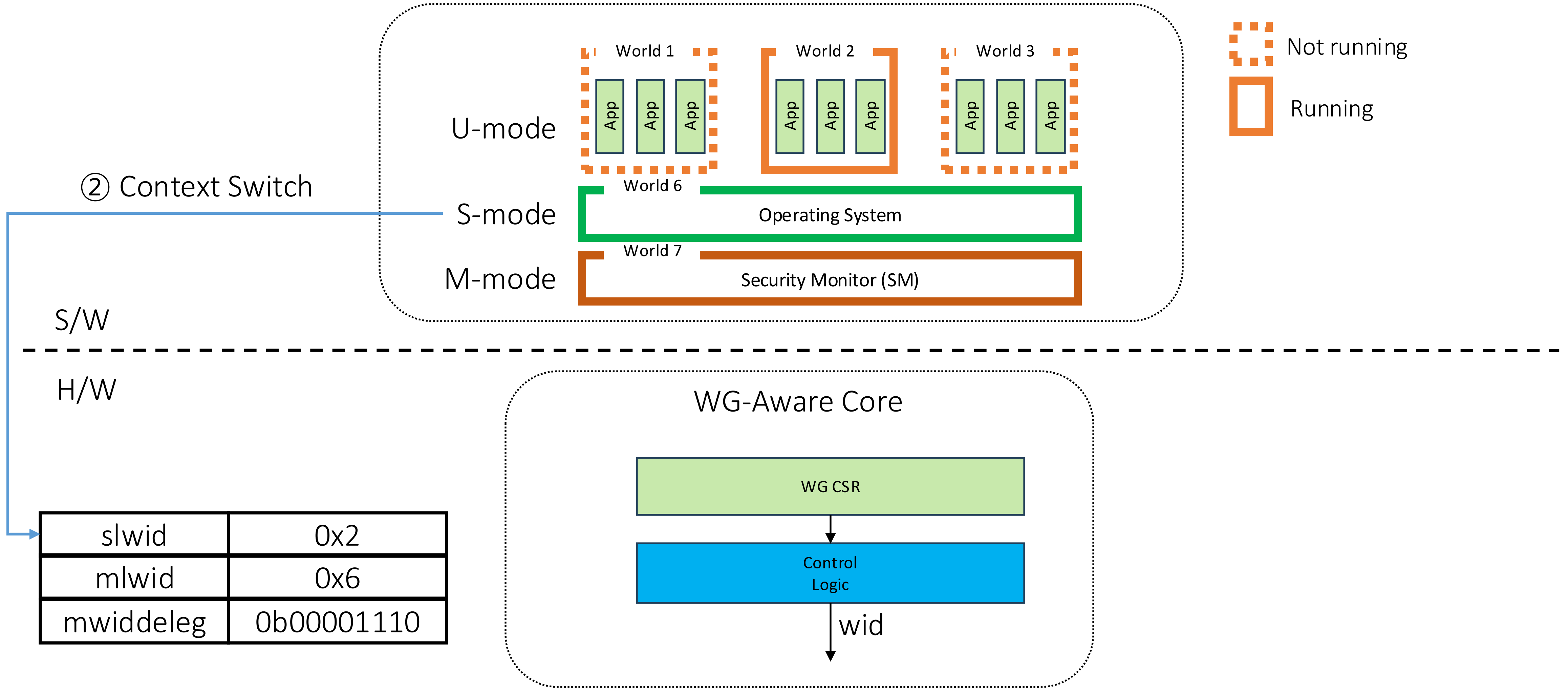


Worlds allocation example



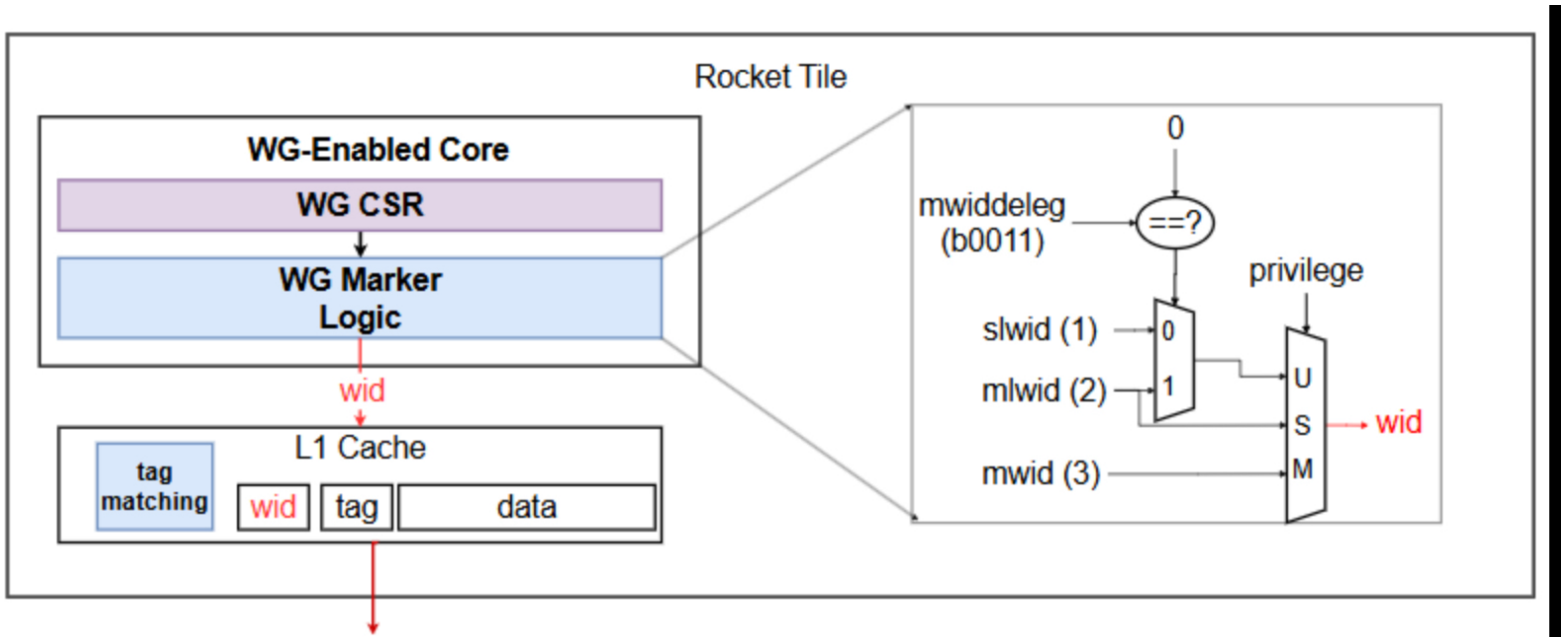
slwid	?
mlwid	0x6
mwiddeleg	0b00001110

Worlds allocation example



slwid	0x2
mlwid	0x6
mwiddeleg	0b00001110

WID selection logic



wgChecker

- The checker monitors a fixed physical address range reject or accept the transaction.
  - Reject: interrupt / exception
  - Accept: send the transaction to downstream
- Slots can specify a rule for a contiguous subset of addresses
  - Read/Write permissions for each world

wgChecker – Configuration Register

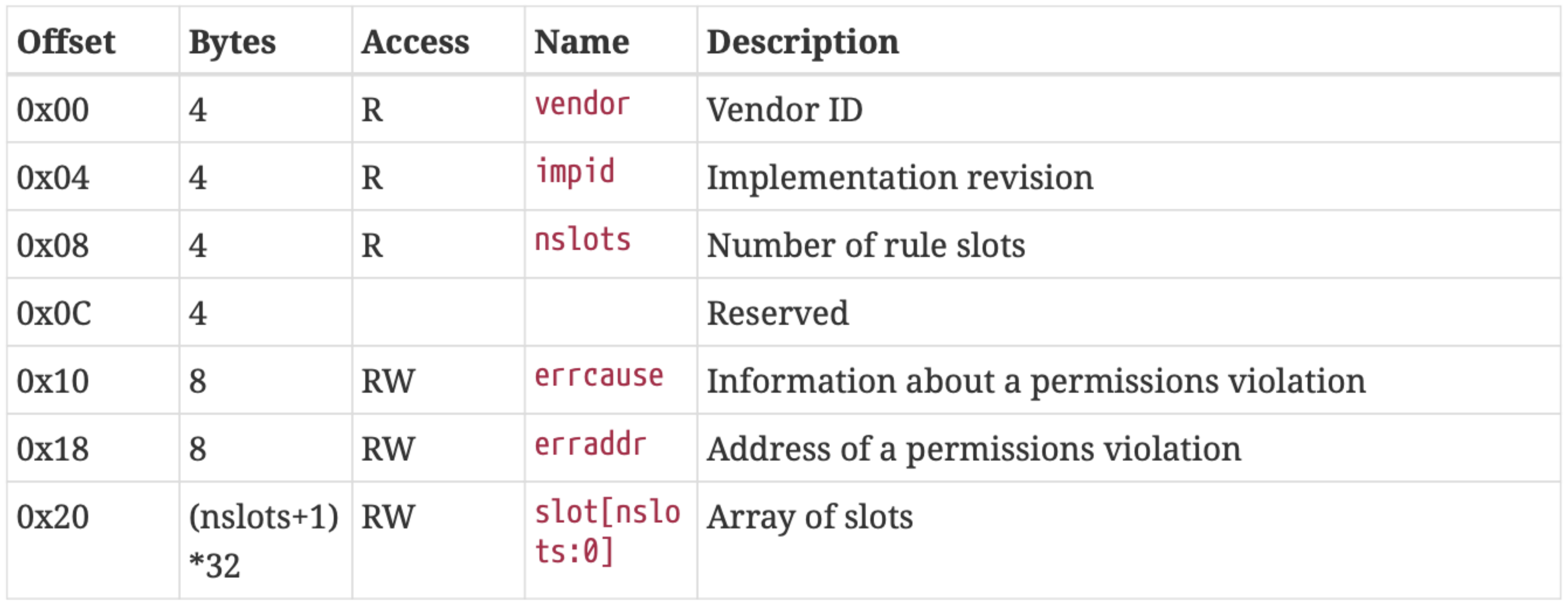


Table 2. WG Generic Checker Configuration Register

Offset	Bytes	Access	Name	Description
0x00	4	R	`vendor`	Vendor ID
0x04	4	R	`impid`	Implementation revision
0x08	4	R	`nslots`	Number of rule slots
0x0C	4			Reserved
0x10	8	RW	`errcause`	Information about a permissions violation
0x18	8	RW	`erraddr`	Address of a permissions violation
0x20	(nslots+1) *32	RW	`slot[nslots:0]`	Array of slots

wgChecker – Slot Register

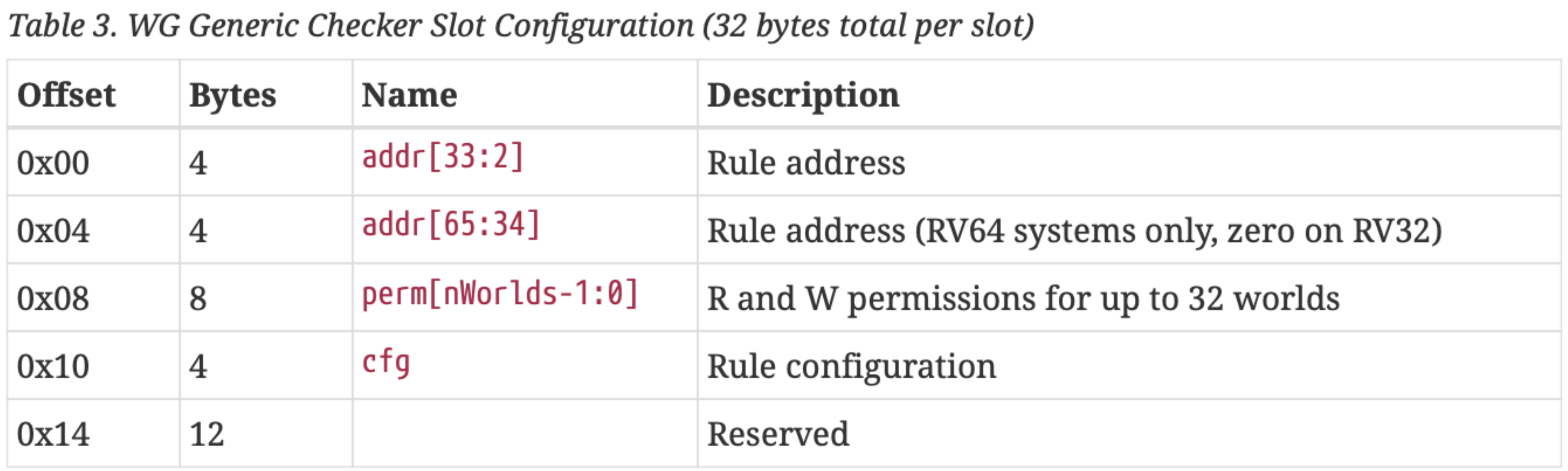


Table 3. WG Generic Checker Slot Configuration (32 bytes total per slot)

Offset	Bytes	Name	Description
0x00	4	`addr[33:2]`	Rule address
0x04	4	`addr[65:34]`	Rule address (RV64 systems only, zero on RV32)
0x08	8	`perm[nWorlds-1:0]`	R and W permissions for up to 32 worlds
0x10	4	`cfg`	Rule configuration
0x14	12		Reserved

wgChecker – Rule register

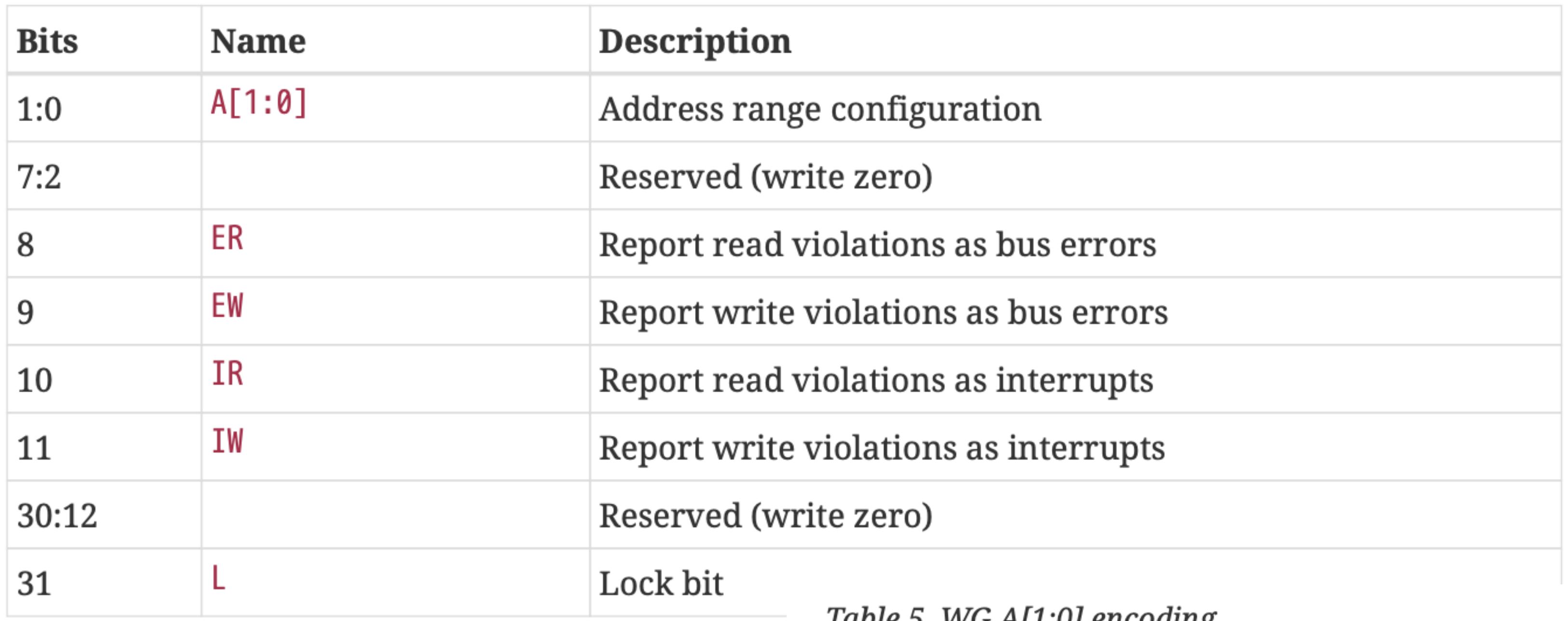


Table 4. WG Rule Configuration Register

Bits	Name	Description
1:0	`A[1:0]`	Address range configuration
7:2		Reserved (write zero)
8	`ER`	Report read violations as bus errors
9	`EW`	Report write violations as bus errors
10	`IR`	Report read violations as interrupts
11	`IW`	Report write violations as interrupts
30:12		Reserved (write zero)
31	`L`	Lock bit

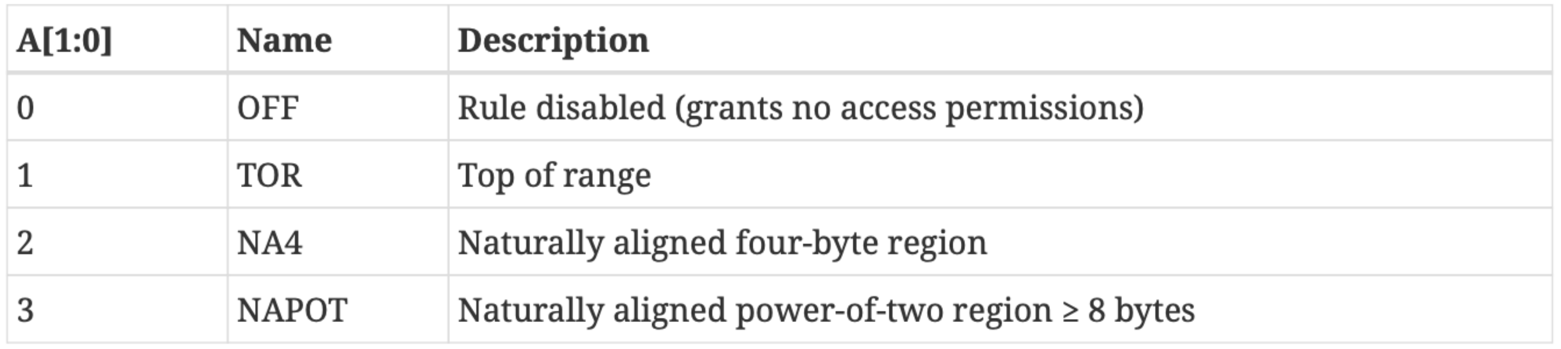
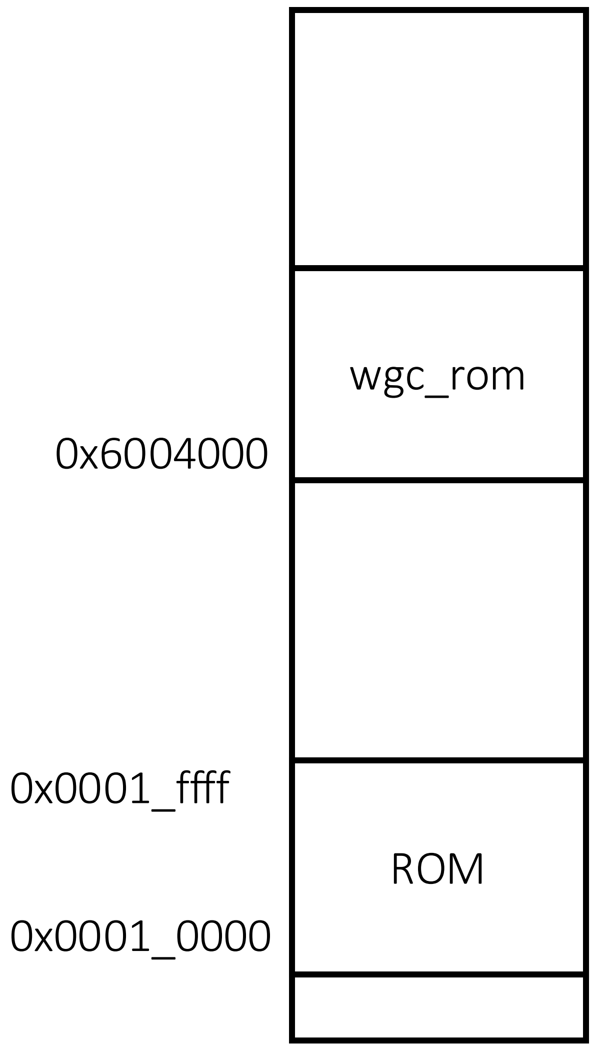


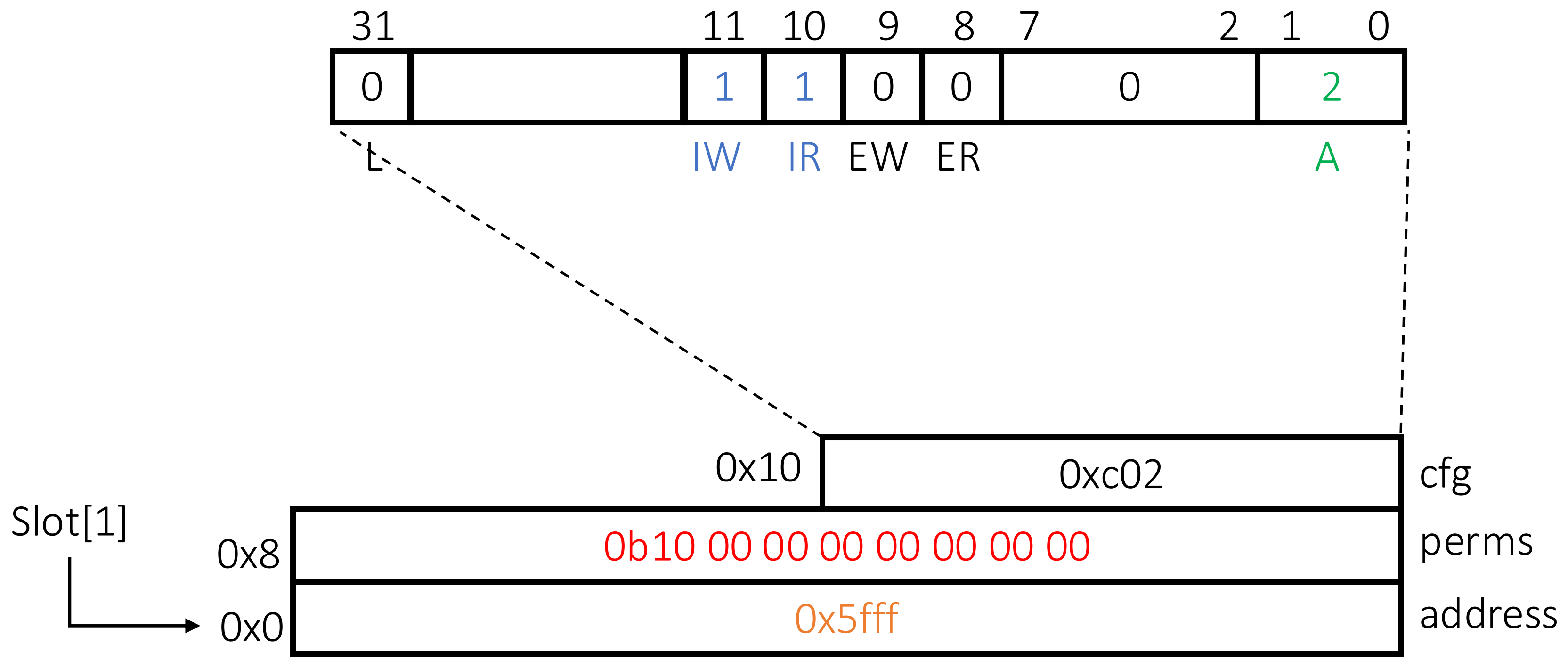
Table 5. WG A[1:0] encoding

A[1:0]	Name	Description
0	OFF	Rule disabled (grants no access permissions)
1	TOR	Top of range
2	NA4	Naturally aligned four-byte region
3	NAPOT	Naturally aligned power-of-two region ≥ 8 bytes

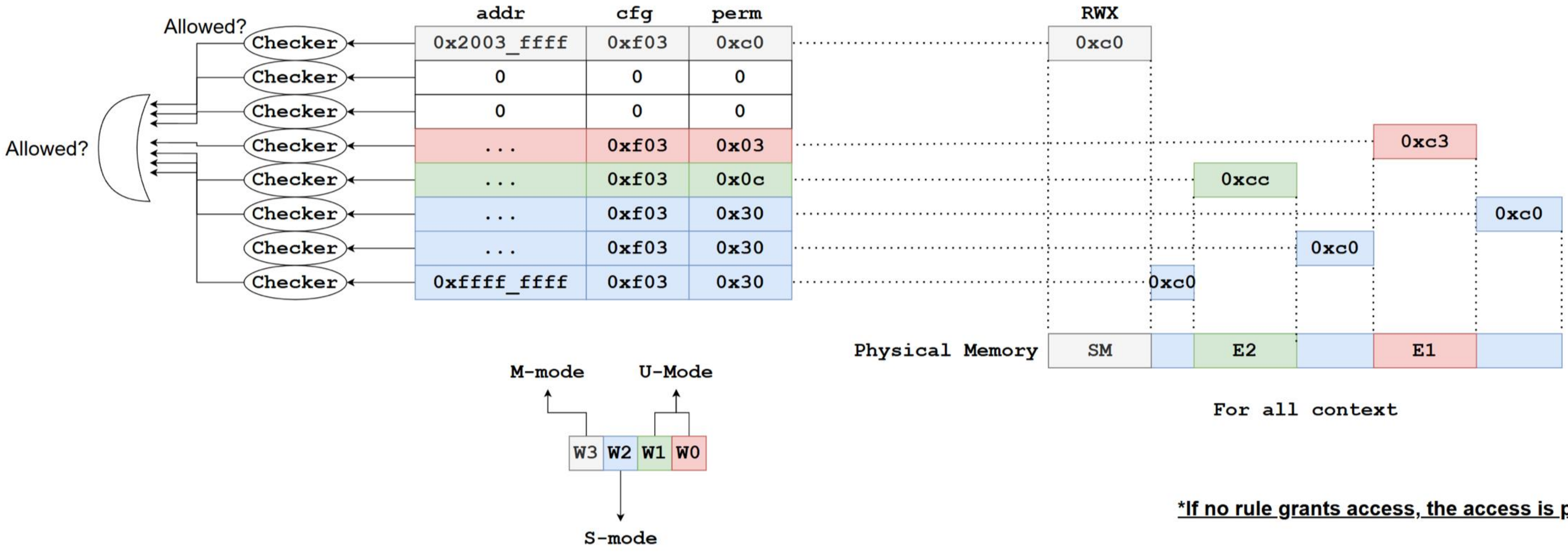
BootROM and wgChecker



Configuration:
m-mode (WID 7) can read the entire ROM region.
For RW without the permission, raise bus interrupt.
NAPOT for addressing encoding



DRAM and wgChecker



addr	cfg	perm
0x2003_ffff	0xf03	0xc0
0	0	0
0	0	0
...	0xf03	0x03
...	0xf03	0x0c
...	0xf03	0x30
...	0xf03	0x30
0xffff_ffff	0xf03	0x30

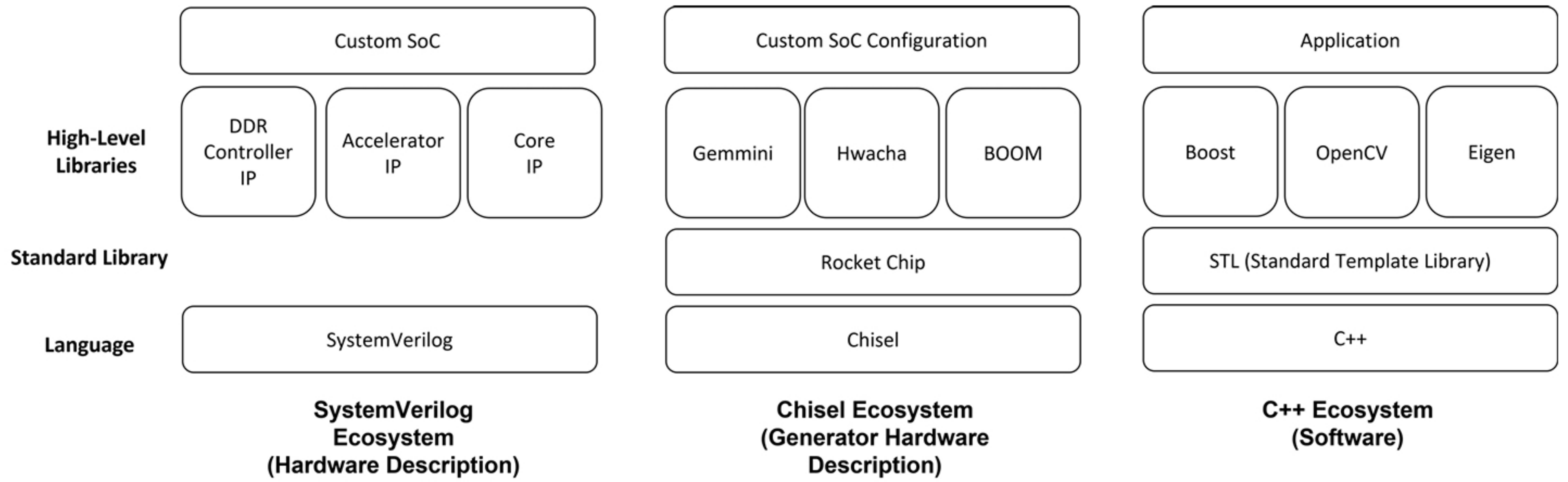
***If no rule grants access, the access is prevented.**

WorldGuard Implementation on Chipyard

What is chipyard?

- A unified framework for agile generator-based SoC development
- It provides a kind of glue for a general SoC design to connect target design flows
- Key features:
  - Chisel - Reusable by a high-level HDL.
  - Single source multiple targets
    - FPGA, Simulation, ASIC
  - Ecosystem – many available generators
    - AI Accelerators – Gemmini, NVDLA
    - RISC-V Cores – Rocket, Boom, Ibex

Quick Prototyping: generators on top of standard library

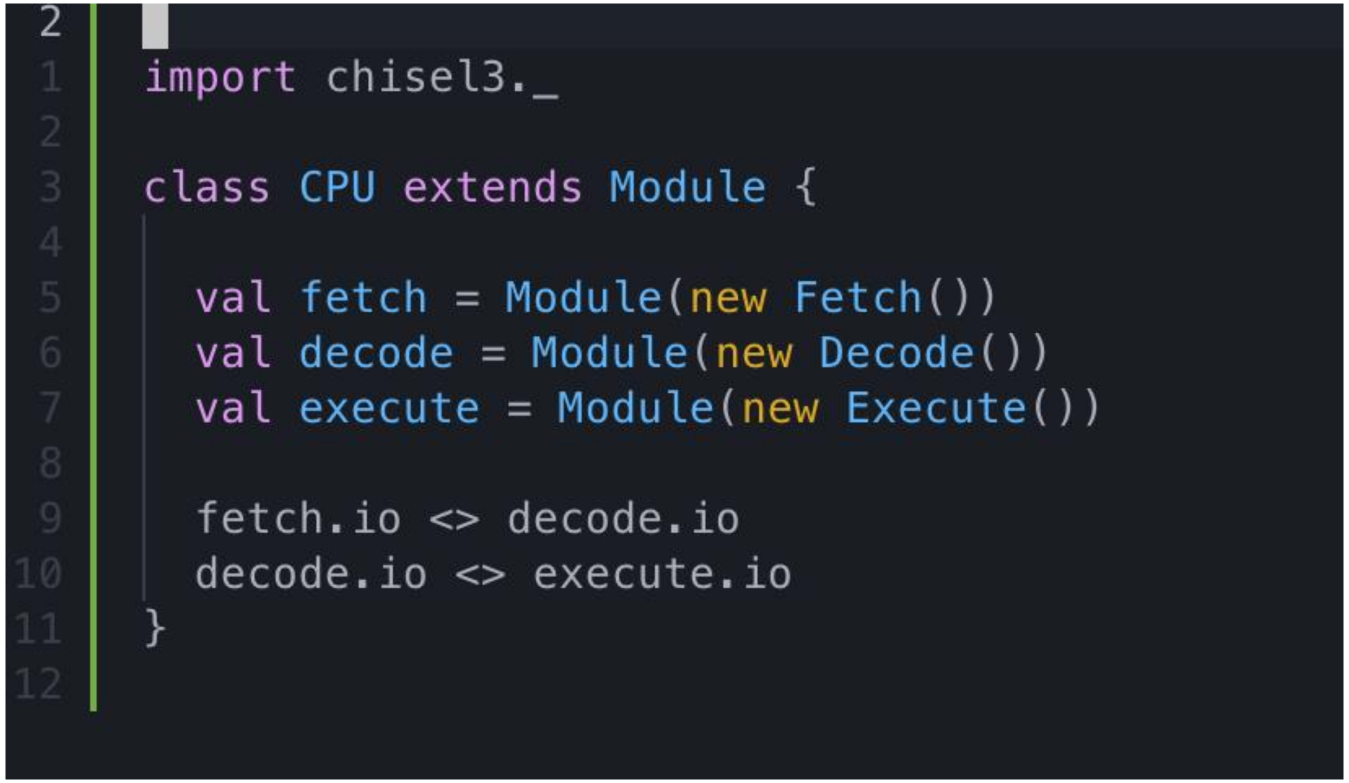


https://www2.eecs.berkeley.edu/Pubs/TechRpts/2022/EECS-2022-247.pdf

Chisel – Hardware Construction Language

- Chisel is a hardware construction language embedded in Scala that lets you write flexible, type-safe, and reusable code to generate Verilog for digital circuit design.
- Clock and Reset is automatically wired-up
- Bulk IO connection is possible (e.g., IO Bundle)
- Parameterized Modules

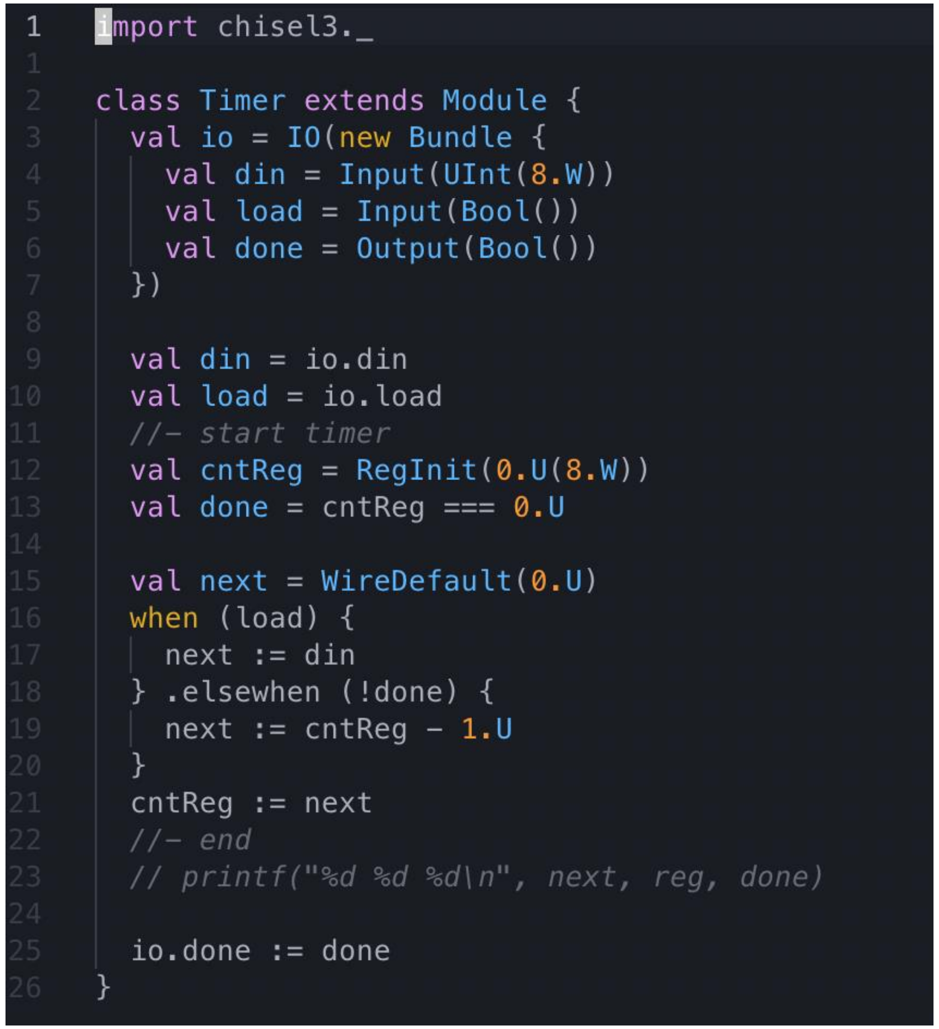
Chisel – Hardware Construction Language (example)



```
import chisel3._

class CPU extends Module {

  val fetch = Module(new Fetch())
  val decode = Module(new Decode())
  val execute = Module(new Execute())

  fetch.io <> decode.io
  decode.io <> execute.io
}

```

CPU Top

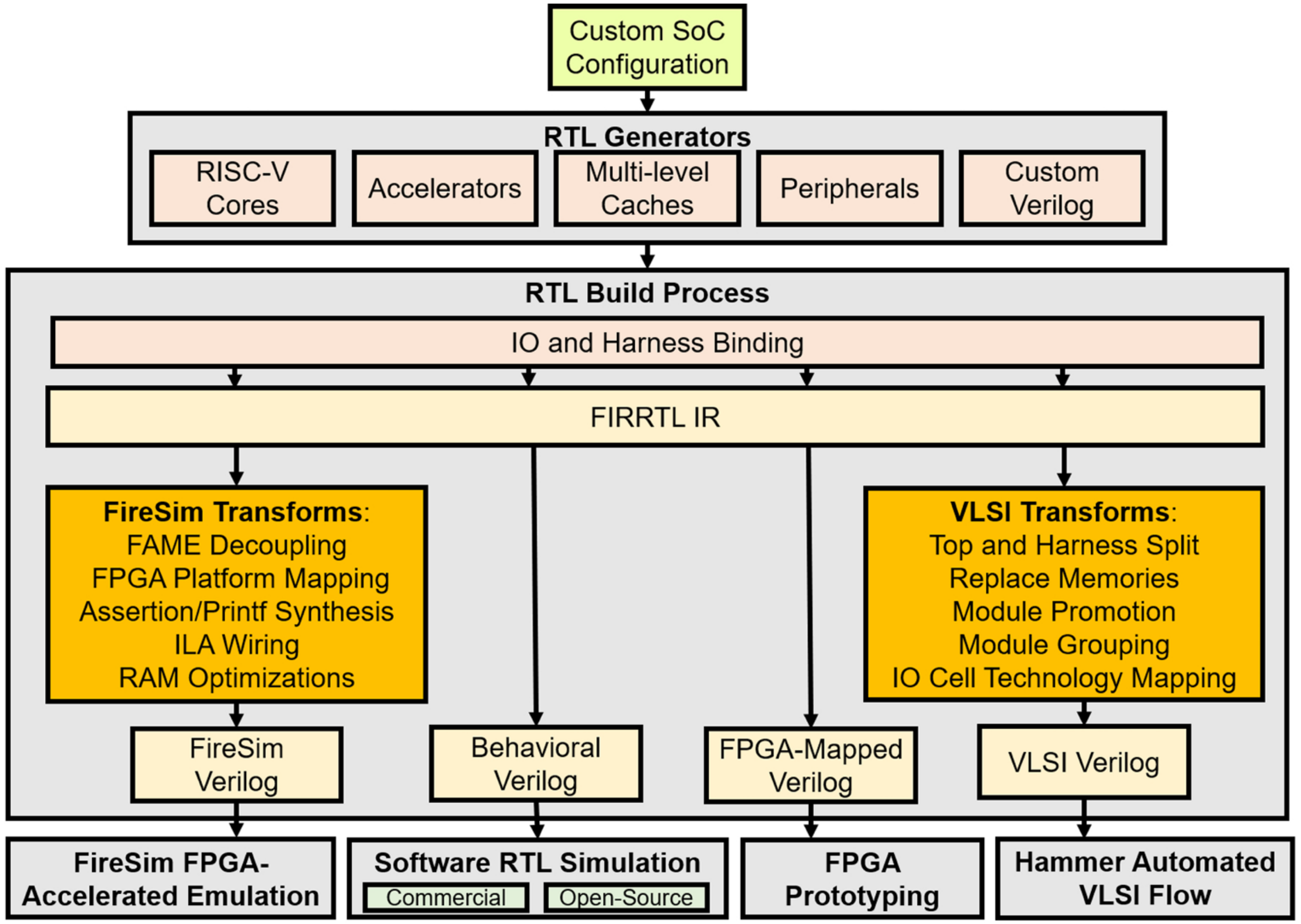
Chisel – Hardware Construction Language (example)



```
import chisel3._

class Timer extends Module {
  val io = IO(new Bundle {
    val din = Input(UInt(8.W))
    val load = Input(Bool())
    val done = Output(Bool())
  })

  val din = io.din
  val load = io.load
  //- start timer
  val cntReg = RegInit(0.U(8.W))
  val done = cntReg === 0.U

  val next = WireDefault(0.U)
  when (load) {
    next := din
  } .elsewhen (!done) {
    next := cntReg - 1.U
  }
  cntReg := next
  //- end
  // printf("%d %d %d\n", next, reg, done)

  io.done := done
}
```

Timer

Single source multiple targets


Custom SoC Configuration
RTL Generators
RISC-V Cores
Accelerators
Multi-level Caches
Peripherals
Custom Verilog
RTL Build Process
IO and Harness Binding
FIRRTL IR
FireSim Transforms: FAME Decoupling FPGA Platform Mapping Assertion/Printf Synthesis ILA Wiring RAM Optimizations
VLSI Transforms: Top and Harness Split Replace Memories Module Promotion Module Grouping IO Cell Technology Mapping
FireSim Verilog
Behavioral Verilog
FPGA-Mapped Verilog
VLSI Verilog
FireSim FPGA-Accelerated Emulation
Software RTL Simulation
Commercial
Open-Source
FPGA Prototyping
Hammer Automated VLSI Flow

https://www2.eecs.berkeley.edu/Pubs/TechRpts/2022/EECS-2022-247.pdf

WorldGuard Implementation

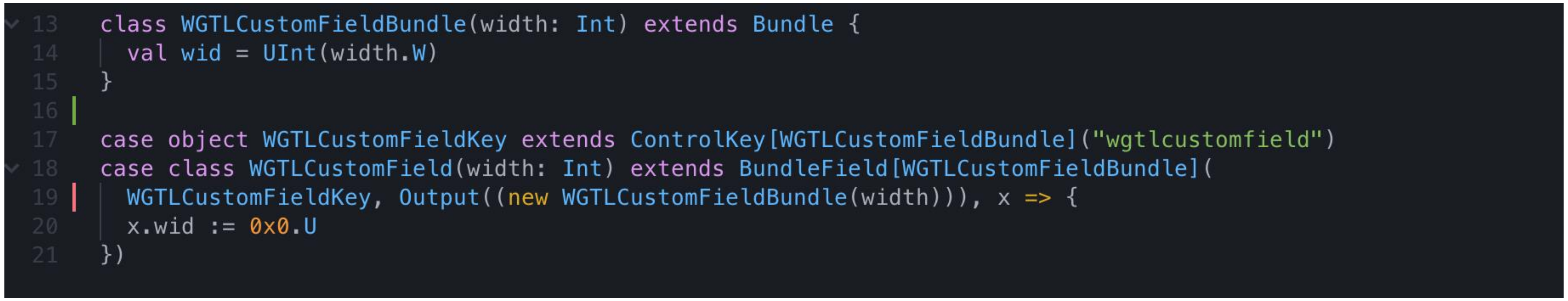
- Rocket Core Extension
  - WG CSR and wgMarker logic
- Bus extension – TileLink
  - New field for WID
- Extension for memory hierarchy
  - TLB, I-Cache, D-Cache, Last Level Cache
- wgMarker and wg checker
  - Standalone generator

WorldGuard Implementation – wg-aware core

- Extend Rocket Core and Tile
- Add code snippet
  - New CSR
  - Control logic
  - Propagate WID to downstream (tlb, icache, cache)

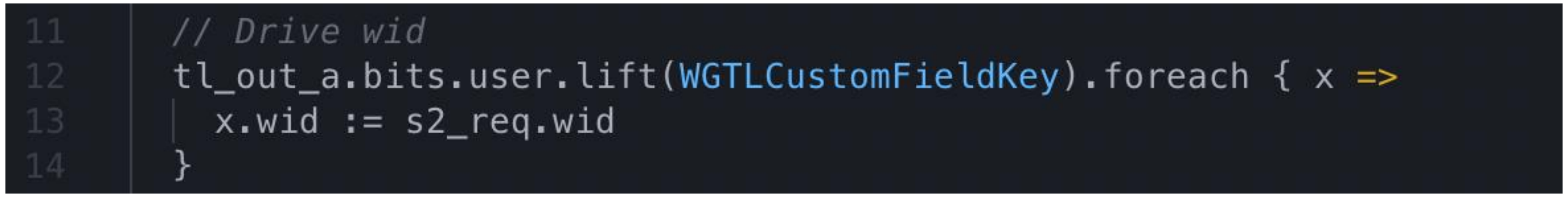
WorldGuard Implementation – bus extension

- Rocket SoC employs TileLink Bus
  - Use reserved field for user extension
  - Simple to extend (no need to modifies all the source files for IO)
  - Show code snippet



```
class WGTLCustomFieldBundle(width: Int) extends Bundle {
  val wid = UInt(width.W)
}

case object WGTLCustomFieldKey extends ControlKey[WGTLCustomFieldBundle]("wgtlcustomfield")
case class WGTLCustomField(width: Int) extends BundleField[WGTLCustomFieldBundle](
  WGTLCustomFieldKey, Output((new WGTLCustomFieldBundle(width))), x => {
  x.wid := 0x0.U
})
```

CustomField

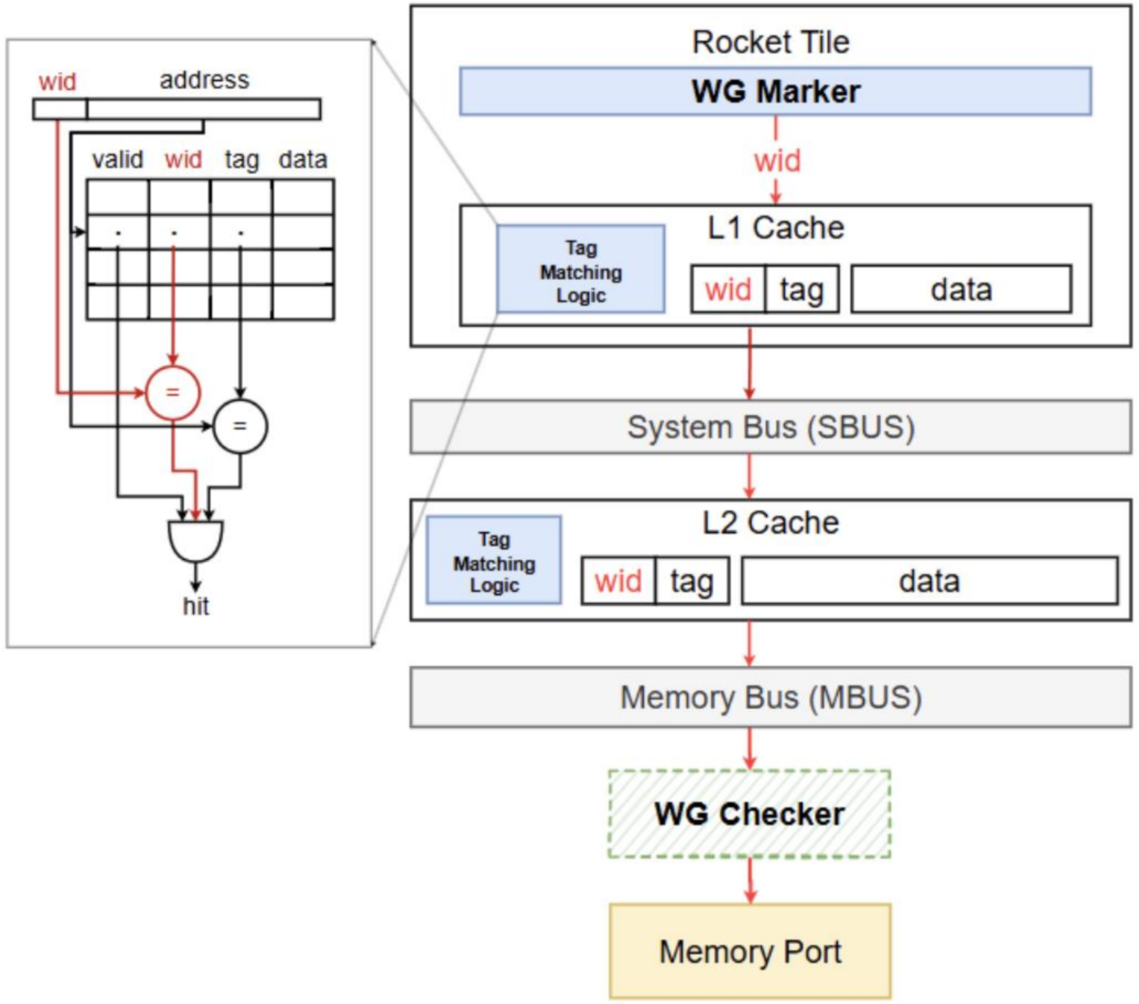


```
// Drive wid
tl_out_a.bits.user.lift(WGTLCustomFieldKey).foreach { x =>
  x.wid := s2_req.wid
}
```

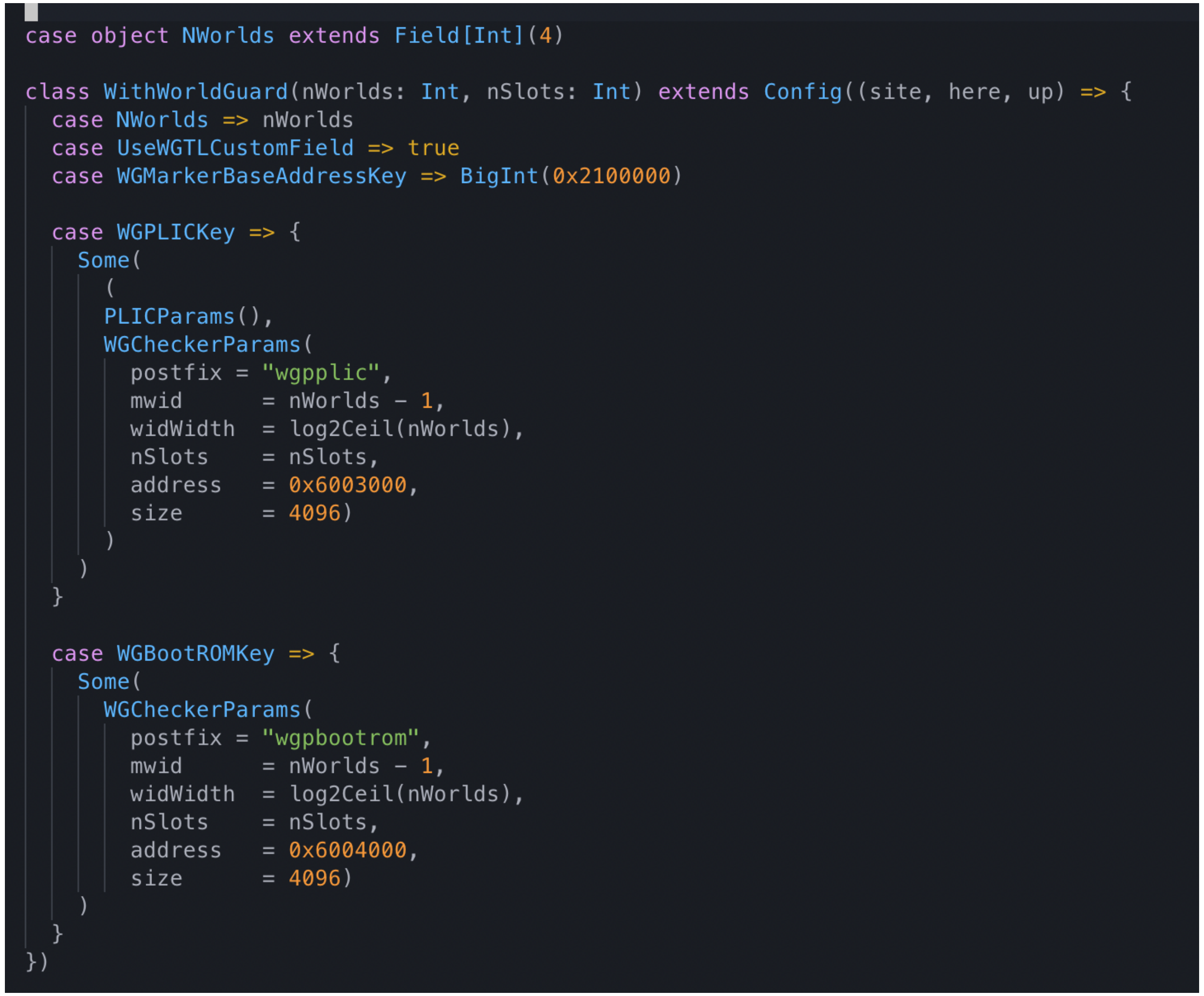
DCache

WorldGuard Implementation – cache extensions

- WID is used for tag matching
  - hit if both tag and wid match

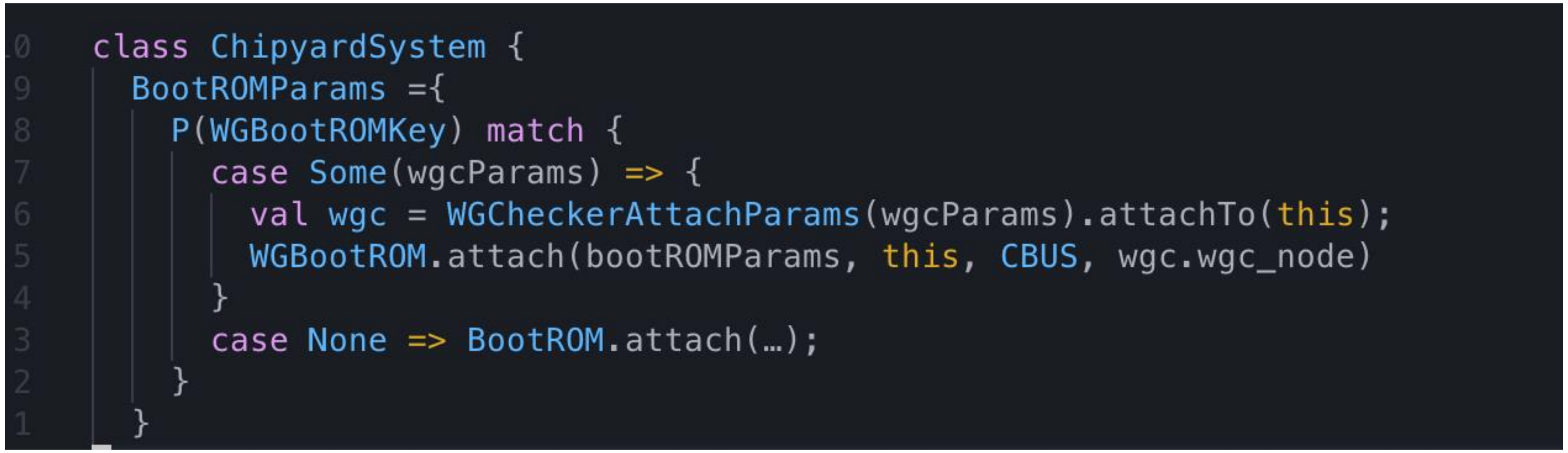


WorldGuard Implementation – Integration to Rocket SoC



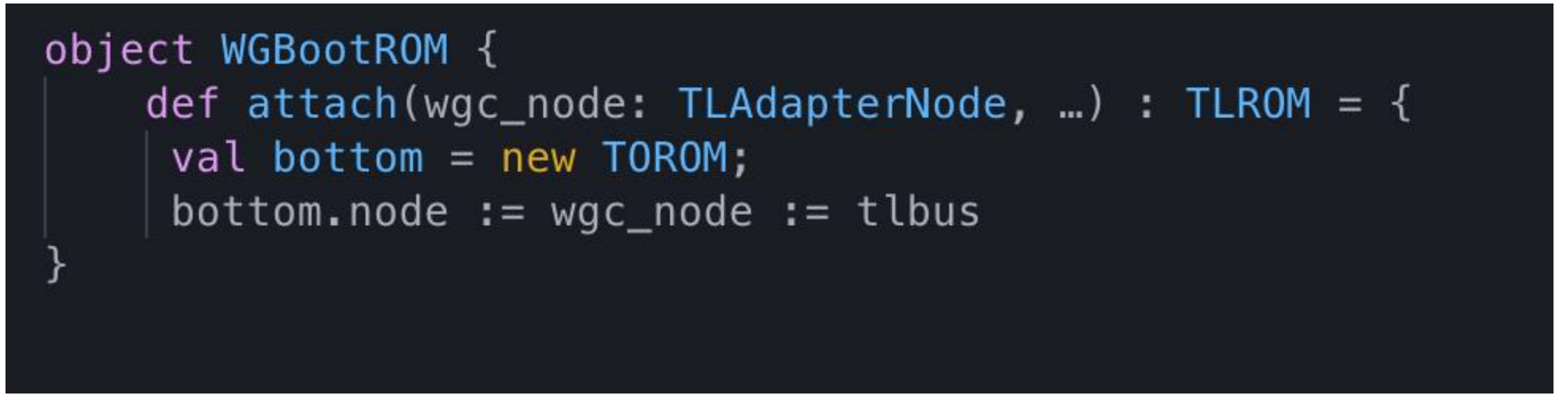
```scala
case object NWorlds extends Field[Int](4)

class WithWorldGuard(nWorlds: Int, nSlots: Int) extends Config((site, here, up) => {
  case NWorlds => nWorlds
  case UseWGTLCustomField => true
  case WGMarkerBaseAddressKey => BigInt(0x2100000)

  case WGPLICKey => {
    Some(
      (
      PLICParams(),
      WGCheckerParams(
        postfix = "wgpplic",
        mwid      = nWorlds - 1,
        widWidth  = log2Ceil(nWorlds),
        nSlots    = nSlots,
        address   = 0x6003000,
        size      = 4096)
      )
    )
  }

  case WGBootROMKey => {
    Some(
      WGCheckerParams(
        postfix = "wgpbootrom",
        mwid      = nWorlds - 1,
        widWidth  = log2Ceil(nWorlds),
        nSlots    = nSlots,
        address   = 0x6004000,
        size      = 4096)
    )
  }
})
```

WorldGuard Implementation – Integration to Rocket SoC



```
class ChipyardSystem {
  BootROMParams ={
    P(WGBootROMKey) match {
      case Some(wgcParams) => {
        val wgc = WGCheckerAttachParams(wgcParams).attachTo(this);
        WGBootROM.attach(bootROMParams, this, CBUS, wgc.wgc_node)
      }
      case None => BootROM.attach(…);
    }
  }
```

Top Module



```
object WGBootROM {
    def attach(wgc_node: TLAdapterNode, …) : TLROM = {
     val bottom = new TOROM;
     bottom.node := wgc_node := tlbus
}
```

Simple to connect components

Related works and future work

PMP and WorldGuard

- *“The WorldGuard solution does not replace the RISC-V core Physical Memory Protection (PMP) mechanism or the Memory Management Unit (MMU), but can coexist with those mechanisms.”* — SiFive WG Tech. Paper

PMP and WorldGuard

	PMP	WorldGuard
Access Check	- Performed at core, before any transaction is initiated on the bus - Only checked accesses from cores	- Performed at the resources, can be out of core - Can check accesses from any initiators
Granularity	Limited to #PMP entries (Not scalable)	- Limited to #wgChecker entries (Not scalable)
Performance	- Need synchronization bet. Cores - Need PMP entry update if enclave switch	- No need synchronization bet. Cores - Need cache sanitization for dynamic config.

IOMMU and WorldGuard

- IOMMU
  - Address Translation (VA to PA)
    - Must access page tables in memory
  - Page-based permission check
  - Supports virtualized environments
  - Itself a device; has own initiator port to access page tables

- WorldGaurd
  - No address translation
  - Region-based permission check
    - Coarse-grain due to limited slots
  - Small code base

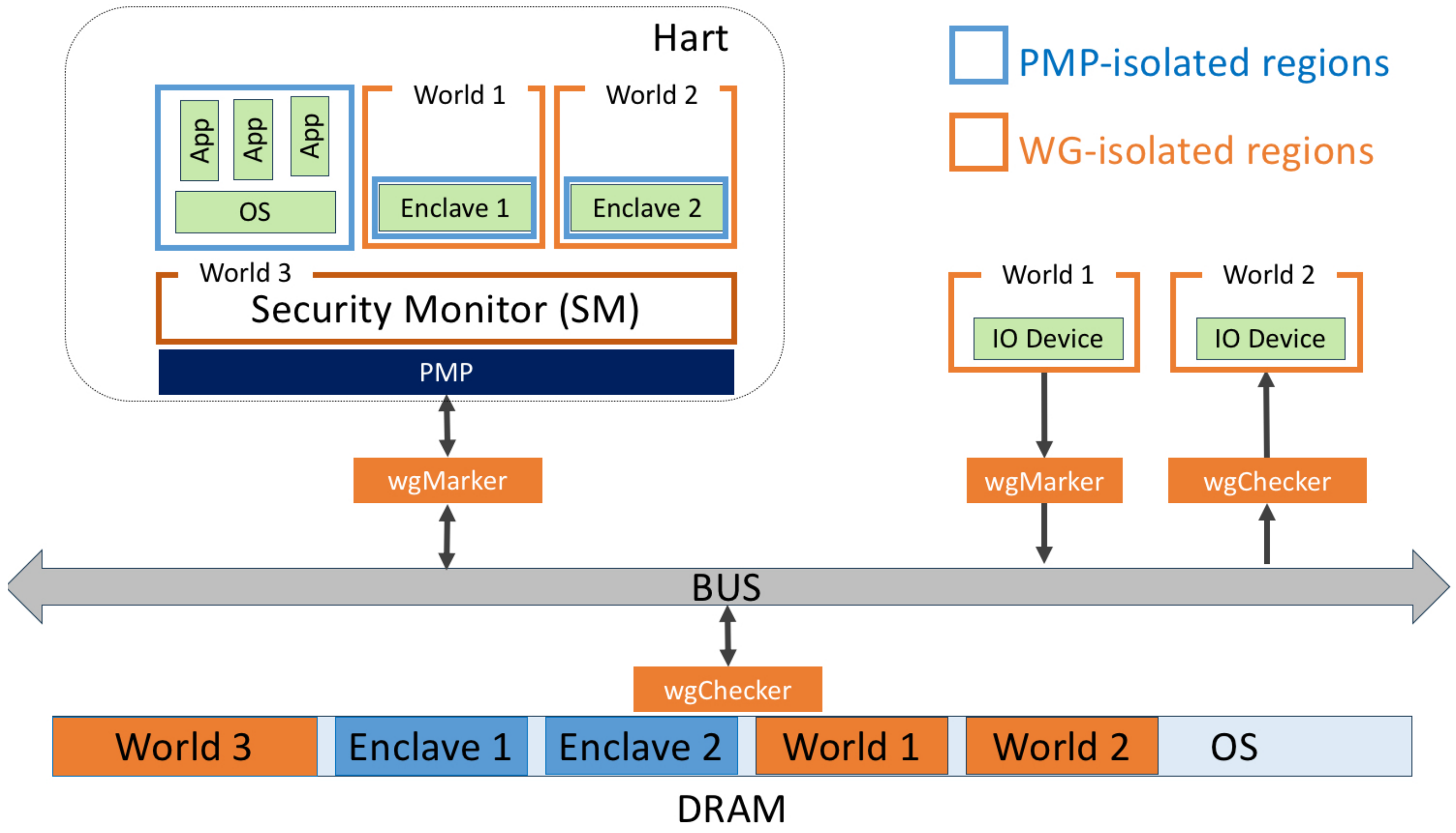
WorldGuard > PMP > MMU/IOMMU

- **WorldGuard** provide the coarsest and least-dynamic configurable isolation scheme, and effectively create a few very large worlds in the system at boot time.
- Within worlds, **PMP** can be used to isolate an arbitrary number of physical memory partitions that may vary dynamically.
- Within PMP physical memory partitions, **MMU and IOMMU** provides flexible dynamic page-level translation and protection for both compute and IO

Collaboration of PMP and WorldGuard for complete isolation

- Isolate statically allocated regions (FW, I/O devices) using WorldGuard
  - No extra works are required for enclave migration (e.g., PMP updates)
  - Useful in Robot, Car, and IoT devices with many sensors
- Isolate dynamically allocated regions (OS, enclaves) using PMP
  - It is easy for security monitor to reallocate memory region for difference enclaves

Collaboration of PMP and WorldGuard for complete isolation



Discussion - Number of World? Number of Slots?

- Current Max #Worlds = 32
- Current #Slots are not limited to the spec.
  - Not increase performan on check
  - But limited to area budg

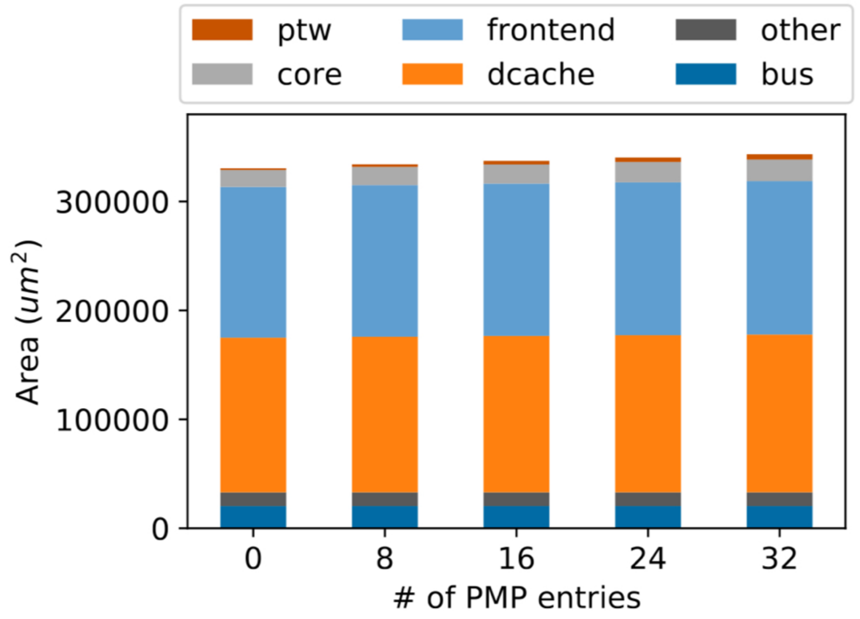


Figure 13: RocketChip Area vs. numbers of PMP entries.

Discussion – Core-driven vs. Process-driven

Discussion – Research topics

Thank you!

Sungkeun Kim
sk84.kim@samsung.com