RISC-V External Debug Security Specification

Version v0.7.1, 2025-04-10: Draft

Table of Contents

Preamble

This document is in the Development state

Expect potential changes. This draft specification is likely to evolve before it is accepted as a standard. Implementations based on this draft may not conform to the future standard.

Copyright and license information

Contributors

This RISC-V specification has been contributed to directly or indirectly by (in alphabetical order): Allen Baum, Aote Jin (editor), Beeman Strong, Gokhan Kaplayan, Greg Favor, Iain Robertson, Joe Xie (editor), Paul Donahue, Ravi Sahita, Robert Chyla, Tim Newsome, Ved Shanbhogue, Vicky Goode

Chapter 1. Introduction

Debugging and tracing are essential for developers to identify and rectify software and hardware issues, optimize performance, and ensure robust system functionality. The debugging and tracing extensions in the RISC-V ecosystem play a pivotal role in enabling these capabilities, allowing developers to monitor and control the execution of programs during the development, testing and production phases. However, the current RISC-V Debug specification grants the external debugger the highest privilege in the system, regardless of the privilege level at which the target system is running. It leads to privilege escalation issues when multiple actors are present.

This specification defines Debug Module Security Extension (non-ISA extension) and Sdsec (ISA extension) to address the above security issues in the current *The RISC-V Debug Specification* [1] and trace specifications [2] [3].

A summary of the changes introduced by *The RISC-V External Debug Security Specification* follows.

- **Per-Hart Debug Control:** Introduce per-hart states to control whether external debug is allowed in M-mode and/or supervisor domains [4].
- **Per-Hart Trace Control:** Introduce per-hart states to control whether tracing is allowed in M-mode and/or supervisor domains.
- **Non-secure debug:** Add a non-secure debug state to relax security constraints.
- **Debug Mode entry:** An external debugger can only halt the hart and enter debug mode when debug is allowed in current privilege mode.
- **Memory Access:** Memory access from a hart's point of view, using the Program Buffer or an Abstract Command, must be checked by the hart's memory protection mechanisms as if the hart is running at debug access privilege level; memory access from the Debug Module using System Bus Access must be checked by a system memory protection mechanism, such as IOPMP or WorldGuard.
- **Register Access:** Register access using the Program Buffer or an Abstract Command works as if the hart is running at debug access privilege level instead of M-mode privilege level. The debug CSRs (`dcsr` and `dpc`) are shadowed in supervisor domains while Smtdeleg [5] and Smstateen [6] extensions expose the trigger CSRs to supervisor domains.
- **Triggers:** Triggers (with action=1) can only fire or match when external debug is allowed in the current privilege mode.

1.1. Terminology

Abstract command	A high-level Debug Module operation used to interact with and control harts
Debug Access Privilege	The privilege with which an Abstract Command or instructions in the Program Buffer access hardware resources
Debug Mode	An additional privilege mode to support off-chip debugging
Hart	A RISC-V hardware thread
IOPMP	Input-Output Physical Memory Protection unit
M-mode	The highest privileged mode in the RISC-V privilege model
PMA	Physical Memory Attributes

PMP	Physical Memory Protection unit
Program buffer	A mechanism that allows the Debug Module to execute arbitrary instructions on a hart
Supervisor domain	An isolated supervisor execution context defined in RISC-V Supervisor Domains Access Protection [4]
Trace encoder	A piece of hardware that takes in instruction execution information from a RISC-V hart and transforms it into trace packets

Chapter 2. External Debug Security Threat Model

Modern SoC development involves several different actors who may not trust each other, resulting in the need to isolate actors' assets during the development and debugging phases. The current RISC-V Debug specification [1] grants external debuggers the highest privilege in the system, regardless of the privilege level at which the target system is running. This leads to privilege escalation issues when multiple actors are present.

For example, the owner of an SoC, who needs to debug their M-mode firmware, may be able to use the external debugger to bypass PMP lock (pmpcfg.L=1) and attack Boot ROM (the SoC creator’s asset).

Additionally, the RISC-V privilege architecture supports multiple software entities, or "supervisor domains," that do not trust each other. These supervisor domains are managed by a secure monitor running in M-mode, are isolated from each other by PMP/IOPMP, and may need different debug policies. The entity that owns the secure monitor wants to disable external debug when shipping the secure monitor; however, the entity that owns the supervisor domain needs to enable external debug to develop the supervisor domain. Since the external debugger is granted the highest privilege in the system, a malicious supervisor domain could compromise the M-mode secure monitor with the external debugger.

Chapter 3. Sdsec (ISA extension)

This chapter introduces the Sdsec ISA extension, which enhances the Sdext extension defined in *The RISC-V Debug Specification* [1]. The Sdsec extension provides privilege-based protection for debug operations and for triggers in Sdtrig [1]. Furthermore, it constrains trace functionality [2] according to RISC-V privilege levels.

3.1. External Debug

Chapter 3 of *The RISC-V Debug Specification* [1] outlines all mandatory and optional debug operations. The operations listed below are affected by the Sdsec extension; other operations remain unaffected. In the context of this chapter, **debug operations** refer to those listed below.

Debug operations affected by Sdsec:

- Halting the hart to enter Debug Mode
- Executing the Program Buffer
- Serving abstract commands (Access Register, Access Memory)

When external debug is disallowed at the current privilege level, the hart behaves as follows:

- The hart will not enter Debug Mode. Halt requests will remain pending until debug is allowed.
- Triggers with `action`=1 (enter Debug Mode) will not match or fire.

The subsequent subsections describe how external debug is authorized by M-mode debug control and supervisor domain debug control.

3.1.1. M-mode Debug Control

A state element in each hart, named `mdbgen`, is introduced to control the debuggability of M-mode for each hart as depicted in Figure 1. When `mdbgen` is set to 1, the following rules apply:

- The debug access privilege for the hart is M-mode. Abstract Commands, including "Quick Access", and Program Buffer execution operate with M-mode privilege.
- The debug operations are allowed when the hart executes in any privilege mode.

When `mdbgen` is set to 0, external debug is disallowed in M-mode. See Section 3.1 for how this impacts hart behavior in M-mode.

The `mdbgen` may be controlled through various methods, such as a new input port to the hart, a handshake with the system Root of Trust (RoT), or other methods. The implementation can choose to group several harts together and use one signal to drive their `mdbgen` state or assign each hart its own dedicated state. For example, a homogeneous computing system can use a signal to drive all `mdbgen` states to enforce a unified debug policy across all harts.

The `mdbgen` state for the Root-of-Trust (RoT) should be managed by SoC hardware, likely dependent on lifecycle fusing. The `mdbgen` for any other harts in the system should be managed by the RoT.

This specification assumes the controlling entity ensures `mdbgen` *shall never be set to* `0` *while the hart is in Debug Mode. Setting* `mdbgen` *to* `0` *while in Debug Mode could lead to undefined behavior; the hart may lose its debug privileges unexpectedly, potentially causing the debug session to fail or become insecure.*

3.1.2. Supervisor Domain Debug Control

The Smsdedbg extension [4] introduces the `sdedbgalw` field (bit 7) in CSR msdcfg to control the debuggability of supervisor domains. When `mdbgen=0`, the `sdedbgalw` field determines both the debug-allowed privilege modes and the debug access privilege, as illustrated in Table 1.

3.1.3. Debug Access Privilege

The **debug access privilege** is the privilege level for state accesses via the hart, such as Abstract Commands and Program Buffer execution. With Sdext, Debug Mode operates as if it has M-mode privilege. When Sdsec is implemented, Debug Mode accesses registers and memory using the **debug access privilege**. Attempts from Debug Mode to access state that requires a privilege level above the **debug access privilege** will fail and set `abstractcs.cmderr` to 3. The **debug access privilege** is derived as shown in Table 1.

Table 1. External Debug Configuration and Privilege

mdbgen	sdedbgalw	Debug allowed privilege modes	Debug access privilege
1	Don't care	All	M-mode
0	1	All except M	S-mode
0	0	None	N/A

The `sdcsr.dmprv` modifies the **effective debug access privilege** for loads and stores in Debug Mode by an S-mode debugger. When `sdcsr.dmprv=0`, the **effective debug access privilege** of loads and stores in Debug Mode follows Table 1; when `sdcsr.dmprv=1`, the **effective debug access privilege** of loads and stores in Debug Mode is represented by `sstatus.spp` and `hstatus.spv`. The `sdcsr.dmprv` does not affect the virtual-machine load/store instructions, HLV, HLVX, and HSV. The `sdcsr.dmprv` only takes effect when `mdbgen` is 0, and it is read-only 0 when `mdbgen` is 1.

The **effective debug access privilege** to memory by an M-mode debugger can be modified by `dcsr.mprven` and `mstatus.mprv` as specified in The RISC-V Debug Specification [1]. With Sdsec, the `dcsr.mprven` only takes effect when `mdbgen=1`, and it is ignored when `mdbgen=0`.

Table 2. Details of the `dmprv` *field in* `sdcsr`

Field	Description	Access	Reset
dmprv	0 (normal): The privilege level in Debug Mode is not modified. 1: In Debug Mode, the privilege level for load and store operations is modified and indicated by `sstatus.spp` and `hstatus.spv`.	WARL	0

3.1.4. Using EBREAK to Enter Debug Mode

EBREAK works as specified in The RISC-V Debug Specification [1] when external debug is allowed at

the running privilege level. When the hart is running at a debug-disallowed privilege level, EBREAK always raises a breakpoint exception.

3.1.5. Privilege Level Changing Instructions

The RISC-V Debug Specification [1] defines that the instructions that change the privilege mode have UNSPECIFIED behavior when executed within the Program Buffer, with the exception of the EBREAK instruction. In Sdsec, privilege-changing instructions (other than EBREAK) executed in the Program Buffer must either act as a NOP or raise an exception (stopping execution and setting `abstractcs.cmderr` to 3).

3.1.6. Interrupt during Single Stepping

Interrupts during single-step can be disabled by setting `dcsr.stepie=0`. When `mdbgen` is 1, `stepie` disables interrupts in all privilege modes for the hart. When `mdbgen` is `0` and `sdedbgalw` is 1, only delegated interrupts are disabled, while interrupts that trap to M-mode are not affected.

When debugging is only allowed for the supervisor domain, M-mode interrupts must not be disabled. Otherwise, debugging might impact the behavior of other parts of the system. For instance, if a crypto engine generates an interrupt to M-mode during single stepping, it will not be disabled if M-mode is debug-disallowed. The interrupt will be served upon exiting Debug Mode.

3.2. Trace

When Sdsec is supported, trace, as a non-intrusive debug method, will be constrained based on RISC-V privilege level. The availability of trace output is indicated through the hart-trace interface (HTI) [2].

3.2.1. M-Mode Trace Control

Each hart must add a new state element, `mtrcen`, which controls the availability of M-mode tracing. Setting `mtrcen` to 1 enables trace for both M-mode and the supervisor domain; setting `mtrcen` to `0` inhibits trace when the hart is running in M-mode.

Similar to M-mode debug control, `mtrcen` *may be controlled through various methods, such as a new input port to the hart, a handshake with the system Root of Trust (RoT), or other methods. The implementation may group several harts together and use one signal to drive their* `mtrcen` *state or assign each hart its own dedicated state.*

3.2.2. Supervisor Domain Trace Control

The Smsdetrc extension introduces the `sdetrcalw` field (bit 8) in CSR msdcfg within a hart. The trace availability for a hart in the supervisor domain is determined by the `sdetrcalw` field and `mtrcen`. If either `sdetrcalw` or `mtrcen` is set to 1, trace can be allowed when the hart runs in the supervisor domain.

When both `sdetrcalw` and `mtrcen` are set to `0`, trace is inhibited at all privilege levels.

3.3. Triggers (Sdtrig)

Triggers configured to enter Debug Mode can only fire or match when external debug is allowed, as outlined in Table 1.

Implementations must ensure that pending triggers intending to enter Debug Mode match or fire only when the hart is in a state where debug is allowed. For example, if an interrupt traps the hart to a debug-disallowed privilege mode, the trigger can only take effect either before the privilege is updated and control flow is transferred to the trap handler, or after the interrupt is completely handled and returns from the trap handler. The implementation must prevent Debug Mode from being entered in an intermediate state where privilege is changed or the PC is updated. This also applies to scenarios where a trigger is configured to enter Debug Mode before instruction execution and an interrupt occurs simultaneously.

3.3.1. M-mode Accessibility to `dmode` in `tdata1`

When the Sdsec extension is implemented, `dmode` is read/write for both M-mode and Debug Mode when `mdbgen` is 0, and remains only accessible to Debug Mode when `mdbgen` is 1.

M-mode is given write access to `dmode` to allow it to save/restore trigger context on behalf of a supervisor debugger. Otherwise, a trigger could serve as a side-channel to debug-disallowed supervisor domains. The trigger may raise a breakpoint exception in a supervisor domain where debugging is disallowed. This could allow the external debugger to indirectly observe the state from the debug-disallowed supervisor domain (PC, data address, etc.) and may even result in a Denial of Service (DoS). By making `dmode` M-mode accessible when `mdbgen` is 0, such an attack can be mitigated by having M-mode firmware switch the trigger context at the supervisor domain boundary.

3.3.2. External Triggers

The external trigger outputs (with `action` = 8/9) will not fire or match when the privilege level of the hart exceeds debug-allowed privilege as specified in Table 1.

The external trigger inputs (`tmexttrigger`) can be driven by any input signals, e.g., the external trigger output from another hart, interrupt signals, etc. The initiators of these signals are responsible for determining whether the signal is allowed to assert. The hart will not acknowledge the input until it is in a debug-allowed state. For example, if the external trigger input of hart *i* is connected to the external trigger output of hart *j*, the assertion of the output signal from hart *j* is determined by its own allowed privilege level for debug. Hart *i* will halt if `tmexttrigger.action` is 1, when it is in a debug-allowed state and hart *j* asserts the output signal.

3.4. CSRs

3.4.1. Extension of Debug Mode CSR

The `dcsr`, `dpc`, and `dscratch0/1` are accessible in Debug Mode only if `mdbgen`=1; otherwise, the access will fail and `abstract.cmderr` is set to 3 (exception). The `sdcsr` and `sdpc` (see Section 3.4.1.2) are always accessible in Debug Mode.

When external debug is disallowed at the current privilege level, the configuration in `dcsr` and `sdcsr`

will be ignored as if they are cleared to 0.

Dcsr

With Sdsec, the maximum privilege level that can be configured in prv and v is determined in Table 3. The fields retain legal values when the prv and v are configured with an illegal privilege level. Illegal privilege levels include unsupported levels and any level higher than the maximum allowed debug privilege.

Table 3. Maximum Allowed Resume Privilege Mode

mdbgen	sdedbgalw	Maximum privilege allowed on resume
1	Don't care	M
0	1	S(HS)
0	0	None

Sdcsr and sdpc

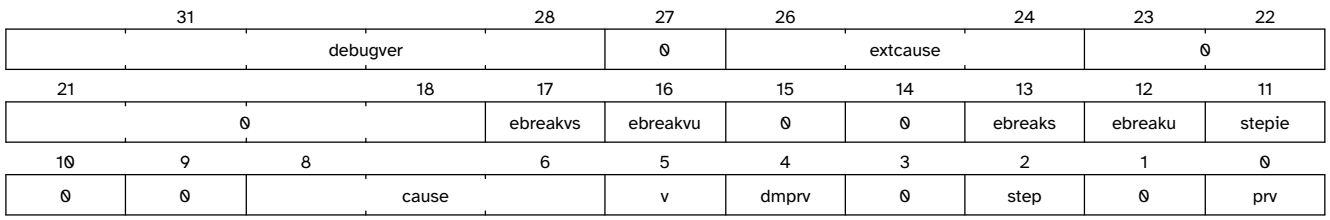
The sdcsr and sdpc registers provide supervisor read/write access to the dcsr and dpc registers respectively. Moreover, the sdcsr adds dmprv to modify the **effective debug access privilege** in S-mode. Both registers are only accessible in Debug Mode.

Table 4. Allocated addresses for supervisor shadow of Debug Mode CSR

Number	Name	Description
0xaaa	sdcsr	Supervisor debug control and status register.
0xaaa	sdpc	Supervisor debug program counter.

The sdcsr register exposes a subset of dcsr, formatted as shown in Register 1, while the sdpc register provides full access to dpc.

Unlike dcsr and dpc, the dscratch0/1 registers do not have a supervisor access mechanism, and external debuggers with S-mode privilege cannot use them.



Register 1: Supervisor debug control and status register (sdcsr)

The nmip, mprven, stoptime, stopcount, ebreakm, and cetrig fields in dcsr are configurable only by M-mode; they are masked in sdcsr, while prv[1] is hardwired to 0 in sdcsr. The field for mprven is reclaimed by dmprv in sdcsr layout to avoid waste of fields.

The dmprv field is added as bit 4 in sdcsr to modify the **effective debug access privilege** for memory load and store accesses, as defined in Section 3.1.3.

3.4.2. Extension of Sdtrig CSR

The Smtdeleg [5] and Smstateen [6] extensions define the process for delegating triggers to modes with lower privilege than M-mode. The Sdsec requires both extensions to securely delegate Sdtrig triggers to the supervisor domain.

When M-mode enables debugging for the supervisor domain, it can optionally delegate the triggers to the supervisor domain, allowing an external debugger with S-mode privilege to configure these triggers.

3.4.3. Debug Control CSR

The CSR `msdcfg`, holding the debug and trace control for the supervisor domain (`sdedbgalw` and `sdetrcalw`), is defined in *RISC-V Supervisor Domains Access Protection* [4]. The Smsdedbg and/or Smsdetrc extensions must be implemented to support security control for debugging and/or tracing in supervisor domains.

Chapter 4. Debug Module Security (non-ISA) Extension

This chapter defines the required security enhancements for the Debug Module. The debug operations listed below are modified by the extension.

- Halt
- Reset
- Keepalive
- Abstract commands (Access Register, Quick Access, Access Memory)
- System bus access

If any hart in the system implements the Sdsec extension, the Debug Module must also implement the Debug Module Security Extension.

4.1. External Debug Security Extensions Discovery

The ISA and non-ISA external debug security extensions impose security constraints and introduce non-backward-compatible changes. The presence of the extensions can be determined by polling the `allsecured` and/or `anysecured` bits in `dmstatus` Table 5. If the field `allsecured` or `anysecured` is set to 1, it indicates that all or any selected harts implement the Sdsec extension. When any hart implements the Sdsec extension, it indicates that the Debug Module implements the Debug Module Security Extension as described in this chapter.

4.2. Halt

The halt behavior for a hart is detailed in Section 3.1. According to *The RISC-V Debug Specification* [1], a halt request must be responded to within one second. However, this constraint must be removed as the request might be pending due to situations where debugging is disallowed. In the case of a halt-on-reset request (`setresethaltreq`), the request is only acknowledged by the hart once it has reached a privilege level in which debug is allowed.

4.3. Reset

The `hartreset` operation resets selected harts. When M-mode is disallowed to be debugged, the hart will raise a security fault error to the Debug Module on `hartreset` operations. The error can be monitored through `allsecfault` and `anysecfault` in `dmstatus`.

The `ndmreset` operation is a system-level reset not tied to hart privilege levels and resets the entire system (excluding the Debug Module). The Debug Module Security Extension makes `ndmreset` read-only `0`. The debugger can determine support for the ndmreset operation by setting the field to 1 and subsequently verifying the returned value upon reading.

4.4. Keepalive

The keepalive bit serves as an optional request for the hart to remain available for debugging. This bit only takes effect when M-mode is allowed to be debugged; otherwise, the hart behaves as if the bit is not set.

4.5. Abstract Commands

The hart's response to abstract commands is detailed in Section 3.1. The following subsection delineates the constraints when the Debug Module issues an abstract command.

4.5.1. Relaxed Permission Check `relaxedpriv`

The `relaxedpriv` field is hardwired to 0.

4.5.2. Address Translation `aamvirtual`

The field `command.aamvirtual` determines whether the Access Memory command uses a physical or virtual address. When `mdbgen`=1, the behavior follows the original RISC-V Debug Specification [1]. When `mdbgen`=0:

- If `aamvirtual`=0, the hart responds with a security fault error (setting `abstractcs.cmderr` to 6).
- If `aamvirtual`=1, addresses are treated as virtual and are translated and protected according to the **effective debug access privilege**, as defined in Section 3.1.3.

4.5.3. Quick Access

When M-mode debugging is disallowed (`mdbgen`=0) for a hart, any Quick Access operation will be discarded by the Debug Module, causing `abstractcs.cmderr` to be set to 6.

Quick Access abstract commands initiate a halt, execute the Program Buffer, and resume the selected hart. These halts cannot remain pending until debugging is allowed because the debugger blocks while waiting for completion. To address this, the hart would need to distinguish between Quick Access and other halt requests. Since Quick Access is an optional optimization, the Debug Module Security Extension avoids this additional hardware complexity by disallowing Quick Access when `mdbgen` *is 0.*

4.6. System Bus Access

The System Bus Access must be checked by bus initiator protection mechanisms such as IOPMP [7], WorldGuard [8]. The bus protection unit can return an error to the Debug Module on illegal accesses; in that case, the Debug Module will set `sberror` of `sbcs` to 6 (security fault error).

Trusted entities like RoT should configure IOPMP or equivalent protection before granting debug access to M-mode. Similarly, M-mode should apply the protection before enabling supervisor domain debug.

4.7. Security Fault Error Reporting

A dedicated error code, security fault error (cmderr 6), is included in `cmderr` of `abstractcs`. Issuance of abstract commands under disallowed circumstances sets `cmderr` to 6. Additionally, the bus security fault error (sberror 6) is introduced in `sberror` of `sbcs` to denote errors related to system bus access.

Error status bits are internally maintained for each hart. The `allsecfault` and `anysecfault` fields in

`dmstatus` indicate the error status of the currently selected harts. These error statuses are sticky and can only be cleared by writing 1 to `acksecfault` in `dmcs2` Table 6.

4.8. Non-secure Debug

The state element `nsecdbg` is introduced to retain full debugging capabilities, as if the extensions in this specification were not implemented. When `nsecdbg` is set to 1:

- All debug operations are executed with M-mode privilege (equivalent to having `mdbgen` set to 1) for all harts in the system.
- The ndmreset operation is allowed.
- The `relaxedpriv` field may be configurable.
- System Bus Access may bypass the bus initiator protections.
- Trace output is enabled in all privilege modes.

4.9. Update of Debug Module Registers

The Debug Module Security Extension introduces new fields in the Debug Module registers. In `dmstatus`, `anysecured` and `allsecured` are added at bit 20 and bit 21 respectively, while `anysecfault` and `allsecfault` are added at bit 25 and bit 26. The `acksecfault` field is added to `dmcs2` at bit 12.

Table 5. Details of newly introduced fields in dmstatus

Field	Description	Access	Reset
allsecured	The field is 1 when all currently selected harts implement the Sdsec extension	R	-
anysecured	The field is 1 when any currently selected hart implements the Sdsec extension	R	-
allsecfault	The field is 1 when all currently selected harts have raised a security fault due to reset or keepalive operation	R	-
anysecfault	The field is 1 when any currently selected hart has raised a security fault due to reset or keepalive operation	R	-

Table 6. Detail of acksecfault in dmcs2

Field	Description	Access	Reset
acksecfault	0 (nop): No effect. 1 (ack): Clears error status bits for any selected harts.	W1	-

Appendix A: Theory of Operation

This chapter explains the theory of operation for the External Debug Security Extension. The subsequent diagram illustrates the reference implementation of security control for the debug and trace, respectively.

A.1. Debug Security Control

As outlined in the specification, the dedicated debug security policy for a hart is enforced by platform state `nsecdbg`, hart state `mdbgen`, and the `sdedbgalw` field inside the `msdcfg` CSR. Both the `nsecdbg` and `mdbgen` states can be accommodated in MMIO outside the harts, such as in the Debug Module registers, or implemented as fuses.

The security control logic validates all debug requests and triggers (with action=1) firing/matching based on `nsecdbg`, `mdbgen`, and `sdedbgalw` against the privilege level of the hart. Debug requests that fail validation will either be dropped or kept pending. Additionally, the platform-specific external trigger inputs must obey platform constraints, which must be carefully handled by the platform implementation.

When `nsecdbg` is set to `0`, the validation process involves two actors, which may lead to a potential Time-of-Check Time-of-Use (TOCTOU) issue. To mitigate this, the implementation must ensure that both the validation and execution of debug requests occur under the same privilege level and the same debug security policy. Failing to do so may allow debug requests to bypass security controls.

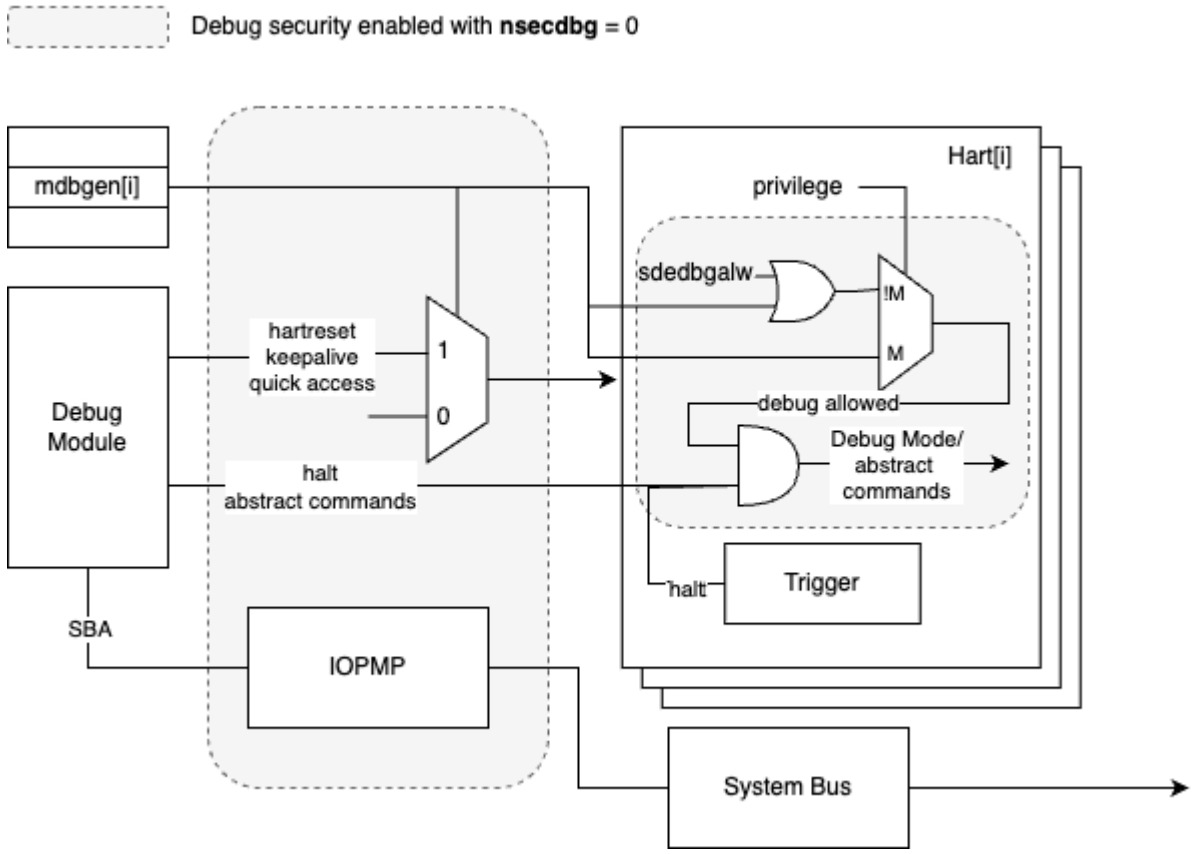


Figure 1. The debug security control

When the external debugger is using `dcsr.step` or `icount` to step over an instruction that triggers a transition to a higher privilege level, the security control logic must check that the resulting privilege level is debuggable (see Table 1). If so, the hart will enter Debug Mode before the first instruction in

the trap handler executes. If not, Debug Mode entry will remain pending until the hart returns to a debuggable privilege level.

Similar to the scenario described in The RISC-V Debug Specification [1] Appendix B.3, the trap handler might restart the instruction in case of page faults, emulated FPU instructions, or interrupts, which can result in single-stepping not passing an instruction. To help users, the external debugger should perform an extra step when the PC does not change during a regular step.

Application-level debugging is primarily accomplished through self-hosted debugging, allowing the management of debug policies by supervisor domains. As a result, user-level debugging management is not addressed within this extension.

A.2. Trace Security Control

Similar to debug security, trace is controlled by platform state `nsecdbg`, hart state `mtrcen`, and `sdetrcalw` in CSR `msdcfg` for each hart. The sec_inhibit sideband signal indicates the availability of trace to the trace encoder.

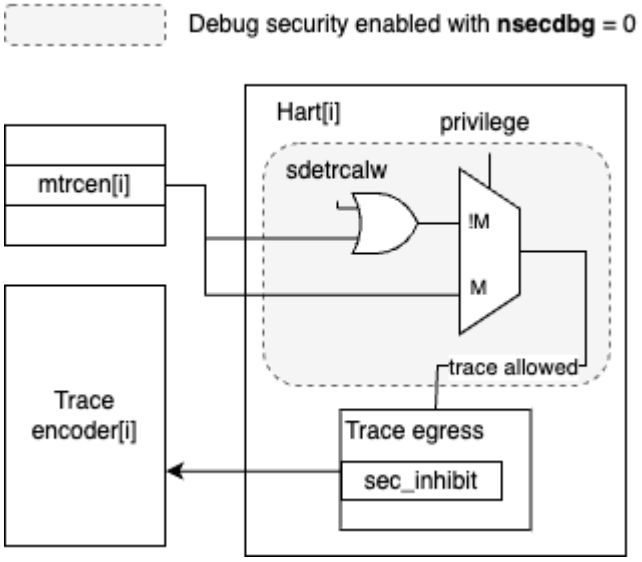


Figure 2. The trace security control

Appendix B: Execution Based Implementation with Sdsec

In an execution-based implementation, the code executing the "park loop" can always run with M-mode privilege to access the memory and CSR. However, once execution is dispatched to an Abstract Command or the program buffer, the privilege level for accessing memory and CSR should be restricted to debug access privilege.

To achieve this, a Debug Mode only state element (e.g., a field in a custom CSR) may be introduced to control the privilege level in Debug Mode. When the state is set to 1, Debug Mode allows M-mode privilege; when cleared to 0, it enforces the debug access privilege. The hardware sets this state to 1 upon entering the park loop and clears it to 0 by the final instruction of the park loop, right before execution is transferred to an Abstract Command or the program buffer.

Bibliography

[1] “RISC-V Debug Specification.” [Online]. Available: github.com/riscv/riscv-debug-spec.

[2] “RISC-V Efficient Trace for RISC-V.” [Online]. Available: github.com/riscv-non-isa/riscv-trace-spec.

[3] “RISC-V N-Trace (Nexus-based Trace) Specification.” [Online]. Available: github.com/riscv-non-isa/tg-nexus-trace.

[4] “RISC-V Supervisor Domains Access Protection.” [Online]. Available: github.com/riscv/riscv-smmtt.

[5] “RISC-V Trigger Delegation Specification.” [Online]. Available: github.com/riscv/ft-trigger-delegation.

[6] “RISC-V State Enable Extension.” [Online]. Available: github.com/riscvarchive/riscv-state-enable.

[7] “RISC-V IOPMP Architecture Specification.” [Online]. Available: github.com/riscv-non-isa/iopmp-spec/releases.

[8] “WorldGuard Specification.” [Online]. Available: github.com/riscv-admin/security/blob/main/papers/worldguard%20proposal.pdf.