RISC-V External Debug Security Specification

Version v0.5.0, 2024-09-24: 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 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 and trace specification grants the external debugger 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 non-ISA extension Debug Module Security Extension (non-ISA extension) and ISA extension 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:** External debugger can only halt the hart and enter debug mode when debug is allowed in current privilege mode; all operations are executed with debug access privilege instead of M-mode privilege.
- **Memory Access:** Memory access from a hart's point of view using a Program Buffer or the Abstract Command must be checked by the hart's memory protection mechanisms as if the hart is running at debug access privilege; memory access from Debug Module using System Bus Access block without involving a hart must be checked by system memory protection mechanism, such as IOPMP or WorldGuard.
- **Register Access:** Register access using Program Buffer or the Abstract Command works as if the hart is running in debug access privilege instead of M-mode privilege. The debug CSRs (`dcsr` and `dpc`) are shadowed in supervisor domains while Smtdeleg/Sstcfg extensions expose the trigger CSRs to supervisor domains through indirect CSR access.
- **Triggers:** Triggers (with action=1) can only fire or match when external debug is allowed in current privilege.

1.1. Terminology

Abstract command	A high-level command in Debug Module used to interact with and control harts
Debug Access Privilege	The privilege with which abstract commands or instructions in program buffers 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 buffer in Debug Module to execute arbitrary instructions on a hart
Supervisor domain	A 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 consists of 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 a 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, RISC-V privilege architecture supports multiple software entities or "supervisor domains" that do not trust each other. The supervisor domains are managed by secure monitor running in M-mode, they are isolated from each other by PMP/IOPMP and they may need different debug policies. The entity that owns 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 will be the granted highest privilege in the system, a malicious supervisor domain will be able to compromise 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 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 Program buffer
- Serving abstract commands (Access Register, Access Memory)

When external debug is disallowed in running privilege level, the hart behaves as the following:

- The hart will not enter Debug Mode. Halt requests will remain pending until debug is allowed.
- Triggers with `action`=1 will not match or fire.
- Abstract commands without halting will be dropped and set `cmderr` to 6.

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

A pending request to enter Debug Mode can dynamically change from a disallowed state to an allowed state due to updates in debug controls. For example, once the software completes executing confidential code, it can grant debuggability for an external debugger. Afterwards, the software can enter a while(1) loop, waiting for the debugger to take control and break out of the loop.

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 can be configured to any legal privilege level
- The debug operations are permitted when the hart executes in all modes
- Abstract commands without halting the hart carries M-mode privilege if supported

When `mdbgen` is set to 0, the debug operations are disallowed and the behaviors applies when the hart runs in M-mode.

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 state to enforce a unified debug policy across all harts.

3.1.2. Supervisor Domain Debug Control

The Smsdedbg extension [4] introduces sdedbgalw field (bit 7) in CSR msdcfg to control the debuggability of supervisor domains. The sdedbgalw along with mdbgen determines the debug allowed privilege levels, as illustrated in Table 1. The debug access privilege can only be configured to debug allowed levels.

Table 1. External debug allowed privilege levels per debug controls

mdbgen	sdedbgalw	Debug allowed privilege levels
1	Don't care	All
0	1	All except M
0	0	None

When debug is allowed in supervisor domain, debug operations are allowed when hart executes in supervisor domain. The abstract commands without halting the hart carries supervisor mode privilege if supported.

3.1.3. Debug Access Privilege

The **debug access privilege** is defined as the privilege level granted to the external debugger to access hardware resources with abstract commands or program buffers. Memory and register accesses from Debug Mode also carry **debug access privilege** instead of always with M-mode. The **debug access privilege** is represented by the prv and v fields in dcsr or sdcsr. The legal privilege levels programmable to the fields in Debug Mode are elaborated in Section 3.1.3.1. Debugger accesses to registers and memory will be checked by permission check mechanisms against **debug access privilege**, and trap if they violate corresponding rules.

Configuring External Debugger Access Privileges

The prv and v fields have been modified to authorize privilege for external debug accesses. Upon transitioning into Debug Mode, the prv and v fields are updated to the privilege level the hart was previously operating in. The maximum debug privilege level that can be configured in prv and v is determined in Table 2. 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. When the hart resumes from Debug Mode, the current privilege mode and virtualization mode are changed to that specified by prv and v.

Table 2. Determining maximum debug access privilege with mdbgen and sdedbgalw

mdbgen	sdedbgalw	Maximum debug privilege allowed
1	Don’t care	M
0	1	S(HS)
0	0	None

As the `prv` and `v` fields are Write Any Read Legal (WARL) fields, the external debugger is able to read back the written value to determine the maximum debug privilege level.

3.1.4. 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 exception of the ebreak instruction. In Sdsec, those instructions including mret, sret, uret, ecall, must either act as NOP or trigger an exception (stopping execution and setting `cmderr` to 3) in Program Buffer. Notably, these instructions retain their normal functionality during single stepping.

3.1.5. Interrupt during Single Stepping

The interrupt can be disabled by `stepie` in `dcsr` during single stepping. When `mdbgen` is 1, `stepie` disables interrupts in all privilege modes for the hart. When `mdbgen` is 0 and `sdedbgalw` is 1, only interrupts delegated to the supervisor domain 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 example, if a context switch for the supervisor domain triggered by a timer interrupt is suppressed, some real-time workloads might not be completed on time, resulting in unexpected errors.

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 interface defined in <[_reference to the trace interface doc_]> to trace module.

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

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

3.3. Trigger (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

When 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.

The dmode being read/write allows M-mode to switch trigger context. The trigger can form a side-channel to debug disallowed supervisor domains from a debug allowed supervisor domain if the trigger context is not switched. Although the trigger cannot fire or match in disallowed supervisor domain to enter Debug Mode, the malicious debugger can exploit it by setting a trigger to raise breakpoint exception (action = 0) when it is in debug allowed supervisor domain. If the trigger hits in debug disallowed supervisor domain, the external debugger can indirectly observe the executed PC, accessed memory address or read/write data in debug disallowed supervisor domain by the checking value in hit0/hit1. As the dmode is accessible when mdbgen is 0, such attack can be mitigated by having M-mode firmware switch the trigger context at 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 input can be driven by any input signals, e.g. the external trigger output from another hart or interrupt signals etc. The input signals cause the trigger (with `action` = 1) to fire only when the hart is allowed to debug. The initiators of these signals are responsible for determining whether the signal is allowed to assert. For example, if the external trigger input of hart i is connected to external trigger output of hart j. The assertion of output signal from hart j is determined by its own allowed privilege level for debug. The output signal of hart j must not assert when debug is disallowed. Similarly, signals from other module in the system are managed by the individual module. When the module is not allowed to debug, the signal connected to external trigger input must not be asserted.

3.3.3. Trigger chain

The privilege level of the trigger chain is determined by the trigger enabled for the highest privilege level inside the chain. The entire trigger chain cannot be modified if the chain privilege level exceeds debug allowed privilege level.

This represents a balance between usability and hardware complexity. There may be instances where the triggers are linked across different privilege levels (e.g., from S-mode to M-mode), while the external debugger may only have access with S-mode privilege. The external debugger should not modify the chain, because it could be suppressed or incorrectly match or fire in M-mode.

3.4. CSRs

3.4.1. Extension of Sdext CSR

The `sdcsr` and `sdpc` registers provide supervisor read/write access to the `dcsr` and `dpc` registers respectively. They are only accessible in Debug Mode.

Table 3. Allocated addresses for supervisor shadow of Debug Mode CSR

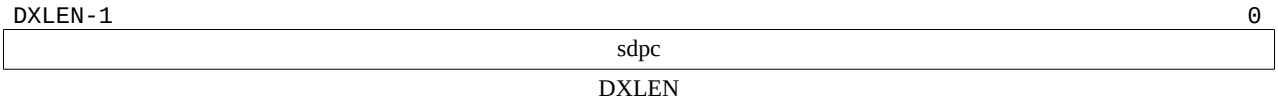
Number	Name	Descirption
0xaaa	sdcsr	Supervisor debug control and status register.
0xaaa	sdpc	Supervisor debug program counter.

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 scratch registers do not have supervisor access, and external debuggers with S-mode privilege cannot not use them as scratch memory.

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, masked from `sdcsr` while the `prv` field is constrained to 1 bit.



Register 2: Supervisor debug program counter (sdpc)

3.4.2. Extension of Sdtrig CSR

The Smtdeleg and Sstcfg 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 supervisor domain.

When M-mode enables debugging for 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 holding the debug and trace contol knobs for supervisor domain are specified in Smsdedbg and Smsdetrc extension respectively in *RISC-V Supervisor Domains Access Protection* [4]. The Smsdedbg and/or Smsdetrc extension must be implemented to support security control for debugging and/or tracing in supervisor domain.

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

This chapter outlines the security enhancements defined for the Debug Module as non-ISA extension. The debug operations listed below are modified by the non-ISA extension. All features in this chapter must be implemented in Debug Module to achieve external debug security. If any hart in the system implements the Sdsec extension, the Debug Module must also implement the non-ISA extension. The debug operations affected by the non-ISA extension include:

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

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` or/and `anysecured` bits in dmstatus Table 4. If the field `allsecured` or `anysecured` is set to 1, it represents that all or any selected harts adopt the Sdsec extension. When any hart adopts the Sdsec extension, it indicates the Debug Module implements 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 within one second. However, this constraint must be removed as the request might be pending due to the situations where debugging is disallowed. In the case of halt-on-reset request, the request is only acknowledged by the hart when it is permitted to debug after the deassertion of reset. Besides, when a Quick Access abstract command is issued to a hart while the hart is not yet allowed to debug, the Quick Access cannot halt the hart, and the Debug Module will receive a security error fault (`cmderr`=6).

The halt action in Quick Access is handled differently because other types of halts can be canceled by external debugger when debugging is disallowed, while the Quick Access command can only complete successfully or respond with an error. Otherwise, the debug interface to the selected hart will appear to be hung. Additionally, reset is not always applicable to the hart to recover from a hang situation. To avoid such situations, the halt action in Quick Access must be handled separately by the hart.

4.3. Reset

The hartreset operation resets selected harts. When M-mode is not allowed to be debugged, the hart will raise a security fault error to Debug Module. The debugger could monitor the error by polling `allsecfault` or/and `anysecfault` in dmstatus.

The ndmreset operation is a system-level reset not tied to hart privilege levels and reset the entire system (excluding the Debug Module). It can only be secured by the system. Thus, it must be de-featured. 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 the abstract commands.

4.5.1. Relaxed Permission Check `relaxedpriv`

The `relaxedpriv` field is hardwired to 0.

4.5.2. Address Translation `aamvirtual`

The field `aamvirtual` in the command (at 0x17 in the Debug Module) determines whether the Access Memory command uses a physical or virtual address. When an Access Memory command is issued with `aamvirtual`=0, the hart must check whether the physical access is allowed to access memory. The hart responds with an exception to the Debug Module when M-mode is not permitted to debug, `tvm` (in mstatus) is set to 1, and `mode` (in satp) enables any kind of virtual translation. In the event of an exception, the Debug Module set `cmderr` to 3 and clear the data registers to 0.

4.5.3. Quick Access

When M-mode debugging is not allowed (`mdbgen`=0) for a hart, any Quick Access operation will be discarded, causing `cmderr` being set to 6.

Quick Access abstract commands effect a halt, execution of Program Buffer, and resume of the selected hart. However, it is undesirable for these Quick Access halts to remain pending until debug is allowed, since the debugger blocks while waiting for the Quick Access to complete. Returning an error only for Quick Access commands received when debug is not allowed would require the hart to distinguish between Quick Access halt requests and other halt requests. Because Quick Access is merely an optimized flow and not required for any usage models,

it was decided to avoid burdening the hart with extra hardware. Therefore, Quick Access is forbiden when `mdbgen` is 0.

4.6. System Bus Access

The System Bus Access must be checked by bus initiator protection mechanisms such as IOPMP [5], WorldGuard [6]. The bus protection unit can return error to Debug Module on illegal access, in that case, Debug Module will set `sberror` 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` (at 0x16 in Debug Module). Issuance of abstract commands under disallowed circumstance sets `cmderr` to 6. Additionally, the bus security fault error (sberror 6) is introduced in `sberror` of `sbcs` (at 0x38 in Debug Module) to denote errors related to system bus access.

The error raised by resethaltreq, reset can be identified through the fields `allsecfault` and `anysecfault` in dmstatus. Error status bits are internally maintained for each hart, with the `allsecfault` and `anysecfault` fields indicating 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.

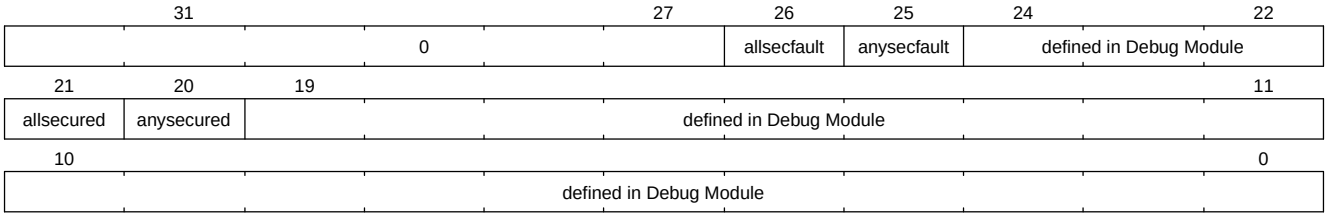
4.8. Non-secure Debug

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.

[NOTE] During the early stages of a chip's lifecycle, such as when developing the boot process, it is desirable to debug from the initial system state. The `nsecdbg` should only be set to 1 when the entire system is authorized for unrestricted debugging and tracing.

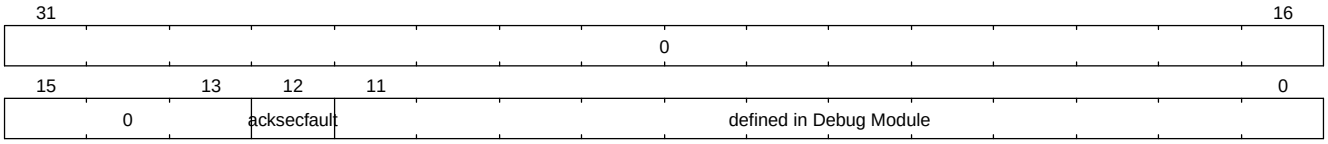
4.9. Update of Debug Module Registers



Register 3: Newly introduced fields in dmstatus

Table 4. Details of newly introduced fields in dmstatus

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



Register 4: Newly introduced acksecfault in dmcs2

Table 5. 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 Module and trace encoder, respectively.

A.1. Debug Module security control

As outlined in the specification, the security control on the Debug Module can vary for each hart. The dedicated security policy for hart i is enforced by the input port mdbgen[i] and the `sdedbgalw` field inside CSR msdcfg. The security control logic examines all debug operations and triggers (with action=1) firing/matching based on mdbgen[i], `sdedbgalw`, and the privilege level of the hart. The failed action will either be dropped or pending. Additionally, the platform-specific external trigger inputs must obey to platform constraints, which must be carefully handled by platform owner. The mdbgen[i] can be bundled in an MMIO (Memory-Mapped I/O) outside the hart, such as in the Debug Module, or implemented as fuses.

The privilege level of the hart is determined by code execution, while the debug requests are validated against the privilege level generated by the hart. This 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 the inspection and execution of debug requests occur within the same privilege level of the hart. Failure to do so could result in debug requests bypassing access controls intended for higher privilege levels. If the accesses fail the security check, it must prompt an immediate termination of access to prevent any information leakage.

When the external debugger is stepping through an instruction that triggers a transition to a higher privilege level, the security control logic must verify against debug capability according to Table 1 before entering Debug Mode. If debugging is permitted, the hart re-enters Debug Mode after executing the instruction. Otherwise, the hart continues executing with the pending single step request until it becomes debuggable and can re-enter Debug Mode. In scenarios where multiple supervisor domains are debuggable, the secure monitor in M-mode may switch the context during single stepping. In such cases, the debugger might stop in a different application than the original one. Users of the debugger should be mindful of this possibility.

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

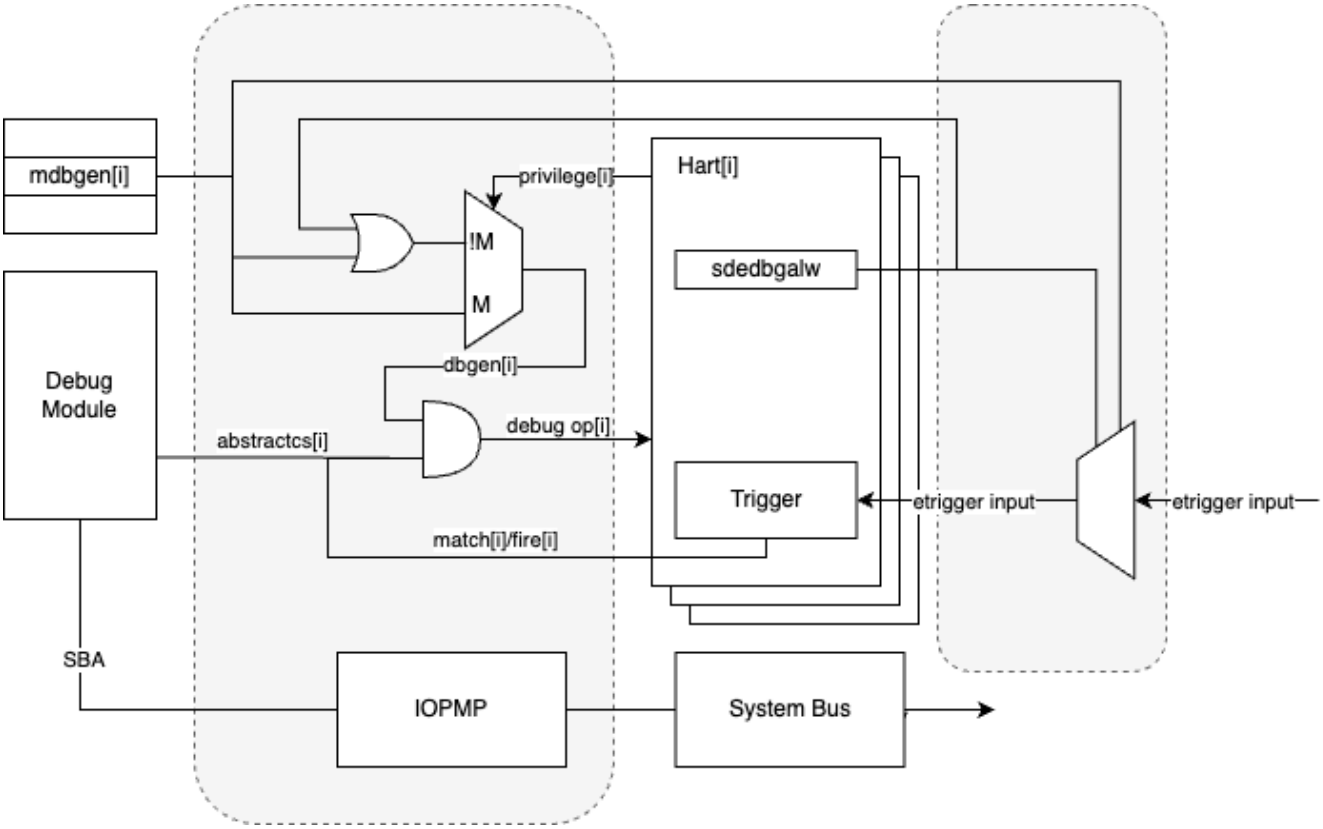


Figure 1. The security control on Debug Module

A.2. Trace Encoder security control

Similar to the Debug Module, the trace encoder is controlled by the mtrcen[i] and `sdetrcalw` in CSR msdcfg for each hart i. The halted sideband signal to the trace encoder is determined by [trcctl].

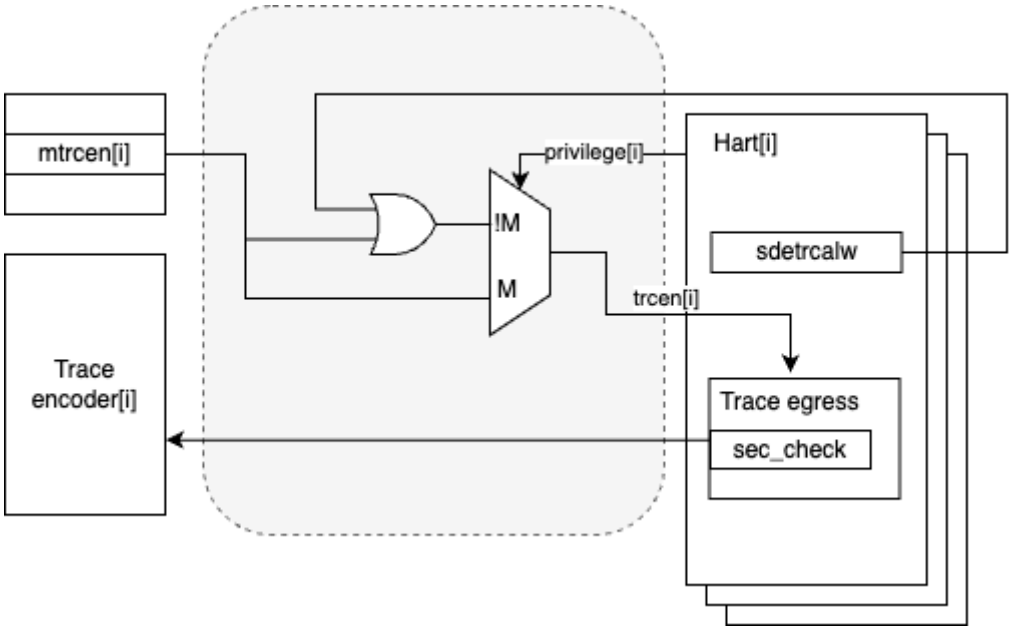


Figure 2. The security control on trace module

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

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