RISC-V External Debug Security Extension

Version v0.5.0, 2024-04-23: 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): Aote Jin (editor), Beeman Strong, Gokhan Kaplayan, Greg Favor, Iain Robertson, Joe Xie (editor), Ravi Sahita, Robert Chyla, Ved Shanbhogue, Vicky Goode

Chapter 1. Introduction

Debugging and tracing are essential tools utilized by 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 and testing phases. However, ensuring the security of this interface is of paramount importance to prevent unauthorized access and potential security breaches. Without adequate security measures, these functions can be exploited by malicious actors to gain unauthorized access, compromise system integrity, and extract sensitive information.

The extension is complementary to RISC-V Debug Specification and Efficient Trace for RISC-V, ensuring robust security. It comprises of ISA part and non-ISA part. The ISA part defines the enhancement inside the hart while the non-ISA part enforces the security rules on Debug Module.

Chapter 2. External Debug Security Threat model

The growing complexity of modern System-on-Chip (SoC) designs has led to a corresponding increase in the need for effective debugging capabilities. However, the use of debugging functions also introduces potential security vulnerabilities that can be exploited by attackers to gain unauthorized access to sensitive information or perform malicious actions on the system.

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 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.L 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.

This specification defines non-ISA extensions and ISA extensions to address the above security issues in the current RISC-V Debug specification.

Chapter 3. Zedsec (ISA extension)

This chapter introduces the Zedsec ISA extension, which extends the ISA section of The RISC-V Debug Specification. The RISC-V hart must implement all features to ensure external debug security. It is designed to enforce access control over operations initiated by the Debug Module, as well as constraints on trigger behaviors [1]. Additionally, it incorporates trace functionality [2], with its output undergoing check based on hart privilege levels.

3.1. External Debug

By default, the extension forbids the following debug operation in The debug operations affected by Zedsec extension unless they are explicitly granted to the external debugger.

The debug operations affected by Zedsec extension

- Entering Debug Mode
- Executing Program buffer
- Serving abstract commands (Access Register, Quick Access, Access Memory)

When the debug operations are not granted, all the above operations issued by Debug Module or trigger will be pending or dropped. The subsequent subsections detail how debug operations are granted.

3.1.1. Debug Access Privilege

The **debug access privilege** is defined as the privilege with which abstract commands or instructions in program buffers access hardware resources such as registers and memory. This privilege operates independently of hart privilege levels and exclusively affects operations within Debug Mode. Memory and register access within Debug Mode are subject to the **debug access privilege**, with all hardware protections, including MMU, PMP, and PMA, checked against it. This privilege is represented by the `prv` and `v` fields in `dcsr`, and it is updated to reflect the hart privilege level upon entering Debug Mode. Each hart has a dedicated **debug access privilege** and it may vary from each other. The permissible privilege levels programmable to `dcsr` in Debug Mode are elaborated in subsequent sections.

In addition, the `mprv` and `mpp` fields take effect exclusively when the **debug access privilege** is in machine mode.

Table 1. Determining maximum debug access privilege with mdbgen[i] and sdedbgalw

mdbgen[i]	sdedbgalw	Maximum debug privilege
1	x	M
0	1	S(HS)
0	0	n/a

3.1.2. Machine Mode Debug Control

An input port, named mdbgen[i], is introduced to control the debuggability of machine mode for each hart i. This signal is transmitted to the hart and its corresponding debug access control logic. When mdbgen[i] is set to 1, the debug operations outlined in The debug operations affected by Zedsec extension are permitted when hart i executes in machine mode. Moreover the following rules apply:

- The debug access privilege for the hart can be configured to any privilege level
- If register access without halting the hart is supported, this access carries the privilege of machine mode.

When mdbgen[i] is set to 0, debug operations in machine mode are prohibited for hart i. Any attempt to halt the hart and bring it into Debug Mode will remain pending, and triggers configured to enter Debug Mode will neither fire nor match in machine mode.

For the homogeneous computing system, the implementation can consolidate all mdbgen[i] into a single port to enforce unified debug policy across all harts.

3.1.3. Submachine Mode Debug Control

The submachine mode debug of hart i is determined by the `sdedbgalw` field of CSR msdcfg within hart i and mdbgen[i]. Debug operations listed in The debug operations affected by Zedsec extension are allowed when hart i executes in submachine mode only if the logical-OR of values in sdedbgalw and mdbgen[i] is 1.

The legal value of debug access privilege for hart i is solely determined by `sdedbgalw` when mdbgen[i] is 0. In the event of `sdedbgalw` being 1 while mdbgen[i] is 0, the debug access privilege can be configured to privilege levels other than machine mode. Any attempt to set debug access privilege to machine mode will result in a security fault error (cmderr 6).

If register access without halting the hart is supported, this access bears the privilege of supervisor/hypervisor mode to access the hart when mdbgen[i] is 0 and `sdedbgalw` is 1.

Debug operations in all modes are prohibited for hart i when the logical-OR of `sdedbgalw` and mdbgen[i] is 0. All halt requests from the Debug Module will remain pending, and triggers configured to enter debug mode will neither match nor fire. The register access without halting is dropped.

Table 2. The debuggable privilege levels per debug controls

mdbgen[i]	sdedbgalw	Debuggable privilege levels
1	x	All
0	1	All except M
0	0	n/a

3.2. Trace

The extension requires that trace availability from each hart is constrained by default. When Zedsec is supported, the optional sideband signal to trace encoder, sec_check[i] [2], must be implemented for each hart i, and this signal must be reset to 1. The sec_check[i] signal is only cleared when trace is permitted by machine mode trace control or submachine mode trace control.

3.2.1. Machine Mode Trace Control

For each hart i, an input port, mtrcen[i], controls machine mode trace availability. Setting mtrcen[i] to 1 enables machine mode and submachine mode trace by clearing the sec_check[i] signal to 0 across all privilege levels. Conversely, if mtrcen[i] is set to 0, the sec_check[i] signal cannot be cleared when the hart operates in machine mode.

For the homogeneous computing system, similarly to machine mode debug control, the implementation can consolidate all mtrcen[i] into a single port to constrain trace capability across all harts.

3.2.2. Submachine Mode Trace Control

The sec_check[i] signal for hart i in submachine mode is determined by the `sdetrcalw` field of CSR msdcfg within hart i, alongside mtrcen[i]. When the logical-OR of `sdetrcalw` and mtrcen[i] is 1, the sec_check[i] signal is cleared while the hart runs in submachine mode.

When both `sdetrcalw` and mtrcen[i] are set to 0, the sec_check[i] signal cannot be cleared at all.

Table 3. The status of the sec_check[i] sideband signal across privilege levels

mtrcen	sdetrcalw	Machine mode	Submachine mode
1	x	sec_check[i] = 0	sec_check[i] = 0
0	1	sec_check[i] = 1	sec_check[i] = 0
0	0	sec_check[i] = 1	sec_check[i] = 1

The sec_check signal serves as an additional signal for the trace module, indicating that trace output is prohibited due to security controls. Functionally, sec_check behaves identically to the halted signal. Both sec_check and halted signals cannot be active simultaneously. Reserved for future applications, the combined state of [sec_check, halted] as 0b11 remains unutilized. In cases where a trace module lacks support for the sec_check signal, the hart may alternatively toggle the halted signal to restrict trace output.

3.3. Trigger

The trigger configured to enter Debug Mode is checked by Zedsec extension. The trigger can fire or match in privilege modes outlined in Table 2.

The extension requires that all pending triggers intending to enter Debug Mode must match or fire

before any hart mode switch to prevent privilege escalation.

3.3.1. Machine mode accessibility to `dmode` accessibility

The RISC-V Debug Specification defines that the `dmode` field is accessible only in Debug Mode. When this field is set, the trigger is allocated exclusively to Debug Mode, and any write access from the hart are disregarded. However, the Debug Mode exclusive trigger could potentially serve as an attack surface for unauthorized submachine mode software where debugging is forbidden. The extension relaxes the constrain to the `dmode`, allowing it to be R/W in machine mode when mdbgen[i] is set to 0. When mdbgen[i] is set to 1, it remains exclusively accessible within Debug Mode.

In this definition, machine mode software assumes responsibility for switching the trigger context according to the debug policy enforced for the submachine mode. As a result, it maintains a clean trigger context for the submachine mode.

3.3.2. External triggers

The external trigger outputs follow the same limitations as other triggers, ensuring they do not fire or match when the privilege level of the hart exceeds the ones specified in Table 2.

The sources of external trigger input (such as machine mode performance counter overflow, interrupts, etc.) require protection to prevent information leakage. The external trigger inputs supported are platform-specific. Therefore, the platform is responsible for enforcing limitations on input sources. As a result, tmexttrigger.intctl and tmexttrigger.select should be restricted to legal values based on mdbgen[i] and `sdedbgalw`. Their definitions are provided in the Table 7 below.

3.3.3. Trigger chain

The privilege level of the trigger chain is determined by the highest privilege level within the chain. The entire trigger chain cannot be modified if the chain privilege level exceeds the debug access privilege.

This represents a balance between usability and hardware complexity. The integrity of the trigger chain set by the hart must be maintained when an external debugger intends to utilize triggers. There may be instances where the triggers are linked across different privilege levels (e.g., from supervisor mode to machine mode), while the external debugger may only have access to supervisor mode privilege. The external debugger should not alter the chain, because it could suppress or incorrectly raise breakpoint exceptions in machine mode.

3.4. Updates of CSR

3.4.1. Sdext CSR

Debug Control and Status (dcsr, at 0x7b0)

The hart must not automatically treat an external debugger with machine mode privilege (or surpassing machine mode privilege) without conditions. The `prv` and `v` fields in the dcsr have been enhanced to authorize privilege for debugger accesses. Upon transitioning into Debug Mode, the `prv` and `v` fields are updated to reflect the privilege level the hart was previously operating in. The dcsr is always permitted to be accessed in Debug Mode and the fields `prv` and `v` could be configured to grant privilege to the debugger other than the privilege level when the harts transitioned to Debug Mode. The maximum debug privilege level that can be configured in prv and v is determined in Table 1. It will generate a security fault error (cmderr 6) if the external debugger attempts to configure `prv` and `v` with a privilege higher than the maximum debug privilege level.

Memory and CSR accesses initiated by abstract commands or from the program buffer will be treated as if they are at the privilege level held in `prv` and `v`. These accesses will undergo protections of PMA, PMP, MMU, and other mechanisms, triggering traps if they violate corresponding rules.

The external debugger has the capability to write to prv and v and subsequently read back the value, thus determining the maximum debug privilege level.

Additionally, the fields in dcsr are further constrained based on their sphere of action. For example, when a field is effective in machine mode, it is accessible only to debugger which is granted with machine mode privilege. The detailed accessibility is listed in the following table.

Table 4. Dcsr fields accessibility against privilege granted to external debugger

Field	Allowed debug access privilege
ebreakvs	M/S/VS
ebreakvu	M/S/VS/VU
ebreakm	M
ebeaks	M/S
ebreaku	M/S/U
stepie	M
stoptime	M
mprven	M
nmip	M

Debug PC (dpc, at 0x7b1) and Debug Scratch Register (dscratch0, at 0x7b2; dscratch1, at 0x7b3)

Debug PC (dpc) and Debug Scratch Register (dscratch0, dscratch1) are not restricted by prv and v fields to simplify the architecture.

3.4.2. Sdtrig CSR

The extension enforces access control in Debug Mode, which complicates trigger usage within Debug Mode. To mitigate these complications, certain trigger CSRs, tselect, tdata1, tdata2, tdata3,

and tinfo are always permitted in Debug Mode, irrespective of the privileges granted to external debuggers. However, the remaining CSRs, tcontrol, scontext, hcontext, mcontext, and mscontext continue to adhere to the debug privileges granted.

Table 5. Trigger CSR accessibility in Debug Mode

Register	**w/o Zedsec**	**w/ Zedsec**
tselect(0x7a0)	Always	No change
tdata1(0x7a1)	Always	No change
tdata2(0x7a2)	Always	No change
tdata3(0x7a3)	Always	No change
tinfo(0x7a4)	Always	No change
tcontrol(0x7a5)	Always	Debug access privilege = M
scontext(0x5a8)	Always	Debug access privilege >= Sub-M
hcontext(0x6a8)	Always	Debug access privilege >= Sub-M
mcontext(0x7a8)	Always	Debug access privilege = M
mscontext(0x7aa)	Always	Debug access privilege = M

Beyond CSR-level accessibility adjustments, the fields within mcontrol, mcontrol6, icount, itrigger, etrigger, and tmexttrigger—variants of tdata1 located at 0x7a1—are redefined to limit the effective scope of triggers as follows.

Table 6. Tdata1 fields accessibility against privilege granted to external debugger

Field	**Allowed debug access privilege**
m	M
s	M/S
u	M/S/U
vs	M/S/VS
vu	M/S/VS/VU

The textra32, textra64 provides additional filtering capability for triggers. They are permitted for access in Debug Mode, as they do not affect the trigger firing/matching as it is constrained by mdbgen[i] and `sdedbgalw`.

The `intctl` and `sselect` field within tmexttrigger are redifined as follows.

Table 7. The redefinition of field `intctl` and `sselect` within tmexttrigger

Field	Description	Access	Reset
intctl	This optional bit, when set, causes this trigger to fire whenever an attached interrupt controller signals a trigger. the field is only configurable when mdbgen[i] is set to 1.	WLRL	0
select	Selects any combination of up to 16 TM external trigger inputs that cause this trigger to fire The legal value must be constrained by mdbgen[i] and `sdedbgalw` according to trigger input type.	WLRL	0

3.4.3. Zedsec CSR

The CSR control knobs in `msdcfg` for submachine mode debug and submachine mode trace are specified in Smsdedbg and Smsdetrc extension respectively in *RISC-V Supervisor Domains Access Protection* [3]. The Smsdedbg and/or Smsdetrc extension must be implemented to activate security enforcement for debugging and/or tracing.

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

This chapter outlines the security enhancements implemented in the Debug Module. The debug operations from external debuggers will be prohibited when the debug capability is not allowed by mdbgen[i] and `sdedbgalw`.

The affected debug operations, including:

- Halt request
- Reset request
- Keepalive request
- Abstract commands

Will be addressed as follows.

4.1. Debug Module Security Extension Discovery

The Debug Module Security Extension imposes security constraints and introduces non-backward-compatible changes. The presence of the Debug Module Security Extension can be determined by polling the allsecured/anysecured bits in dmstatus.

4.2. Halt

The hart can only be halted when it is running in privilege levels specified in Table 2. Otherwise, the halt request will remain pending. Consequently, the response to the external debugger request may take longer than one second, and this constraint must be eliminated.

When machine mode is not permitted (mdbgen[i] set to 0) to engage in debugging, the halt-on-reset (resethaltreq) operation must fail and raise security fault error. The debugger could check the error by polling allsecfault/anysecfault bits in dmstatus as specified in Table 8.

4.3. Reset

The reset operations must be safeguarded against various attacks. The RISC-V Debug Specification [1] defines the hartreset operations, which reset selected harts. The reset operations must be prohibited when machine mode is not allowed to be debugged. The security fault error will be raised if the operation is issued when mdbgen[i] is 0. The debugger could monitor the error by polling allsecfault/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 operation serves as an optional request for the hart to remain available for debugger. It is only allowed when machine mode is permitted to debug. Otherwise, it causes a security fault error when mdbgen[i] is 0, indicated by allsecfault/anysecfault bits in dmstatus.

4.5. Abstract Commands

The hart's response to abstract commands is details in Section 3.1.2 and Section 3.1.3. The following subsection delineates the constrains when the Debug Module issues the abstract commands.

4.5.1. Relaxed Permission Check `relaxedpriv`

The field `relaxedpriv` in abstractcs (at 0x16) allows for relaxed permission checks, such as bypassing PMA, PMP, MMU, etc. However, this relaxation violates security requirements, and the extension mandates that `relaxedpriv` be hardwired to 0.

4.5.2. Address Translation `aamvirtual`

The field `aamvirtual` in command (at 0x17) determines whether physical or virtual address translation is employed. However, when mdbgen[i] is 0, the extension mandates that `aamvirtual` is hardwire to 1 and memory access addresses are processed as if initiated by the hart in debug access privilege.

4.6. System Bus Access

System Bus Access enables direct reading/writing of memory space without involving the hart. However, it must always be checked by bus initiator protection mechanisms such as IOPMP, WorldGuard, etc. If these protections are not implemented or not deployed for Debug Module, System Bus Access must not be Supported. Failed system bus access attempts result in a bus security fault error (sberror 6).

In scenarios where a Debug Module lacks System Bus Access, memory access by the debugger can be achieved through the use of abstract commands. These commands provide secure means of accessing memory.

Trusted entities like RoT should configure IOPMP or equivalent protection before granting debug access to machine mode. Similarly, machine mode should apply the protection before enabling submachine mode debug.

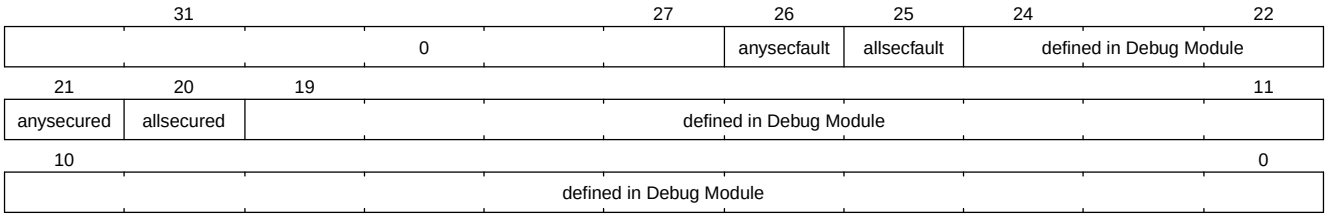
4.7. Security Fault Error Reporting

A dedicated error code, security fault error (cmderr 6), is included in cmderr of abstractcs (at 0x16) to signal any breaches of security enforcement by abstract commands. Additionally, the bus security fault error (sberror 6) is introduced in sberror of sbcs (at 0x38) to denote errors related to system bus access.

The error raised by other debug operations which could be applied to selected harts such as resethaltreq, reset and keepalive can be identified through the fields allsecfault/anysecfault in dmstatus. The security fault errors must be detectable prior to any subsequent read of the register responsible for reporting the error. Error statuses are internally maintained for each hart, with the allsecfault/anysecfault fields indicating the error status of the currently selected harts. Any error indicated by allsecfault/anysecfault remains until updated through a successful resethaltreq, reset or keepalive operation.

While the resethaltreq, reset, and keepalive operations can potentially take a significant amount of time to complete depending on the implementation, the error status can be immediately reported via following read of allsecfault/anysecfault if the operation is prohibited. Therefore, if a read of allsecfault/anysecfault indicates no error, it suggests that the operation is allowed and either currently in progress or has been successfully executed.

4.8. Update of Debug Module Status (dmstatus, at 0x11)



Register 1: The `allsecured` and `anysecured` fields in dmstatus

Table 8. The updates in dmstatus

Field	Description	Access	Reset
allsecured	The field is 1 when all currently selected harts implement Zedsec extension	R	-
anysecured	The field is 1 when any currently selected hart implements Zedsec 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	-

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 2 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 submachine mode software are debuggable, the secure monitor in machine 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.

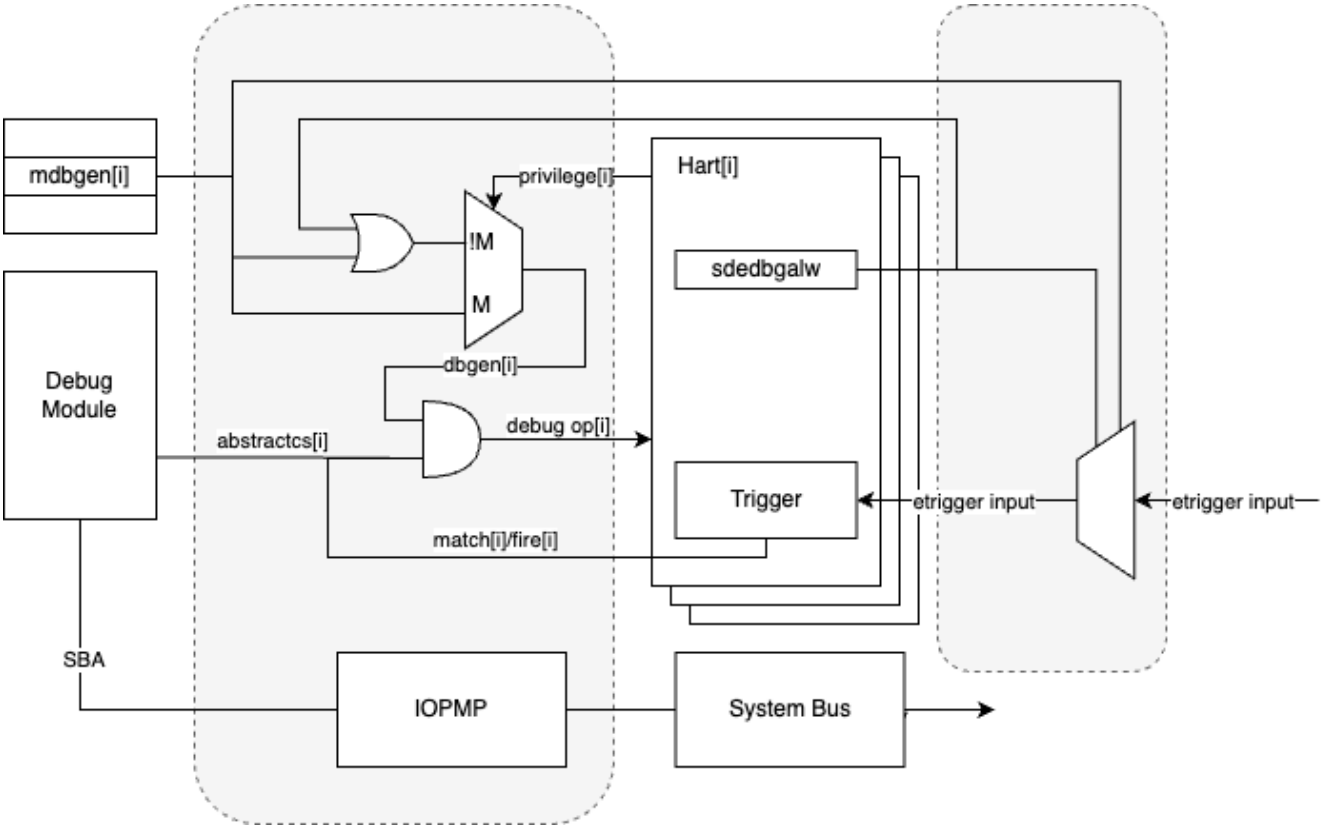


Figure 1. The security control on Debug Module

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 Table 3.

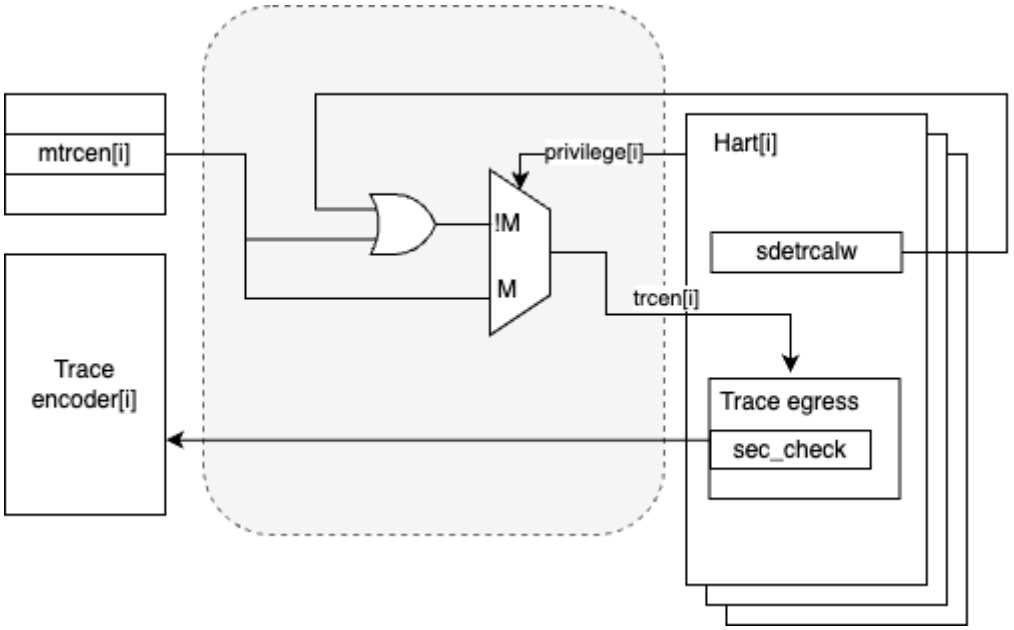


Figure 2. The security control on trace module

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