RISC-V Advanced Core Local Interruptor Specification

RISC-V Platform Specification Task Group

Version 1.0-rc4 (stable), January 10, 2022: This document is in Draft state. Change should be expected.

Table of Contents

Preamble

This document is in the Draft state

Assume everything can change. This draft specification will change before being accepted as standard, so implementations made to this draft specification will likely not conform to the future standard.

Copyright and license information

This RISC-V ACLINT specification has been contributed directly or indirectly by:

- Andrew Waterman <andrew@sifive.com>
- Greg Favor <gfavor@ventanamicro.com>
- John Hauser <jh.riscv@jhauser.us>
- Anup Patel <anup.patel@wdc.com>
- Bin Meng <bmeng.cn@gmail.com>
- Wesley Norris <repnop@outlook.com>

NOTE: Please add yourself to the above list if you have contributed to the RISC-V ACLINT specification.

Change Log

Version 1.0

- Dedicated chapter on synchronizing multiple MTIMER devices
- Initial release with MTIMER, MSWI, and SSWI devices

Chapter 1. Introduction

This RISC-V ACLINT specification defines a set of memory mapped devices which provide inter-processor interrupts (IPI) and timer functionalities for each HART on a multi-HART RISC-V platform. These HART-level IPI and timer functionalities are required by operating systems, bootloaders and firmwares running on a multi-HART RISC-V platform.

The SiFive Core-Local Interruptor (CLINT) device has been widely adopted in the RISC-V world to provide machine-level IPI and timer functionalities. Unfortunately, the SiFive CLINT has a unified register map for both IPI and timer functionalities and it does not provide supervisor-level IPI functionality.

The RISC-V ACLINT specification takes a more modular approach by defining separate memory mapped devices for IPI and timer functionalities. This modularity allows RISC-V platforms to omit some of the RISC-V ACLINT devices for when the platform has an alternate mechanism. In addition to modularity, the RISC-V ACLINT specification also defines a dedicated memory mapped device for supervisor-level IPIs. The Table 1 below shows the list of devices defined by the RISC-V ACLINT specification.

Table 1. ACLINT Devices

Name	Privilege Level	Functionality
MTIMER	Machine	Fixed-frequency counter and timer events
MSWI	Machine	Inter-processor (or software) interrupts
SSWI	Supervisor	Inter-processor (or software) interrupts

1.1. Backward Compatibility With SiFive CLINT

The RISC-V ACLINT specification is defined to be backward compatible with the SiFive CLINT specification. The register definitions and register offsets of the MTIMER and MSWI devices are compatible with the timer and IPI registers defined by the SiFive CLINT specification. A SiFive CLINT device on a RISC-V platform can be logically seen as one MSWI device and one MTIMER devices placed next to each other in the memory address space as shown in Table 2 below.

Table 2. One SiFive CLINT device is equivalent to two ACLINT devices

SiFive CLINT Offset Range	ACLINT Device	Functionality
0x0000_0000 - 0x0000_3fff	MSWI	Machine-level inter-processor (or software) interrupts
0x0000_4000 - 0x0000_bfff	MTIMER	Machine-level fixed-frequency counter and timer events

Chapter 2. Machine-level Timer Device (MTIMER)

The MTIMER device provides machine-level timer functionality for a set of HARTs on a RISC-V platform. It has a single fixed-frequency monotonic time counter (**MTIME**) register and a time compare register (**MTIMECMP**) for each HART connected to the MTIMER device. A MTIMER device not connected to any HART should only have a MTIME register and no MTIMECMP registers.

On a RISC-V platform with multiple MTIMER devices:

- Each MTIMER device provides machine-level timer functionality for a different (or disjoint) set of HARTs. A MTIMER device assigns a HART index starting from zero to each HART associated with it. The HART index assigned to a HART by the MTIMER device may or may not have any relationship with the unique HART identifier (**hart ID**) that the RISC-V Privileged Architecture assigns to the HART.
- Two or more MTIMER devices can share the same physical MTIME register while having their own separate MTIMECMP registers.
- The MTIMECMP registers of a MTIMER device must only compare against the MTIME register of the same MTIMER device for generating machine-level timer interrupt.

The maximum number of HARTs supported by a single MTIMER device is 4095 which is equivalent to the maximum number of MTIMECMP registers.

2.1. Register Map

A MTIMER device has two separate base addresses: one for the MTIME register and another for the MTIMECMP registers. These separate base addresses of a single MTIMER device allows multiple MTIMER devices to share the same physical MTIME register.

The Table 3 below shows map of the MTIME register whereas the Table 4 below shows map of the MTIMECMP registers relative to separate base addresses.

Table 3. ACLINT MTIMER Time Register Map

Offset	Width	Attr	Name	Description
0x0000_0000	8B	RW	MTIME	Machine-level time counter

Table 4. ACLINT MTIMER Compare Register Map

Offset	Width	Attr	Name	Description
0x0000_0000	8B	RW	MTIMECMP0	HART index 0 machine-level time compare
0x0000_0008	8B	RW	MTIMECMP1	HART index 1 machine-level time compare
...	...	...	...	...
0x0000_7FF0	8B	RW	MTIMECMP4094	HART index 4094 machine-level time compare

2.2. MTIME Register (Offset: 0x00000000)

The MTIME register is a 64-bit read-write register that contains the number of cycles counted based on a fixed reference frequency.

On MTIMER device reset, the MTIME register is cleared to zero.

2.3. MTIMECMP Registers (Offsets: 0x00000000 - 0x00007FF0)

The MTIMECMP registers are per-HART 64-bit read-write registers. It contains the MTIME register value at which machine-level timer interrupt is to be triggered for the corresponding HART.

The machine-level timer interrupt of a HART is pending whenever MTIME is greater than or equal to the value in the corresponding MTIMECMP register whereas the machine-level timer interrupt of a HART is cleared whenever MTIME is less than the value of the corresponding MTIMECMP register. The machine-level timer interrupt is reflected in the MTIP bit of the `mip` CSR.

On MTIMER device reset, the MTIMECMP registers are in unknown state.

2.4. Synchronizing Multiple MTIME Registers

A RISC-V platform can have multiple HARTs grouped into hierarchical topology groups (such as clusters, nodes, or sockets) where each topology group has it's own MTIMER device. Further, such RISC-V platforms can also allow clock-gating or powering off for a topology group (including the MTIMER device) at runtime.

On a RISC-V platform with multiple MTIMER devices residing on the same die, each device must satisfy the RISC-V architectural requirement that all the MTIME registers with respect to each other, and all the per-HART `time` CSRs with respect to each other, are synchronized to within one MTIME tick period. For example, if the MTIME tick period is 10ns, then the MTIME registers, and their associated time CSRs, should respectively be synchronized to within 10ns of each other.

On a RISC-V platform with multiple MTIMER devices on different die, the MTIME registers (and their associated `time` CSRs) on different die may be synchronized to only within a specified interval of each other that is larger than the MTIME tick period. A platform may define a maximum allowed interval.

To satisfy the preceding MTIME synchronization requirements:

- All MTIME registers should have the same input clock so as to avoid runtime drift between separate MTIME registers (and their associated `time` CSRs)
- Upon system reset, the hardware must initialize and synchronize all MTIME registers to zero
- When a MTIMER device is stopped and started again due to, say, power management actions, the software should re-synchronize this MTIME register with all other MTIME registers

When software updates one, multiple, or all MTIME registers, it must maintain the preceding synchronization requirements (through measuring and then taking into account the differing latencies of performing reads or writes to the different MTIME registers).

As an example, the below RISC-V 64-bit assembly sequence can be used by software to synchronize a MTIME register with reference to another MTIME register.

Listing 1. Synchronizing a MTIME Registers On RISC-V 64-bit Platform

```
/*
 * unsigned long aclint_mtime_sync(unsigned long target_mtime_address,
 *                                 unsigned long reference_mtime_address)
 */
        .globl aclint_mtime_sync
aclint_mtime_sync:
        /* Read target MTIME register in T0 register */
        ld        t0, (a0)
        fence     i, i

        /* Read reference MTIME register in T1 register */
        ld        t1, (a1)
        fence     i, i

        /* Read target MTIME register in T2 register */
        ld        t2, (a0)
        fence     i, i

        /*
         * Compute target MTIME adjustment in T3 register
         * T3 = T1 - ((T0 + T2) / 2)
         */
        srli      t0, t0, 1
        srli      t2, t2, 1
        add       t3, t0, t2
        sub       t3, t1, t3

        /* Update target MTIME register */
        ld        t4, (a0)
        add       t4, t4, t3
        sd        t4, (a0)

        /* Return MTIME adjustment value */
        add       a0, t3, zero

        ret
```

***NOTE**: On some RISC-V platforms, the MTIME synchronization sequence (i.e. the `aclint_mtime_sync()` function above) will need to be repeated few times until delta between target MTIME register and reference MTIME register is zero (or very close to zero).*

Chapter 3. Machine-level Software Interrupt Device (MSWI)

The MSWI device provides machine-level IPI functionality for a set of HARTs on a RISC-V platform. It has an IPI register (**MSIP**) for each HART connected to the MSWI device.

On a RISC-V platform with multiple MSWI devices, each MSWI device provides machine-level IPI functionality for a different (or disjoint) set of HARTs. A MSWI device assigns a HART index starting from zero to each HART associated with it. The HART index assigned to a HART by the MSWI device may or may not have any relationship with the unique HART identifier (**hart ID**) that the RISC-V Privileged Architecture assigns to the HART.

The maximum number of HARTs supported by a single MSWI device is 4095 which is equivalent to the maximum number of MSIP registers.

3.1. Register Map

Table 5. ACLINT MSWI Device Register Map

Offset	Width	Attr	Name	Description
0x0000_0000	4B	RW	MSIP0	HART index 0 machine-level IPI register
0x0000_0004	4B	RW	MSIP1	HART index 1 machine-level IPI register
...	...	...	...	...
0x0000_3FFC	4B		RESERVED	Reserved for future use.

3.2. MSIP Registers (Offsets: 0x00000000 - 0x00003FF8)

Each MSIP register is a 32-bit wide WARL register where the upper 31 bits are wired to zero. The least significant bit is reflected in MSIP of the `mip` CSR. A machine-level software interrupt for a HART is pending or cleared by writing `1` or `0` respectively to the corresponding MSIP register.

On MSWI device reset, each MSIP register is cleared to zero.

Chapter 4. Supervisor-level Software Interrupt Device (SSWI)

The SSWI device provides supervisor-level IPI functionality for a set of HARTs on a RISC-V platform. It provides a register to set an IPI (**SETSSIP**) for each HART connected to the SSWI device.

On a RISC-V platform with multiple SSWI devices, each SSWI device provides supervisor-level IPI functionality for a different (or disjoint) set of HARTs. A SSWI device assigns a HART index starting from zero to each HART associated with it. The HART index assigned to a HART by the SSWI device may or may not have any relationship with the unique HART identifier (**hart ID**) that the RISC-V Privileged Architecture assigns to the HART.

The maximum number of HARTs supported by a single SSWI device is 4095 which is equivalent to the maximum number of SETSSIP registers.

4.1. Register Map

Table 6. ACLINT SSWI Device Register Map

Offset	Width	Attr	Name	Description
0x0000_0000	4B	RW	SETSSIP0	HART index 0 set supervisor-level IPI register
0x0000_0004	4B	RW	SETSSIP1	HART index 1 set supervisor-level IPI register
...	...	...	...	...
0x0000_3FFC	4B		RESERVED	Reserved for future use.

4.2. SETSSIP Registers (Offsets: 0x00000000 - 0x00003FF8)

Each SETSSIP register is a 32-bit wide WARL register where the upper 31 bits are wired to zero. The least significant bit of a SETSSIP register always reads `0`. Writing `0` to the least significant bit of a SETSSIP register has no effect whereas writing `1` to the least significant bit sends an edge-sensitive interrupt signal to the corresponding HART causing the HART to set SSIP in the `mip` CSR. Writes to a SETSSIP register are guaranteed to be reflected in SSIP of the corresponding HART but not necessarily immediately.

> ***NOTE**: The RISC-V Privileged Architecture defines SSIP in `mip` and `sip` CSRs as a writeable bit so the M-mode or S-mode software can directly clear SSIP.*