芯来科技
NUCLEI

Datasheet

for Bumblebee Processor Core

www.riscv-mcu.com

Revision History

Rev.	Revision Date	Revised Content
1.0	2019/8/28	**1. Initial Release.**

Table of Contents

Table List

Figure List

1. Introduction of the Bumblebee Processor Core

The Bumblebee Processor Core, or Bumblebee core for short， is a commercial RISC-V processor core customized by Nuclei System Technology and Gigadevice for general-purpose MCU products for IoT or other ultra-low-power applications. It is dedicated to MCU products of model GD32VF103.

Note: The Bumblebee core used for this MCU is jointly developed by Nuclei System Technology and Andes Technology. Nuclei System Technology provides authorization services and technical support.

At present, Nuclei System Technology can authorize fully domestically controllable N200 series ultra-low power commercial processor core IPs, as well as other multiple series (300/600/900 series) 32-bit or 64-bit high-performance embedded processor core IPs, and provide customers with processor IP customization services.

Note: For more detailed information on RISC-V MCU chips, boards and solutions, please visit the website: www.riscv-mcu.com

1.1. Bumblebee Core Feature List

The Bumblebee core has the following features:

- CPU Core
  - 2-stage variable pipeline architecture, using state-of-the-art processor architecture to deliver the highest degree of performance efficiency and lowest cost.
  - Simple dynamic branch predictor.
  - The instruction fetch unit (IFU) can prefetch the following two instructions to mask the instruction memory access latency.
  - Support Machine Mode and User Mode.
- Supported Instruction Set Architecture (ISA)
  - The Bumblebee core is a 32-bit RISC-V-based processor, supporting a combination of RISCV 32IMAC instruction subsets.

- Hardware supports misaligned memory access operations (Load/Store instructions)

- Bus Interface
  - Supports 32-bit wide standard AHB-Lite system bus interface for accessing external instruction and data.
  - Supports 32-bit wide Instruction Local Memory (ILM) bus interface (supports standard AHB-Lite or SRAM interface protocol) for private instruction local memory.
  - Supports 32-bit wide Data Local Memory (DLM) bus interface (supports standard AHB-Lite or SRAM interface protocol) for private data local memory.
  - Supports 32-bit wide Private Peripheral Interface (PPI) bus interface, and supports standard APB interface protocol for private peripherals.
- Debugging System
  - Supports standard JTAG interface.
  - Supports standard RISC-V debugging systems.
  - Supports up to 4 Hardware Breakpoints.
  - Supports mature interactive debugging tools.
- Low-Power Management
  - Supports WFI（Wait For Interrupt）and WFE（Wait For Event）to enter sleep mode.
  - Supports two-level sleep modes: shallow sleep mode and deep sleep mode.
- Core-Private Timer Unit
  - 64-bit real-time timer, supporting the generation of the timer interrupt defined by the RISC-V standard.
- Enhanced Core Level Interrupt Controller (ECLIC)
  - Supports the RISC-V architecturally defined software, timer and external interrupts.
  - Supports dozens of external interrupt sources. For the supported number and allocation of interrupts, please refer to the datasheet of the specific MCU chip.

- Supports up to 16 interrupt levels and priorities, and supports software dynamic programmable modification of values of interrupt levels and priorities.
- Supports interrupt preemption based on interrupt levels.
- Supports fast vectored interrupt processing.
- Supports fast interrupt tail-chaining mechanism

- Supports NMI（Non-Maskable Interrupt）。
- Software development tools:
  - The Bumblebee core supports the RISC-V standard compilation tool chain and Integrated Development Environment (IDE) on Linux/Windows systems.

1.2. Bumblebee Core Instruction Set and Architecture

Please refer to the Bumblebee Processor Core Instruction Set Architecture Manual for details about the instruction sets supported and architectures of the core.

1.3. Bumblebee Core Hierarchy Diagram

The top level of the Bumblebee core is shown in Figure 1-1. The structure of the Bumblebee core contains the following points:

- The Core is the top-level of the whole processor.
- The uCore is located under the Core hierarchy and is the main part of the processor core.
- In addition to the uCore, the following components are included in the Core”
  - DEBUG：handle the JTAG interface and related debugging features.
  - ECLIC：the Enhanced Core Level Interrupt Controller
  - TIMER：the timer unit.
  - LM Ctrl：the control unit of external ILM and DLM interface.

- BIU：the control unit of external PPI interface and MEM interface.
- Misc Ctrl：the control unit of other components.

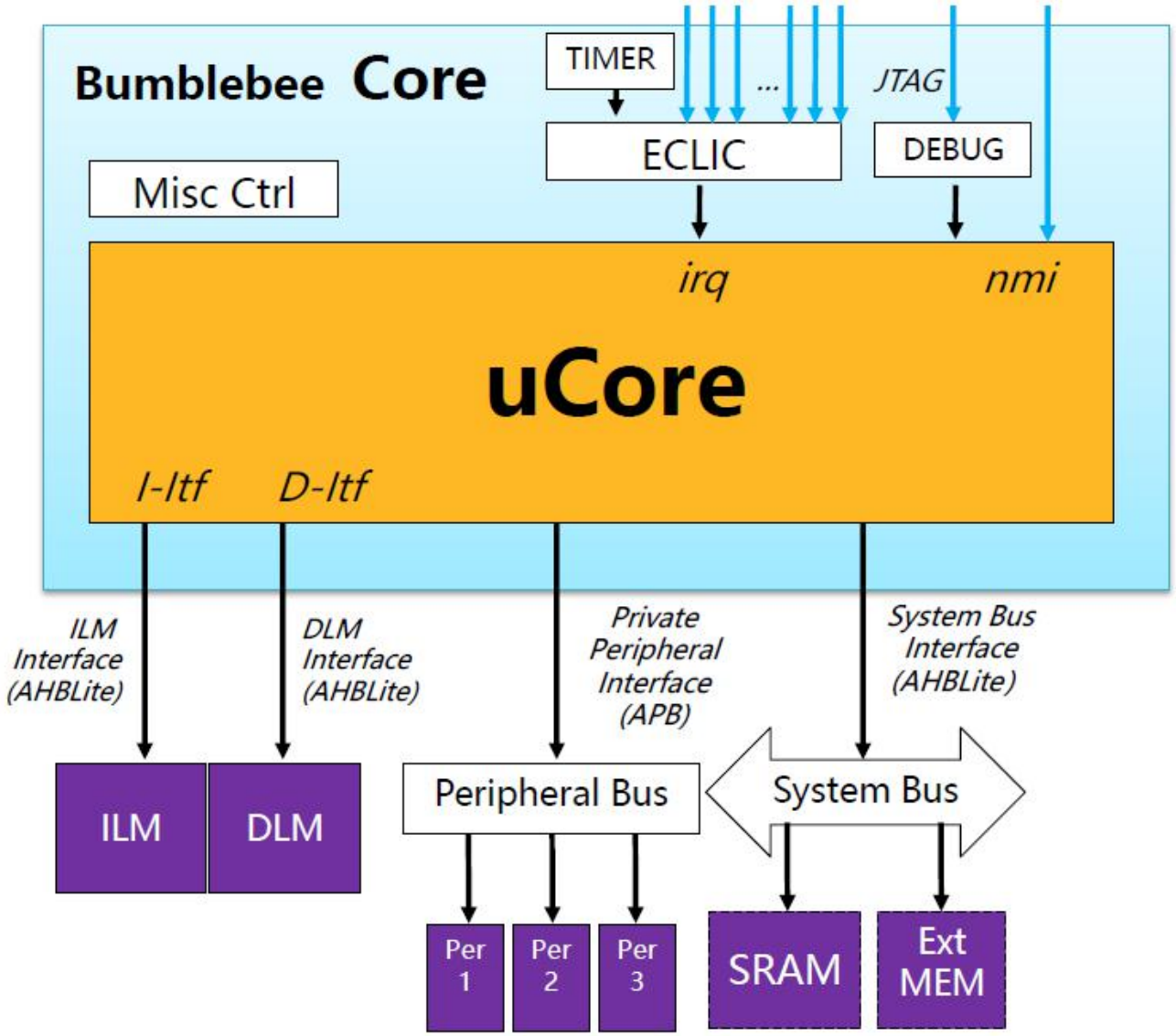

Figure 1-1 Top-level of the Bumblebee Core

2. Introduction to the Function of the Bumblebee Core

2.1. Introduction to the Bumblebee Core Clock Domain

The clock domains of the Bumblebee core are shown in Figure 2-1. The entire processor is divided into two asynchronous clock domains:

- The core clock domain includes the core_clk and core_clk_aon, which drive most of the functional logic of the processor core. Note:
  - core_clk and core_clk_aon have the same frequency and phases as they are clocks from the same clock source.
  - core_clk is the main working clock that drives the main function logic inside the processor core and is clock-gated at the system level.
  - core_clk_aon is an always-on clock that drives the always-on logic in the core, including the ECLIC, the Timer, and the Debugger. Please refer to the Bumblebee Processor Core Instruction Set Architecture Manual for details about the ECLIC and the Timer.
- JTAG clock domain includes the jtag_TCK, which drives the corresponding logic of JTAG debugging system.

The above two clock domains are asynchronous, so asynchronous cross-clock domain processing has been implemented in the processor core.

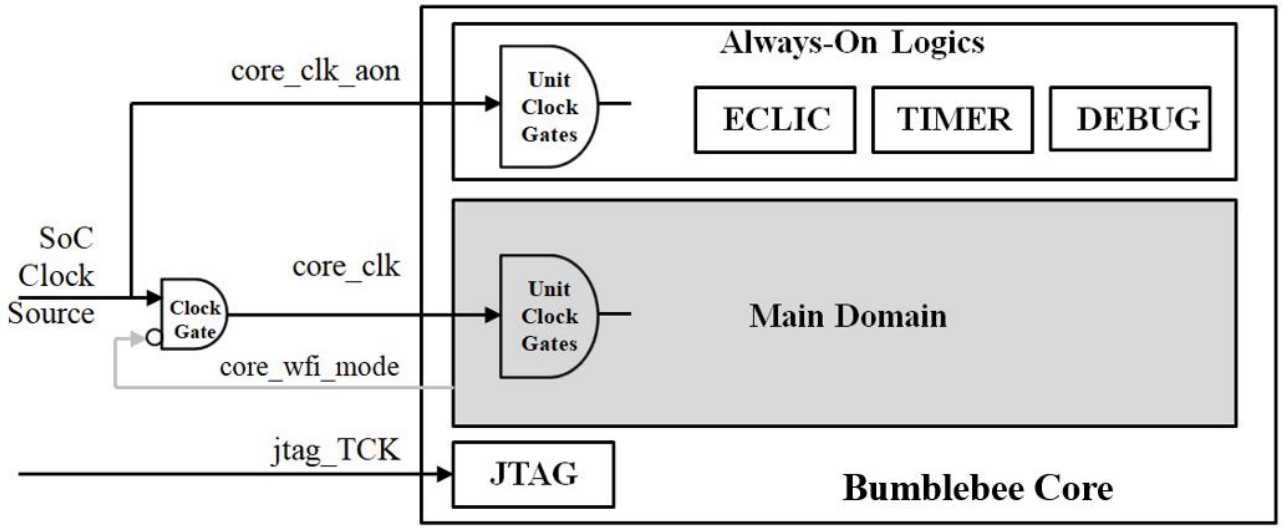

Figure 2-1 Clock domains of the Bumblebee Core

2.2. Introduction to the Bumblebee Core Power Domain

The Bumblebee Core does not have a power domain inside. The SoC system integrator can divide the power domain and perform cross-power domain processing according to the Bumblebee Core hierarchy.

2.3. The Memory Map of the Bumblebee Core

For the memory map of the Bumblebee Core, please refer to the datasheet of the specific MCU chip.

2.4. The Privilege Modes of the Bumblebee Core

The Bumblebee Core supports two privilege modes: Machine mode and User mode. For the detailed information about privilege modes, please refer to the Bumblebee Processor Core Instruction Set Architecture Manual.

2.5. Memory resources of the Bumblebee Core

Bumblebee core supports the following memory resources:

- ILM：
  - The Bumblebee Core supports Instruction Local Memory (ILM) access via a proprietary AHB-Lite Bus or SRAM interface if an ILM interface is configured.
  - The size of the ILM is configurable. The ILM interface has separate address space, and the user can configure a specific base address for it. Please see Section 2.3 for details.
  - The ILM is implemented by the SoC system integrator and can generally be an on-chip SRAM or Flash for storing instructions. If the AHB-Lite interface is implemented, to achieve optimal performance, the ILM should be implemented to support the AHB protocol and return instructions at the next cycle after receiving the read request.
- DLM：
  - The Bumblebee Core supports Data Local Memory (DLM) access via a proprietary AHB-Lite Bus or SRAM interface if a DLM interface is configured.
  - The size of the DLM is configurable. The DLM interface has separate address space, and the user can configure a specific base address for it. Please see Section 2.3 for details.
  - The DLM is implemented by the SoC system integrator and can generally be an on-chip SRAM or Flash for storing instructions. If the AHB-Lite interface is implemented, to achieve optimal performance, the DLM should be implemented to support the AHB protocol and return instructions at the next cycle after receiving the read request.

2.6. Private Peripherals of the Bumblebee Core

As shown in Figure 1-1, under the Core hierarchy of the Bumblebee Core, in addition to the

uCore, the following private peripherals are included:

- DEBUG: handle the JTAG interface and related debugging features.
- ECLIC：the Enhanced Core-Level Interrupt Controller
- TIMER: the private TIMER unit.

The above devices are private to the processor core and are accessed using memory address. See Section 2.3 for the details of their specific address space allocation.

2.7. Physical Memory Protection of the Bumblebee Core

Since the Bumblebee Core is a low-power core designed for microcontrollers, it does not support the Memory Management Unit, so all the address access operations are using physical addresses. In order to perform memory access protection and isolation according to memory physical address of different devices and execution privilege mode, the RISC-V standard architecture defines a physical memory protection mechanism: Physical Memory Protection (PMP) unit.

Note: the Bumblebee Core does not support PMP unit.

2.8. Debugging Mechanism of the Bumblebee Core

The Bumblebee Core supports standard JTAG debugging interface, and the interactive debugging tool GDB. Note:

- The Bumblebee Core supports up to 4 hardware breakpoints. Hardware breakpoints are primarily used to set breakpoints to read-only memory space such as FLASH.

The Bumblebee Core defines an input signal i_dbg_stop, which can be controlled by the external drive:

- If the value of the i_dbg_stop signal is 1, the debug mode of the processor is off.
- If the value of the i_dbg_stop signal is 0, the debug mode of the processor is working

properly.

2.9. Interrupt and Exception Mechanism of the Bumblebee Core

For a detailed description of the Bumblebee Core's interrupt and exception mechanisms, please refer to the Bumblebee Processor Core Instruction Set Architecture Manual.

.

2.10. NMI Mechanism of the Bumblebee Core

NMI (Non-Maskable Interrupt) is a special input signal of the processor, often used to indicate system-level emergency errors (such as external hardware failures, etc.) After encountering the NMI, the processor should abort execution of the current program immediately and process the NMI error instead. For a detailed description of the NMI mechanism of the Bumblebee Core, please refer to the Bumblebee Processor Core Instruction Set Architecture Manual.

2.11. CSRs of the Bumblebee Core

Some control and status registers (CSRs) are defined in the RISC-V architecture to configure or record the status of execution. CSR registers are registers internal to the processor core that uses their proprietary 12-bit encoding space to access. Please see the Bumblebee Processor Core Instruction Set Architecture Manual for more details.

2.12. Perfomance Monitor's Counters of the Bumblebee Core

The RISC-V architecture defines the following two performance counters:

- Cycle Counter：
  - A 64-bit wide clock cycle counter that reflects how many clock cycles the processor has executed. This counter will continuously increment as long as the processor is in the execution state.

- The CSR mcycle stores the lower 32 bits of the counter, and the CSR mcycleh stores the upper 32 bits of the counter. Please see the Bumblebee Processor Core Instruction Set Architecture Manual for more details.

- Instruction Retirement Counter：

  - The RISC-V architecture defines a 64-bit wide instruction completion counter that reflects how many instructions the processor successfully executed. This counter will increment if the processor executes an instruction successfully.

  - The CSR minstret stores the lower 32 bits of the counter, and the CSR minstreth stores the upper 32 bits of the counter. Please see the Bumblebee Processor Core Instruction Set Architecture Manual for more details.

The Cycle Counter and the Instruction Retirement Counter are typically used to measure performance.

By default, the counter is 0 after a reset and then increments itself continuously. Because the counter count consumes some dynamic power consumption, the Bumblebee Core implements additional control fields which software can configure the corresponding bits to enable or disable these counters. These bits can be used to stop the corresponding counter to save power when you do not need to use them.

Please see the Bumblebee Processor Core Instruction Set Architecture Manual for more details about the CSR mcountinhibit.

2.13. TIMER Unit of the Bumblebee Core

The RISC-V architecture defines a 64-bit Timer Counter which is clocked by the system's low-speed Real Time Clock frequency. The value of this timer is reflected in the register mtime. The RISC-V architecture also defines a 64-bit mtimecmp register that used as a comparison value for the timer. A timer interrupt is generated if the value of mtime is greater than or equal to the value of mtimecmp.

Note: The RISC-V architecture does not define the mtime and mtimecmp registers as CSR

registers, but rather as Memory Address Mapped system registers. The specific memory mapped address is not defined by the RISC-V architecture, so it is defined by the implementation. In the Bumblebee Core, mtime/mtimecmp are both implemented in the TIMER Unit. Please see the Bumblebee Processor Core Instruction Set Architecture Manual for more details about the TIMER Unit.

2.13.1. TIMER Behavior in Debug Mode

When the Bumblebee Core is in debug mode, it occasionally executes some programs designated by the debugger, which is stored in the DEBUG unit and inivisible to the user, to support the debugger's function. If the timer keeps counting while executing these programs designated by the debugger, it does not truly reflect the true behavior of the program being debugged. Therefore, when the Bumblebee Core executes these programs, the timer will automatically stop counting.

2.13.2. TIMER Behavior in Normal Mode

By default, the timer has a value of 0 after a reset and then continue to increment itself. Because the counter count consumes some dynamic power consumption, the Bumblebee Core implements additional control fields which software can configure the corresponding bits to enable or disable these counters. These bits can be used to stop the corresponding counter to save power when you do not need to use them. Please see the Bumblebee Processor Core Instruction Set Architecture Manual for more details about the CSR mcountinhibit.

2.14. Low-Power Mechanism of the Bumblebee Core

The low-power mechanism of the Bumblebee Core is reflected in the following aspects:

- The clock of the main units in the Bumblebee Core are automatically gated off when they are in idle to reduce static power consumption.
- The Bumblebee Core supports sleep mode through common WFI (Wait for Interrupt)

and WFE (Wait for Event) mechanisms to achieve lower dynamic and static power consumption. Please see the Bumblebee Processor Core Instruction Set Architecture Manual for more details about “Wait for Interrupt” and “Wait for Event”.

2.14.1. Clock Control Entering Sleep Mode

The Bumblebee Core can go to sleep by executing the WFI instruction. Please see the Bumblebee Processor Core Instruction Set Architecture Manual for more details about how to enter the sleep mode.

The output signal core_sleep_value of the Bumblebee Core can be used to indicate current sleep mode (0 or 1). Sleep mode 0 indicate the core is in shallow sleep mode, while sleep mode 1 means in deep sleep mode. Note: When entering to the deep sleep mode, the processor core will no longer be able to be debugged by JTAG interface.

The key points of the clock control (reference scheme) of the core in sleep mode are shown as the followings:

- As shown in Figure 2-1, when the WFI is successfully executed, the output signal core_wfi_mode of the Bumblebee Core is asserted, indicating that the processor core has entered to the sleep mode after executing the WFI instruction; from the sight of SoC, the signal core_wfi_mode can be used to control the external gate logic to disable the core_clk.
- If the Bumblebee Core entered the deep sleep mode (core_sleep_value is 1), then SoC can decide whether to disable the always on clock core_clk_aon according to its actual scenario.

2.14.2. Clock Control of Exiting the Sleep Mode

The core can be waked up by interrupt, event, or NMI. Please see the Bumblebee Processor Core Instruction Set Architecture Manual for more details about how to exit the sleep mode.

The key points of the clock control when the core exits the sleep mode are shown as the followings:

- For the case that the core is waiting for an interrupt to wake up, because the Bumblebee Core can only handle the interrupt processed and distributed by ECLIC unit, then only the interrupt, which is enabled and has greater interrupt level than the interrupt threshold level, can wake up the core. Furthermore, whether enable the core_clk_aon inside the core needs to be handled carefully:

  - As mentioned in Section 2.1, the TIMER unit is clocked by core_clk_aon.
    - Assuming the SoC system has disabled the always-on clock core_clk_aon, the TIMER unit cannot generate timer or software interrupt because it has no clock.
  - As mentioned in Section 2.1, the ECLIC unit is clocked by core_clk_aon.
    - Assuming the SoC system has disabled the always-on clock core_clk_aon, then the external interrupt signal must kept asserted until the SoC system enable the signal core_clk_aon again. Otherwise, the ECLIC unit cannot sample the external interrupt signal because there is no clock, and the core cannot be woken up.

- For the case that the core is waiting for an event or NMI to wake up, if the core samples (by the core_clk_aon) the input signal rx_evt (keep as "1" to indicate there is one event) or the input signal nmi (a rising edge to indicate there is one NMI), the core will be woken up. Furthermore, whether enable the core_clk_aon inside the core needs to be handled carefully:

  - Assuming the SoC system has disabled the always-on clock core_clk_aon, then the input signal rx_evt or nmi must keep as 1 once be set until the SoC system turns on the clock core_clk_aon. Otherwise, the core cannot sample the Even or NMI as the sample logic has no clock, and the core will not wake up.

- The output signal core_wfi_mode will be 0 immediately after the core being waked up. Assuming the SoC system control the gate of core_clk using the signal core_wfi_mode, the working clock of core, core_clk will be enabled as soon as the signal

core_wfi_mode is cleared to 0.

3. Introduction of Interfaces of the Bumblebee Core

3.1. Introduction of Interfaces of the Bumblebee Core

The Bumblebee Core has the following groups of interfaces:

- Clock and reset interface
- Debugging interface
- External interrupt interface
- Bus interfaces，including the following interfaces:
  - ILM Master Interface: the interface used to access the external ILM.
  - DLM Master Interface：the interface used to access the external DLM.
  - Private Peripheral Interface：the interface used to access the Private Peripheral.
  - System Memory Interface：the interface used to access the System Memory.
- Other function interface

Since the most interface signals of the Bumblebee Core are the internal details of the MCU, we do not discuss about them too much herein.

This document only describes an input signal mtime_toggle_a, because this signal is critical for MCU embedded software development.

Table 3-1 mtime_toggle_a Interface Signals

Signal Name	Direction	Width	Description
mtime_toggle_a	Input	1	■ mtime_toggke_a is a pulse signal from the SoC system, and used to drive the timer of the internal TIMER unit of the Core. ■ Note: ● This signal is treated as an asynchronous input signal. ● This signal is synchronized within the Core (using several FF synchronizer). ● After the synchronization, both the rising edge and falling edge of the signal are sampled by the main clock of the core, and any detected edge will trigger the TIMER incrementation. ● It is recommended that use the output of the register driven by the slow clock (rtc_clk, whose frequency is halved frequency of core_clk_aon) as the input of this

			signal. Then the self-increment frequency is equal to the frequency of the slow clock, as shown in the figure 3-1. Hence, the lower the frequency of the slow clock, the lower the self-increment frequency of the internal timer, the lower the dynamic power consumption.

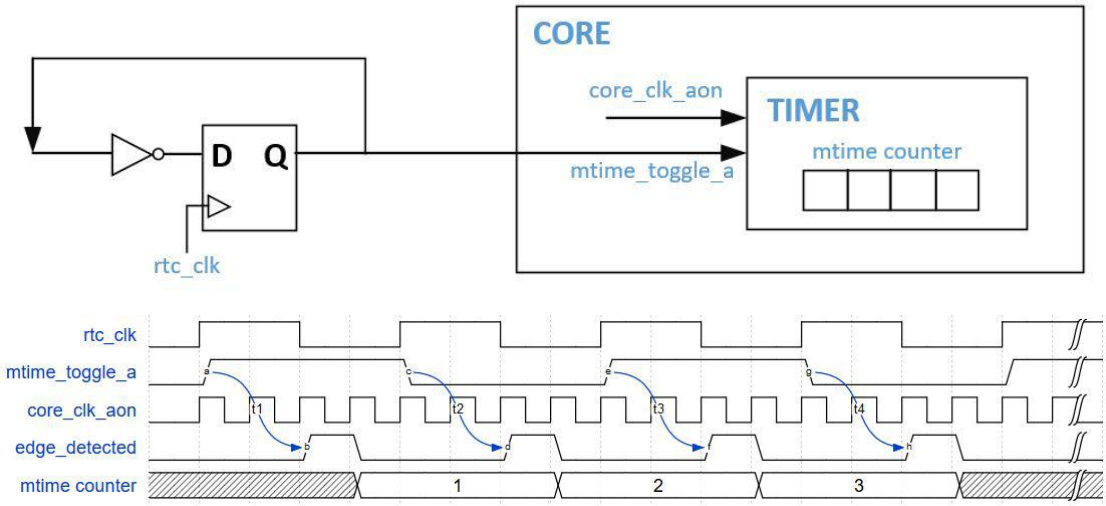

Figure 3-1 Generation Diagram of the signal mtime_toggle_a

Note: in the MCU GD32VF103, rtc_clk runs at the frequency which is a quarter of the frequency of core_clk_aon, which means the internal TIMER counter is triggered by the signal mtimt_toggle_a, running at a self-increment frequency which is a quarter of the frequency of core_clk_aon.