Bumblebee Processor Core Concise Data Sheet

revise history

version number	Revision date	Revised chapter	Revised content
1.0	2019/6/21	N/A	Initial version

table of Contents

Picture list

1. Bumblebee kernel overview

The Bumblebee Processor Core, or Bumblebee core, is a commercial custom made by Nuclei System Technology in conjunction with Gigadevice for its general purpose MCU products for IoT or other ultra-low power scenarios. RISC-V processor core.

Note: The Bumblebee core used for this MCU is jointly developed by Nuclei System Technology and Taiwanese Andes Technology, and Nuclei System Technology provides authorization and technical support services.At present, Nuclei System Technology can license the mass-proven N200 series of ultra-low-power commercial processor cores, as well as research a variety of high-performance embedded processor series, and provide customers with customized services.

1.1. Bumblebee kernel feature list

The list of features of the Bumblebee kernel is as follows:

- CPU core (CPU Core)
  - The 2-stage variable-length pipeline architecture uses state-of-the-art processor architecture to achieve the industry's highest energy efficiency and lowest cost.
  - Simple dynamic branch predictor.
  - The instruction prefetch unit can prefetch two instructions in order to hide the instruction fetch delay.
  - Supports Machine Mode and User Mode.
- Support for Instruction Set Architecture (ISA, Instruction Set Architecture)
  - The Bumblebee kernel supports the 32-bit RISC-V instruction set architecture and supports a combination of RV32IMAC instruction subsets.
  - Hardware supports misaligned memory access operations (Load/Store instructions)
- Bus interface
  - Supports a 32-bit wide standard AHB-Lite system bus interface for accessing external commands and data.

- Supports 32-bit wide Instruction Local Memory (ILM) bus interface (supports standard AHB-Lite or SRAM interface protocols) for connecting private instruction local memory.
- Supports 32-bit wide Data Local Memory (DLM) bus interface (supports standard AHB-Lite or SRAM interface protocols) for connecting private data local memory.

- Supports 32-bit wide Private Peripheral Interface (PPI) and supports standard APB Interface protocol for connecting private peripherals.

- Debugging function
  - Support for standard jtag interfaces.
  - Support for the risc-v debugging standard.
  - Support for 4 hardware breakpoints (Hardware Breakpoints).
  - Support for sophisticated interactive debugging tools.
- Low power management
  - Supports WFI (Wait For Interrupt) and WFE (Wait For Event) to enter sleep mode.
  - Two-level sleep mode is supported: shallow sleep and deep sleep.
- Kernel-owned timer unit (Machine Timer, TIMER for short)
  - A 64-bit wide real-time timer that supports the generation of timer interrupts defined by the risc-v standard.
- Enhanced Core Level Interrupt Controller (ECLIC)
  - Supports software interrupts, timer interrupts, and external interrupts defined by the risc-v standard.
  - Support dozens of external interrupt sources. For the number and allocation of interrupt sources, please refer to the data sheet of the specific mcu chip.
  - Supports 16 interrupt levels and priorities, and supports software dynamic programmable modification of interrupt levels and interrupt priority values.
  - Interrupt nesting based on interrupt levels is supported.
  - Support for fast vector interrupt handling mechanisms.
  - Support for fast interrupt biting mechanism.
- Support NMI (Non-Maskable Interrupt).
- Software development tools:
  - The Bumblebee kernel supports the RISC-V standard build toolchain and the Linux/Windows Graphical Integrated Development Environment (IDE).

1.2. Bumblebee kernel instruction set and architecture

See the Bumblebee Kernel Instruction Architecture Handbook for details on the instruction set and architecture supported by the Bumblebee kernel.

1.3. Bumblebee kernel hierarchy diagram

The top of the Bumblebee kernelFigure 1-1 Shown.The organizational structure of the Bumblebee kernel mainly includes the following points:

- Core is the top level of the entire processor core.
- uCore is under the Core hierarchy and is the main part of the processor core.
- In addition to uCore, the following main components are included under the Core hierarchy:
  - Debug: Handles the jtag interface and related debugging features.
  - Eclic: Interrupt control unit.
  - Timer: timer unit.
  - LM Ctrl: Control of the external ILM and DLM interfaces.
  - Biu: Control of the external ppi interface and mem interface.
  - Misc Ctrl: Other control modules.

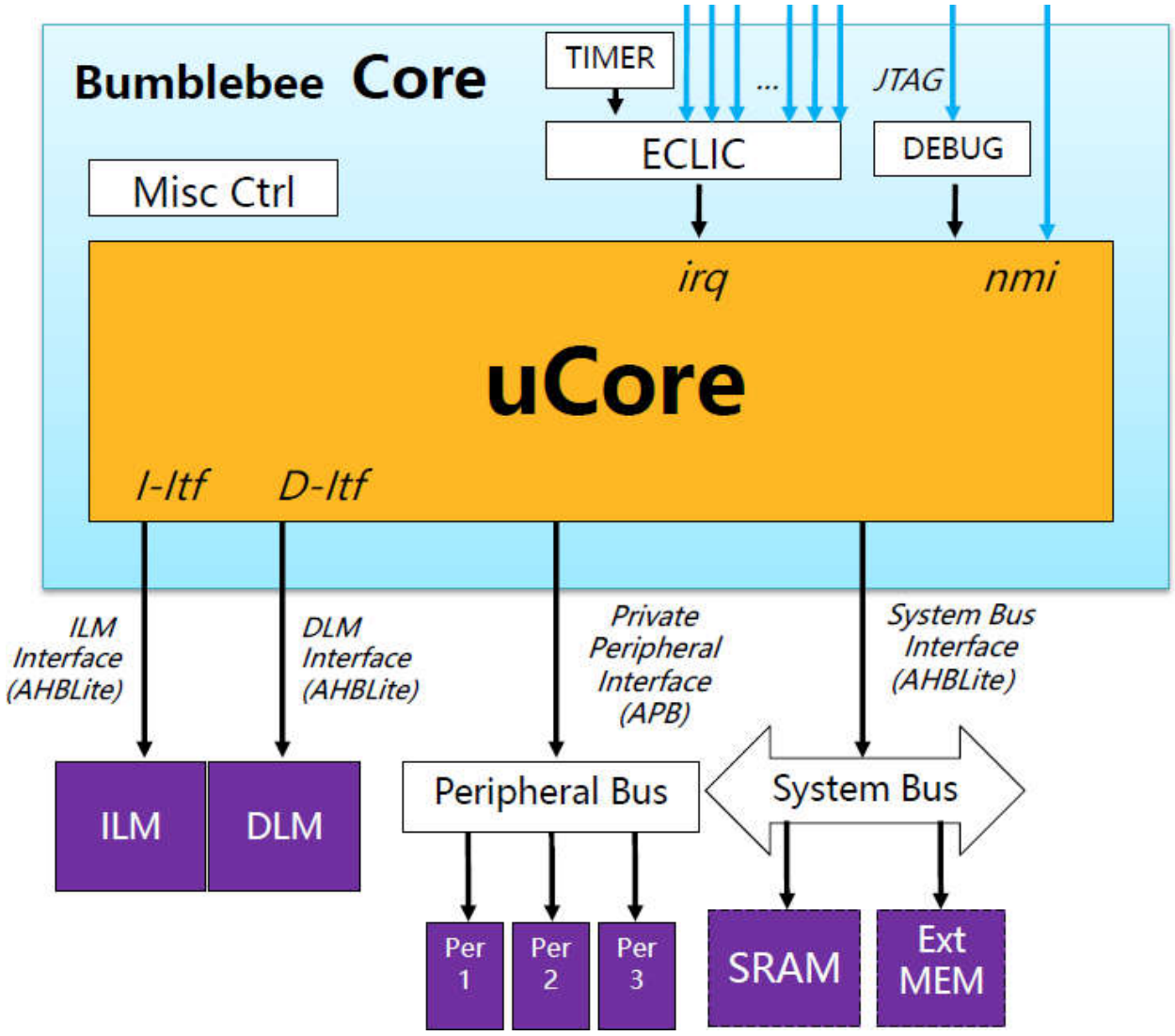

Figure 1-1 Top view of the Bumblebee kernel

2. Introduction to Bumblebee Kernel Features

2.1. Bumblebee kernel clock domain introduction

Bumblebee kernel clockDomain division shown in Figure 2-1 As shown, the entire processor core is divided into two clock domains that are asynchronous to each other:

- The working clock domain, driven by the input clocks core_clk and core_clk_aon, functions most of the processor core.note:
  - Core_clk and core_clk_aon are co-frequency, in-phase clocks from the same clock source.
  - Core_clk is the main operating clock that drives the main working logic inside the processor core and can be globally gated at the system level.
  - Core_clk_aon is a normally open clock that drives the Always-On logic in the core, mainly including ECLIC.
    TIMER and DEBUG.See the Bumblebee Kernel Instruction Architecture Handbook for more information on ECLIC and TIMER.
- The JTAG clock domain, driven by the input signal jtag_TCK, drives the JTAG debug-related logic of the processor core.

The above two clock domains are completely asynchronous, and asynchronous cross-clock domain processing has been performed in the internal implementation of the processor core.

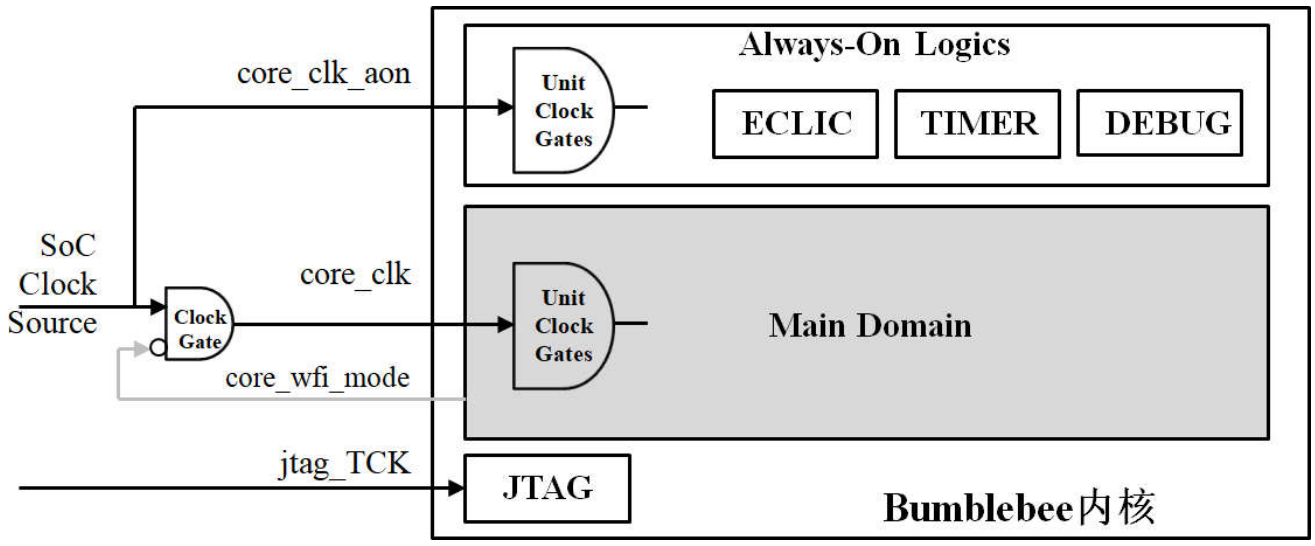

Figure 2-1 Bumblebee kernel clock domain diagram

2.2. Bumblebee kernel power domain introduction

The Bumblebee kernel does not have a power domain inside, and SoC system integrators can divide the power domain and cross-power domain processing according to the Bumblebee kernel hierarchy.

2.3. Introduction to the Bumblebee Kernel Interface

The Bumblebee kernel contains the following types of interfaces:

- Clock and reset interface
- Debug interface
- External interrupt interface
- Bus interface, including the following interfaces:
  - ILM Master Interface: An interface to access an external ILM.
  - DLM Master Interface: An interface that accesses an external DLM.
  - PPI Priperhal Interface: An interface that accesses an external private peripheral bus.
  - MEM interface (System Memory Interface): The interface to access the system bus.
- Other function interface

This document is not extensively described in detail on interface signals.

2.4. Bumblebee kernel address space allocation

Please refer to the data sheet of the specific MCU chip for the address space of the Bumblebee kernel.

2.5. Privileged mode of the Bumblebee kernel

The Bumblebee kernel supports two Privilege Modes: Machine Mode and User Mode.See the Bumblebee Kernel Instruction Architecture Handbook for more information on

2.6. Memory resources of the Bumblebee kernel

The Bumblebee kernel supports the following types of memory resources:

- ILM:
  - The Bumblebee kernel supports ILM access via a proprietary AHB-Lite bus or SRAM interface if an Instructon Local Memory (ILM) interface is configured.
  - The size of ilm can be configured.The ilm interface has independent address ranges. You can configure the specific base address. See section 2.4 Learn more about it.
  - The ILM is implemented by the SoC system integrator and can generally be an on-chip SRAM or on-chip for storing instructions.
    Flash.If you use the AHB-Lite interface, to achieve optimal performance, ILM should follow the AHB protocol rules and return instructions in the next cycle after receiving the address.
- DLM:
  - The Bumblebee kernel supports DLM access via a proprietary AHB-Lite bus or SRAM interface if a Data Local Memory (DLM) interface is configured.
  - The size of dlm can be configured.The dlm interface has a separate address range, and the user can configure a specific base address. 2.4 Learn more about it.
  - The DLM is implemented by the SoC system integrator and can generally be an on-chip SRAM for storing data.If you use the AHB-Lite interface, for optimal performance, the DLM should follow the AHB protocol rules and return data in the next cycle after receiving the address.

2.7. Bumblebee kernel private device

Figure 1-1 As shown in the Bumblebee kernel's Core hierarchy, in addition to uCore, it also includes the following private devices:

- Debug: Handles the jtag interface and related debugging features.
- Eclic: Kernel interrupt control unit.
- Timer: The kernel private timer unit.

The above devices belong to the processor core and are accessed by using the memory address addressing mode. For details on the specific address range allocation, see the 2.4 Section.

2.8. Physical storage protection for the Bumblebee kernel

Since the Bumblebee core is a low-power core for the microcontroller domain, it does not support virtual address management units.
(Memory Management Unit), so all address access operations are physical addresses used. In order to isolate and protect permissions based on different memory physical address ranges and different Privilege Modes, the RISC-V architecture standard defines a physical memory protection mechanism (Physical Memory Protection (PMP) unit.

Note: The Bumblebee kernel does not support PMP units.

2.9. Bumblebee kernel debugging mechanism

The Bumblebee kernel supports the standard JTAG debug interface and the proven interactive debugging tool GDB. note:

- The number of hardware breakpoints supported by the Bumblebee kernel is four. Hardware breakpoints are primarily used to set breakpoints to read-only intervals such as Flash.

The Bumblebee kernel defines an input signal, and i_dbg_stop can be controlled by the value of its input signal:

- If the value of the i_dbg_stop signal is 1, the debug function of the processor core is turned off.
- If the value of the i_dbg_stop signal is 0, the debug function of the processor core is working properly.

2.10. Bumblebee kernel interrupt and exception mechanism

For a detailed description of the Bumblebee kernel's interrupt and exception mechanisms, see the Bumblebee Kernel Instruction Architecture Handbook.

2.11. NMI mechanism of the Bumblebee kernel

NMI (Non-Maskable Interrupt) is a special input signal of the processor core, often used to indicate the system layer.

Emergency errors (such as external hardware failures, etc.).After encountering the NMI, the processor core should immediately abort execution of the current program and instead process the NMI error.For a detailed description of the NMI mechanism of the Bumblebee kernel, see the Bumblebee Kernel Instruction Architecture Handbook.

2.12. Bumblebee kernel CSR register

Some control and status registers (CSRs) are defined in the RISC-V architecture to configure or log the status of some operations.The CSR register is a register internal to the processor core and uses its proprietary 12-bit address encoding space.See the Bumblebee Kernel Instruction Architecture Handbook for details.

2.13. Performance counters for the Bumblebee kernel

The risc-v architecture defines the following two performance counters:

- Clock Counter:
  - A 64-bit wide clock cycle counter that reflects how many clock cycles the processor has executed.This counter continuously increments as long as the processor is in the execution state.
  - The CSR register mcycle reflects the lower 32 bits of the counter, and the CSR register mcycleh register reflects the 32-bit high value of the counter.See the Bumblebee Kernel Instruction Architecture Handbook for more information on mcycle and mcycleh.
- 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 increments as long as the processor completes an instruction every successful execution.

- The CSR register minstret reflects the lower 32 bits of the counter, and the CSR register minstreth reflects the 32-bit high value of the counter. See the Bumblebee Kernel Instruction Architecture Handbook for more information on minstret and minstreth.

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

By default, the counter has a value of 0 after a kernel reset and then continues to increment itself.Since the counter count consumes some dynamic power consumption, in the implementation of the Bumblebee kernel, a number of additional control fields are added to the custom CSR register mcountinhibit. The software can configure the corresponding control fields to separate the different counters. Stop, so stop counting when you don't need to use them to save power.

See the Bumblebee Kernel Instruction Architecture Manual for details on the CSR register mcountinhibit.

2.14. Timer unit of the Bumblebee kernel

The RISC-V architecture defines a 64-bit Timer Counter that is clocked by the system's low-speed Real Time Clock frequency.The value of this timer is reflected in the mtime register in real time.The RISC-V architecture also defines a 64-bit mtimecmp register that serves as a comparison value for the timer. A timer interrupt is generated assuming that the value mtime of the timer 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 system registers for Memory Address Mapped. The specific memory mapped address RISC-V architecture is not specified, but is instead The kernel designer implements it on its own.In the implementation of the Bumblebee kernel, mtime/mtimecmp is
TIMER unit implementation, see the Bumblebee Kernel Instruction Architecture Handbook for details on the TIBR unit of the Bumblebee kernel.

2.14.1. Timer behavior when debugging mode

When the Bumblebee kernel is in debug mode, it occasionally executes some debugger (Debugger) set code (in the DEBUG unit, which is invisible to the user) to support

the debugger's functionality. If the timer still counts while executing the code set by these debuggers, it does not truly reflect the true behavior of the program being debugged. Therefore, when the Bumblebee kernel executes the code set by the debugger, the timer will automatically stop counting.

2.14.2. Timer behavior in normal mode

By default, the timer has a value of 0 after a kernel reset and then continues to increment itself. In view of the fact that the timer count consumes some dynamic power, in the implementation of the Bumblebee kernel, an additional bit control field is added to the custom CSR register mcountinhibit, and the software can configure the control field to shut down the timer. So stop when you don't need to use them

Stop counting to achieve power saving.See the Bumblebee Kernel Instruction Architecture Manual for details on the CSR register mcountinhibit.

2.15. Low-power mechanism of the Bumblebee core

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

- The clocks of the main units inside the Bumblebee core are automatically gated off when idle to save static power.
- The Bumblebee kernel supports Sleep mode through common WFI (Wait for Interrupt) and WFE (Wait for Event) mechanisms for low dynamic and static power consumption. For "Wait for Interrupt" and "Wait for Event" See the Bumblebee Kernel Instruction Architecture Handbook for details.

2.15.1. Clock control into sleep state

The Bumblebee kernel can go to sleep by executing the WFI instruction. See the Bumblebee Kernel Instruction Architecture Manual for details on how to go to sleep.

The output signal core_sleep_value of the Bumblebee core can be used to indicate different sleep modes (0 or 1). Sleep mode 0 can usually be used for shallow sleep and sleep mode 1 is used for deep sleep.Note: After entering Deep Sleep mode, the processor core will no longer be able to be debugged by the JTAG debug interface.

The key points of the clock control (reference scheme) when the processor core enters the sleep state are as follows:

- Figure 2-1 When the WFI is successfully executed, the output signal core_wfi_mode of the Bumblebee core is pulled high, indicating that the processor core is in a sleep state after executing the WFI instruction; the SoC system level can be used.
  Core_wfi_mode Controls the external total gated clock to turn off the processor core's main operating clock, core_clk.
- If the Bumblebee kernel enters deep sleep mode (core_sleep_value is 1), the SoC system can decide whether to turn off the core's normally open clock core_clk_aon depending on its actual situation.

2.15.2. Clock control to exit sleep

The processor core can be interrupted, interrupted, or NMI awake. See the Bumblebee Kernel Instruction Architecture Manual for details on how to exit hibernate.

The key points of clock control when the processor core exits the sleep state are as follows:

- If the wakeup is waiting for an interrupt, the interrupt of the Bumblebee core needs to be processed and distributed by the ECLIC unit. The interrupt can only wake up the kernel after passing the conditions such as the enable and priority thresholds. In addition, pay special attention to whether the processor core's normally open clock (core_clk_aon) is off:
  - Such as the first **2.1** As described in the section, since TIMER is driven by core_clk_aon,
    - Assuming the SoC system level has turned off the processor core's normally open clock (core_clk_aon), then

      The timer unit cannot generate timer interrupts and software interrupts because it has no clock.
  - Such as the first **2.1** As described in the section, since ECLIC is driven by core_clk_aon, :
    - Assume that the SoC system level has turned off the processor core's normally-on clock (core_clk_aon), and the external interrupt signal line must be held high until the SoC system level turns the processor core's normally-on clock (core_clk_aon) back on. Otherwise, the ECLIC unit of the processor core cannot sample the external interrupt signal because there is no clock, and the processor core cannot be woken up.
- If it is a wake-up event (Event) or NMI wake-up, the kernel once (by the core_clk_aon clock) samples the input signal rx_evt (Event signal, active high) or nmi (NMI signal, rising edge active), from the sleep state Was awakened. In addition, pay special attention to whether the processor core's normally open clock (core_clk_aon) is off:
  - Assume that the SoC system level has turned off the processor core's normally open clock (core_clk_aon), the input signal

    Rx_evt or nmi must be held high after it is pulled high until the SoC system level turns the processor core's normally open clock (core_clk_aon) back on. Otherwise, the processor core's Event and NMI sampling logic cannot be sampled to Event and NMI because there is no clock and cannot be woken up.

- The output signal core_wfi_mode is pulled low immediately after the processor wakes up. Assume that the SoC system level is used.
  Core_wfi_mode controls the core's core_clk gated clock, and as the core_wfi_mode signal is pulled low, the processor core's working clock core_clk will be reopened.