



Copyright Notice

Copyright © 2018–2021 Nuclei System Technology. All rights reserved.

Nuclei™ are trademarks owned by Nuclei System Technology. All other trademarks used herein are the property of their respective owners.

The product described herein is subject to continuous development and improvement; information herein is given by Nuclei in good faith but without warranties.

This document is intended only to assist the reader in the use of the product. Nuclei System Technology shall not be liable for any loss or damage arising from the use of any information in this document, or any incorrect use of the product.

Contact Information

Should you have any problems with the information contained herein or any suggestions, please contact Nuclei System Technology by email support@nucleisys.com, or visit “Nuclei User Center” website <http://user.nucleisys.com> for supports or online discussion.

Revision History

| Rev. | Revision Date | Revised Section | Revised Content |
|--------------|---------------|-----------------|---|
| 1.0.0 | 2020/2/5 | N/A | 1. First version as the full English |
| 1.1.0 | 2020/3/9 | 2.1, 3.7, 3.8 | 1. Add more details of jtag_TCK and jtag_TMS 2. Fix icache_data1_wem width typo 3. Add more constraints for port sysrstreq |
| 1.2.0 | 2020/3/11 | 3.4.1, 3.7 | 1. Add CLK signal for SRAM of ILM/DLM/I-Cache |
| 1.3.0 | 2020/4/1 | 1.1, 2.4, 2.10 | 1. Add D-Cache and CCM |
| 1.3.1 | 2020/6/1 | 3.2 | 1. Add jtag_dwbypass to disable/bypass the 2-wire jtag mode in hardware |
| 1.3.2 | 2020/7/9 | 3.9 | 1. Add override_dm_sleep to support debug in low-power mode |
| 1.3.3 | 2020/7/24 | 3.6 | 1. Add trace_bjp_taken and trace_dmode in trace interface |
| 1.3.4 | 2020/11/30 | 4 | 1. Add I/D-LM_Slave_Port configuration option |
| 1.3.5 | 2021/1/20 | 3.6, 3.9 | 1. Add dbg_stoptime in core_wrapper to stop the SoC timer if necessary. 2. Add trace_cmt_en in core_wrapper to indicate a valid instruction is committing. |
| 1.4.0 | 2021/2/4 | 4 | 1. Update the Configuration options |
| 1.5.0 | 2021/6/30 | 4 | 1. To support 64-bit wide NICE operands without needing the DSP configuration. |

| Rev. | Revision Date | Revised Section | Revised Content |
|-------|---------------|-----------------|---|
| 1.6.0 | 2021/7/15 | 3 | 1. Add Notes that the ILM/DLM/I-Cache/D-Cache SRAM should keep the dout unchanged before the next request comming. |
| 1.7.0 | 2021/8/2 | 3.9 | 1. Add stop_on_reset input signal to stop the core after reset to allow other master (DMA) to preload the ILM/DLM through the slave port after reset. |

Table of Contents

| | |
|---|-----------|
| COPYRIGHT NOTICE..... | 0 |
| CONTACT INFORMATION..... | 0 |
| REVISION HISTORY..... | 1 |
| TABLE OF CONTENTS..... | 3 |
| LIST OF TABLES..... | 5 |
| LIST OF FIGURES..... | 6 |
| 1. OVERVIEW..... | 7 |
| 1.1. FEATURE LIST..... | 7 |
| 1.2. SUPPORTED INSTRUCTION SET AND PRIVILEGED ARCHITECTURE..... | 10 |
| 1.3. TOP DIAGRAM..... | 11 |
| 1.4. DIFFERENT CORES OF N300 SERIES..... | 13 |
| 2. FUNCTIONAL INTRODUCTIONS..... | 15 |
| 2.1. CLOCK DOMAINS..... | 15 |
| 2.2. POWER DOMAINS..... | 16 |
| 2.3. CORE INTERFACES..... | 16 |
| 2.4. MEMORY RESOURCES..... | 16 |
| 2.5. PRIVATE PERIPHERALS..... | 18 |
| 2.6. ADDRESS SPACES OF INTERFACES AND PRIVATE PERIPHERALS..... | 18 |
| 2.7. DEBUG SUPPORT..... | 20 |
| 2.8. INTERRUPT AND EXCEPTION MECHANISM..... | 20 |
| 2.9. NMI MECHANISM..... | 20 |
| 2.10. CONTROL AND STATUS REGISTERS (CSRs)..... | 21 |
| 2.11. PERFORMANCE MONITOR..... | 21 |
| 2.12. TIMER UNIT..... | 22 |
| 2.13. LOW-POWER MECHANISM..... | 23 |
| 2.13.1. Clock Control of Entering Sleep Modes..... | 23 |
| 2.13.2. Clock Control of Exiting Sleep Mode..... | 23 |
| 2.14. PHYSICAL MEMORY PROTECTION..... | 25 |
| 2.15. FPU FEATURE..... | 25 |
| 2.16. DSP FEATURE..... | 25 |
| 2.17. NICE FEATURE..... | 25 |
| 2.18. TEE FEATURE..... | 26 |
| 3. CORE INTERFACES..... | 27 |
| 3.1. CLOCK AND RESET INTERFACE..... | 27 |

| | | |
|-----------|--|-----------|
| 3.2. | 4-WIRE AND 2-WIRE JTAG DEBUG INTERFACE..... | 27 |
| 3.3. | INTERRUPT INTERFACE..... | 30 |
| 3.4. | BUS INTERFACES..... | 31 |
| 3.4.1. | ILM and DLM Interface..... | 31 |
| 3.4.2. | MEM Interface..... | 34 |
| 3.4.3. | Instruction Fetch Bus Interface for I-Cache..... | 35 |
| 3.4.4. | PPI Interface..... | 36 |
| 3.4.5. | FIO Interface..... | 37 |
| 3.5. | NICE INTERFACE..... | 37 |
| 3.6. | TRACE INTERFACE..... | 38 |
| 3.7. | I-CACHE SRAM INTERFACE..... | 38 |
| 3.8. | D-CACHE SRAM INTERFACE..... | 39 |
| 3.9. | OTHER FUNCTIONAL INTERFACE..... | 40 |
| 4. | CONFIGURABLE OPTIONS..... | 43 |

List of Tables

| | |
|---|----|
| TABLE 1-1 CONFIGURABLE FEATURES OF DIFFERENT CORES..... | 13 |
| TABLE 2-1 ADDRESS SPACES OF THE CORE..... | 18 |
| TABLE 3-1 CLOCK AND RESET SIGNALS..... | 27 |
| TABLE 3-2 JTAG SIGNALS..... | 28 |
| TABLE 3-3 INTERRUPT AND NMI SIGNALS..... | 30 |
| TABLE 3-4 ILM AHB-LITE SIGNALS..... | 32 |
| TABLE 3-5 ILM SRAM SIGNALS..... | 33 |
| TABLE 3-6 DLM AHB-LITE SIGNALS..... | 33 |
| TABLE 3-7 DLM SRAM SIGNALS..... | 34 |
| TABLE 3-8 MEM SIGNALS..... | 34 |
| TABLE 3-9 I-CACHE BUS SIGNALS..... | 36 |
| TABLE 3-10 PPI SIGNALS..... | 36 |
| TABLE 3-11 FIO SIGNALS..... | 37 |
| TABLE 3-12 TRACE INTERFACE SIGNALS..... | 38 |
| TABLE 3-13 I-CACHE SRAM INTERFACE..... | 38 |
| TABLE 3-14 D-CACHE SRAM INTERFACE..... | 39 |
| TABLE 3-15 OTHER INTERFACE SIGNALS..... | 40 |
| TABLE 4-1 THE CONFIGURABLE OPTION OF N300 SERIES..... | 43 |

List of Figures

| | |
|--|----|
| FIGURE 1-1 THE TOP DIAGRAM OF N300 SERIES CORE..... | 12 |
| FIGURE 2-1 CLOCK DOMAINS OF THE N300 SERIES CORE..... | 15 |
| FIGURE 3-1 THE EXAMPLE OF JTAG IO PADS..... | 30 |
| FIGURE 3-2 THE IO PAD WITH CONTROLLABLE PULL-UP AND PULL-DOWN..... | 30 |
| FIGURE 3-3 MTIME_TOGGLE_A SIGNAL GENERATION..... | 42 |

1. Overview

The Nuclei N300 Series Processor Core, or N300 Series Core for short , is a commercial RISC-V Processor Core Series designed by Nuclei System Technology for MCU, IoT or other low-power applications. The N300 Series is a competitive rival to ARM Cortex-M3/M4/M4F/M33 Processor Cores.

1.1. Feature List

The N300 Series have the following features:

- CPU Core
 - 3-pipeline stages , using state-of-the-art processor micro-architecture to deliver the best-of-class performance efficiency and lowest cost.
 - Support dynamic branch predictor.
 - Configurable instruction prefetch logic, which can prefetch subsequent two instructions to hide the instruction memory access latency.
- Supported Instruction Set Architecture (ISA)
 - The N300 Series is a 32-bit RISC-V Processor Core Series, supporting the combination of RV32I/M/A/F/D/C/P instruction extensions.
 - Support misaligned memory access (Load/Store instructions).
- Supported Privileged Modes
 - Support Machine Mode, Supervisor Mode (configurable) and User Mode (configurable).
- Bus Interfaces
 - Support 32-bit wide standard AHB-Lite system bus interface for accessing system instruction and data.
 - Support 32-bit wide standard AHB-Lite I-Cache bus interface (configurable, if not configured, instruction fetch and data load will share the system bus) for instruction fetch when I-Cache miss.
 - Support 32-bit wide Instruction Local Memory (ILM) bus interface (configurable with standard AHB-Lite or SRAM interface protocol) for accessing private instruction local memory.
 - Support 32-bit wide Data Local Memory (DLM) bus interface (configurable with standard AHB-Lite or SRAM interface protocol) for accessing private data local memory.
 - Support 32-bit wide Private Peripheral Interface (PPI) bus interface (with standard APB interface protocol) for accessing private

- peripherals.
- Support 32-bit wide Fast-IO Interface (FIO) bus interface (with simple zero-cycle interface) for accessing private fast peripherals, e.g., GPIO for fast IO manipulation.
- Low-Power Management
 - Support WFI (Wait For Interrupt) and WFE (Wait For Event) scheme to enter sleep mode.
 - Support two-level sleep modes: shallow sleep mode, and deep sleep mode.
- Core-Private Timer Unit (TIMER)
 - 64-bits wide real-time counter.
 - Support the generation of the timer interrupt defined by the RISC-V standard.
 - Support the generation of the precise periodic timer interrupt (can be used as System Tick) with auto clear-to-zero mode.
 - Support the generation of software interrupt defined by the RISC-V standard.
- Enhanced Core Level Interrupt Controller (ECLIC)
 - Support the RISC-V architecturally defined software, timer and external interrupts.
 - Support configurable number of external interrupt sources.
 - Support configurable number of interrupt levels and priorities, and support software dynamically programmable division of interrupt levels and priorities values.
 - Support interrupts preemption based on interrupt levels.
 - Support vectored interrupt processing mode for extremely fast interrupt response (6 cycles).
 - Support fast interrupts tail-chaining mechanism.
- Support NMI (Non-Maskable Interrupt)
- Memory Protection
 - Support configurable Physical Memory Protection (PMP) to protect the memory.
- Security with Trust Execution Environment
 - Support configurable Trust Execution Environment (TEE) feature.
- Support Instructions Extended by User
 - Support Nuclei Instruction Co-unit Extension (NICE) scheme to support

user to extend custom instructions according to their applications.

■ Support Instruction Cache (I-Cache)

- The Cache size is configurable.
- 2-way associative structure.
- Cache Line Size is 32 Bytes.
- Support configurable Scratchpad mode.
- Support to LOCK, INVAL cachelines.

■ Support Data Cache (D-Cache)

- The Cache size is configurable.
- 2-way associative structure.
- Cache Line Size is 32 Bytes.
- Support to LOCK, FLUSH and/or INVAL cachelines.

■ Debugging System

- Support standard 4-wire or 2-wire JTAG interface.
- Support standard RISC-V debug specification (v0.13).
- Support configurable number of Hardware Breakpoints.
- Support mature interactive debugging software/hardware tools, such as GDB, OpenOCD, Lauterbach TRACE32, SEGGER J-Link, IAR, etc.

■ Software Development Tools

- The N300 Series support the RISC-V standard compilation toolchain and Integrated Development Environment (IDE) on Linux/Windows systems, such as GCC, MCU-Eclipse, Nuclei-Studio, SEGGER Embedded Studio, IAR, etc.

1.2. Supported Instruction Set and Privileged Architecture

The N300 Series is following the Nuclei RISC-V Instruction Set and Privileged Architecture Specification (Nuclei_RISCV_ISA_Spec), user can easily get the specification from “Nuclei User Center” website <http://user.nucleisys.com>.

1.3. Top Diagram

The top diagram of N300 Series is as depicted in Figure 1-1, the key points of which are:

- Core Wrapper is the top module of the Core, the sub-modules of it are:
 - Core: The core part.
 - Reset Sync: The module to sync external async reset signal to synced reset with “Asynchronously assert and synchronously de-assert” style.
 - DEBUG: The module to handle the debug functionalities.
- uCore is under Core hierarchy, it is the main part of Core.
- Besides the uCore, there are several other sub-modules:
 - LM Ctrl: The control module for ILM and DLM.
 - ECLIC: The private interrupt controller.
 - TIMER: The private timer unit.
 - BIU: The bus interface unit.
 - Misc Ctrl: Other miscellaneous modules.

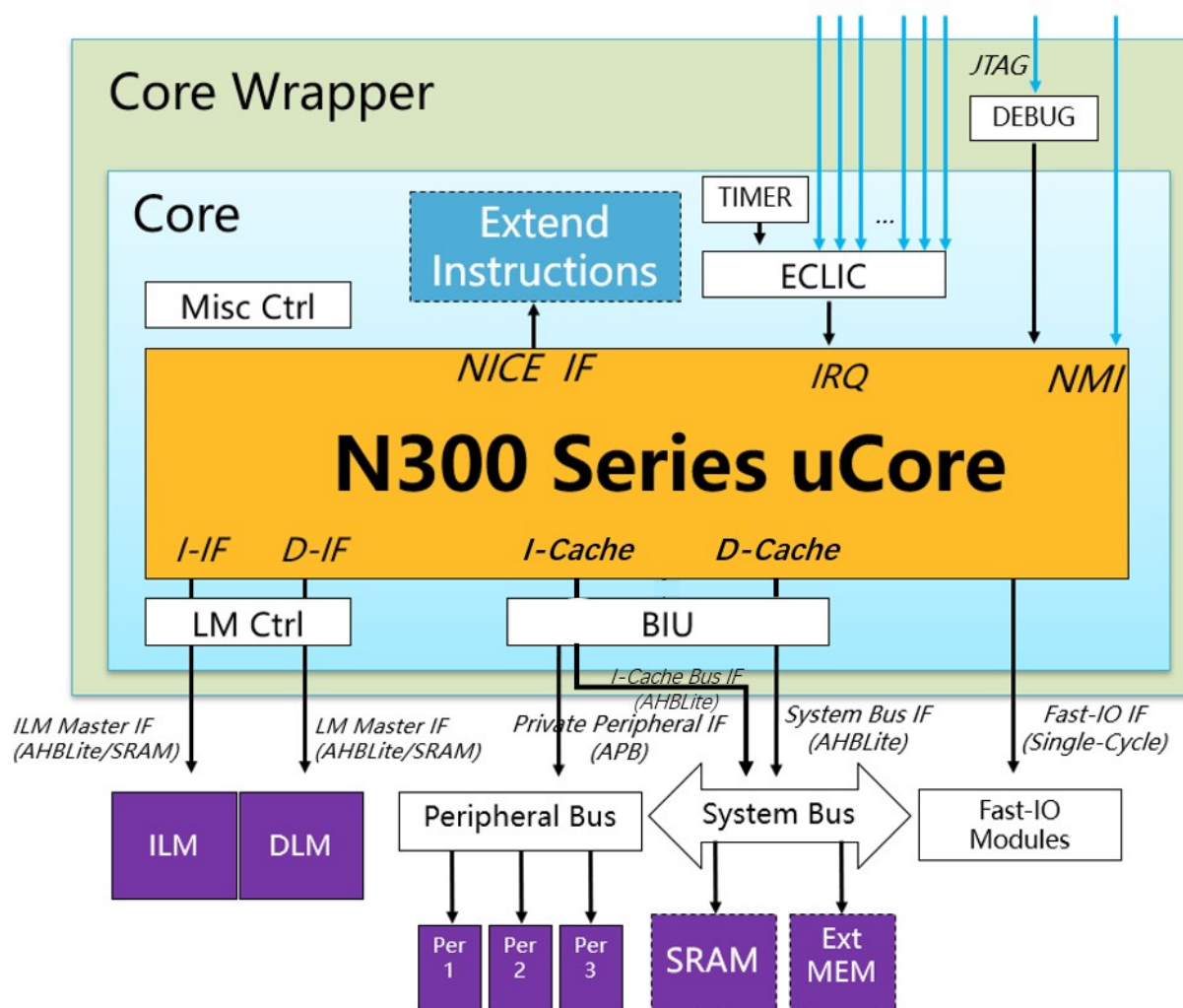


Figure 1-1 The top diagram of N300 Series Core

1.4. Different Cores of N300 Series

The N300 is a series of Cores with different configuration templates. Different Cores have different configurable features, as briefly summarized in Table 1-1.

Table 1-1 Configurable features of different Cores

| | N305 | N307 | N308 | N309 |
|--|-----------------------|-----------------------|-----------------------|-----------------------|
| Baseline Instruction Set | IMAC | IMAC/ IMACFDP | IMAC/ IMACFDP | IMAC/ IMACFDP |
| Hardware Multiplier | YES (Single-Cycle) | YES (Single-Cycle) | YES (Single-Cycle) | YES (Single-Cycle) |
| Hardware Divider | YES | YES | YES | YES |
| A (Atomic) Instruction Extension | YES | YES | YES | YES |
| Unaligned Load/Store Hardware Support | YES | YES | YES | YES |
| Low Power Features | YES | YES | YES | YES |
| Interrupts Number | Configurable | Configurable | Configurable | Configurable |
| NMI | YES | YES | YES | YES |
| ILM and DLM Interface | Configurable | Configurable | Configurable | Configurable |
| PPI Interface | Configurable | Configurable | Configurable | Configurable |
| Fast-IO Interface | Configurable | Configurable | Configurable | Configurable |
| Instruction Cache (I-Cache) | Configurable | Configurable | Configurable | Configurable |
| Scratchpad Mode for I-Cache | Configurable | Configurable | Configurable | Configurable |
| User Mode and PMP | Configurable | Configurable | Configurable | Configurable |
| Debugging System | Configurable | Configurable | Configurable | Configurable |
| User Instruction Extendibility (NICE) | Configurable | Configurable | Configurable | Configurable |
| Timing Enhancing Options | Configurable | Configurable | Configurable | Configurable |
| Performance Enhancing Options | Configurable | Configurable | Configurable | Configurable |
| Single-Precision Floating point Unit | NO | Configurable | Configurable | Configurable |
| Double-Precision Floating point Unit | NO | Configurable | Configurable | Configurable |
| Packed-SIMD DSP | NO | Configurable | Configurable | Configurable |
| TEE Support | NO | NO | Configurable | Configurable |
| Data Cache (D-Cache) | NO | NO | NO | Configurable |

2.Functional Introductions

2.1. Clock Domains

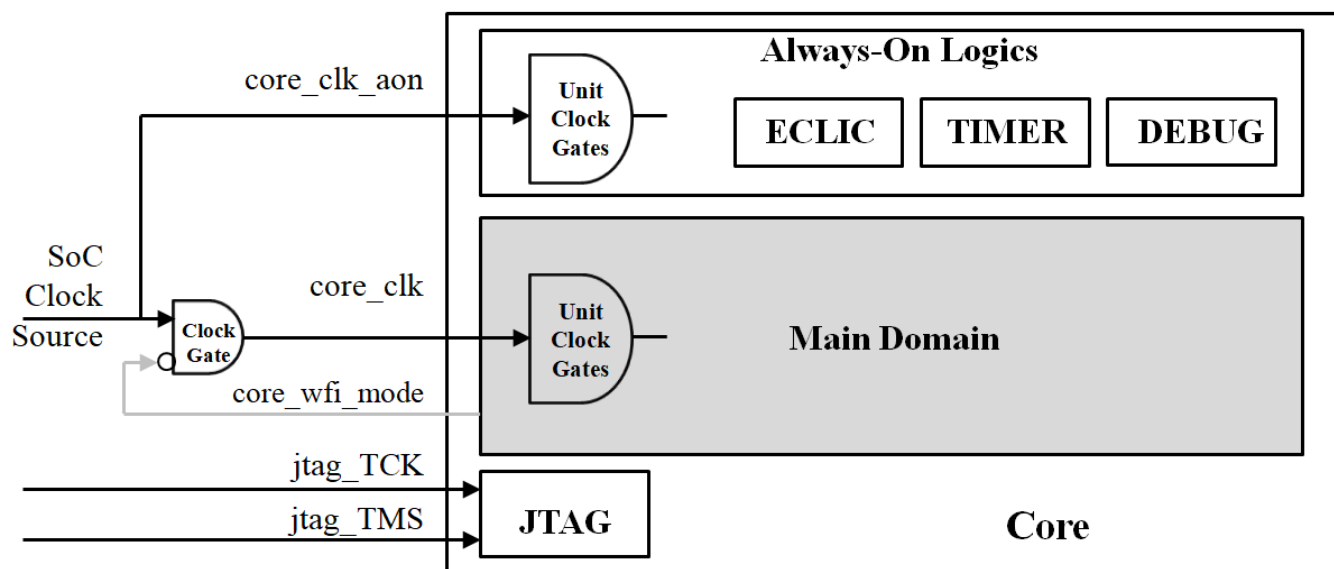


Figure 2-2 Clock domains of the N300 Series Core

The clock domains of the N300 Series Core are shown in Figure 2-1. The entire Core is divided into three asynchronous clock domains:

- The main clock domain (for the `core_clk` and `core_clk_aon`), which drive most of the functional logics of the Core. Note:
 - `core_clk` and `core_clk_aon` have the same frequency and same phases as they are supposed to be clocks from the same clock source.
 - `core_clk` is the main working clock that drives the main domain inside the Core, `core_clk` is supposed to be clock-gated at the SoC level during the sleep mode.
 - `core_clk_aon` is an always-on clock that drives the always-on logic in the Core, including the ECLIC, TIMER, DEBUG, etc.
- The JTAG_CLK clock domain (for the `jtag_TCK`), which drives the JTAG logics of DEBUG unit. Note:
 - For generated clock `tck_s` inside JTAG, which is a divide-by-3 clock of `jtag_TCK`, should be also created in synthesis. More details could be referred to in <Nuclei_Processor_Integration_Guide> section 7.2.
- The JTAG_TMS clock domain (for the `jtag_TMS`), since the `jtag_TMS` will be used as clock when switching between 4-wire and 2-wire JTAG modes,

hence the jtag_TMS is also a clock driving very few JTAG logics of DEBUG unit. In synthesis, the frequency of jtag_TMS could be set to half of jtag_CLK.

The above three clock domains are asynchronous with each other, so asynchronous cross-clock domain processing has been implemented in the Core.

2.2. Power Domains

There is no power domains specified inside N300 Series Core. Per different applications, the SoC integrator can choose to divide the power domains according to the convenience of the hierarchies inside the Core.

2.3. Core Interfaces

Please refer to Chapter 3. for the details of Core interfaces.

2.4. Memory Resources

N300 Series Core supports the following memory resources:

■ ILM:

- The Core supports Instruction Local Memory (ILM) access via an independent AHB-Lite or SRAM interface if the ILM interface is configured.
- The address space of the ILM is configurable. Please see Section 2.6. for details.
- The ILM is implemented by the SoC integrator and can generally be an SRAM or Flash for storing instructions. If the AHB-Lite interface is implemented, to achieve best performance, the interface should return response at the next cycle after receiving the read request.

■ DLM:

- The Core supports Data Local Memory (DLM) access via an independent AHB-Lite or SRAM interface if the DLM interface is configured.
- The address space of the DLM is configurable. Please see Section 2.6. for details.
- The DLM is implemented by the SoC integrator and can generally be an SRAM for storing data. If the AHB-Lite interface is implemented, to

achieve best performance, the interface should return response at the next cycle after receiving the read/write request.

■ I-Cache:

- The Core supports I-Cache (Instruction Cache) to cache the instruction fetched from MEM interface. Note: the instruction fetched from ILM interface will not be cached, i.e., if the instruction fetch address fall into ILM address space, it will directly access ILM interface with the I-Cache bypassed.
- I-Cache is 2-ways associative structure, with Line Size as 32 Bytes. The size of I-Cache is configurable. Please refer to Chapter 4. for more details of configurations.
- Some extended CSR registers are defined for I-Cache Control and Maintenance (CCM) operations, which can be used to INVAL, LOCK and UNLOCK I-Cache by ADDR or by ALL. Please refer to the “Nuclei_CCM_Mechanism” for more details, user can easily get the specification from “Nuclei User Center” website <http://user.nucleisys.com>.
- If the core is not configured with ILM, just configured with I-Cache, then the I-Cache can be configured to support the Scratchpad Mode.
- When under Scratchpad Mode, the Data SRAM of I-Cache will be reused and downgraded to memory mapped SRAM which can be accessed by instruction fetch and data access, like ILM and DLM, but called as Scratchpad here.
- When under Scratchpad Mode, the base address of Scratchpad is configurable. Please refer to Chapter 4. for more details of configurations.
- This Scratchpad Mode is controlled by CSR register mcache_ctl, please refer to “Nuclei_RISCV_ISA_Spec” for more details, user can easily get the specification from “Nuclei User Center” website <http://user.nucleisys.com>.

■ D-Cache:

- The Core supports D-Cache (Data Cache) to cache the data fetched from MEM interface with cacheable attribute address space. Note: the data fetched from ILM and DLM interface will not be cached.
- D-Cache uses Write-Back and Write-Allocate mechanism, which is 2-ways associative structure, with Line Size as 32 Bytes. The size of D-Cache is configurable. Please refer to Chapter 4. for more details of configurations.
- Some extended CSR registers are defined for D-Cache Control and Maintenance (CCM) operations, which can be used to FLUSH, INVAL, LOCK and UNLOCK D-Cache by ADDR or by ALL. Please refer to “Nuclei_CCM_Mechanism” for more details, user can easily get the specification from “Nuclei User Center” website

<http://user.nucleisys.com>.

2.5. Private Peripherals

As shown in Figure 1 -1, under the Core hierarchy, 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 peripherals are private to the Core and are accessed using memory mapped address. See Section 2.6. for the details of their specific address space allocation.

2.6. Address Spaces of Interfaces and Private Peripherals

There are quite several interfaces and private peripherals for the Core, the address spaces of them are shown in Table 2 -2.

Table 2-2 Address Spaces of the Core

| Unit | Base Address | Offset | Description |
|-------|--------------|---|---|
| DEBUG | Configurable | 0x000~ 0xFFFF | <ul style="list-style-type: none"> ■ Address space of DEBUG unit. ■ Note: DEBUG is private inside the core. And DEBUG is used for debugging functionality. The regular application software should not access this space. |
| ECLIC | Configurable | 0x0000 ~ 0xFFFF | <ul style="list-style-type: none"> ■ Address space of ECLIC unit. ■ Note: ECLIC is private inside the core. |
| TIMER | Configurable | 0x000 ~ 0xFFFF | <ul style="list-style-type: none"> ■ Address space of TIMER unit. ■ Note: TIMER is private inside the core. |
| ILM | Configurable | Depends on the configuration of ILM address space | <ul style="list-style-type: none"> ■ Address space of ILM interface. |
| DLM | Configurable | Depends on the configuration of DLM address space | <ul style="list-style-type: none"> ■ Address space of DLM interface. |
| FIO | Configurable | Depends on the configuration of FIO address | <ul style="list-style-type: none"> ■ Address space of FIO interface. |

| | | space | |
|--|--------------|---|---|
| PPI | Configurable | Depends on the configuration of PPI address space | ■ Address space of PPI interface. |
| MEM | N/A | N/A | ■ The address space other than the above mentioned spaces are all to MEM (System Memory) interface. |
| Note: please refer to Chapter 4. for more details of configurations. | | | |

There are several key points:

- The Core's instruction fetches will not go to DLM, ECLIC, TIMER, FIO, or PPI anyway.
- Since the Core have instruction fetch and data access paths independent inside the Core, the address space of ILM and DLM can be overlapped.
 - If the Core has not been configured with "N300_CFG_LSU_ACCESS_ILM", then the data access cannot access ILM interface anyway.
 - If the Core has been configured with "N300_CFG_LSU_ACCESS_ILM", then the data access can access ILM interface if the data access address fall into the ILM address space. Note: if the data access address fall into both ILM and DLM address spaces (since address spaces of ILM and DLM could be overlapped), then the data access still go to ILM interface.
- The total address space of "ILM and DLM" should not overlap with the total address space of "DEBUG, TIMER, ECLIC, FIO and PPI", otherwise it is configuration error.
- The address spaces of DEBUG, TIMER, ECLIC, FIO, and PPI should not overlap; otherwise it is the configuration error.

2.7. Debug Support

N300 Series Core supports standard 4-wire or 2-wire JTAG interface, standard RISC-V debug specification (v0.13), configurable number of Hardware Breakpoints, and mature interactive debugging software/hardware tools, such as GDB, OpenOCD, Lauterbach TRACE32, Segger J-Link, IAR, etc.

N300 Series Core defines an input signal `i_dbg_stop`, which can be controlled by the SoC level to disable the debug functionality or not:

- If the value of the `i_dbg_stop` signal is 0, the debug functionality of the

Core is working properly.

- If the value of the `i_dbg_stop` signal is 1, the debug functionality of the Core is off, and the external Debug Hardware Probe cannot debug the Core through JTAG interface anymore.

2.8. Interrupt and Exception Mechanism

For a detailed description of the Core's interrupt and exception mechanisms, please refer to Nuclei ISA specification (Nuclei_RISCV_ISA_Spec), user can easily get the specification from “Nuclei User Center” website <http://user.nucleisys.com>.

2.9. NMI Mechanism

NMI (Non-Maskable Interrupt) is a special input signal of the Core, often used to indicate system-level emergency errors (such as external hardware failures, etc.). After encountering the NMI, the Core should abort execution of the current program immediately and process the NMI handler instead. For a detailed description of the NMI mechanism of the N300 Series Core, please refer to Nuclei ISA specification (Nuclei_RISCV_ISA_Spec), user can easily get the specification from “Nuclei User Center” website <http://user.nucleisys.com>.

2.10. Control and Status Registers (CSRs)

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 Core that uses their private 12-bit encoding space to access.

Some extended CSR registers are defined for I/D-Cache Control and Maintenance (CCM) operations, which can be used to FLUSH, INVALID, LOCK and UNLOCK I/D-Cache by ADDR or by ALL.

For a detailed description of the Core's CSRs, please refer to Nuclei ISA specification (Nuclei_RISCV_ISA_Spec), user can easily get the specification from “Nuclei User Center” website <http://user.nucleisys.com>.

2.11. Performance Monitor

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 Core has executed. This counter will continuously increment as long as the Core's core_clk_aon is ON.
- The CSR mcycle reflect the lower 32 bits of the counter, and the CSR mcycleh reflect the upper 32 bits of the counter. Please refer to Nuclei ISA specification (Nuclei_RISCV_ISA_Spec) for the details; user can easily get the specification from "Nuclei User Center" website <http://user.nucleisys.com>.

■ Instruction Retirement Counter:

- The RISC-V architecture defines a 64-bit wide instruction completion counter that reflects how many instructions the Core successfully executed. This counter will increment if the processor executes an instruction successfully.
- The CSR minstret reflect the lower 32 bits of the counter, and the CSR minstreth reflect the upper 32 bits of the counter. Please refer to Nuclei ISA specification (Nuclei_RISCV_ISA_Spec) for more details; user can easily get the specification from "Nuclei User Center" website <http://user.nucleisys.com>.

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

By default, the counter is zero value after a reset and then increments itself continuously. But in Nuclei N300 Series Core, considering the counter increases the power consumption, there is an extra bit in the customized CSR mcountinhibit that can pause the counter to save power when users don't need to monitor the performance through the counter. Please refer to Nuclei ISA specification (Nuclei_RISCV_ISA_Spec) for more details; user can easily get the specification from "Nuclei User Center" website <http://user.nucleisys.com>.

2.12. TIMER Unit

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 N300 Series Core, mtime and mtimecmp are

both implemented in the TIMER Unit.

Besides, the TIMER Unit of N300 Series Core can also generate the periodic timer interrupt (normally as System Tick) and the software interrupt, please refer to Nuclei ISA specification (Nuclei_RISCV_ISA_Spec) for more details; user can easily get the specification from “Nuclei User Center” website <http://user.nucleisys.com>.

2.13. Low-Power Mechanism

The low-power mechanism of the N300 Series Core is as below:

- The clocks of the main units in the Core are automatically gated off when they are in idle state to reduce static power consumption.
- The Core supports different sleep modes (shallow sleep mode or deep sleep mode) through WFI (Wait for Interrupt) and WFE (Wait for Event) mechanisms to achieve lower dynamic and static power consumption. For more details about “Wait for Interrupt” and “Wait for Event” mechanism, please refer to Nuclei ISA specification (Nuclei_RISCV_ISA_Spec) for more details; user can easily get the specification from “Nuclei User Center” website <http://user.nucleisys.com>.

2.13.1. Clock Control of Entering Sleep Modes

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

- As shown in Figure 2-1, when the WFI/WFE is successfully executed, the output signal `core_wfi_mode` of the Core is asserted, indicating that the Core has entered to the sleep mode; at the SoC level, the signal `core_wfi_mode` should be used to control the external gate logic to disable the `core_clk`.
- If the 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.

2.13.2. Clock Control of Exiting Sleep Mode

The Core can be waked up by interrupt, event, or NMI. Please refer to Nuclei ISA specification (Nuclei_RISCV_ISA_Spec) for more details; user can easily get the specification from “Nuclei User Center” website <http://user.nucleisys.com>.

The key points of the clock control of the Core exiting the sleep mode are as the followings:

- The output signal `core_wfi_mode` will be de-asserted immediately after the core being waked up. Assuming the SoC 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 de-asserted.
- For the case that the core is waiting for an interrupt to wake up, because the 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`, so if 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 working clock, and the core cannot be woken up.
 - As mentioned in Section 2.1., the ECLIC unit is clocked by `core_clk_aon`, so if the SoC system has disabled the always-on clock `core_clk_aon`, then the external interrupt signal must kept asserted until the SoC enable the `core_clk_aon` again. Otherwise, the ECLIC unit cannot sample the external interrupt signal because it has no working 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 sampled (by the `core_clk_aon`) the input signal `rx_evt` (indicate there is one event) or the input signal `nmi` (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:

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

2.14. Physical Memory Protection

Since the N300 Series Core is 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 and execution privilege mode, the RISC-V standard architecture defines a physical memory protection mechanism: Physical Memory Protection (PMP).

N300 Series Core can be configured to support the PMP feature. About the programming mode of PMP, please refer to Nuclei ISA specification (Nuclei_RISCV_ISA_Spec), user can easily get the specification from “Nuclei User Center” website <http://user.nucleisys.com>.

2.15. FPU Feature

N300 Series Core can be configured to support single-precision (F extension) or double-precision (D extension) floating point instructions. For the details of F/D extension, please refer to Nuclei RISC-V Instruction Set and Privileged Architecture Specification (Nuclei_RISCV_ISA_Spec), user can easily get the specification from “Nuclei User Center” website <http://user.nucleisys.com>.

2.16. DSP Feature

N300 Series Core can be configured to support Packed-SIMD DSP instructions, please refer to another document <Nuclei_DSP_QuickStart> for the details of DSP instructions. User can easily get the copy from “Nuclei User Center” website <http://user.nucleisys.com>.

2.17. NICE Feature

N300 Series Core can be configured to support user to add their custom instructions with NICE (Nuclei Instruction Co-unit Extension) interface. Please refer to another document <Nuclei_NICE_Extension> for the details. User can easily get the copy from “Nuclei User Center” website <http://user.nucleisys.com>.

2.18. TEE Feature

N300 Series Core can be configured to support the TEE (Trust Execution Environment) feature. About the details of TEE, please refer to another document <Nuclei_TEE_Architecture>. User can easily get the copy from “Nuclei User Center” website <http://user.nucleisys.com>.

3. Core Interfaces

3.1. Clock and Reset Interface

The clock and reset signals of N300 Series Core are as depicted Table 3-3.

Table 3-3 Clock and Reset Signals

| Signal Name | Direction | Width | Description |
|----------------|-----------|-------|--|
| core_clk_aon | Input | 1 | ■ This clock is to drive the Always-On Logics of the Core, please refer to Section 2.1. for more details. |
| core_clk | Input | 1 | ■ This clock is to drive the Main Logics of the Core, please refer to Section 2.1. for more details. |
| por_reset_n | Input | 1 | ■ Power on Reset. This signal is active low signal. This reset will reset the entire N300 Core including the JTAG logics. ■ Note: this reset signal will be synced inside N300 Core to make it as “asynchronously asserted and synchronously de-asserted” style. |
| core_reset_n | Input | 1 | ■ System Reset. This signal is active low signal. This reset will reset the N300 Core except the JTAG logics. ■ Note: this reset signal will be synced inside N300 Core to make it as “asynchronously asserted and synchronously de-asserted” style. |
| reset_bypass | Input | 1 | ■ If the reset_bypass signal is high, then the internal generated reset will be bypassed, to allow DFT (Design For Test) rules. ■ Note: if the reset_bypass is high, the core_reset_n will be bypassed and disabled, only the por_reset_n will really take effects. |
| clkgate_bypass | Input | 1 | ■ If the clkgate_bypass is high, the clock gater will be bypassed, to allow DFT (Design For Test) rules. |

3.2. 4-wire and 2-wire JTAG Debug Interface

N300 Series Core can be configured to support the standard 4-wire JTAG and 2-wire JTAG interface (two modes can be switched dynamically); it is compliant to IEEE 1149.1 T4 Wide protocol. For the 2-wire mode, the Oscan0 and Oscan1 mode are supported.

The JTAG signals of N300 Series Core are as depicted Table 3-4.

Table 3-4 JTAG Signals

| Signal Name | Direction | Width | Description |
|----------------|-----------|-------|--|
| jtag_TCK | Input | 1 | <ul style="list-style-type: none"> JTAG TCK signal. Note: this signal needs to be constrained as Clock (asynchronous to the Core's main clock). |
| jtag_TMS_in | Input | 1 | <ul style="list-style-type: none"> 4-wire JTAG TMS signal Or 2-wire JTAG TMS input signal from IO PAD. Note: if support 2-wire mode, this signal need to be constrained as Clock (asynchronous to the Core's main clock and jtag_TCK). |
| jtag_TMS_out | Output | 1 | <ul style="list-style-type: none"> 2-wire JTAG TMS output signal to IO PAD. Note: if the 2-wire mode does not need to be supported, then this signal is unused. |
| jtag_BK_TMS | Output | 1 | <ul style="list-style-type: none"> 2-wire JTAG TMS bus keep control signal to IO PAD. The functionality of this signal will be detailed at late parts. Note: if the 2-wire mode does not need to be supported, then this signal is unused. |
| jtag_DRV_TMS | Output | 1 | <ul style="list-style-type: none"> 2-wire JTAG TMS output enable signal to IO PAD. When the TMS is outputting, this DRV_TMS signal will be high to enable IO PAD as output. Note: if the 2-wire mode does not need to be supported, then this signal is unused. |
| jtag_TDI | Input | 1 | <ul style="list-style-type: none"> 4-wire JTAG TDI signal. |
| jtag_TDO | Output | 1 | <ul style="list-style-type: none"> 4-wire JTAG TDO signal. |
| jtag_DRV_TDO | Output | 1 | <ul style="list-style-type: none"> 4-wire JTAG TDO output enable signal to IO PAD. When the TDO is outputting, this DRV_TDO signal will be high to enable IO PAD as output. |
| jtag_dwen | Output | 1 | <ul style="list-style-type: none"> Indicate it is in 2-wire mode. Note: if the SoC level is chaining several JTAG TAPs with Core's JTAG, then when Core's JTAG TAP is in 2-wire mode, it may break other standard 4-wire JTAG TAPs on the chain to mis-function. This jtag_dwen signal can be used to disable other standard 4-wire JTAG TAPs on the chain to make sure they functioned correctly. |
| jtag_dwbyypass | Input | 1 | <ul style="list-style-type: none"> To disable the 2-wire mode. 1 is to disable/bypass the 2-wire mode in hardware; 0 is to reserve the 2-wire mode, software can enable/disable this 2-wire mode. |

The example JTAG IO PADs are depicted as in Figure 3-3. Note:

- The "Optional" part (TDI and TDO) can just not be implemented. If not implemented, the SoC will just support only 2-wire JTAG mode, i.e., IEEE 1149.7 T4 Narrow standard.

- In the IEEE 1149.7 specification, the 2-wire JTAG require JTAG_TMS_PAD to have the “Bus Keep” functionality inside the SoC Chip.
 - Nuclei designed “HBird Debugger Kit” will have the “Bus Keeper Circuit” included, so the SoC Chip does not need to implement the “Bus Keep” functionality inside it, i.e., JTAG_TMS_PAD no need to have “Bus Keep” feature.
 - But other third part Debugger Hardware Probes may not have the “Bus Keeper Circuit” included, hence, it is still recommended to have JTAG_TMS_PAD implemented with “Bus Keep” feature.
- If user SoC Chip’s IO PAD does not have “Bus Keep” feature, it can be implemented with “IO PAD with controllable pull-up and pull-down of” to achieve it, as depicted in Figure 3-4.
 - User can manipulate the pull-up and pull-down control signal as below logic codes:

```
// If the Bus Keep is not enabled, the IO Pad will be pull-up by default.
// If the Bus Keep is enabled, then the pull-up enable is driven from PAD_IN, and pull-down enable is driven from ~PAD_IN, to make the
// IO PAD weakly keep the value of PAD_IN.

assign RPUEN = bk_en? PAD_IN : 1'b1;
assign RPDEN = bk_en? ~PAD_IN : 1'b0;

// Note: the above bk_en control signal may come from jtag_BK_TMS directly, or combined with other software programmable register bit. For
// example,

bk_en = jtag_BK_TMS & ctrl_bk_enable.

// ctrl_bk_enable is from SoC level programmable register bit, such that user can use software to disable “Bus Keep” functionality.
```

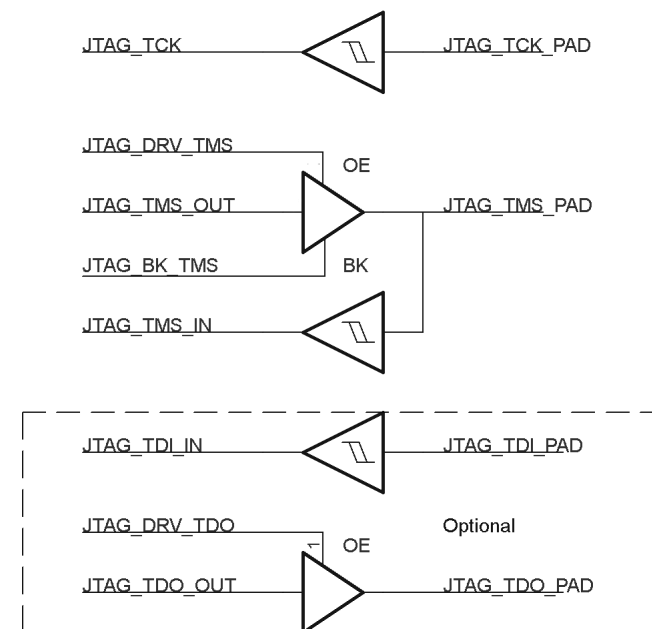


Figure 3-3 The example of JTAG IO PADS

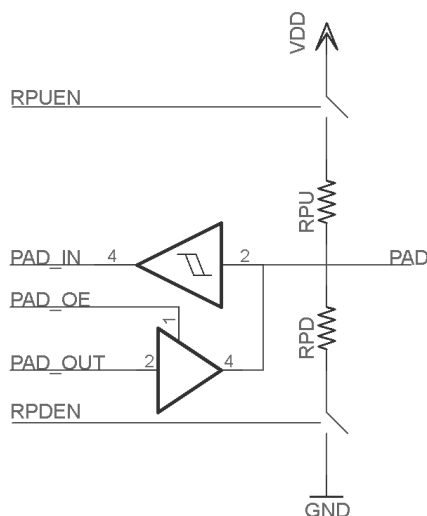


Figure 3-4 The IO PAD with controllable pull-up and pull-down

3.3. Interrupt Interface

The interrupt signals of N300 Series Core are as depicted Table 3-5

Table 3-5 Interrupt and NMI signals

| Signal Name | Direction | Width | Description |
|-------------|-----------|--------------|--|
| nmi | Input | 1 | <ul style="list-style-type: none"> NMI (Non-Maskable Interrupt) input. Note: <ul style="list-style-type: none"> nmi_irq signal will not be synced inside the Core, so this signal need to be synced at the SoC level if it is from different clock domain at SoC. Please refer to Section 2.9. for the details of NMI. |
| clic_irq | Input | Configurable | <ul style="list-style-type: none"> External Interrupt input, each bit of which can be used to connect an interrupt source. Note: <ul style="list-style-type: none"> clic_irq signal will not be synced inside the Core, so these signals need to be synced at the SoC level if it is from different clock domain at SoC. Please refer to Section 2.9. for the details of interrupt. |

3.4. Bus Interfaces

N300 Series Core support several bus interfaces, including:

- (Configurable) ILM Master Interface.
- (Configurable) DLM Master Interface.
- (Configurable) PPI Interface.
- (Configurable) FIO Interface.
- (Configurable) Instruction fetch Interface for I-Cache.
- MEM (System Memory) Interface.

3.4.1. ILM and DLM Interface

ILM interface is used to access Instruction Local Memory, and DLM interface is used to access Data Local Memory. Both the ILM and DLM interface can be configured as AHB-Lite or SRAM protocol interface. If configured as AHB-Lite, it should be noted:

- If the DLM AHB-Lite and ILM AHB-Lite interfaces need to be arbitrated at SoC level, the SoC need to make sure DLM interface have the higher priority than ILM interface.
- N300 Series provide a configurable option “N300_CFG_ILM_DLM_EXCLUSIVE”, if this option is configured, the Core will guarantee the DLM AHB-Lite and ILM AHB-Lite interfaces will not issue transactions at the same cycle. This configuration can easy the SoC to just use the simple MUX to arbitrate the ILM and DLM AHB-Lite interfaces, without worrying the priority.

3.4.1.1 ILM Master Interface

ILM interface can be configured as AHB-Lite or SRAM protocol.

- If it is configured as AHB-Lite, the signal list is as shown in Table 3-6.
- If it is configured as SRAM, the signal list is as shown in Table 3-7.

(Note: ILM SRAM should keep the dout unchanged before the next request coming.)

Table 3-6 ILM AHB-Lite signals

| Signal Name | Direction | Width | Description |
|-------------|-----------|-------|-------------|
|-------------|-----------|-------|-------------|

| | | | |
|---------------|--------|--------------------------|---|
| ilm_htrans | Output | 2 | <ul style="list-style-type: none"> ■ AHB-Lite protocol's HTRANS signal. ■ Note: in ILM interface, only the IDLE and NONSEQUENTIAL transactions will be issued. |
| ilm_haddr | Output | Depends on configuration | <ul style="list-style-type: none"> ■ AHB-Lite protocol's HADDR signal. |
| ilm_hsize | Output | 3 | <ul style="list-style-type: none"> ■ AHB-Lite protocol's HSIZE signal. ■ Note: in ILM interface, only the 32bits (HSIZE is 3'b010) transactions will be issued. |
| ilm_hburst | Output | 3 | <ul style="list-style-type: none"> ■ AHB-Lite protocol's HBURST signal. ■ Note: in ILM interface, only the SINGLE (HBURST is 3'b000) transactions will be issued. |
| ilm_hprot | Output | 4 | <ul style="list-style-type: none"> ■ AHB-Lite protocol's HPROT signal. ■ Note: in ILM interface, <ul style="list-style-type: none"> ● HPROT[3] is tied to 1, means Cacheable. ● HPROT[2] is tied to 0, means Non-bufferable. ● HPROT[1] can be 1 (Machine Mode) or 0 (User Mode). ● HPROT[0] can be 1 (Data access) or 0 (Instruction access). |
| ilm_hrdata | Input | 32 | <ul style="list-style-type: none"> ■ AHB-Lite protocol's HRDATA signal. |
| ilm_hresp | Input | 2 | <ul style="list-style-type: none"> ■ AHB-Lite protocol's HRESP signal. ■ Note: in ILM interface, only the OKAY and ERROR response will be supported. |
| ilm_hready | Input | 1 | <ul style="list-style-type: none"> ■ AHB-Lite protocol's HREADY signal. |
| ilm_hwrite | output | 1 | <ul style="list-style-type: none"> ■ AHB-Lite protocol's HWRITE signal. |
| ilm_hmastlock | output | 1 | <ul style="list-style-type: none"> ■ AHB-Lite protocol's HLOCK signal. |
| ilm_hwdata | output | 32 | <ul style="list-style-type: none"> ■ AHB-Lite protocol's HWDATA signal. |

Table 3-7 ILM SRAM signals

| Signal Name | Direction | Width | Description |
|-------------|-----------|--------------------------|--|
| ilm_cs | output | 1 | <ul style="list-style-type: none"> ■ SRAM's CS signal. |
| ilm_we | output | 1 | <ul style="list-style-type: none"> ■ SRAM's Write Enable signal. |
| ilm_wem | output | 4 | <ul style="list-style-type: none"> ■ SRAM's Write Byte Mask signal. |
| ilm_addr | output | Depends on configuration | <ul style="list-style-type: none"> ■ SRAM's ADDR signal. |
| ilm_din | output | 32 | <ul style="list-style-type: none"> ■ SRAM's Write Data signal. |
| ilm_dout | input | 32 | <ul style="list-style-type: none"> ■ SRAM's Read Data signal. |
| clk_ilm | output | 1 | <ul style="list-style-type: none"> ■ SRAM's CLK signal |

3.4.1.2 DLM Master Interface

DLM interface can be configured as AHB-Lite or SRAM protocol.

- If it is configured as AHB-Lite, the signal list is as shown in Table 3-8.
- If it is configured as SRAM, the signal list is as shown in Table 3-9.

(Note: DLM SRAM should keep the dout unchanged before the next request coming.)

Table 3-8 DLM AHB-Lite signals

| Signal Name | Direction | Width | Description |
|---------------|-----------|--------------------------|---|
| dln_htrans | Output | 2 | <ul style="list-style-type: none"> ■ AHB-Lite protocol's HTRANS signal. ■ Note: in DLM interface, only the IDLE and NONSEQUENTIAL transactions will be issued. |
| dln_hwrite | Output | 1 | <ul style="list-style-type: none"> ■ AHB-Lite protocol's HWRITE signal. |
| dln_haddr | Output | Depends on configuration | <ul style="list-style-type: none"> ■ AHB-Lite protocol's HADDR signal. |
| dln_hsize | Output | 3 | <ul style="list-style-type: none"> ■ AHB-Lite protocol's HSIZE signal. ■ Note: in DLM interface, the 8bits, 16bits or 32bits transactions will be issued. |
| dln_hburst | Output | 3 | <ul style="list-style-type: none"> ■ AHB-Lite protocol's HBURST signal. ■ Note: in DLM interface, only the SINGLE (HBURST is 3'b000) transactions will be issued. |
| dln_hprot | Output | 4 | <ul style="list-style-type: none"> ■ AHB-Lite protocol's HPROT signal. ■ Note: in DLM interface, <ul style="list-style-type: none"> ● HPROT[3] is tied to 1, means Cacheable. ● HPROT[2] is tied to 0, means Non-bufferable. ● HPROT[1] can be 1 (Machine Mode) or 0 (User Mode). ● HPROT[0] is tied to 1 (Data access). |
| dln_hmastlock | output | 1 | <ul style="list-style-type: none"> ■ AHB-Lite protocol's HLOCK signal. |
| dln_master | Output | 2 | <ul style="list-style-type: none"> ■ This is not AHB-Lite standard signal. ■ Note: in DLM interface, the value of this signal can be 2'b01 (Data access under debug mode) or 2'b00 (Regular data access). |
| dln_hwdata | Output | 32 | <ul style="list-style-type: none"> ■ AHB-Lite protocol's HWDATA signal. |
| dln_hrdata | Input | 32 | <ul style="list-style-type: none"> ■ AHB-Lite protocol's HRDATA signal. |
| dln_hresp | Input | 2 | <ul style="list-style-type: none"> ■ AHB-Lite protocol's HRESP signal. ■ Note: in DLM interface, only the OKAY and ERROR response will be supported. |

| | | | |
|------------|-------|---|--------------------------------------|
| d1m_hready | Input | 1 | ■ AHB-Lite protocol's HREADY signal. |
|------------|-------|---|--------------------------------------|

Table 3-9 DLM SRAM signals

| Signal Name | Direction | Width | Description |
|-------------|-----------|--------------------------|------------------------------------|
| d1m_cs | output | 1 | ■ SRAM's CS signal. |
| d1m_we | output | 1 | ■ SRAM's Write Enable signal. |
| d1m_wem | output | 4 | ■ SRAM's Write Enable Mask signal. |
| d1m_addr | output | Depends on configuration | ■ SRAM's ADDR signal. |
| d1m_din | output | 32 | ■ SRAM's Write Data signal. |
| d1m_dout | input | 32 | ■ SRAM's Read Data signal. |
| clk_d1m | output | 1 | ■ SRAM's CLK signal |

3.4.2. MEM Interface

MEM interface is used to access system memory for instruction and data. MEM interface is AHB-Lite protocol interface as shown in Table 3-10.

Table 3-10 MEM signals

| Signal Name | Direction | Width | Description |
|-------------|-----------|-------|--|
| htrans | Output | 2 | <ul style="list-style-type: none"> ■ AHB-Lite protocol's HTRANS signal. ■ Note: in MEM interface, if there is no Cache configured in the core, then only the IDLE and NONSEQUENTIAL transactions will be issued; if there are Cache configured, then there could be BUSY and SEQUENTIAL transactions. |
| hwrite | Output | 1 | ■ AHB-Lite protocol's HWRITE signal. |
| haddr | Output | 32 | ■ AHB-Lite protocol's HADDR signal. |
| hsize | Output | 3 | <ul style="list-style-type: none"> ■ AHB-Lite protocol's HSIZE signal. ■ Note: in MEM interface, the 8bits, 16bits or 32bits transactions will be issued. |
| hburst | Output | 3 | <ul style="list-style-type: none"> ■ AHB-Lite protocol's HBURST signal. ■ Note: in MEM interface, <ul style="list-style-type: none"> ● If the instruction fetch transaction is not from Cache miss, it is SINGLE (HBURST is 3'b000). ● If the data access transaction is not from Cache miss, it is INCR (HBURST is 3'b001). ● If the instruction fetch or data access transaction is from Cache miss, it is INCR8 (HBURST is 3'b101). |
| hprot | Output | 4 | <ul style="list-style-type: none"> ■ AHB-Lite protocol's HPROT signal. ■ Note: in MEM interface, |

| | | | |
|-----------|--------|----|---|
| | | | <ul style="list-style-type: none"> ● HPROT[3] is 1 (Cacheable) if the transaction is from Cache miss; otherwise is 0 (Non-Cacheable). ● HPROT[2] is 1 (Bufferable) if the transaction is from Cache miss; otherwise is 0 (Non-Bufferable). ● HPROT[1] can be 1 (Machine Mode) or 0 (User Mode). ● HPROT[0] can be 1 (Data access) or 0 (Instruction fetch). |
| hmastlock | output | 1 | ■ AHB-Lite protocol's HLOCK signal. |
| master | Output | 2 | ■ This is not AHB-Lite standard signal. ■ Note: in MEM interface, the value of this signal can be 2'b01 (Data access under debug mode), 2'b00 (Regular data access), or 2'b10 (Regular instruction fetch). |
| hwddata | Output | 32 | ■ AHB-Lite protocol's HWDATA signal. |
| hrdata | Input | 32 | ■ AHB-Lite protocol's HRDATA signal. |
| hresp | Input | 2 | ■ AHB-Lite protocol's HRESP signal. |
| hready | Input | 1 | ■ AHB-Lite protocol's HREADY signal. |

3.4.3. Instruction Fetch Bus Interface for I-Cache

A dedicate I-Cache bus is configurable for instruction fetch, if not configured, instruction fetch and data load will share the system MEM bus. I-Cache bus is AHB-Lite protocol interface as shown below.

Table 3-11 I-Cache bus signals

| Signal Name | Direction | Width | Description |
|------------------|-----------|-------|---|
| icache_htrans | Output | 2 | ■ AHB-Lite protocol's HTRANS signal. |
| icache_hwrite | Output | 1 | ■ AHB-Lite protocol's HWRITE signal. |
| icache_haddr | Output | 32 | ■ AHB-Lite protocol's HADDR signal. |
| icache_hsize | Output | 3 | ■ AHB-Lite protocol's HSIZE signal. ■ Note: in MEM interface, the 8bits, 16bits or 32bits transactions will be issued. |
| icache_hburst | Output | 3 | ■ AHB-Lite protocol's HBURST signal. |
| icache_hprot | Output | 4 | ■ AHB-Lite protocol's HPROT signal. |
| icache_hmastlock | output | 1 | ■ AHB-Lite protocol's HLOCK signal. |
| icache_master | Output | 2 | ■ This is not AHB-Lite standard signal. |
| icache_hwddata | Output | 32 | ■ AHB-Lite protocol's HWDATA signal. |
| icache_hrdata | Input | 32 | ■ AHB-Lite protocol's HRDATA signal. |
| icache_hresp | Input | 2 | ■ AHB-Lite protocol's HRESP signal. |

| | | | |
|---------------|-------|---|--------------------------------------|
| icache_hready | Input | 1 | ■ AHB-Lite protocol's HREADY signal. |
|---------------|-------|---|--------------------------------------|

3.4.4. PPI Interface

The configurable PPI (Private Peripheral Interface) is used to access private peripherals. PPI is APB protocol interface as shown in Table 3-12.

Table 3-12 PPI signals

| Signal name | Dir | Bit width | Description |
|-----------------|--------|-----------|---|
| ppi_clk_en | Input | 1 | ■ PPI interface clock ratio |
| ppi_hmasterlock | Output | 1 | ■ AHB-Lite protocol HMASTERLOCK signal. |
| ppi_hburst | Output | 1 | ■ AHB-Lite protocol HBURST signal. |
| ppi_htrans | Output | 2 | ■ AHB-Lite protocol HTRANS signal. |
| ppi_hwrite | Output | 1 | ■ AHB-Lite protocol HWRITE signal. |
| ppi_haddr | Output | 32 | ■ AHB-Lite protocol HADDR signal. |
| ppi_hsize | Output | 3 | ■ AHB-Lite protocol HSIZE signal. |
| ppi_hprot | Output | 4 | ■ AHB-Lite protocol HPROT signal. ° |
| ppi_hwdata | Output | 32 | ■ AHB-Lite protocol HWDATA signal. |
| ppi_hrdata | Input | 32 | ■ AHB-Lite protocol HRDATA signal. |
| ppi_hresp | Input | 2 | ■ AHB-Lite protocol HRESP signal. ■ Note: support OKAY and ERROR only. |
| ppi_hready | Input | 1 | ■ AHB-Lite protocol HREADY signal. |

3.4.5. FIO Interface

The configurable FIO (Fast-IO) interface is used to access private fast peripherals, e.g., GPIO for fast IO manipulation. FIO is simple zero-cycle protocol interface as shown in Table 3-13.

Table 3-13 FIO signals

| Signal Name | Direction | Width | Description |
|---------------|-----------|-------|---|
| fio_cmd_valid | Output | 1 | ■ Indicate this is a valid transaction. |
| fio_cmd_addr | Output | 32 | ■ Indicate the address of transaction. |

| | | | |
|---------------|--------|----|---|
| fio_cmd_read | Output | 1 | ■ Indicate this is a read (1'b1) or write (1'b0). |
| fio_cmd_dmode | Output | 1 | ■ Indicate this transaction is accessed under debug mode. |
| fio_cmd_mmode | Output | 1 | ■ Indicate this transaction is accessed under machine mode. |
| fio_cmd_wdata | Output | 32 | ■ Indicate the write data of transaction. |
| fio_cmd_wmask | Output | 4 | ■ Indicate the write mask of transaction. |
| fio_rsp_rdata | Input | 32 | ■ The read-data returned from peripheral. ■ Note: this read-data must take effect at the same cycle as fio_icb_cmd_valid is high. |
| fio_rsp_err | Input | 1 | ■ The error flag returned from peripheral. ■ Note: this read-data must take effect at the same cycle as fio_icb_cmd_valid is high. |

3.5. NICE Interface

The configurable NICE interface is used to allow user to extend the custom instructions according to their applications. Please refer to another document <Nuclei NICE Extension> for the details. User can easily get the copy from “Nuclei User Center” website <http://user.nucleisys.com>.

3.6. Trace Interface

The Trace interface is used to output the internal key information from the Core, as shown in Table 3-14.

Table 3-14 Trace Interface signals

| Signal Name | Direction | Width | Description |
|------------------|-----------|-------|--|
| trace_ivalid | Output | 1 | ■ Indicate there is a valid instruction is committing or entering trap (exception, NMI, and interrupts). |
| trace_iexception | Output | 1 | ■ Indicating the Core is entering exception or NMI mode. |
| trace_interrupt | Output | 1 | ■ Indicating the Core is entering interrupt mode. |
| trace_cause | Output | 32 | ■ Indicating the cause of trap (same as mcause). |
| trace_tval | Output | 32 | ■ Indicating the value of exception (same as mtval). |
| trace_iaddr | Output | 32 | ■ Indicating the PC of current instruction. |
| trace_instr | Output | 32 | ■ Indicating the instruction code of current |

| | | | |
|-----------------|--------|---|---|
| | | | instruction. |
| trace_priv | Output | 2 | ■ Indicating the current privilege mode. |
| trace_bjp_taken | Output | 1 | ■ Indicating the final result of branch or jump instruction is taken. |
| trace_dmode | Output | 1 | ■ Indicating whether current instruction is in debug mode. |
| trace_cmt_en | Output | 1 | ■ Indicating a valid instruction is committing. |

3.7. I-Cache SRAM Interface

The I-Cache SRAM interface is the interface of Data RAM and Tag RAM used in Instruction Cache, as shown in Table 3-15.

(Note: I-Cache SRAM should keep the dout unchanged before the next request coming.)

Table 3-15 I-Cache SRAM Interface

| Signal Name | Direction | Width | Description |
|---------------------|-----------|-------|---|
| icache_disable_init | Input | 1 | <ul style="list-style-type: none"> ■ If this signal is 0, normally after reset, the I-Cache's Tag RAM will be cleared to zero with hundreds or thousands of cycles (depends on the cache size). During this period, all the instruction fetches in the I-Cache/System-Bus path will be blocked. ■ If this signal is 1, then the I-Cache's Tag RAM clear-to-zero operations will be skipped. |
| clk_icache_tagn | Output | 1 | ■ Tag RAMn's CLK signal |
| icache_tagn_cs | Output | 1 | ■ CS signal of Tag RAMn |
| icache_tagn_we | Output | 1 | ■ Write enable of Tag RAMn |
| icache_tagn_addr | Output | IM | ■ Address of Tag RAMn |
| icache_tagn_din | Output | IM | ■ Write data of Tag RAMn |
| icache_tagn_dout | Input | IM | ■ Read data of Tag RAMn |
| clk_icache_datan | Output | 1 | ■ Data RAMn's CLK signal |
| icache_datan_cs | Output | 1 | ■ CS signal of Data RAMn |
| icache_datan_we | Output | 1 | ■ Write enable of Data RAMn |
| icache_datan_addr | Output | IM | ■ Address of Data RAMn |
| icache_datan_din | Output | IM | ■ Write data of Data RAMn |
| icache_datan_dout | Input | IM | ■ Read data of Data RAMn |

Note:

- IM indicates Implementation dependent, replies on configurations;
- I-Cache has 2 ways, so n can be “0/1” accordingly;

3.8. D-Cache SRAM Interface

The D-Cache SRAM interface is the interface of Data RAM and Tag RAM used in Data Cache, as shown below.

(Note: D-Cache SRAM should keep the dout unchanged before the next request coming.)

Table 3-16 D-Cache SRAM Interface

| Signal Name | Direction | Width | Description |
|---------------------|-----------|-------|---|
| dcache_disable_init | Input | 1 | <ul style="list-style-type: none"> ■ If this signal is 0, normally after reset, the D-Cache's Tag RAM will be cleared to zero with hundreds or thousands of cycles (depends on the cache size). During this period, all the instruction fetches in the D-Cache/System-Bus path will be blocked. ■ If this signal is 1, then the D-Cache's Tag RAM clear-to-zero operations will be skipped. |
| clk_dcache_wn_tram | Output | 1 | ■ Tag RAMn's CLK signal |
| dcache_wn_tram_cs | Output | 1 | ■ CS signal of Tag RAMn |
| dcache_wn_tram_we | Output | 1 | ■ Write enable of Tag RAMn |
| dcache_wn_tram_addr | Output | IM | ■ Address of Tag RAMn |
| dcache_wn_tram_din | Output | 32 | ■ Write data of Tag RAMn |
| dcache_wn_tram_dout | Input | 32 | ■ Read data of Tag RAMn |
| clk_dcache_dram_bn | Output | 1 | ■ Data RAMn's CLK signal |
| dcache_dram_bn_cs | Output | 1 | ■ CS signal of Data RAMn |
| dcache_dram_bn_we | Output | 1 | ■ Write Enable of Data RAMn |
| dcache_dram_bn_wem | Output | 4 | ■ Write Byte Mask of Data RAMn |
| dcache_dram_bn_addr | Output | IM | ■ Address of Data RAMn |
| dcache_dram_bn_din | Output | 32 | ■ Write data of Data RAMn |
| dcache_dram_bn_dout | Input | 32 | ■ Read data of Data RAMn |

Note:

- IM indicates Implementation dependent, replies on configurations;
- D-Cache has 2 ways, so n can be “0/1” accordingly;

3.9. Other Functional Interface

Table 3-17 Other Interface signals

| Signal Name | Direction | Width | Description |
|----------------|-----------|-------|--|
| tx_evt | Output | 1 | <ul style="list-style-type: none"> ■ N300 Series Core use this txevt output a pulse signal as the transmitting Event signal. ■ Please refer to “Nuclei_RISCV_ISA_Spec” for more details of CSR register txevt. |
| rx_evt | Input | 1 | <ul style="list-style-type: none"> ■ The receiving Event as the event to wake up Core from WFE mode. ■ Please refer to Section 2.13. for more details of WFE. |
| mtime_toggle_a | Input | 1 | <ul style="list-style-type: none"> ■ The mtime_toggle_a is a periodic pulse signal (normally as System Tick) from the SoC system, and used to drive the counter of the internal TIMER unit inside the Core. ■ Note: <ul style="list-style-type: none"> ● This signal is treated as an asynchronous input signal, and is synchronized within the Core (using several DFF synchronizers). ● After the synchronization, both the rising edge and falling edge of the signal are sampled by the core_clk_aon of the Core, and any detected edge will trigger the TIMER counter to increment. ● It is recommended that use the output of the register driven by the “slow clock” (e.g., rtc_clk, whose frequency is the divided 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 -5. 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. |
| dbg_toggle_a | Input | 1 | <ul style="list-style-type: none"> ■ The dbg_toggle_a is a periodic pulse signal from the SoC system, and used to drive the time-out counter of the DEBUG unit inside the Core. ■ The time-out counter of DEBUG unit is used to protect the case that if the JTAG Debugger Probe is unexpectedly pulled out which leave the DEBUG unit in an uncertain state. ■ Note: |

| | | | |
|-------------------|--------|----|---|
| | | | <ul style="list-style-type: none"> ● This DEBUG time-out protection feature is configurable. Hence, this signal will only be there if this feature is configured. ● This signal is treated as an asynchronous input signal, and is synchronized within the Core (using several DFF synchronizers). ● In order to make the time-out upper limit to around 170-320ms, it is recommended to use 25kHz~50kHz system real-time-clock as the slow clock to generate this dbg_toggle_a signal, the generation scheme of which is similar to Figure 3-5. |
| hart_id | Input | 32 | <ul style="list-style-type: none"> ■ The Core's HART ID indication signal, when integrated in SoC, this input can be tied to a specific constant value, and the value of it will be reflected in CSR register mhartid inside the Core. ■ In single Core case, this signal should be tied as zero. |
| reset_vector | Input | 32 | <ul style="list-style-type: none"> ■ This signal is to indicate the PC value of the first instruction to be fetched after reset. ■ User can use this signal at SoC level to control the PC address of first instruction executed after reset. |
| hart_halted | Output | 1 | <ul style="list-style-type: none"> ■ If this output signal is 1, it is indicating the Core is under debug mode. |
| i_dbg_stop | Input | 1 | <ul style="list-style-type: none"> ■ If this input signal is 1, then the Core's Debug functionality will be disabled, and the external Debug Hardware Probe cannot debug the Core through JTAG interface. |
| sysrstreq | Output | 1 | <ul style="list-style-type: none"> ■ This output signal is the System Reset request generated from the Core. The SoC integrator can use this signal to trigger the Core's core_reset_n to reset the Core except the JTAG logics (Note: 1. please must not trigger por_reset_n; 2. please must not disable core_clk_aon; 3. please must delay it one cycle on SoC). |
| core_wfi_mode | Output | 1 | <ul style="list-style-type: none"> ■ If this signal is 1, then indicating the Core is under sleep mode. ■ Please refer to Section 2.13. for more details of sleep modes. |
| core_sleep_value | Output | 1 | <ul style="list-style-type: none"> ■ When the core_wfi_mode signal is 1, this signal is to indicate the shallow sleep or deep sleep mode. ■ Please refer to Section 2.13. for more details of sleep modes. |
| override_dm_sleep | Input | 1 | <ul style="list-style-type: none"> ■ This signal can be configured in SoC to enable the clock of debug module when deep sleep mode (core_sleep_value=1), to support debug in low-power. |
| dbg_stoptime | Output | 1 | <ul style="list-style-type: none"> ■ dcsr[9], stoptime in debug control and status register: 0: increase timers as usual; |

| | | | |
|---------------|-------|---|--|
| | | | 1: don't increase timer while in debug mode; Can be used to stop the SoC timer if necessary. |
| stop_on_reset | Input | 1 | ■ This signal is used to STOP the core after reset, to allow other master such as DMA to preload the ILM/DLM through the slave port, this signal is only effective after reset once, it cannot stop the core during running. |

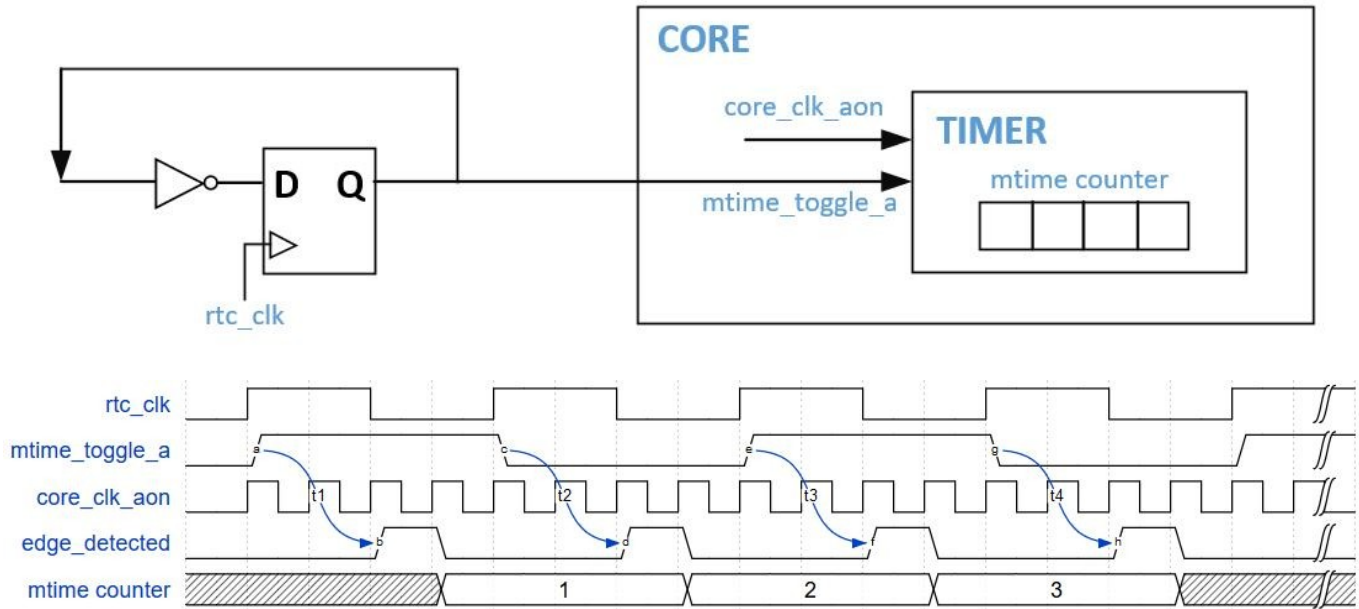


Figure 3-5 mtime_toggle_a signal generation

4. Configurable Options

N300 Series Core is fully configurable. The configurable options are as shown in Table 4-18.

Note: about how to configure the processor IP package with GUI tools, please refer to another document <Nuclei_Processor_Integration_Guide> for the details. User can easily get the copy from “Nuclei User Center” website <http://user.nucleisys.com>.

Table 4-18 The configurable option of N300 Series

| Categories | Macro | Description |
|---------------------------|------------------------|---|
| CORE PFX | CORE Prefix | ■ To configure the Prefix name of the generated files and modules, default prefix name is 'n300'. |
| User Mode | N300_CFG_HAS_UMODE | ■ This Macro configures to have the user mode. |
| 2-cycle Multiplier | N300_CFG_MUL_2CYC | ■ To configure 2-cycle multiplier for better timing, default is 1-cycle multiplier. |
| ADV_DIV | N300_CFG_ADV_DIV | ■ If N300_CFG_ADV_DIV is configured, then SRT-4 based hardware divider are configured, or 33-cycle divider is used. |
| PMP | N300_CFG_HAS_PMP | ■ This Macro configures to have the PMP feature. ■ Note: this option only appeared if the User Mode have been configured. |
| | N300_CFG_PMP_ENTRY_NUM | ■ This Macro configures the PMP entries number. ● 8: Means PMP has 8 entries. ● 16: Means PMP has 16 entries. |
| | N300_CFG_PMP_GRAIN | ■ To configure the minimum granularity supported for PMP. ■ G is the configured num, the minimum granularity is $2^{(G+2)}$ byte. ■ Such as, G=0, which sets the minimum granularity $2^{(G+2)}=4B$; G=10, which sets the minimum granularity $2^{(G+2)}=4KB$; ■ G can be 0-10, then the minimum granularity can be 4B to 4KB. |
| | N300_CFG_HAS_PMP_TOR | ■ To configure PMP TOR (Top Of Range) feature: $pmpaddr_{i-1} \leq y <$ |

| | | |
|-----------------------|-----------------------------|---|
| | | pmpaddr _i |
| Debug Features | N300_CFG_HAS_DEBUG | <ul style="list-style-type: none"> ■ This Macro configures to have the Debug functionalities. ■ Note: the Debug unit cost about 4K Gates resource. |
| | N300_CFG_DEBUG_BASE_ADDR | <ul style="list-style-type: none"> ■ This Macro to configure the base address of the Debug unit. ■ Note: the Debug unit will occupy 4K address space starting from its base address. |
| | N300_CFG_DEBUG_TRIGM_NUM | <ul style="list-style-type: none"> ■ This Macro to configure the number of Hardware Trigger (2/4/8). ■ Note: each Trigger cost about 64bits DFFs resource. |
| | N300_CFG_DEBUG_PROGBUF_SIZE | <ul style="list-style-type: none"> ■ This Macro to configure the number of Program Buffer (2~16). |
| | N300_CFG_HAS_SBA | <ul style="list-style-type: none"> ■ This Macro configures to have SBA (System Bus Access) feature, with this feature the Debugger can directly access the memory without halting the Core. |
| | N300_CFG_DEBUG_TIMEOUT | <ul style="list-style-type: none"> ■ This Macro configures to have “DEBUG time-out protection” feature. Please refer to the dbg_toggle_a signal in Table 3 - 17 for more details of this feature. |
| | N300_CFG_DEBUG_COUNTLEN | <ul style="list-style-type: none"> ■ This Macro to configure the time-out counter’s width of “DEBUG time-out protection” feature. ■ Note: the timer-out upper limit is calculated by $2^{N300_CFG_DEBUG_COUNTLEN}/(2 \times FREQ_dbg_toggle_a)$. User should configure this Macro to make the upper limit to around 170-320ms. |
| I-Cache | N300_CFG_HAS_ICACHE | <ul style="list-style-type: none"> ■ This Macro configures to have I-Cache. |
| | N300_CFG_ICACHE_ADDR_WIDTH | <ul style="list-style-type: none"> ■ This Macro to configure the I-Cache Size. (Range is [2-20], I-cache is fixed as 2-way) ■ It is using the ADDR_WIDTH as the Cache Size metric, for example, if the Cache Size wants to be 1Kbytes, then the ADDR_WIDTH should be configured as 10. |
| | N300_CFG_HAS_ICACHE_BUS | <ul style="list-style-type: none"> ■ To configure a dedicate AHB-Lite bus for I-Cache to fetch instructions. ■ If not configured, instruction |

| | | |
|---------------------|---|--|
| | | fetch and data load will share the system bus. |
| | N300_CFG_SCRATCHPAD_MODE | <ul style="list-style-type: none"> ■ This Macro to configure the I-Cache has Scratchpad Mode. Please refer to Section 2.4. for the details of Scratchpad Mode. ■ Note: this configuration only allowed when ILM is not configured. |
| | N300_CFG_SCRATCHPAD_BASE_ADDR | <ul style="list-style-type: none"> ■ This Macro to configure the base address of the Scratchpad. |
| D-Cache | N300_CFG_HAS_DCACHE | <ul style="list-style-type: none"> ■ This Macro configures to have D-Cache. |
| | N300_CFG_DCACHE_ADDR_WIDTH | <ul style="list-style-type: none"> ■ This Macro to configure the address space of Dcache. (<i>Range is [7-17], D-cache is fixed as 2-way</i>) ■ For example, if the ADDR_WIDTH is 10, then Dcache size is 1KB; |
| | N300_CFG_DEVICE_REGION_NUM | <ul style="list-style-type: none"> ■ To configure the num of Device Region; |
| | N300_CFG_DEVICE_REGIONn_BASE | <ul style="list-style-type: none"> ■ Configure Device Region <i>n</i> base addr (<i>n</i> can be 0-7); |
| | N300_CFG_DEVICE_REGIONn_MASK | <ul style="list-style-type: none"> ■ Configure Device Region <i>n</i> addr mask (<i>n</i> can be 0-7); ■ For example, if BASE is 0x10000000, MASK is 0xffff000, then this REGION will be 0x10000000 - 0x10000fff; |
| | N300_CFG_NC_REGION_NUM | <ul style="list-style-type: none"> ■ To configure the num of Non-Cacheable region; |
| | N300_CFG_NC_REGIONn_BASE | <ul style="list-style-type: none"> ■ Configure Non-Cacheable Region <i>n</i> base addr (<i>n</i> can be 0-7); |
| | N300_CFG_NC_REGIONn_MASK | <ul style="list-style-type: none"> ■ Configure Non-Cacheable Region <i>n</i> addr mask (<i>n</i> can be 0-7); ■ For example, if BASE is 0x10000000, MASK is 0xffff000, then this REGION will be 0x10000000 - 0x10000fff; |
| Local Memory | N300_CFG_HAS_ILM | <ul style="list-style-type: none"> ■ This Macro configures to have ILM. |
| | N300_CFG_LM_ITF_TYPE_AHB N300_CFG_LM_ITF_TYPE_SRAM | <ul style="list-style-type: none"> ■ This Macro to configure the interface protocol of LM interfaces as the AHB-Lite or SRAM Style. |
| | N300_CFG_ILM_BASE_ADDR | <ul style="list-style-type: none"> ■ This Macro to configure the base address of the ILM. |
| | N300_CFG_ILM_ADDR_WIDTH | <ul style="list-style-type: none"> ■ This Macro to configure the address space of ILM. (<i>Range is [7-31]</i>) ■ For example, if the ADDR_WIDTH is 20, and the BASE_ADDR is 0x1000_0000, then the address space of ILM is 0x1000_0000 |

| | | |
|------------|----------------------------|---|
| | | ~0x100F_FFFF. |
| | N300_CFG_HAS_DLM | ■ This Macro configures to have DLM. |
| | N300_CFG_DLM_BASE_ADDR | ■ This Macro to configure the base address of the DLM. |
| | N300_CFG_DLM_ADDR_WIDTH | ■ This Macro to configure the address space of DLM. (<i>Range is [7-31]</i>) ■ For example, if the ADDR_WIDTH is 20, and the BASE_ADDR is 0x1000_0000, then the address space of DLM is 0x1000_0000 ~0x100F_FFFF. |
| | N300_CFG_ILM_DLM_EXCLUSIVE | ■ If this option is configured, the Core will guarantee the DLM AHB-Lite and ILM AHB-Lite interfaces will not issue transactions at the same cycle. ■ This Feature is Valid only when CFG_LSU_ACCESS_ILM option is not configured. ■ Please refer to Section 3.4.1. for more details. |
| | N300_CFG_LSU_ACCESS_ILM | ■ If this option is configured, the Data accessing can directly goes to ILM interface if the accessing address is within ILM space. ■ Please refer to Section 2.6. for more details. |
| | N300_CFG_LM_HAS_SLAVE_PORT | ■ AHB-Lite slave port for ILM/DLM ■ Then DMA can also preload ILM/DLM with this slave port when ILM/DLM is configured as SRAM interface |
| PPI | N300_CFG_HAS_PPI | ■ This Macro configures to have PPI. |
| | N300_CFG_PPI_BASE_ADDR | ■ This Macro to configure the base address of the PPI interface. |
| | N300_CFG_PPI_ADDR_WIDTH | ■ This Macro to configure the address space of PPI. ■ For example, if the ADDR_WIDTH is 20, and the BASE_ADDR is 0x1000_0000, then the address space of PPI is 0x1000_0000 ~0x100F_FFFF. |
| FIO | N300_CFG_HAS_FIO | ■ This Macro configures to have FIO. |
| | N300_CFG_FIO_BASE_ADDR | ■ This Macro to configure the base address of the FIO interface. |
| | N300_CFG_FIO_WIDTH | ■ This Macro to configure the address space of FIO. ■ For example, if the ADDR_WIDTH is 20, and the BASE_ADDR is 0x1000_0000, then the address |

| | | |
|---------------------|-----------------------------|---|
| | | space of FIO is 0x1000_0000 ~0x100F_FFFF. |
| ECLIC | N300_CFG_CLIC_BASE_ADDR | ■ This Macro to configure the base address of the ECLIC. Please refer to Section 2.6. for more details. |
| | N300_CFG_CLIC_IRQ_NUM | ■ This Macro to configure the number of external interrupts; it can be up to 1005 interrupts. |
| | N300_CFG_CLICINTCTLBITS | ■ This Macro to configure the bits width (range from 1 to 8) of level registers in ECLIC. ■ For example, if this Macro is configured as 3, then ECLIC can support 8 levels; if this Macro is configured as 8, then ECLIC can support 256 levels. |
| | N300_CFG_CLIC_CUT_TIMING | ■ This Macro to add one more register flop stage in ECLIC to easy the timing of interrupt arbitrations when big num of interrupts configured, but added 1 more cycle for interrupt latency. |
| TIMER | N300_CFG_TMR_BASE_ADDR | ■ This Macro to configure the base address of the TIMER. Please refer to Section 2.6. for more details. |
| Timing Boost | N300_CFG_HAS_PREFETCH | ■ If this Macro is configured, then I-Fetch unit will be equipped with a Prefetch Unit for ILM/I-Cache/BIU. ■ Note: this option will help the timing path of “from ILM/I-Cache/BIU to Instruction-fetch interface”. ■ Note: this option will add around 2K Gates resource and around 8% performance drop. |
| | N300_CFG_DELAY_BRANCH_FLUSH | ■ If this Macro is configured, then the branch mis-prediction flush will be added with 1 more cycle penalty, it will help the timing path of “from ALU to branch-resolve to Instruction-fetch interface”. |
| NICE | N300_CFG_HAS_NICE | ■ This Macro configures to have NICE feature, to allow user to extend their custom instructions. ■ Note: if this Macro is configured, then N300_CFG_REGFILE_2WP must also be configured. |
| | N300_CFG_NICE_64BITS | ■ To configure 64-bit NICE wide operands. |

| | | |
|------------|---------------------|--|
| FPU | N300_CFG_FPU_SINGLE | ■ This Macro configures to have single-precision floating point support. |
| | N300_CFG_FPU_DOUBLE | ■ This Macro configures to have double-precision floating point support. |
| | N300_CFG_3CYC_FPU | ■ This Macro configures to have floating point MAC unit as 3 cycles latency (better timing and worse performance), otherwise it is 2 cycles. |
| DSP | N300_CFG_HAS_DSP | ■ This Macro configures to have packed-SIMD DSP support. |